Intel® Visual Compute Accelerator Product Family

Software Guide

Reference for setting up Intel® Visual Compute Accelerator product family software.

Rev 2.3

December 2018

Intel® Server Products and Solutions

<Blank page>

Document Revision History

Date	Revision	Changes
December 2018	2.3	Aligned with Intel® VCA Software 2.3 public release.
July 2018	2.2	Aligned with Intel® VCA Software 2.2 public release.
May 2018	2.1.5	Aligned with Intel® VCA Software 2.1 public release.
Feburary 2018	2.1.1	Added clarifications, details and rearranged sections.
November 2017	2.1	Updated sections A.2 and A.4.2.3.
November 2017	2.0	Aligned with 2.0 VCA software public release. New features: supports XEN and KVM; support Windows OS baremetal images; support blockIO devices; support kernel crash dump; support m.2.
July 2017	1.6	Reorganized chapters and sections. Applied new formatting and edited for clarity.
March 2017	1.5.0	Aligned with 1.5 VCA Software public Release. VCA Software 1.5 is the first release which supports both VCA1283LVV and VCA1585LMV.
October 2016	1.3.0	Aligned with 1.3 VCA Software Public Release.
July 2016	1.2.2	Fixed incorrect documentation in section 7.18; Fixed typos; Changed vca_os_repo to local_repo in section 14 to synch with build_scripts values.
July 2016	1.2.1	Fixed typos and broken references; Added Bridge Configuration section; Expanded vcactl config section
May 2016	1.2.0	Aligned with 1.2 VCA Software Public Release
December 2015	1.01	Minor corrections
November 2015	1.0	First Public Release

Disclaimers

Table of Contents

List of Figures

List of Tables

1. Introduction

This document is intended to be a reference for installing and using Intel® Visual Compute Accelerator (Intel® VCA) product family software.

1.1 Notes on Example Commands

When performing the steps throughout this guide, the following prompts may be encountered.

- Request for user password.

  ```
  [sudo] password for user
  ```

  Enter the user's password to complete the command. Requires that the user be allowed to run commands with `sudo`. This prompt will not necessarily be displayed every time `sudo` is invoked, but is usually displayed upon first use and, by default, after a failure to issue a subsequent `sudo` command within five minutes.

- Check for host authenticity.

  ```
  The authenticity of host 'hostname' can't be established.
  ECDSA key fingerprint is he:x :st:ri:ng: o:f :ch:ar:ac:te:rs.
  Are you sure you want to continue connecting (yes/no)?
  ```

  If this is a trusted remote system, type `yes` and press **<Enter>**. Otherwise, press **<Ctrl+C>** or type `no` and press **<Enter>**.

Finally, please keep in mind that many commands are too long to fit on a single line of this formatted document, but should be entered on the command line without carriage returns or line feeds where this document format arbitrarily places those.

2. Intel® Visual Compute Accelerator Overview

The Intel® Visual Compute Accelerator (Intel® VCA) family equips Intel® Xeon® Scalable processor and Intel® Xeon® processor E5-based platforms with Iris® Pro Graphics and Intel® Quick Sync Video media transcode capabilities. Comprised of three Intel Xeon processors, this PCIe* add-in card delivers outstanding total cost of ownership and is supported by a rich ecosystem of server OEMs, ISVs, and solutions. Applications include:

- Broadcast – Ultra-high channel density with high visual quality.
- Remotely rendered graphics – High video quality, low latency graphics for enterprise productivity and anytime anywhere gaming.
- Multi-party communication – Video-enabled B2B, B2C, and C2C communication with massive scaling

Table 1. Intel® VCA (VCA1283LVV) card specifications

Feature	Description
Form factor	Full-length, full-height, double-width PCIe* card
CPU	(3) Intel® Xeon® E3-1200v4 product family processors, 47 W TDP, 2.9 GHz, 4 cores
Graphics	GT3e, Iris® Pro Graphics P6300, 128 MB eDRAM
Memory	DDR3L (1.35V) ECC SODIMMs, 2 channels per CPU, Up to 32 GB per CPU
PCIe* configuration	Gen3, x16, 4 lanes per CPU
BIOS	16 MB SPI flash
Operating system support on host	• CentOS* 7.2 for Intel® VCA software versions 1.3 and 1.5 • CentOS* 7.3 for Intel® VCA software version 2.0 • CentOS* 7.4 for Intel® VCA software versions 1.3 Hot Fix (HF), 1.5 HF, and 2.1 • Ubuntu* 16.04.3 for Intel® VCA software version 2.1
Operating system support on card	• CentOS* 7.2 for Intel® VCA software versions 1.3, 1.5, and 2.0 • CentOS* 7.4 for Intel® VCA software versions 1.3 HF, 1.5 HF, and 2.1 • Ubuntu* 16.04.3 for Intel® VCA software version 2.1 • Windows Server* 2016 • Windows* 10
Hypervisor support on card	• Xen (Intel® VCA software versions 1.3 or earlier, and 2.0) • KVM (Intel® VCA software versions 1.3 or earlier, and 2.0 or later)

Table 2. Intel® VCA 2 (VCA1585LMV) card specifications

Feature	Description
Form factor	Full-length, full-height, double-width PCIe* card
CPU	(3) Intel® Xeon® E3-1500v5 product family processors, 45 W TDP, 3.0 GHz, 4 cores
Graphics	GT4e, Iris® Pro Graphics P580, 128 MB eDRAM
Memory	DDR4 ECC SODIMMs or non-ECC SODIMMS, 2 channels per CPU, pp to 64 GB per CPU
PCIe* configuration	Gen3, x16, 8 lanes per CPU
BIOS	16 MB SPI flash
Operating system support on host	• CentOS* 7.2 for Intel® VCA software version 1.5 • CentOS* 7.3 for Intel® VCA software version 2.0 • CentOS* 7.4 for Intel® VCA software versions 1.3 Hot Fix (HF), 1.5 HF, 2.1, 2.2 and 2.3 • Ubuntu* 16.04.3 for Intel® VCA software version 2.1, 2.2 and 2.3
Operating system support on card	• CentOS* 7.2 for Intel® VCA software versions 1.5 and 2.0 • CentOS* 7.4 for Intel® VCA software versions 1.3 HF, 1.5 HF, 2.1, 2.2 and 2.3 • Ubuntu* 16.04.3 for Intel® VCA software version 2.1, 2.2 and 2.3 • Windows Server* 2016 • Windows* 10

Feature	Description
Hypervisor support on card	• Xen (Intel® VCA software versions 1.3 or earlier, and 2.0) • KVM (Intel® VCA software versions 1.3 or earlier, and 2.0 or later) • KVMGT (Intel® VCA software versions 2.1 or later)

3. Architecture Overview

The Intel® VCA card consists of three Intel® Xeon® E3 processors with Iris® Pro graphics and Intel® Quick Sync Video. Each Intel Xeon E3 processor is an independent, fully functioning embedded server on the card.

3.1 Hardware Architecture

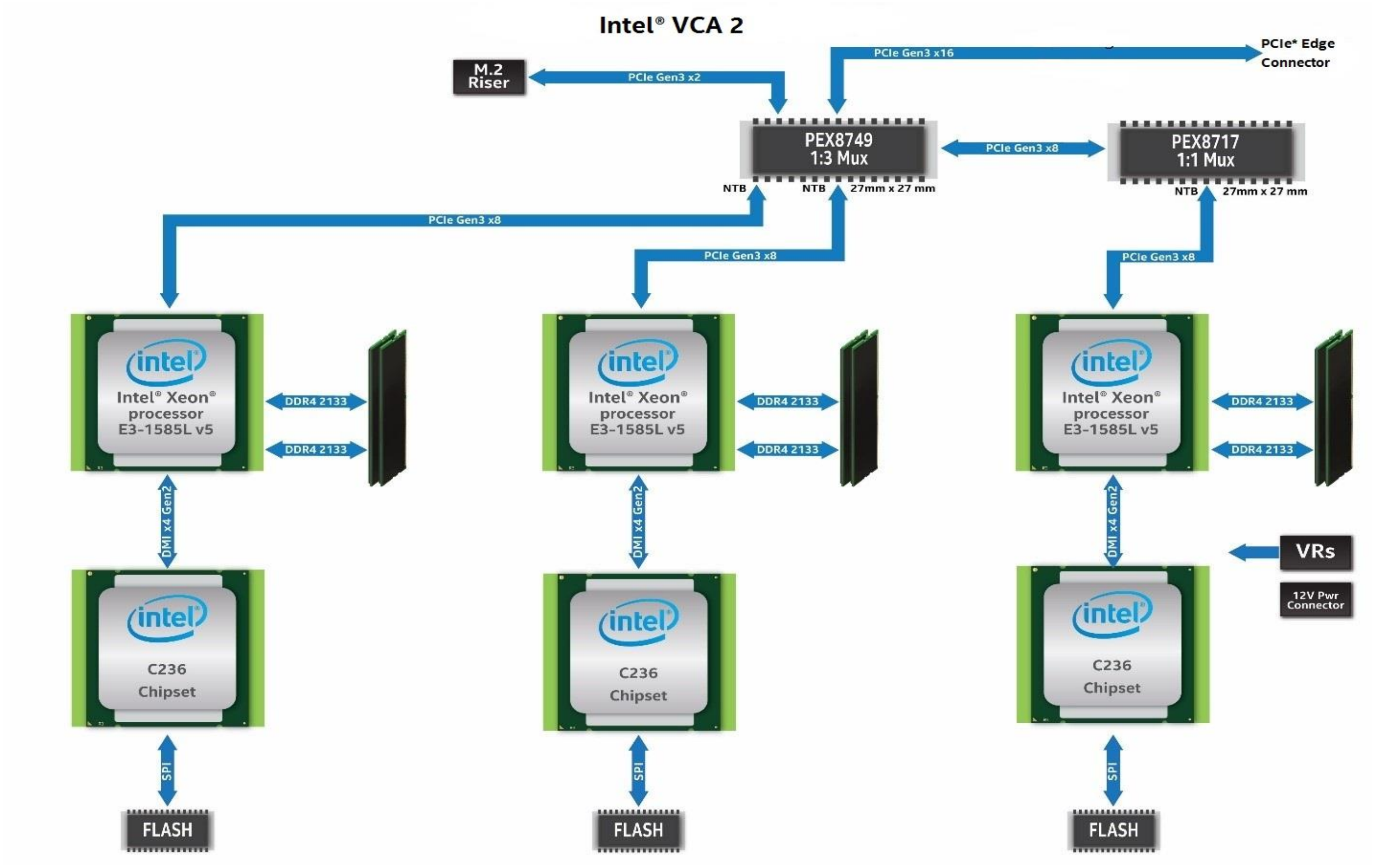


Figure 1. Hardware architecture of Intel® VCA 2 card

Figure 1 illustrates the hardware architecture of Intel® VCA 2 card. The hardware architecture of Intel® VCA is similar. The Intel® VCA connects with the host through a PCIe* bridge (PEX8749). Each Intel® VCA card consists of three Intel Xeon E3-1585L processors (with Iris Pro graphics P580) with PCH, DIMM, and boot flash. These three Intel Xeon E3-1585L processors work independently and can be used as three media/graphics servers. Throughout this document each of these processors is referred to as an Intel® VCA node. Thus, each Intel® VCA card consists of three nodes, which act and process data as separate computers, each governed by its own, independent copy of the operating system. Figure 2 illustrates this concept with two Intel® VCA cards in the host server.

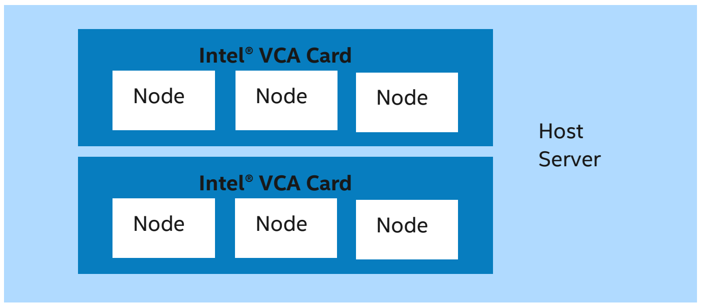


Figure 2. The relation between host server, Intel® VCA card, and Intel® VCA nodes

3.2 Software Architecture

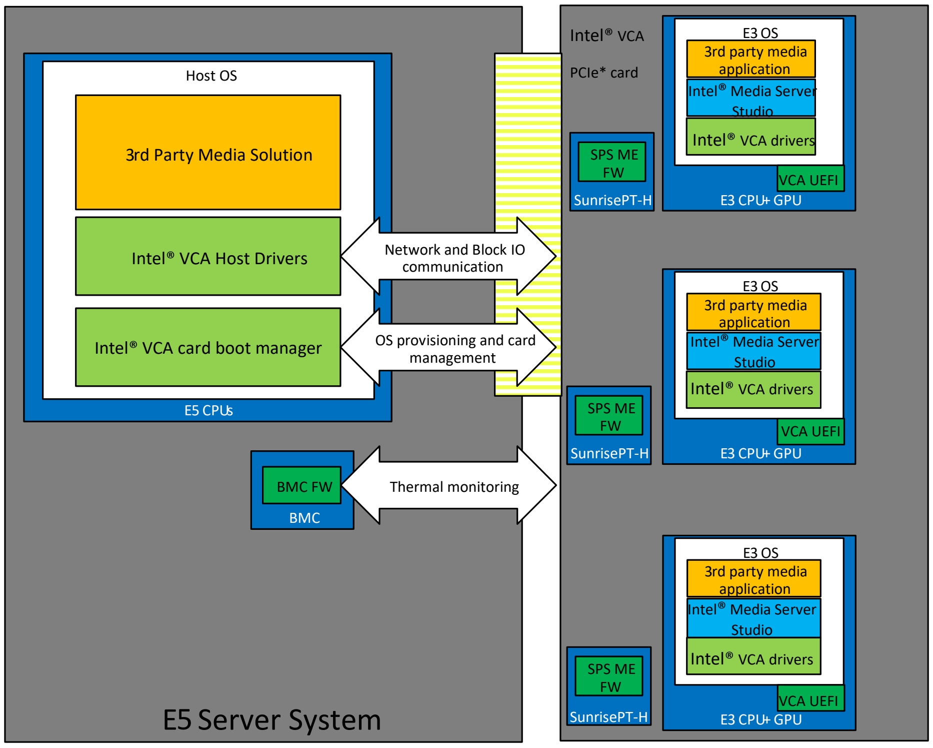


Figure 3. Software architecture of Intel® VCA

Figure 3 illustrates the software architecture of Intel® VCA product family. It should be noted that the Intel® VCA communicates with host through a virtual network over PCIe* and over block IO over PCIe*. So, logically, the Intel® VCA has a three Intel Xeon E3 processor systems, each running its own OS instance that are connected over virtual network and block IO to host the system. Therefore, the programming model of Intel® VCA is not the same as a usual PCIe* accelerator. Instead, Intel® VCA should be used as a network-based accelerator. Figure 3 also shows the software stack of Intel® VCA product family.

The host initiates booting and serves as the management point for the nodes by means of a dedicated, out of band management interface (Figure 4). The host is also responsible for providing configuration for the nodes and for ensuring their network connectivity. The nodes are connected to the host by dedicated virtual Ethernet links. The IP addresses of the nodes can be assigned either by the host from a predefined, non-routable IP address pool or by an external DHCP server. The latter requires creating a virtual bridge from the node virtual Ethernet interface and the host network interface.

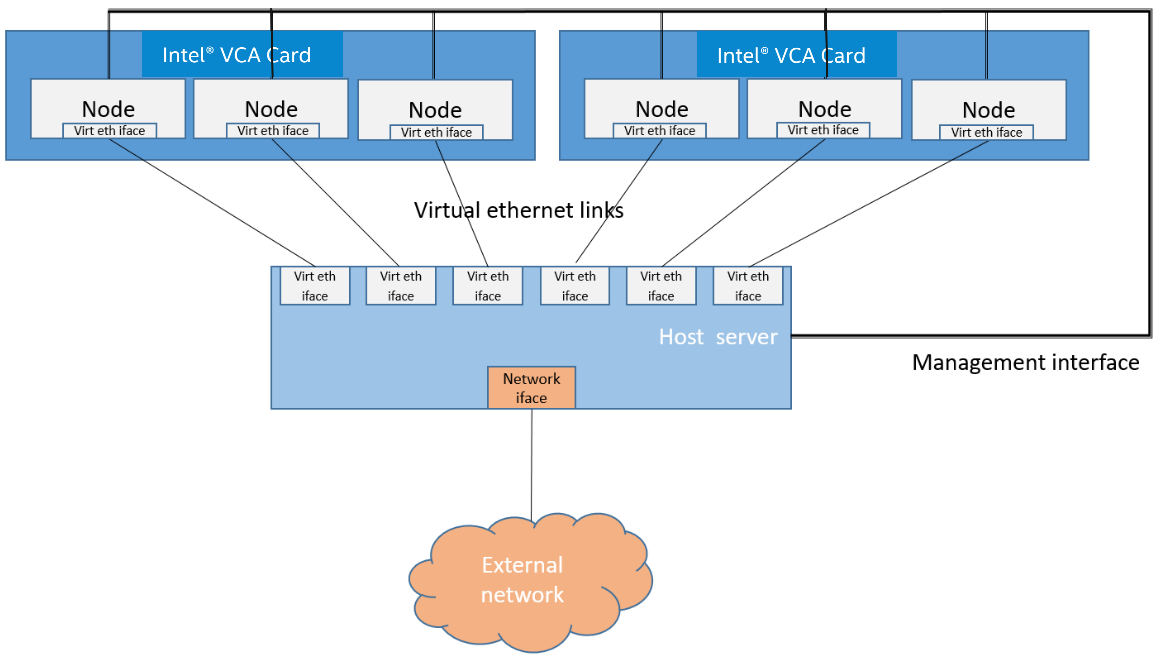


Figure 4. Network connectivity and management interface of the Intel® VCA

The host stores disk images with the operating system for the nodes. Thorough out this document, these operating system images are referred to as **VCA boot images**, or just **images**. Different mechanisms are used to provide these images for the nodes to boot, resulting in persistent VCA boot images and volatile VCA boot images. Persistent VCA boot images can be modified by the booted node; the images preserve these modifications across node reboots and therefore the images resemble a traditional disk from the node's point of view. The volatile VCA boot images are provided by the host by means of a RAMDisk and therefore, while allowing modifications by a running node, they do not preserve any state across node reboots.

The framework for Intel® VCA application software is Intel® Media Server Studio (https://software.intel.com/en-us/intel-media-server-studio), which provides software to innovate and optimize enterprise-grade video delivery and cloud graphics applications and solutions for data center, cloud, and network usages. Intel Media Server Studio consists of Intel® Media SDK and Intel® SDK for OpenCL*. Application programs are on top of Intel Media Server Studio. The bottom layers are operating system media framework (eg, libva for Linux*) and Intel® graphics drivers.

Intel® VCA strategy is to test the Intel® VCA software stack against the most recent, publicly released Intel Media Server Studio to provide the reference Intel® VCA software stack. But some customers may want to keep older Intel Media Server Studio releases, which have been well tested in their environment, or may have access to non-public (e.g. obtained under NDA) Intel Media Server Studio releases. To cover such needs, the production VCA software stack does not include Intel Media Server Studio.

Every node may either run a simple operating system, or may run a hypervisor to provide some number of virtual machines with hardware support for video transcoding applications (Figure 5). Only hypervisors running in Linux operating system are supported in Intel® VCA software. The hypervisors currently available are KVM, KVMGT (also known as KVM GVT-G), and Xen and are provided by CentOS* or Ubuntu*.

The Intel® VCA software includes the Intel® VCA boot images with modified kernels of the operating systems which run directly on the nodes, and the Intel® VCA boot images for the guest operating systems.

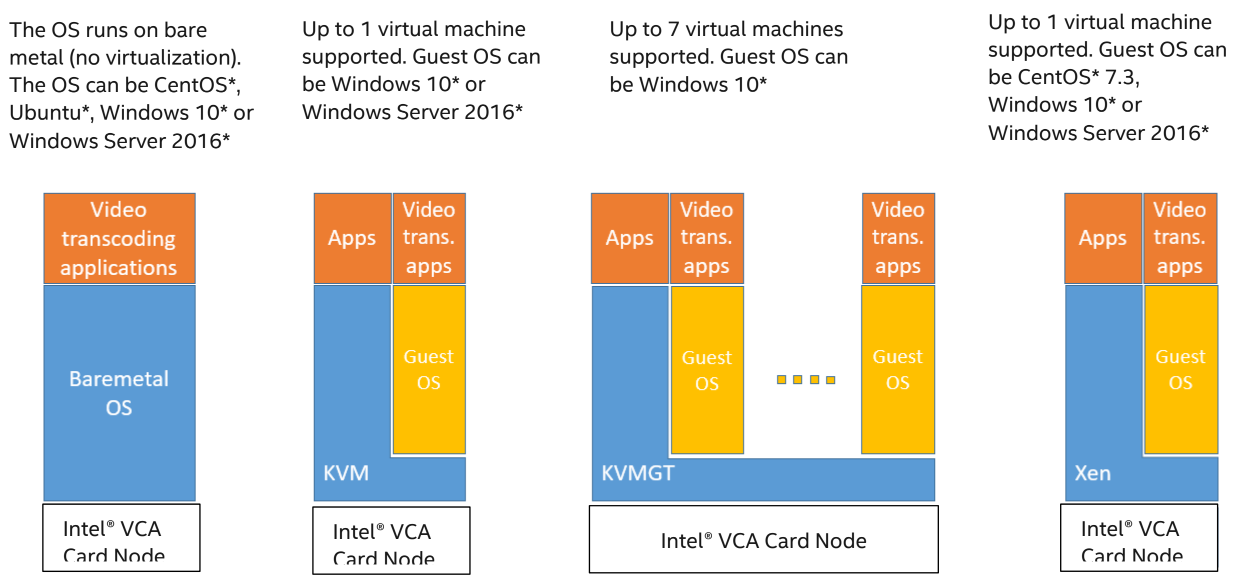

The OS runs on bare metal (no virtualization). The OS can be CentOS*, Ubuntu*, Windows 10* or Windows Server 2016*
Up to 1 virtual machine supported. Guest OS can be Windows 10* or Windows Server 2016*
Up to 7 virtual machines supported. Guest OS can be Windows 10*
Up to 1 virtual machine supported. Guest OS can be CentOS* 7.3, Windows 10* or Windows Server 2016*
Video transcoding applications
Baremetal OS
Intel® VCA Card Node
Apps
Video trans. apps
Guest OS
KVM
Intel® VCA Card Node
Apps
Video trans. apps
Guest OS
Video trans. apps
Guest OS
KVMGT
Intel® VCA Card Node
Apps
Video trans. apps
Guest OS
Xen
Intel® VCA Card Node

Figure 5. Software stack with the virtualization technologies supported by Intel® VCA

3.2.1 Software Support Matrix

Following tables summarize the host and node OSes supported with Intel® VCA and Intel® VCA 2. Table 5 and Table 6 summarizes the supported node OS software versions for Intel® VCA and Intel® VCA 2. The Intel Media Server Studio column indicates if Intel Media Server Studio is enabled (reference) or disabled (production). The Features column designates persistent baremetal images by VCA_DISK. Volatile images are described by the name of the virtualization technology used (KVM, KVMGT, or XEN) or by baremetal if no virtualization is provided. For the virtualized volatile images, the guest operating system is further indicated as Windows 10 Enterprise*, Windows Server 2016*, or DomU (Linux*).

Table 3. Intel® VCA software host operating system support matrix

Operating System	Kernel Version	Intel® VCA Software Support
CentOS* 7.4	4.14.20	2.3
	3.10.0-693.11.6	1.3 HF, 1.5 HF, 2.1, and 2.2
Debian* 8.7	4.4.115	1.5 HF
	4.4.122	2.1, 2.2
	4.14.20	2.3
Ubuntu* 16.04.3 LTS	4.14.20	2.3
	4.4.98 (4.4.0-112.135)	2.1, and 2.2
CentOS* 7.0	Only kernel patches for kernel 3.10.107	2.1, 2.2, and 2.3

Table 4. Intel® Media Server Studio version support matrix

Intel® VCA Software Version	Intel® Media Server Studio Version in Reference Package
1.3, 1.5, 2.0	Community Edition 2017R2, SDK 2017 16.5.2
1.3 HF, 1.5 HF	Community Edition 2017R3, SDK 2017 16.5.2
2.1	Community Edition 2017R3, SDK 2017 16.5.2
2.2	Community Edition 2018R1, SDK 2018 16.8
2.3	Community Edition 2018R2 HF1, SDK 2018 16.9

Table 5. Intel® VCA software version 2.3 support matrix Node OS

Operating System (kernel version)	Intel® Media Server Studio Support	Feature (image type)	Guest Operating System	Supported Intel® VCA Card
CentOS* 7.4 (4.14.20)	Reference	baremetal (volatile)	-	Intel® VCA 2
		vca-disk (persistent, 24 GB)	-	Intel® VCA 2
Ubuntu* 16.04.3 LTS (4.14.20)	Reference	baremetal (volatile)	-	Intel® VCA 2
		vca-disk (persistent, 24 GB)	-	Intel® VCA 2
Windows 10 (GPT) [1]	Production	Baremetal (volatile)	-	Intel® VCA 2
Windows Server 2016 (GPT) [1]	Production	Baremetal (volatile)	-	Intel® VCA 2

[1] The operating system image needs to be built by the user. Details are provided in Appendix A.

[2]. Size of the image can be changed using the script provided. Details are provided in

Table 6. Intel® VCA software version 2.2 support matrix Node OS

Operating System (kernel version)	Intel® Media Server Studio Support	Feature (image type)	Guest Operating System	Supported Intel® VCA Card
CentOS* 7.4 (3.10.0-693.11.6.el7)	Reference	baremetal (volatile)	-	Intel® VCA, Intel® VCA 2
		vca-disk (persistent, 50 GB)	-	Intel® VCA, Intel® VCA 2
	Production	KVM hypervisor	Windows 10* (GPT)	Intel® VCA, Intel® VCA 2
			Windows Server 2016* (GPT)	Intel® VCA, Intel® VCA 2
Ubuntu* 16.04.3 LTS (4.13.0-32.35)	Production	KVMGT hypervisor	Windows 10 (MBR)	Intel® VCA, Intel® VCA 2
			Windows Server 2016 (MBR)	Intel® VCA, Intel® VCA 2
			Linux* [1]	Intel® VCA, Intel® VCA 2
	Reference	baremetal (volatile)	-	Intel® VCA, Intel® VCA 2
		vca-disk (persistent, 24 GB)	-	Intel® VCA, Intel® VCA 2
Windows 10 (GPT) [1]	Production	Baremetal (volatile)	-	Intel® VCA, Intel® VCA 2
Windows Server 2016 (GPT) [1]	Production	Baremetal (volatile)	-	Intel® VCA, Intel® VCA 2

[1] The operating system image needs to be built by the user. Details are provided in Appendix A.

Table 7. Intel® VCA software version 2.1 support matrix Node OS

Operating System (kernel version)	Intel® Media Server Studio Support	Feature (image type)	Guest Operating System	Supported Intel® VCA Card
CentOS* 7.4 (3.10.0-693.11.6.el7)	Reference	baremetal (volatile)	-	Intel® VCA, Intel® VCA 2
		vca-disk (persistent, 50 GB)	-	Intel® VCA, Intel® VCA 2
	Production	KVM hypervisor	Windows 10* (GPT)	Intel® VCA, Intel® VCA 2
			Windows Server 2016* (GPT)	Intel® VCA, Intel® VCA 2

Operating System (kernel version)	Intel® Media Server Studio Support	Feature (image type)	Guest Operating System	Supported Intel® VCA Card
Ubuntu* 16.04.3 LTS (4.13.0-32.35)	Production	KVMGT hypervisor	Windows 10 (MBR)	Intel® VCA, Intel® VCA 2
			Windows Server 2016 (MBR)	Intel® VCA, Intel® VCA 2
			Linux* [1]	Intel® VCA, Intel® VCA 2
Ubuntu* 16.04.3 LTS (4.4.98 (4.4.0-112.135))	Reference	baremetal (volatile)	-	Intel® VCA, Intel® VCA 2
		vca-disk (persistent, 24 GB)	-	Intel® VCA, Intel® VCA 2
Windows 10 (GPT) [1]	Production	Baremetal (volatile)	-	Intel® VCA, Intel® VCA 2
Windows Server 2016 (GPT) [1]	Production	Baremetal (volatile)	-	Intel® VCA, Intel® VCA 2

[1] The operating system image needs to be built by the user. Details are provided in Appendix A.

Table 8. Intel® VCA software version 1.5 HF support matrix Node OS

Operating System (kernel version)	Intel® Media Server Studio Support	Feature (image type)	Guest Operating System	Supported Intel® VCA Card
CentOS* 7.4 (3.10.0-693)	Reference	Baremetal (volatile)	-	Intel® VCA, Intel® VCA 2
		Persistent over NFS	-	Intel® VCA, Intel® VCA 2
	Production	Baremetal (volatile)	-	Intel® VCA, Intel® VCA 2
CentOS* 7.4 (4.4.115)	Reference	Baremetal (volatile)	-	Intel® VCA, Intel® VCA 2
		Persistent over NFS	-	Intel® VCA, Intel® VCA 2

Table 9. Intel® VCA software version 1.3 HF support matrix Node OS

Operating System (kernel version)	Intel® Media Server Studio Support	Feature (image type)	Guest Operating System	Supported Intel® VCA Card
CentOS* 7.4 (3.10.0-693)	Reference	Baremetal (volatile)	-	Intel® VCA
		Persistent over NFS	-	Intel® VCA
	Production	Baremetal (volatile)	-	Intel® VCA
		KVM hypervisor	Windows 10* (GPT) [1]	Intel® VCA
			Windows Server 2016* (GPT) [1]	Intel® VCA

[1] The operating system image needs to be built by the user. Details are provided in Appendix A.

4. System Setup

Generally setting up Intel® VCA card for use requires following steps:

- Configure Host System (Refer to Section 4.1 for details)
  - Changing Host Server BIOS settings
  - Host OS Configuration
- Setting up Host Server with required software (Refer to Section 4.3 for details)
- Enabling Virtual Network (Refer to Section 4.4 for details)
- Booting the CPUs on the VCA card with OS image (refer to Section 4.5 for details)
- Configuration of network for nodes (Refer to Section 4.4 for details)
- Install Intel Media Server Studio and Applications (refer to Section 5 For details)

4.1 Configuring the Host System

4.1.1 Configure the Host BIOS

The host system BIOS must be configured to enable large Memory-Mapped Input/Output (MMIO), and allow for large per-device BAR allocations. BAR must have 64-bit address enabled.

The minimum requirements for BAR and MMIO are:

- MMIO mapping above 4 GB is enabled
- Minimum MMIO size is 4 GB/CPU (node)

For example, on Intel® Server Board S2600WT based systems, this can be enabled in BIOS setup by configuring the following two options on the PCI Configuration screen.

- Set **Memory Mapped Above 4 GB** to **Enabled**
- Set **Memory Mapped IO size** to **256 GB or higher**

The specific wording in the BIOS setting of non-Intel produced motherboards may vary, but the ability to enable large MMIO range is required for proper operation of Intel® VCA. Please see the *Intel® VCA VCA1283LVV/VCA1585LMV Product Specification and Hardware User Guide* document for further details regarding host system BIOS feature requirements. The minimum requirements for BAR size and MMIO size are dependent up on the VCA SW version as listed below.

For VCA SW 2.0 on host:

- MMIO >= (12 GB + 768 KB) per Intel® VCA card Installed in the host
- BAR >= 4 GB per CPU

For VCA SW 1.5 or less on host:

- MMIO >= (3 GB + 768 KB) per Intel® VCA card Installed in the Host
- BAR >= 1GB per CPU

4.1.2 Install the Operating System

For supported operating system versions, refer to Table 1, Table 2, and Table 6.

4.1.2.1 Install CentOS*

When installing CentOS* on the host system, make sure to select **Development and Creative Workstation** as the base environment. (Refer to Appendix B.1 for steps and details.) Make sure the user created during install has administrator rights and the root partition is big enough to store operating system images for multiple nodes.

4.1.2.2 Install Ubuntu*

When installing Ubuntu* on the host system, refer to Appendix C.1 for steps and details. Additional software modules listed in Appendix C.1 should be installed.

4.1.3 Configure Network Settings

Use the following guidelines to configure the host network settings.

- Ensure that the host can access the external network. Add necessary proxy settings if the host requires a proxy to connect to external network.
  - For CentOS* follow steps in Appendix B.1 step 18.
  - For Ubuntu* follow steps in Appendix C.1 step 13.
- Disable the `NetworkManager` service. This is critical since the service may block communication between the host and Intel® VCA card. Use the following commands to disable the service.

```
sudo systemctl disable NetworkManager
sudo systemctl stop NetworkManager
```

If the network is no longer present in Ubuntu*, add following lines to `/etc/network/interfaces`.

```
auto networkdevname #replace networkdevname to proper dev name
iface networkdevname inet dhcp #example provided here is with dhcp, for
   static additional steps are required
```

4.2 Physical Installation of the Intel® VCA Card

Refer to the *Intel® VCA Quick Start Guide* included in the box for instructions on how to quickly install the Intel® VCA card into the host server. The *Intel® VCA Quick Start Guide* can also be found at https://www.intel.com/ . Search for "*Support for Intel® Visual Compute Accelerator*" and select corresponding Intel® VCA version. Click on "*Install & setup*" to take you to a link to quick start guide. Make sure host BIOS setup is complete before inserting the Intel® VCA card in to the server.

4.2.1 Optional Hardware (Intel® VCA 2 only)

The Intel® VCA 2 card provides an M.2 socket for memory card size 42 mm. M.2 drives can be used like ordinary storage to, for example, boot an operating system image. Below are steps to prepare the M.2 drive.

1. Check available disk :

   ```
   fdisk -l
   ```

2. Find disk named `/dev/nvme0n1`
3. Mount disk. For example:

   ```
   mount /dev/nvme0n1p1 /mnt/m.2
   ```

4. Check if disk is mounted:

   ```
   df
   ```

4.3 Installing Intel® VCA Software

Download the Intel® VCA product family software from https://downloadcenter.intel.com. There are several packages available to download:

- Intel® Visual Compute Accelerator VCA1283LVV/VCA1585LMV Host Files
- Intel® Visual Compute Accelerator EEPROM Update
- Intel® Visual Compute Accelerator BIOS Update
- Intel® Visual Compute Accelerator Volatile Reference Package
- Intel® Visual Compute Accelerator Persistent Reference Package

- Intel® Visual Compute Accelerator Persistent Production Package
- Intel® Visual Compute Accelerator Kernel Development Package

The host files (kernel, Intel® VCA daemon, and VCASS drivers), EEPROM and BIOS update packages are always needed for a new Intel® VCA software installation. Other packages can be downloaded based on the system design after going through the detailed explanation in Section 4.5. Typically, only one of the volatile and persistent reference packages are necessary. The kernel development package is only necessary for kernel development.

4.3.1 Required Software

- Intel® Visual Compute Accelerator VCA1283LVV/VCA1585LMV host files appropriate for the operating system in use. Refer to the release notes for proper filenames.
  - Updated kernel
    - For CentOS*: `kernel-<kernel_ver>_<version>.VCA.x86_64.rpm`
    - For Ubuntu*: `linux-image-<kernel_ver>.<version>_amd64.deb`
  - Intel® VCA user-space applications
    - For CentOS*: `daemon-vca-<version>.x86_64.rpm`
    - For Ubuntu*: `daemon-vca-<version>-amd64.deb`
  - VCASS drivers (`vcass-modules-<kernel_ver>-<version>.VCA-<version>-0.x86_64.rpm`)
  - Boot images (can be downloaded later; refer to Section 4.5 for details)

4.3.2 Required Firmware

- Basic I/O system for Intel® VCA card (`VCA-BIOS-<version>.img`)
  - "Intel Visual Compute Accelerator VCA1283LVV/VCA1585LMV BIOS Update"
- EEPROM for Intel® VCA card for PLX configuration (`eeprom_<version>.bin`)
  - "Intel Visual Compute Accelerator EEPROM Update"

4.3.3 Optional Software

To run sample transcodes, download Intel® Media Server Studio and sample code from https://software.intel.com/en-us/intel-media-server-studio.

4.3.4 Install VCA Software on Host

To install the software, perform the following steps:

1. Boot the host system and log in to operating system.
2. Copy the downloaded packages to a temporary folder on the host server.
3. If booted into a graphical interface, open a terminal session.
4. Change directory to the temporary folder where the packages were copied.
5. If updating from a previous version, remove the older RPMs with the following command:
   - For CentOS*:

     ```
     rpm -qa | grep -e daemon-vca -e vcass-modules | xargs yum -y erase
     ```

   - For Ubuntu*:

     ```
     dpkg --list | grep vca | xargs apt-get remove
     ```

   Purging of these packages also required with `apt-get purge pkg-name`.
6. Unzip the downloaded packages.
7. Install the updated kernel packages.
   - For CentOS*

     ```
     sudo yum -y localinstall kernel-*rpm
     ```

- For Ubuntu*

  ```
  sudo dpkg -i linux-image-<kernel_ver>.<version>.amd64.deb
  ```

8. Configure the new kernel to be the default at boot. This step is very important to boot the operating system with proper kernel.
   - For CentOS*

     ```
     sudo grub2-set-default 0
     ```

   - For Ubuntu*
     - Set default kernel `GRUB_DEFAULT` in `/etc/default/grub`. Find the name of kernel name and versions in `/boot/grub/grub.cfg`. (Search for `gnulinux-advanced` and `vca-advanced`.)

       ```
       GRUB_DEFAULT="gnulinux-advanced-716b90f8-9b3c-428a-aaca-
          318038803998>gnulinux-4.4.0-1.2.1.106.vca-advanced-716b90f8-
          9b3c-428a-aaca-318038803998"
       ```

     - Reload grub

       ```
       sudo update-grub
       ```

     - Reboot host
9. Install the Intel® VCA driver and utility.
   - For CentOS*

     ```
     sudo yum -y localinstall vcass-modules*rpm
     sudo yum -y localinstall daemon-vca*rpm
     ```

   - For Ubuntu*

     ```
     sudo dpkg -i vcass-modules*.deb
     sudo dpkg -i daemon-vca*.deb
     ```

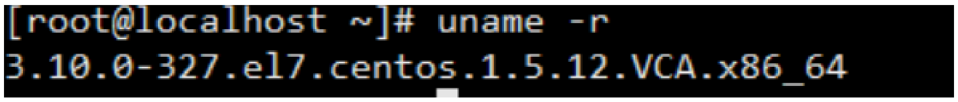
10. Reboot the system to enable the new kernel.

    ```
    sudo reboot
    ```

11. After reboot, confirm that the expected Intel® VCA kernel is being used.

    ```
    uname –r | grep –i vca
    ```

    The command should return the Intel® VCA kernel as shown in Figure 6. In this example, the Intel® VCA software version is `1.5.12.VCA` and the CentOS*/Ubuntu* kernel version is `3.10.0-327.el7.centos`.



```
[root@localhost ~]# uname -r
3.10.0-327.el7.centos.1.5.12.VCA.x86_64
```

Figure 6. Check for Intel® VCA kernel

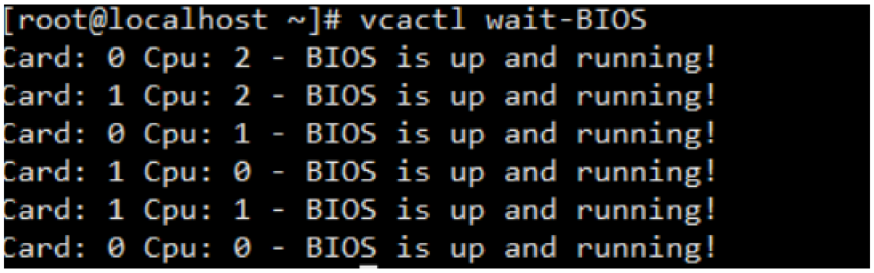
12. Add a user to the `vcausers` group so that `sudo` is not required for `vcactl` use. Refer to section 6.1.

    ```
    usermod -aG vcausers <username>
    ```

Note: To add a username to `vcausers`, a relogin or reboot is required.

13. Confirm that the Intel® VCA cards are detected and the BIOS is up and running as shown in Figure 7.

    ```
    vcactl wait-BIOS
    ```



```
[root@localhost ~]# vcactl wait-BIOS
Card: 0 Cpu: 2 - BIOS is up and running!
Card: 1 Cpu: 2 - BIOS is up and running!
Card: 0 Cpu: 1 - BIOS is up and running!
Card: 1 Cpu: 0 - BIOS is up and running!
Card: 1 Cpu: 1 - BIOS is up and running!
Card: 0 Cpu: 0 - BIOS is up and running!
```

Figure 7. Intel® VCA card in `BIOS is up and running` status

If Intel® VCA is not in `BIOS is up and running` status, do not continue with the procedure. Instead, double check the hardware and software installation.

4.3.5 Updating EEPROM

To update the EEPROM, perform the following steps.

1. Unzip the downloaded Intel® Visual Compute Accelerator EEPROM Update package
2. Reset the node using the following command:

   ```
   sudo vcactl reset
   ```

3. Wait for the reset to complete; the command `sudo vcactl wait-BIOS` or `sudo vcactl status` returns `bios_up` when ready.
4. Run the following command:

   ```
   sudo vcactl update-EEPROM <EEPROM file name>
   ```

5. When the last command finishes successfully, reboot the host. After rebooting, the software installation is complete.

4.3.6 Updating BIOS

To update the BIOS, perform the following steps.

1. Power on the host.
2. Unzip the downloaded Intel® Visual Compute Accelerator BIOS update package
3. Reset the node using the following command:

   ```
   sudo vcactl reset
   ```

4. Wait for the reset to complete; the command `sudo vcactl wait-BIOS` or `sudo vcactl status` returns `bios_up` when ready.
5. Run the following command:

   ```
   sudo vcactl update-BIOS <BIOS file name>
   ```

6. Wait for the update to complete; the command `sudo vcactl wait-BIOS` or `sudo vcactl status` returns `bios_up` when ready.
7. After BIOS update is complete, the node(s) reboot to come online with the new BIOS version.
8. When the last command finishes successfully, G3 (mechanical off) is necessary. To perform G3, execute the power off command on the host and then disconnect from the power supply for 15 seconds.

4.4 Enabling Virtual Network

VCA uses host server's Ethernet Network to communicate with external devices. Intel® VCA can work in two network configurations. Bridged mode or NAT, With DHCP or with static IP addressing. Following sections provide details on how to setup the Host and Nodes for these operations. An Intel® VCA network stack implements standard network device and thus is capable of sending and receiving packets to multicast addresses as well as Internet Group Management Protocol (IGMP) packets. Instructions on how to setup multicast are available in Section 7.

4.4.1 Configuring Nodes for Bridged Mode Operation

Configuring a network bridge containing virtual Ethernet adapters to Intel® VCA nodes and some physical network interfaces allows for direct connecting Intel® VCA nodes to the external network without need of configuring routing mechanisms at the host to which the Intel® VCA card is inserted. Such a network is flatter and nodes must use IP addresses from a common pool. The addresses can be configured statically per each node, or dynamically assigned by a DHCP server located somewhere in local network.

The Intel® VCA stack is flexible and setting for each node is independent. In effect different combinations are possible:

- All nodes can be assigned to same bridge or to different ones; some nodes can be connected to bridges, while others use only a local virtual network.
- Some nodes can be configured to use dynamic addressing (DHCP), while other use static addressing. Do not There is no sense to combine node not connected to bridge with DHCP, as DHCP server would be required to be place on VCA card host server

Several steps must be taken to properly configure nodes for bridged network operation. The host must configure a bridge interface and bring that interface online. The node configuration must be updated to point to the bridge interface, take a new IP information, and, if using persistent boot image, be provided with the NFS server and shared directory information. The steps are detailed below. Steps 1 to 3 are different for different OSes, from step 4 they are common.

4.4.1.1 Set Up Bridge for CentOS*

1. Create a bridge configuration file (e.g. `/etc/sysconfig/network-scripts/ifcfg-virbr0-vca`) containing the following lines.

```
DEVICE=virbr0-vca
TYPE=Bridge
BOOTPROTO=dhcp
ONBOOT=yes
NM_CONTROLLED=no
```

2. Modify the main interface file (e.g. `/etc/sysconfig/network-scripts/ifcfg-enp3s0f0`) to contain the following lines. The main interface is the interface physically connected to the network.

```
DEVICE=enp3s0f0
TYPE=Ethernet
BOOTPROTO=dhcp
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=virbr0-vca
UUID=UUID_OF_DEVICE
```

3. Restart the network.

   ```
   systemctl restart network
   ```

   Continue with step 4 below.

4.4.1.2 Set Up Bridge for Ubuntu*

Make sure bridge-utils are installed on the system before following these steps.

1. Create a bridge configuration in file `/etc/network/interfaces` by adding new network interface:

   ```
   auto virbr0-vca
   iface virbr0-vca inet dhcp
   bridge_ports ieth0  #Replace ieth0 with the network card name
   ```

2. Bring up the bridge

   ```
   ifup virbr0-vca
   ```

3. Restart the host.

   ```
   reboot
   ```

4. Configure Intel® VCA nodes to use the new network bridge.

   ```
   vcactl config bridge-interface virbr0-vca
   ```

5. Increase the MTU size of the main network interface used in step 2.

   Note: This steps needs to be done every time host server is rebooted.

   Virtual Ethernet links to Intel® VCA nodes already set to a high MTU value; the default value for normal Ethernet is only 1500. Use the following command, where `enp3s0f0` is the main network interface, to set the maximum MTU value (`ABCD`); for a high enough value, if `ifconfig` returns an error and look at the `dmesg` output for Ethernet to find the maximum value for `ABCD`. Typically 9000 is a good value for `ABCD`.

   ```
   ifconfig enp3s0f0 mtu ABCD #replace enp3s0f0 with main network interface
      name
   ```

6. Configure the IP scheme for the Intel® VCA nodes.
   - For DHCP configuration, use the following command.

     ```
     vcactl config ip dhcp
     ```

   - For static configuration, the IP addresses for nodes can be set after booting the nodes. Use the following commands replacing `II.JJ.KK.NN` with the appropriate IP addresses for each `card_id` and `cpu_id`. Replace `YY` with the bitmask of the node.

     ```
     vcactl config [<card_id> <cpu_id>] ip II.JJ.KK.NN
     vcactl config mask YY
     ```

     Set the gateway `II.JJ.KK.GG` and host `II.JJ.KK.HH` IP addresses and replace `ZZ` with the bitmask of the host.

     ```
     vcactl config gateway II.JJ.KK.GG
     vcactl config host-ip II.JJ.KK.HH
     vcactl config host-mask ZZ
     ```

7. Configure the Intel® VCA nodes with NFS host information (for persistent images).
    - If the host is used as the NFS server, follow the steps (1 to 4) listed in Appendix B.5.
    - If using the default host directory, use the following command replacing `II.JJ.KK.SS` with the server's IP address or fully-qualified domain name if DNS is supported on the network.

      ```
      vcactl config nfs-server II.JJ.KK.SS
      ```

    - If using a custom NFS host directory, use the following command to configure the node with that information. Note that a separate file system should be used for each node, so it is strongly recommended that the %s variable (node-name) be included in the path on the server. Also `X` and `Y` should be replaced with the card and node values, respectively, for the specific node being modified.

      ```
      vcactl config X Y nfs-path /path/on/server
      ```

8. Copy configuration of the host to each node

   ```
   scp /etc/resolv.conf root@<node ip address>:/etc/resolv.conf
   ```

9. Setup proxies on the nodes

   ```
   export https_proxy=http://<your_proxy>:<your_proxy_port>
   ```

4.4.2 Creating NAT

To create NAT, follow the steps below. Some steps need to be run after booting the nodes.

On the host:

1. Disable the firewall. (It is possible to add rules to the firewall instead of disabling firewall, but there is no one rule for all applications. It depends on firewall configuration.)

   ```
   systemctl stop firewalld.service
   systemctl disable firewalld.service
   systemctl status firewalld.service
   ```

2. Enable forwarding in kernel.

   ```
   echo 1 > /proc/sys/net/ipv4/ip_forward
   ```

3. Following steps needs to be performed after booting the nodes and getting IP address of each node
    - Add rules to NAT (Change X to the appropriate value (e.g. 3 for 172.31.3.1).)

      ```
      iptables -t nat -A POSTROUTING -s 172.31.X.0/24 -d 0/0  -j MASQUERADE
      ```

    - Copy configuration to the host. Change X to the appropriate value (e.g. 3 for 172.31.3.1).

      ```
      scp /etc/resolv.conf root@172.31.X.1:/etc/resolv.conf
      ```

On the card:

After booting the nodes, set up proxies for the nodes:

```
export https_proxy=http://<your_proxy>:<your_proxy_port>
```

4.5 Booting the Intel® VCA Card

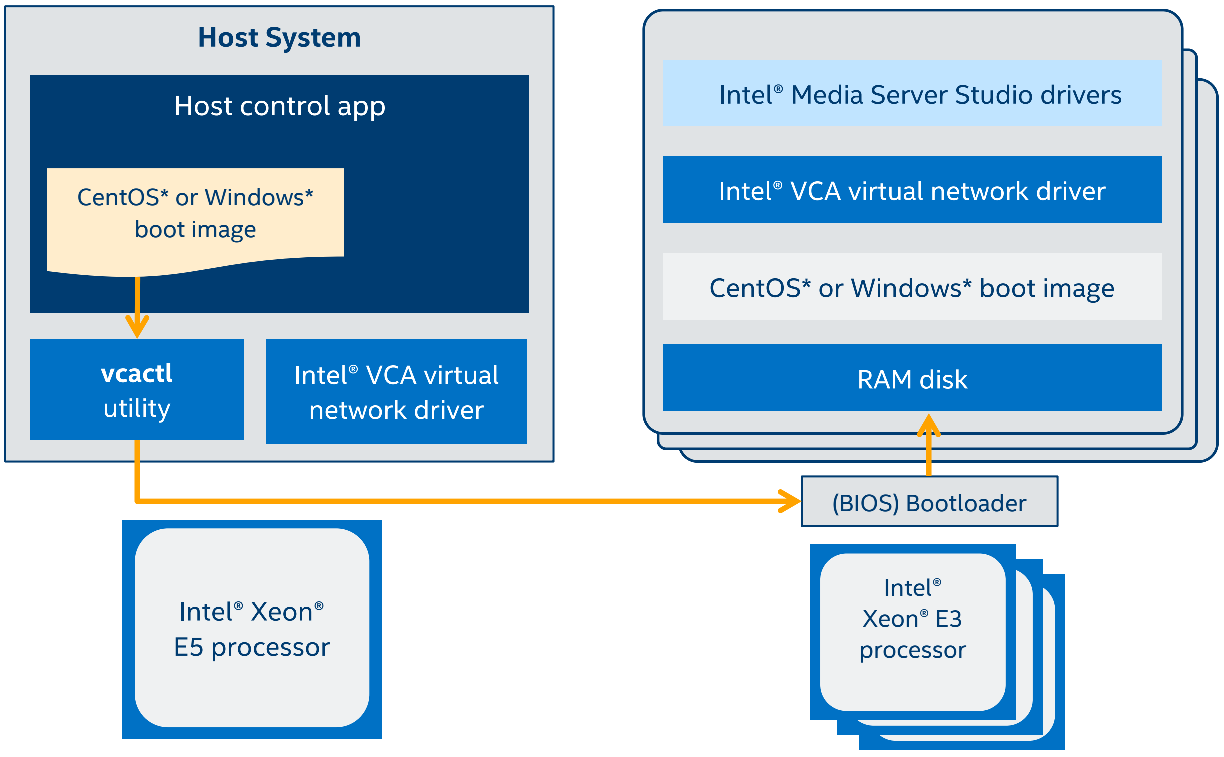

Host System
Host control app
CentOS* or Windows* boot image
vcactl utility
Intel® VCA virtual network driver
Intel® Xeon® E5 processor
Intel® Media Server Studio drivers
Intel® VCA virtual network driver
CentOS* or Windows* boot image
RAM disk
(BIOS) Bootloader
Intel® Xeon® E3 processor

Figure 8. Process to boot Intel® VCA card

During boot, the provided Intel® VCA control utility (`vcactl`) takes the boot image and, with help from a bootloader incorporated into the firmware on the card, loads it into a RAM disk and initiates boot.

Following explains different terms in the package names available at Intel download center:

- **Volatile Packages**: Images from these packages load the entire operating system within a RAMDisk. As this is read only memory, changes are not persistent across reboot. If one or more folders need to be persistent, use a persistent package or see Section 4.5.4 on to manually create a persistent file system using NFS.
- **Persistent Packages**: Images from these packages boot a minimal image and all other folders are mounted back to a share on the host. The names of the folders and share must be exactly as described in each section.
- **Production Packages**: Images in these packages do not contain Intel Media Server Studio drivers and are built using production kernel (without Intel Media Server Studio patches). Images in these packages are for customers who want to install their own Intel Media Server Studio.
- **Reference Packages**: Images in these packages contain reference kernel (including Intel Media Server Studio patches) and contains the latest publicly released Intel Media Server Studio Community Edition (without media samples package) at the time of image creation. If a later version of Intel Media Server Studio drivers are available, it is customer responsibility to update to later versions.

Intel provides boot images with different combinations of the above definitions, in the Intel® VCA Volatile Reference Package, the Intel® VCA Persistent Reference Package, and the Intel® VCA Persistent Production Packages. All of these boot images also create a virtual network between the devices.

The above packages contain different boot image types:

- Custom image – See Appendix A for instructions on how to build a custom image.
- Baremetal images (basic option, plain image) – Baremetal CentOS* 7.x image. This image loads the entire operating system within a RAMDisk. As this is read only memory, changes are not persistent. If

one or more folders need to be persistent, use a persistent package or see Section 4.5.4 on to manually create a persistent file system using NFS.

- Persistent images with BlockIO - These type of images have `.vcad` file extensions and are valid from Intel® VCA software version 2.0 and higher. The persistent image keeps operating system configurations made on the node even after power-cycle through `blockio` drivers.
- Persistent image with `rootfs` – Persistent image with option of remounting root filesystem via NFS service, with Intel® VCA software version 1.5 and lower.
- Xen image – This image boots Xen hypervisor and CentOS* in domain 0. The user can then load any supported guest operating system to DomU. The image contains a reference kernel, but does not contain Intel Media Server Studio (as graphics are accessed from the DomU operating system).
- KVM image – This image boots KVM hypervisor and CentOS* in domain 0. The user can then load any supported guest operating system to DomU. The image contains a reference kernel, but does not contain Intel Media Server Studio (as graphics are accessed from the DomU operating system).
- KVM-GT image – This image boots KVM-GT hypervisor and CentOS*/Ubuntu* in domain 0. The user can then load any supported guest operating system to DomU. The image contains a reference kernel, but does not contain Intel Media Server Studio (as graphics are accessed from the DomU operating system).

General boot sequence after the BIOS is up and running includes:

- Pre-boot setup (BlockIO, NFS, etc.)
  - Baremetal – No specific setup required.
    - BlockIO – Step 2 of Section 4.5.2.
    - NFS – Steps 1 to 6 of Section 4.5.3 or 4.5.4.
- Boot image.

  ```
  vcactl boot [<card_id> <cpu_id>] [os_image_file.img/vcablkN]
  ```

- Get Network IP. Wait for vcactl wait to return net_device_ready.

  ```
  vcactl network ip <card-ID> <CPU-ID>
  ```

- Nodes are ready to be used.

General shutting down sequence after the operating system is up and running includes:

- Shutdown operating system gracefully.
  - For Intel® VCA 2 (operating system shutdown and power off)

    ```
    vcactl pwrbtn-short [<card_id> <cpu_id>]
    ```

  - For Intel® VCA

    ```
    vcactl os-shutdown <card_id> <cpu_id>
    ```

- Post-shutdown setup (close BlockIO, etc.). Wait for operating system to shut down on nodes.
  - BlockIO

    ```
    vcactl blockio close <card_id> <cpu_id> [vcablk<N>]
    ```

- Reset
  - For Intel® VCA 2 (to power on node from power off)

    ```
    vcactl pwrbtn-short [<card_id> <cpu_id>]
    ```

  - For Intel® VCA

    ```
    vcactl reset [<card_id> <cpu_id>]
    ```

4.5.1 Intel® VCA Volatile Reference Package

Before using the Intel® VCA Volatile Reference Package, verify that the nodes are ready for boot by opening a terminal session and using the command `sudo vcactl wait-BIOS`. All nodes must be indicated as up and running before continuing.

Perform the following steps to use the Intel® VCA Volatile Reference Package. These steps use the Intel® VCA utility `vcactl`. For detailed information on the `vcactl` utility, refer to Section 6.

1. Use the following commands, depending on the card model, to boot the OS on Intel® VCA.
   - For Intel® VCA 2:

     ```
     vcactl boot <card_ID> <CPU_ID> vca_baremetal_4p4_<version>.img
     vcactl wait <card_ID> <CPU_ID>
     ```

   - For Intel® VCA:

     ```
     vcactl boot <card_ID> <CPU_ID> vca_baremetal_<version>.img
     vcactl wait <card_ID> <CPU_ID>
     ```

   The `<card_ID>` and `<CPU_ID>` parameters are optional. If they are not provided, the command boots all Intel® VCA cards and all CPUs on those cards.

   If the `vcactl wait` command returns `net_device_ready`, the Intel® VCA booted successfully. Otherwise, it reports a boot failure and related error messages.

2. Determine the Intel® VCA card IP address.

   ```
   vcactl network ip <card-ID> <CPU-ID>
   ```

3. Log in to Intel® VCA. Login password `vistal`.

   ```
   ssh root@<IP addr of VCA>
   ```

The bare metal volatile image loads the entire OS within a RAMDisk. As this is read-only memory, changes are not persistent. If one or more folders need to be persistent, please refer to section 4.5.2, 4.5.3, or 4.5.4.

4.5.2 Intel® VCA Persistent over BlockIO (Intel® VCA software version 2.0 and above)

Intel® VCA software version 2.0 introduces a new type of persistent image. This new image does not need NFS protocol to work; instead, all read/write operations are handled by the BlockIO driver. Using BlockIO devices is explained in Section 4.5.9.

The boot process is as follows:

1. Extract the persistent image copy for each node. Make sure that each node has its own copy; the extracted `.vcad` file needs 50 GB (24 GB for Ubuntu* OS image) of free disk space.

   ```
   tar -xf vca_disk_X_Y_Z.tar.gz –C <destination directory>
   ```

   or

   ```
   gzip –d vca_disk_X_Y_Z.gz
   ```

   where `X_Y_Z` represents the Intel® VCA software version, kernel version, and operating system type.

2. Add the extracted `vca_disk` image (eg. `vca_disk_4p4_centos7.2_2.0.283.vcad`) as the primary BlockIO device. (The `vcablk0` device is used to boot nodes and for simple storage purposes.) All the nodes can be booted with same `vcablk0`, but each node needs to have its own copy of the `.vcad` file and the following command needs to be issued for each card-ID and node-ID along with the corresponding path to each node file.

```
sudo vcactl blockio open <card-ID> <node-ID> vcablk0 RW <path to vca disk
   (.vcad) file>
```

3. Boot the card using the `vcactl boot` command.

   ```
   sudo vcactl boot <card-ID> <node-ID> vcablk0
   ```

4.5.3 Intel® VCA Persistent over NFS (Intel® VCA software version 1.5 and lower only)

Note: After setup is complete and the operating system using a persistent file image is booted on Intel® VCA nodes, it is important to not restart the NFS on the host. This could cause a kernel panic for the nodes booted on the card.

This section describes the required configuration of the host to enable use of the persistent file image for Intel® VCA software versions 1.5 and lower. The setup here is required exactly as it is listed, including folder names. Failure to set up exactly as described causes the operating system to not boot properly and be unusable. See Section 4.5.4 for information on how to manually set up persistent storage for a bare metal image file.

1. Set up NFS server on the host or any server, for information on setting up a CentOS* 7 NFS server on host, see Section B.5.
2. Create NFS mount points with the node's file system.

   ```
   sudo mkdir /mnt/vca_node_00
   sudo mkdir /mnt/vca_node_01
   sudo mkdir /mnt/vca_node_02
   sudo mkdir /mnt/vca_node_10
   sudo mkdir /mnt/vca_node_11
   sudo mkdir /mnt/vca_node_12
   ```

   The directory, which is used for NFS exports, is hard-coded in a persistent file system boot image for the card. It consists of `/mnt/` and the hostname of the node (`vca_node_00`, `vca_node_01` and so on). So as seen above, the full path of the exported mount point for node 0 on card 0 is `/mnt/vca_node_00`. The hostname of the node can be changed in the `/etc/vca_config.d/vca_*_default.sh` scripts for each of the node; if changed here, the NFS mount point must be changed to match. The `/mnt/` prefix cannot be changed.

   A configuration with two Intel® VCA cards is assumed. If using more than two cards, create additional folders as necessary. If using only one card, enter only the first three lines above.

3. Create or update `/etc/exports` with:

   ```
   /mnt/vca_node_00   *(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_01   *(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_02   *(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_10   *(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_11   *(rw,sync,no_root_squash,no_all_squash)
   /mnt/vca_node_12   *(rw,sync,no_root_squash,no_all_squash)
   ```

   A configuration with two Intel® VCA cards is assumed. If using more than two cards, append additional lines as necessary. If using only one card, enter only the first three lines above.

4. Export the directories in `/etc/exports`.

   ```
   sudo exportfs –a
   ```

5. Copy the contents of `vca_rootfs_{build_version}-tree.tar.gz` into each mount point.

   ```
   sudo tar xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_00
   sudo tar xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_01
   sudo tar xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_02
   ```

```
sudo tar xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_10
sudo tar xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_11
sudo tar xf vca_rootfs-tree.tar.gz -C /mnt/vca_node_12
```

A configuration with two Intel® VCA cards is assumed. If using more than two cards, append additional lines as necessary. If using only one card, enter only the first three lines above.

6. Start the server.

```
sudo systemctl enable rpcbind
sudo systemctl enable nfs-server
sudo systemctl start rpcbind
sudo systemctl start nfs
```

7. Make sure the firewall is properly configured.
   - o Determine which zones are active.

```
sudo firewall-cmd --get-active-zones
```

   - o Add NFS to the active zones.

```
sudo firewall-cmd --permanent --zone=<ZONES_FROM_PREVIOS_COMMAND> --add-
   service=nfs
sudo firewall-cmd --reload
```

8. Boot the card using the `vcactl boot` command.
   - o For Intel® VCA 2:

     ```
     sudo vcactl boot <card-ID> <node-ID> VCA_persistent_4.4_<version>.img
     ```

   - o For Intel® VCA:

     ```
     sudo vcactl boot <card-ID> <node-ID> VCA_persistent_<version>.img
     ```

9. Verify `net_device_ready` for each node by using the `vcactl wait` command.

4.5.4 Manually Creating a Persistent File System Using NFS

Note: After setup is complete and the OS using a persistent file image is booted on Intel® VCA nodes, it is important that users do not restart the NFS on the host. This could cause a kernel panic for the nodes booted on the card.

This section describes how to manually set up a persistent file system if using the persistent image described in Section 4.5.3 cannot be used. Unlike using the persistent image, if following these steps, the user needs to mount back to the host on each reboot of the card. All commands are written with IPs from static configurations; for DHCP configurations, proper IP addresses need to be used along with the NFS server address and host address.

1. Set up either host or any other server as the NFS server. For information on setting up a CentOS* 7 host as NFS server, see Section B.5.
2. Boot the nodes.

   ```
   sudo vcactl boot vca_baremetal_<version>.img
   ```

   Wait for the nodes to be ready. (The `sudo vcactl status` command should show all nodes as `net_device_ready`.)

   Get IP address of nodes with following command

   ```
   vcactl network ip <card-ID> <CPU-ID>
   ip a
   ```

   This command should show ethX network devices for each Intel® VCA node.
3. Configure CentOS* for passwordless SSH as detailed in section B.2. Test the configuration by using the following command. Change the `1 2 3` to match the number of nodes in the system and 172.31.x.1 to the IP address of the node. (e.g. `1 2 3 4 5 6` for six nodes).

   ```
   for i in 1 2 3 ; do ssh root@172.31.${i}.1 hostname ; done
   ```

   The command should not prompt for a password and return the hostname for each node.
4. Set up the NFS folders, update `/etc/exports`, and export the file system with a single command. Change the `1 2 3` to match the number of nodes in the system (e.g. `1 2 3 4 5 6` for six nodes).

Note: The following is a helpful script to perform steps 4-5. If using this script, skip to step 8.

```
sudo bash
for i in 1 2 3 ; do for j in root etc home opt srv usr var; do mkdir -p
   /media/share/node${i}/${j} ; if [ `grep -c /media/share/node${i}
   /etc/exports` -eq "0" ] ; then echo "/media/share/node${i}
   *(no_root_squash,rw)" >> /etc/exports; fi ; if [ `grep -c
   /media/share/node${i}/${j} /etc/exports` -eq "0" ] ; then echo
   "/media/share/node${i}/${j}     *(no_root_squash,rw)" >> /etc/exports ;
   fi ; done ; done ; exportfs –a
exit
```

5. (Skip this step if the script in step 3 was executed.) Set up NFS shares folders.

   ```
   sudo mkdir –p /media/share/node1/root
   sudo mkdir –p /media/share/node1/etc
   sudo mkdir –p /media/share/node1/home
   sudo mkdir –p /media/share/node1/opt
   sudo mkdir –p /media/share/node1/srv
   ```

```
sudo mkdir -p /media/share/node1/usr
sudo mkdir -p /media/share/node1/var
```

Repeat for each folder that is to be made persistent. Repeat for each node in the system.

6. (Skip this step if the script in step 3 was executed.) Create or update `/etc/exports` with:

```
/media/share/node1   *(rw,sync,no_root_squash,no_all_squash)
/media/share/node1/root   *(rw,sync,no_root_squash,no_all_squash)
/media/share/node1/etc   *(rw,sync,no_root_squash,no_all_squash)
/media/share/node1/home   *(rw,sync,no_root_squash,no_all_squash)
/media/share/node1/opt   *(rw,sync,no_root_squash,no_all_squash)
/media/share/node1/srv   *(rw,sync,no_root_squash,no_all_squash)
/media/share/node1/usr   *(rw,sync,no_root_squash,no_all_squash)
/media/share/node1/var   *(rw,sync,no_root_squash,no_all_squash)
```

Repeat for each node in the system.

7. (Skip this step if the script in step 3 was executed.) Export the file systems.

```
sudo exportfs -a
```

8. Copy the directories from the nodes to the host shares. The following script goes into each node and each folder and copies the content back to the host. Again, change the `1 2 3` to match the number of nodes to be processed and change the list of directories (`root etc home opt srv usr var`) to include all folders to be made persistent. `172.31.${i}.1` is the IP address of the nodes and `172.31.${i}.254` is the NFS server in use, either the host or other NFS server.

```
for i in 1 2 3 ; do for j in root etc home opt srv usr var ; do ssh
   root@172.31.${i}.1 "mount 172.31.${i}.254:/media/share/node${i}   /mnt ;
   cp -af /${j}   /mnt/ ; umount   /mnt" ; done ; done
```

9. Mount the directories to the nodes. This step must be done on every reboot. Again, change the `1 2 3` to match the number of nodes to be processed and change the list of directories (`root etc home opt srv usr var`) to include all folders to be made persistent.

```
for i in 1 2 3 ; do for j in root home opt srv usr var etc; do ssh
   root@172.31.${i}.1 "cd / ; mount
   172.31.${i}.254:/media/share/node${i}/${j} /${j}" ; done ; done
```

10. Verify that the shares are mounted. This step is recommended on every reboot. Again, change the `1 2 3` to match the number of nodes to be processed. Review the output to verify that the expected directories and mounted via NFS.

```
for i in 1 2 3 ; do ssh root@172.31.${i}.1 "mount" ; done
```

Software can now be installed to the nodes normally. If using yum, the steps in section B.4 should be followed to configure a local yum repository. On each subsequent reboot, step 9 is required and step 10 is recommended.

4.5.5 Running Baremetal Image on Intel® VCA

The baremetal version is available for Linux (CentOS*), Windows* 10, and Windows Server* 2016 images.

4.5.5.1 Running Linux* Baremetal Image

1. Run image on selected nodes.

   ```
   vcactl boot <card_ID> <node_ID> vca_baremetal_<version>.img
   ```

2. Check IP address of nodes.

   ```
   vcactl network ip <card-ID> <node-ID>
   ```

3. Connect to the node via ssh (default password is `vista1`).

   ```
   ssh root@<ip-address>
   ```

Note: It is recommended to change the password using the `passwd` command.

4.5.5.2 Running Windows* Baremetal Image

Intel does not distribute Windows baremetal images. For information on creating a Windows baremetal image, refer to section A.4.

1. Unzip the image (e.g. in `/home/Win_images`). Make sure that each VCA node has its own copy of `vca_windows*.img`.

   ```
   mkdir /home/Win_images
   cd /home/Win_images
   unzip vca_windows_10_*<version>.zip
   ```

2. Open image via blockIO.

   ```
   vcactl blockio open <card_id> <cpu_id> vcablk0 RW
      vca_windows_10_baremetal_<version>.img
   ```

3. Run Windows on node. After first boot to Windows, the OS reboots. Wait for `net_device_ready` status.

   ```
   vcactl boot <card_id> <cpu_id> vcablk0
   ```

4. Check IP address of nodes.

   ```
   vcactl network ip <card-ID> <node-ID>
   ```

5. Connect to the node (e.g. via Remote Desktop Connection). The default user is `Administrator` and the password is `vista12#`.

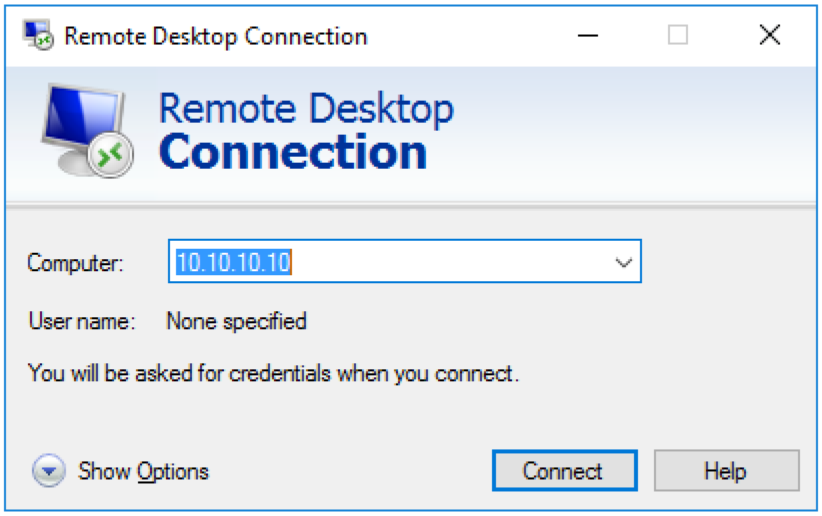


Figure 9. Connect to node using Remote Desktop Connection

Note: It is recommended to change the password.

4.5.6 Running KVM Virtulization on Intel® VCA

4.5.6.1 Running Windows* Baremetal on Node via KVM Hypervisor

1. Set up either host or any other server as the NFS server. For information on setting up a CentOS* 7 host as NFS server, see Section B.5.
2. Create NFS mount points in the node's filesystem.

   ```
   sudo mkdir /mnt/kvm
   ```

Note: Any folder can be created to host the Dom U files. For this document, `/mnt/kvm` is used as an example. If a different folder is selected, change the below occurrences of `/mnt/kvm`, as well.

---

3. Create or update `/etc/exports`. This is needed for the nodes to have access.

   ```
   /mnt/kvm *(rw,sync,no_root_squash,no_all_squash)
   ```

4. Export the directories in `/etc/exports`.

   ```
   sudo exportfs -a
   ```

5. Run KVM on node.

   ```
   vcactl boot <card_ID> <node_ID> vca_kvm_<OS_version>_<version>.img
   ```

6. Run Windows Baremetal.
   a. On Host run following command to start Windows VM on node.

      ```
      vca_start_card_kvm.sh -p /mnt/kvm/ -c <card_ID> -n <node_ID> -i
         vca_windows_10_baremetal_<version>.img --kvm_vnc_port 5900 -b --
         boot-uefi
      ```

   `boot-uefi` is required, if the Windows image is created with steps specified in Appendix A.4.

   b. Log in to node system (hypervisor system – Linux) using SSH.
   c. Turn off firewall on node system (on Linux) (necessary if using RDP).

      ```
      sudo sysctld firewall stop
      ```

d. Log into Windows (e.g. via RDP). The default user is `Administrator` and the password is `vista12#` (if the Windows image is created with steps specified in Appendix A.4).

4.5.6.2 Configuring and Creating an Image

Note: This method of creating a Windows guest operating system for KVM is deprecated. Instead, use the Windows image created as a Windows baremetal operating system as described in Section A.4.

The Intel® VCA can be run in a virtualized environment. Users may want to run in virtualized environment to run applications in an OS other than CentOS* 7.2 or to use virtualization orchestration tools to start their application. A single VM per node on the Intel® VCA card is the only supported configuration at this time. This section describes how to boot the KVM hypervisor and load a guest OS.

IMPORTANT: After setup is completed and operating system booted on Intel® VCA nodes, it is important to not restart the NFS service on the host. This could cause a kernel panic for the nodes booted on the card.

Configure NFS Server
To set-up CentOS* 7 NFS server, see Section B.5 .

1. Create NFS mount points in the node's filesystem.

   ```
   sudo mkdir /mnt/kvm
   ```

Note: Any folder can be created to host the DomU files. For this document, `/mnt/kvm` is used as an example. If a different folder is selected, change the below occurrences of `/mnt/kvm`, as well.

2. Create or update `/etc/exports`.

   ```
   /mnt/kvm *(rw,sync,no_root_squash,no_all_squash)
   ```

3. Export the directories in `/etc/exports`.

   ```
   sudo exportfs -a
   ```

4. Copy scripts to boot DomU to mount folder.

   ```
   sudo cp /usr/sbin/vca_start_card_kvm.sh /mnt/kvm/
   ```

5. Start the NFS server.

   ```
   sudo systemctl enable rpcbind
   sudo systemctl enable nfs-server
   sudo systemctl start rpcbind
   sudo systemctl start nfs
   ```

6. Make sure the firewall is properly configured.

   ```
   sudo firewall-cmd --permanent --zone=`sudo firewall-cmd --get-default-zone`
      --add-service=nfs
   firewall-cmd --permanent --zone=public --add-service vnc-server
   sudo firewall-cmd –-reload
   ```

Create Bootable DomU Image

To create a bootable DomU image using an installation ISO file of an OS, follow the instructions in this section. This example creates a DomU image using Windows* 10; the installation of Windows Server 2016 is analogous.

1. Boot KVM hypervisor on VCA.

   ```
   sudo vcactl boot <card_ID> <node_ID> vca_kvm_<version>.img
   ```

2. Find the IP address of the KVM hypervisor.

   ```
   sudo vcactl network ip <card_ID> <node_ID>
   ```

3. SSH to the VCA node with above IP address. The default user name is `root` and default password is `vista1`.
4. On the VCA node, Configure the firewall to allow VNC service by executing the following commands on both the VCA node and host.

   ```
   firewall-cmd --permanent --zone=public --add-service vnc-server
   firewall-cmd --reload
   ```

5. Create an empty disk image at shared folder `/mnt/kvm`. The container needs to be at least 10 GB, but at least 20 GB is recommended. The sample name (`win10.img`) may be changed.

   ```
   sudo dd if=/dev/zero of=win10.img bs=1 count=0 seek=20G
   ```

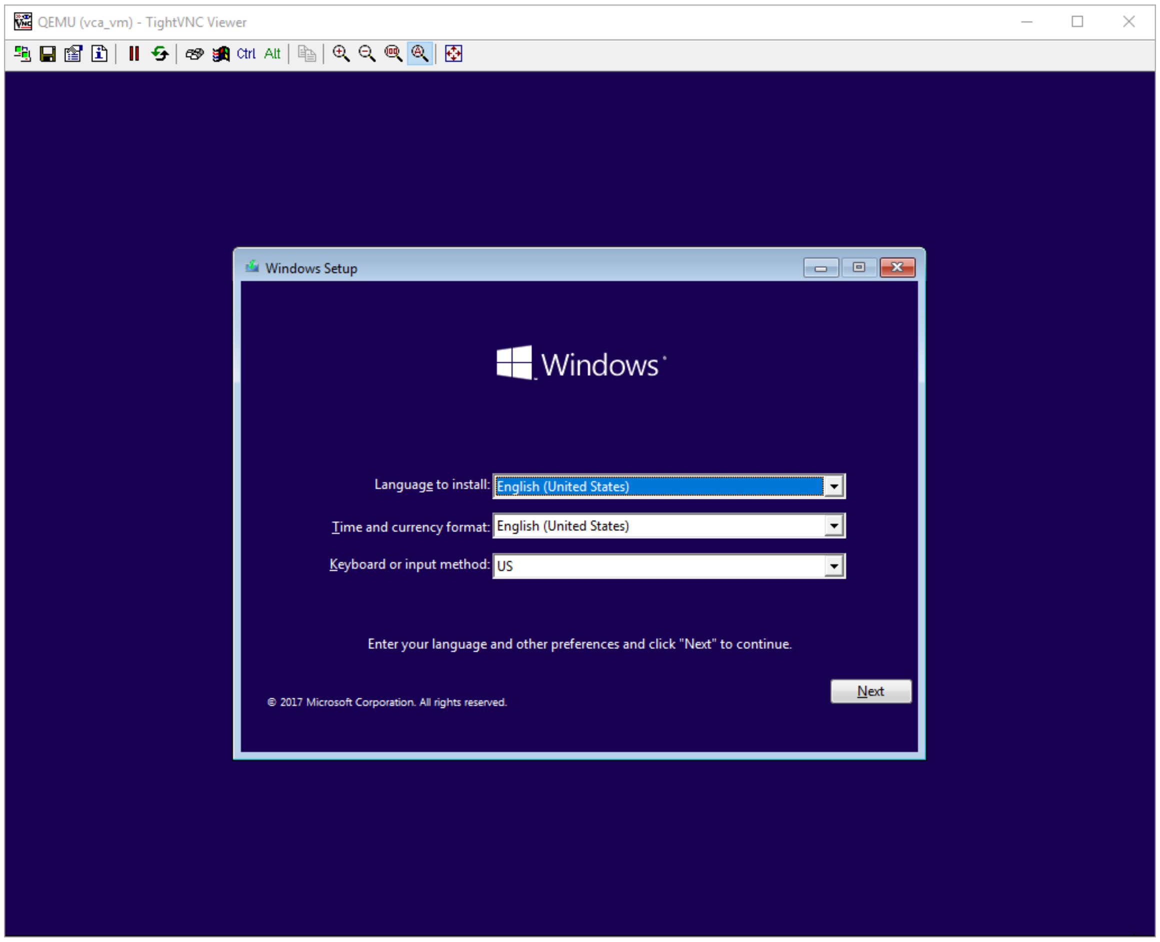
6. Assuming an ISO file of Windows 10 named `en_windows_10_enterprise_x64_dvd.iso`, copy the iso file to shared folder `/mnt/kvm` and run following commands as root.

   ```
   sudo ./vca_start_card_kvm.sh -c <card_ID> -n <node_ID> -i win10.img --
      install en_windows_10_enterprise_x64_dvd.iso --kvm_vnc_port 5900 -g -p
      /mnt/kvm
   ```

Note: The command `./vca_start_card_kvm.sh –help` prints a list of command options.

7. When `Creating domain...   | 0 B 00:02` is printed out by the `./vca_start_card_kvm.sh` command, connect a VNC client (e.g. tightVNC viewer) to `host_ip:5900` where `host_ip` is the IP address of VCA host platform. The VNC client must support **Ctrl-Alt-Del**.

8. In Windows setup, press **Next** (Figure 10) and then choose **Install now**.



Language to install: English (United States)

Time and currency format: English (United States)

Keyboard or input method: US

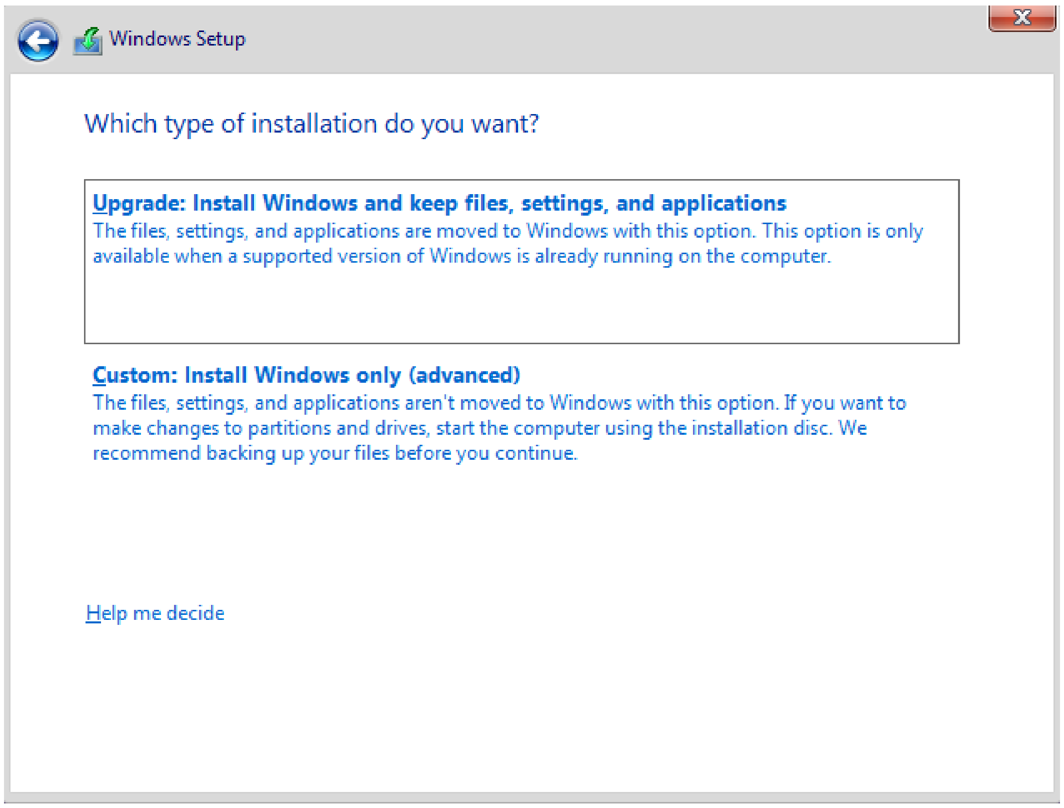
Enter your language and other preferences and click "Next" to continue.

© 2017 Microsoft Corporation. All rights reserved.

Figure 10. Windows* setup screen

9. Enter the Windows activation key and accept the license terms.

10. Choose **Install Windows only (advanced)** for the type of installation (Figure 11).


Windows Setup

Which type of installation do you want?

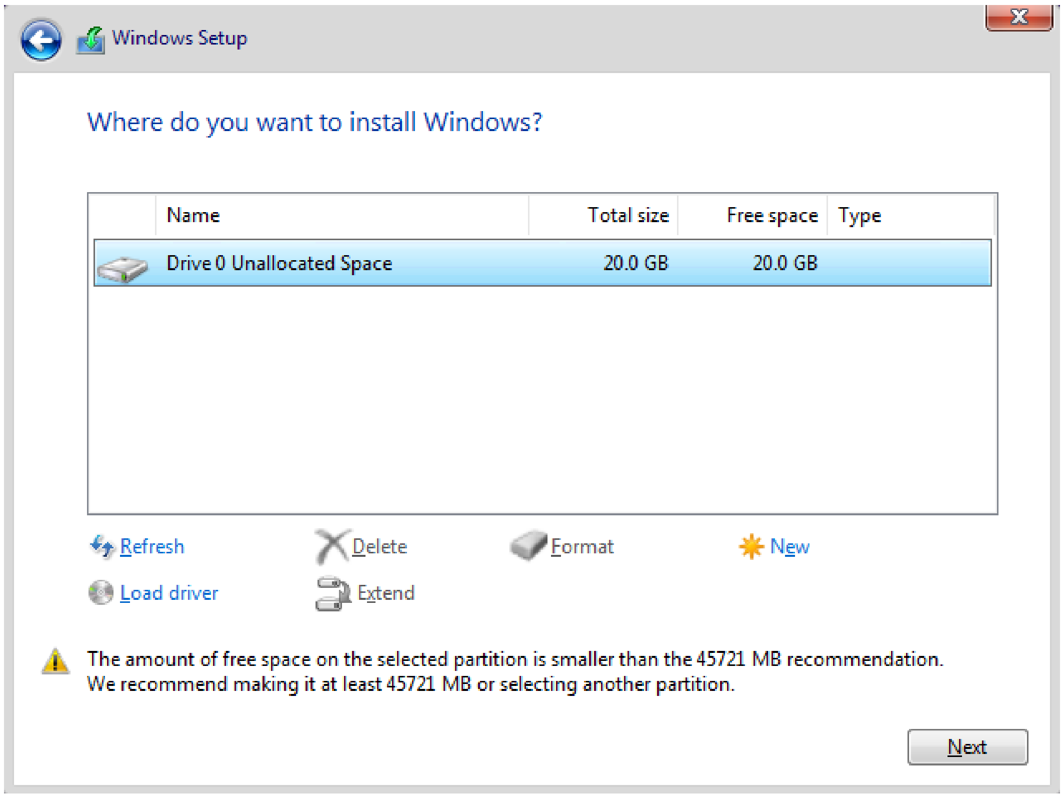
Upgrade: Install Windows and keep files, settings, and applications
The files, settings, and applications are moved to Windows with this option. This option is only available when a supported version of Windows is already running on the computer.

Custom: Install Windows only (advanced)
The files, settings, and applications aren't moved to Windows with this option. If you want to make changes to partitions and drives, start the computer using the installation disc. We recommend backing up your files before you continue.

Help me decide

Figure 11. Select type of installation

11. On the screen with available drives, click **Next** to start installation (Figure 12).


Windows Setup

Where do you want to install Windows?

Name	Total size	Free space	Type
Drive 0 Unallocated Space	20.0 GB	20.0 GB	

The amount of free space on the selected partition is smaller than the 45721 MB recommendation. We recommend making it at least 45721 MB or selecting another partition.

Figure 12. Select installation location

12. After Windows is installed, configure the personalization settings as they are presented such as region, keyboard layout, network connection, login information, and privacy settings.
13. Right click on the Windows button and shut down the VM. This triggers KVM hypervisor to destroy the current Windows 10 guest.

Now we have a bootable image of Windows 10 (`win10.img` in the example).

Configure Bootable Image

To further configure this image, perform the following steps.

1. On the host platform download the latest stable drivers in form of virtio-win iso file (`virtio-win-<version>.iso`) from https://fedoraproject.org/wiki/Windows_Virtio_Drivers.
2. Copy `virtio-win-<version>.iso` to shared folder `/mnt/kvm` and create KVM guest with `win2k12.img` and `virtio-win-<version>.iso`.

   ```
   sudo ./vca_start_card_kvm.sh -n <node-ID> -c <card-ID> -i win10.img --
      kvm_vnc_port 5900 -g -p /mnt/kvm --cdrom virtio-win-<version>.iso
   ```

   After booting, `virtio-win-<version>.iso` is seen as a CD-ROM.

3. Right click on the Windows button and choose **Device Manager**.
4. Right click **Ethernet Controller** in **Other devices** and select **Update driver** (Figure 13).

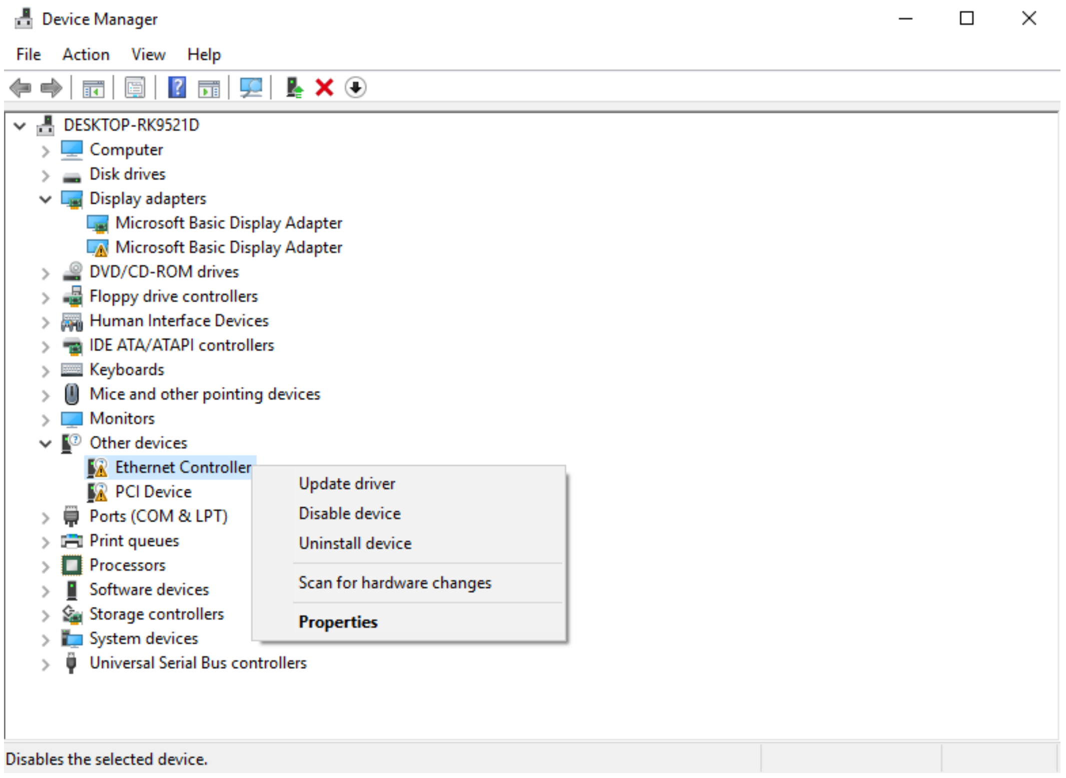
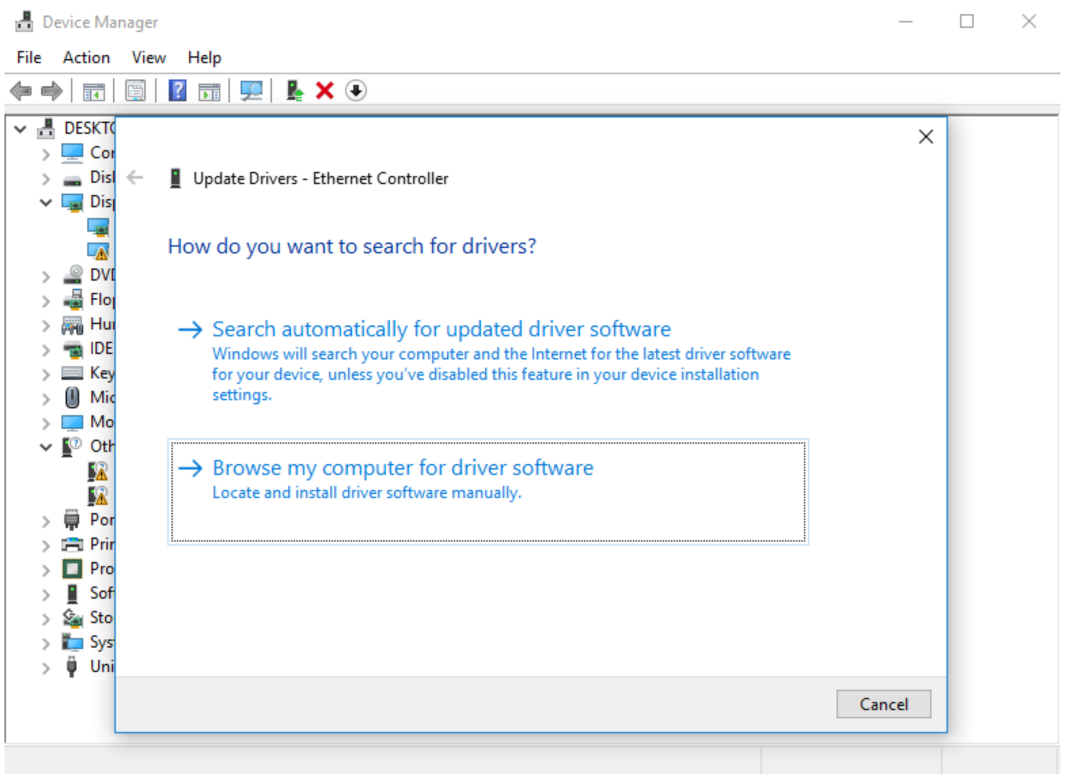


Figure 13. Update Ethernet controller driver

5. Select **Browse my computer for driver software** (Figure 14).



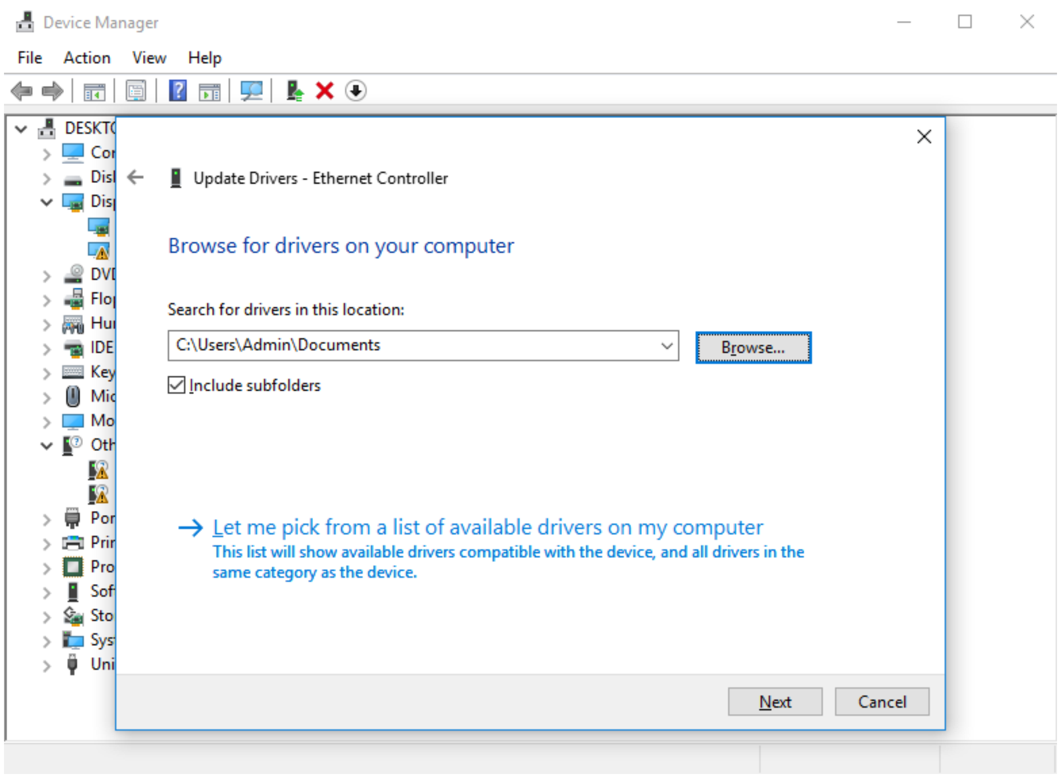
How do you want to search for drivers?

→ Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

→ Browse my computer for driver software
Locate and install driver software manually.

Figure 14. Browse for Ethernet controller driver software

6. Check the box **Include subfolders** and click the **Browse** button (Figure 15).



Browse for drivers on your computer

Search for drivers in this location:
C:\Users\Admin\Documents

☑ Include subfolders

→ Let me pick from a list of available drivers on my computer
This list will show available drivers compatible with the device, and all drivers in the same category as the device.

Figure 15. Browse for driver on computer

7. Select the CD drive with `virtio-win` (Figure 16). Click **OK**, then **Next**, then **Install**.

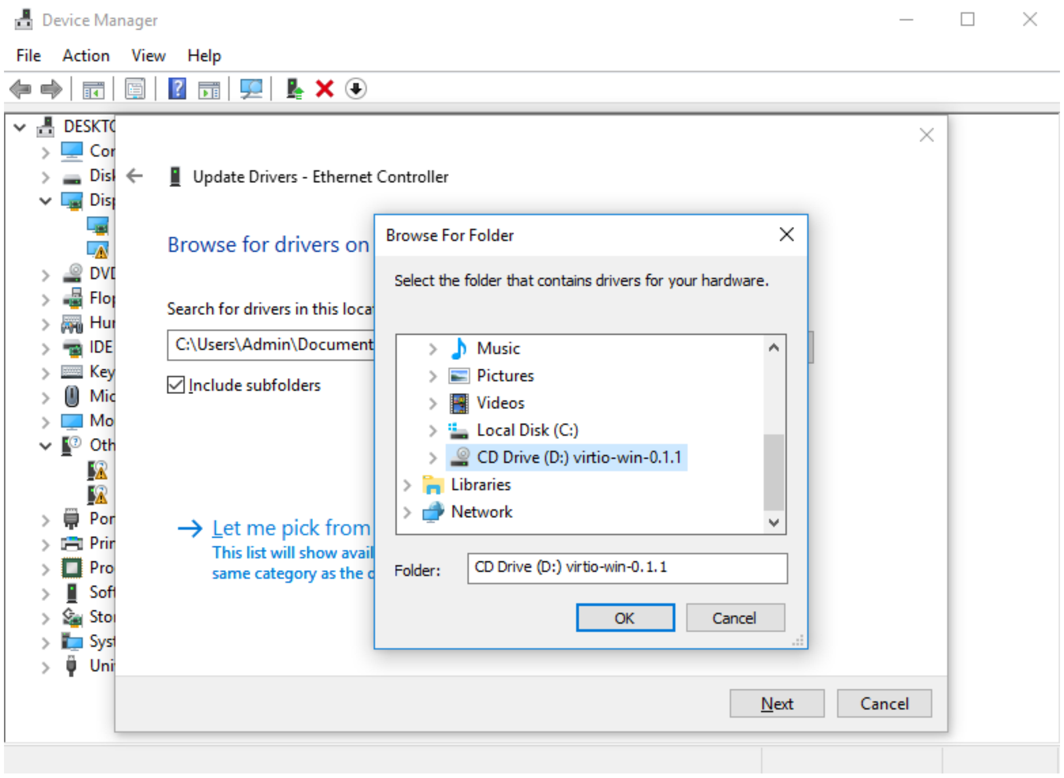


Figure 16. Select `virtio-win` CD drive

8. After installing the Ethernet driver, repeat the same steps for **PCI Device** in **Other devices** to install the VirtIO Ballon driver.

9. Open the Device Manager. Right click **Red Hat VirtIO Ethernet Adapter** under **Network Adapters** and select **Properties** (Figure 17).

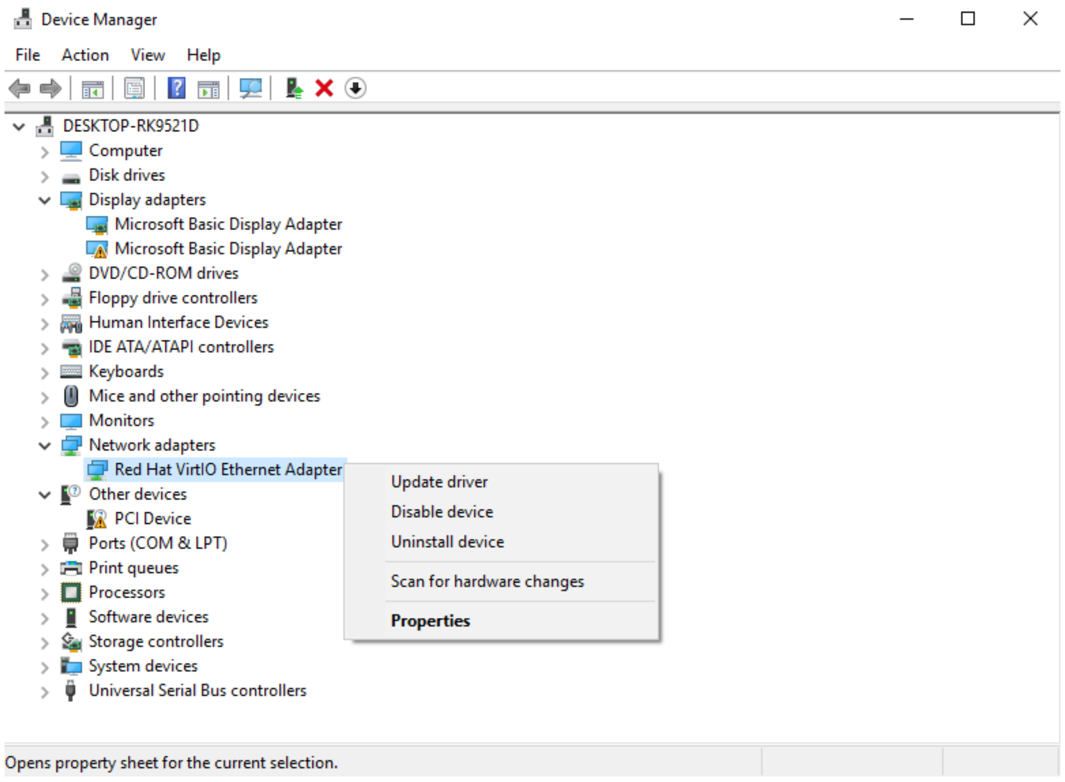
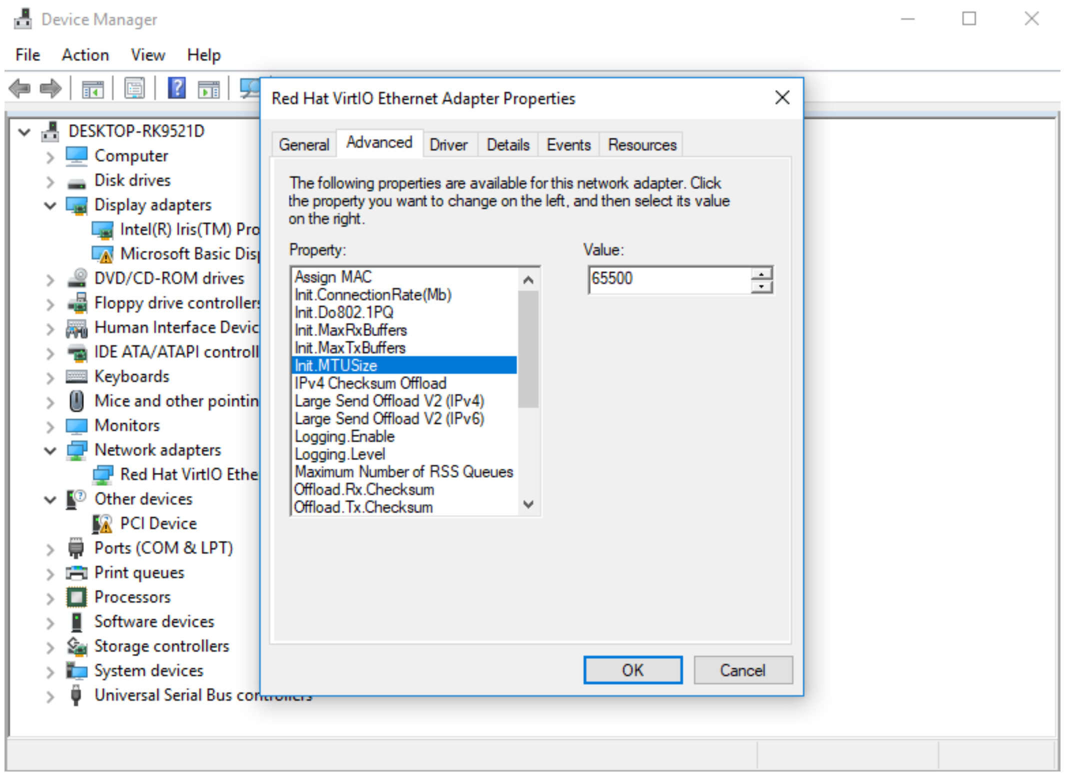


Figure 17. Select Red Hat VirtIO Ethernet Adapter

10. On the **Advanced** tab select **Init MTU Size** and specify a value of `65500` (Figure 18).



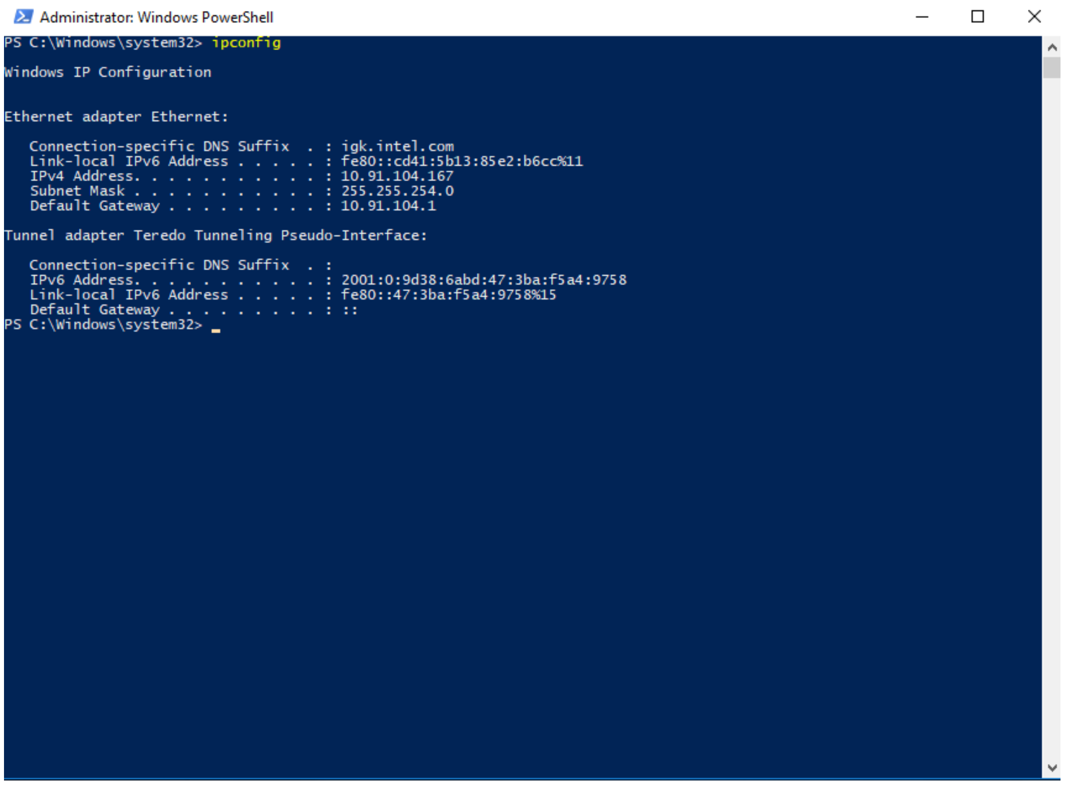
The following properties are available for this network adapter. Click the property you want to change on the left, and then select its value on the right.

Property:
Assign MAC
Init.ConnectionRate(Mb)
Init.Do802.1PQ
Init.MaxRxBuffers
Init.MaxTxBuffers
Init.MTUSize
IPv4 Checksum Offload
Large Send Offload V2 (IPv4)
Large Send Offload V2 (IPv6)
Logging.Enable
Logging.Level
Maximum Number of RSS Queues
Offload.Rx.Checksum
Offload.Tx.Checksum

Value: 65500

Figure 18. Red Had VirtIO properties

11. The KVM guest obtains its address from the KVM DHCP server. To verify, open Windows PowerShell* and invoke `ipconfig` command (Figure 19).

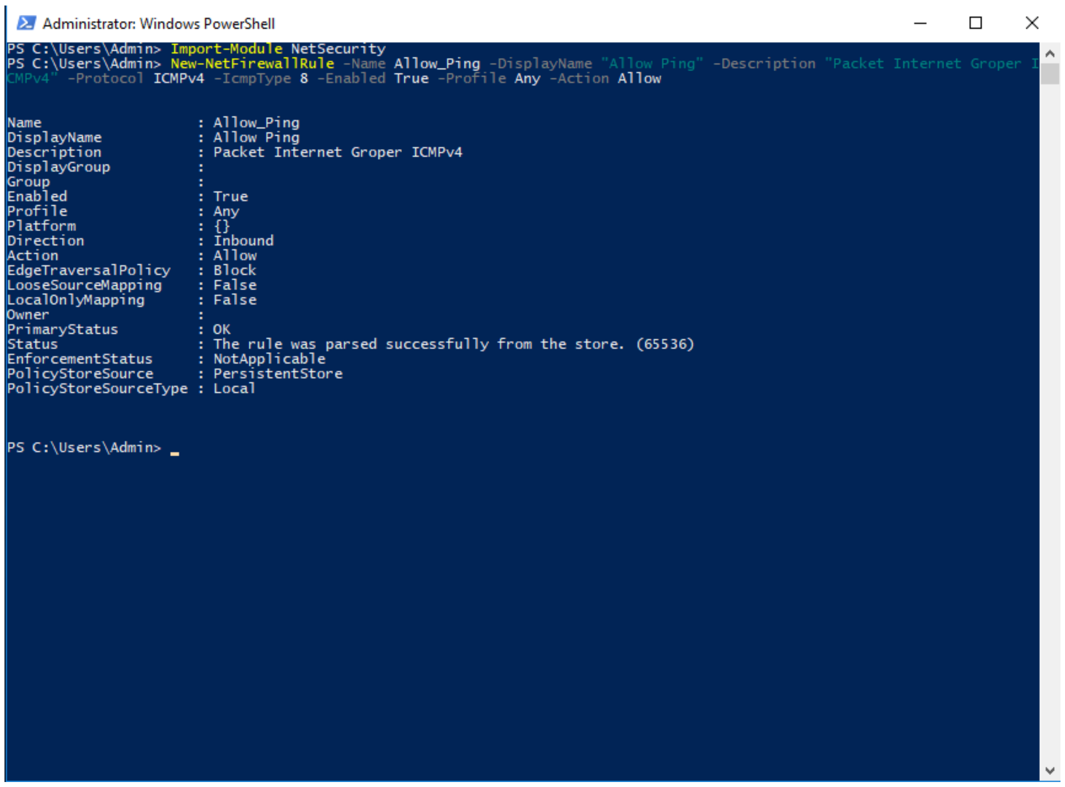

Administrator: Windows PowerShell

```
PS C:\Windows\system32> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : igk.intel.com
   Link-local IPv6 Address . . . . . : fe80::cd41:5b13:85e2:b6cc%11
   IPv4 Address. . . . . . . . . . . : 10.91.104.167
   Subnet Mask . . . . . . . . . . . : 255.255.254.0
   Default Gateway . . . . . . . . . : 10.91.104.1

Tunnel adapter Teredo Tunneling Pseudo-Interface:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:0:9d38:6abd:47:3ba:f5a4:9758
   Link-local IPv6 Address . . . . . : fe80::47:3ba:f5a4:9758%15
   Default Gateway . . . . . . . . . : ::
PS C:\Windows\system32> _
```

Figure 19. `ipconfig` in Windows PowerShell*

12. By default, KVM guest (Win10 in this example) does not respond to pings. To enable pinging, add proper firewall rule using PowerShell (Figure 20).

```
Import-Module NetSecurity
New-NetFirewallRule -Name Allow_Ping -DisplayName “Allow Ping” -Description
   “Packet Internet Groper ICMPv4” -Protocol ICMPv4 -IcmpType 8 -Enabled
   True -Profile Any -Action Allow
```

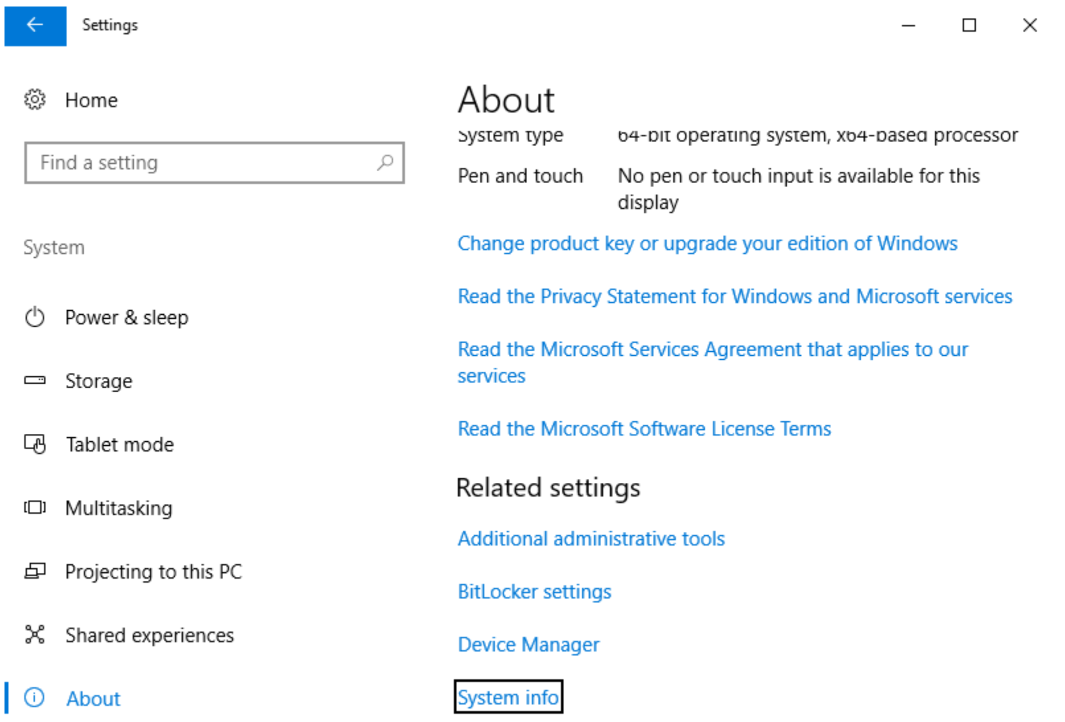


Administrator: Windows PowerShell

```
PS C:\Users\Admin> Import-Module NetSecurity
PS C:\Users\Admin> New-NetFirewallRule -Name Allow_Ping -DisplayName "Allow Ping" -Description "Packet Internet Groper I
CMPv4" -Protocol ICMPv4 -IcmpType 8 -Enabled True -Profile Any -Action Allow


Name                  : Allow_Ping
DisplayName           : Allow Ping
Description           : Packet Internet Groper ICMPv4
DisplayGroup          :
Group                 :
Enabled               : True
Profile               : Any
Platform              : {}
Direction             : Inbound
Action                : Allow
EdgeTraversalPolicy   : Block
LooseSourceMapping    : False
LocalOnlyMapping      : False
Owner                 :
PrimaryStatus         : OK
Status                : The rule was parsed successfully from the store. (65536)
EnforcementStatus     : NotApplicable
PolicyStoreSource     : PersistentStore
PolicyStoreSourceType : Local


PS C:\Users\Admin> _
```

Figure 20. Enable pinging in Windows PowerShell*

13. Enable Remote Desktop Connection at KVM guest. In VNC, right click on the Windows icon and select **System**. Under **Related settings**, select **System info** (Figure 21) and then select **Advanced system settings** (Figure 22).



About

System type 64-bit operating system, x64-based processor

Pen and touch No pen or touch input is available for this display

Change product key or upgrade your edition of Windows

Read the Privacy Statement for Windows and Microsoft services

Read the Microsoft Services Agreement that applies to our services

Read the Microsoft Software License Terms

Related settings

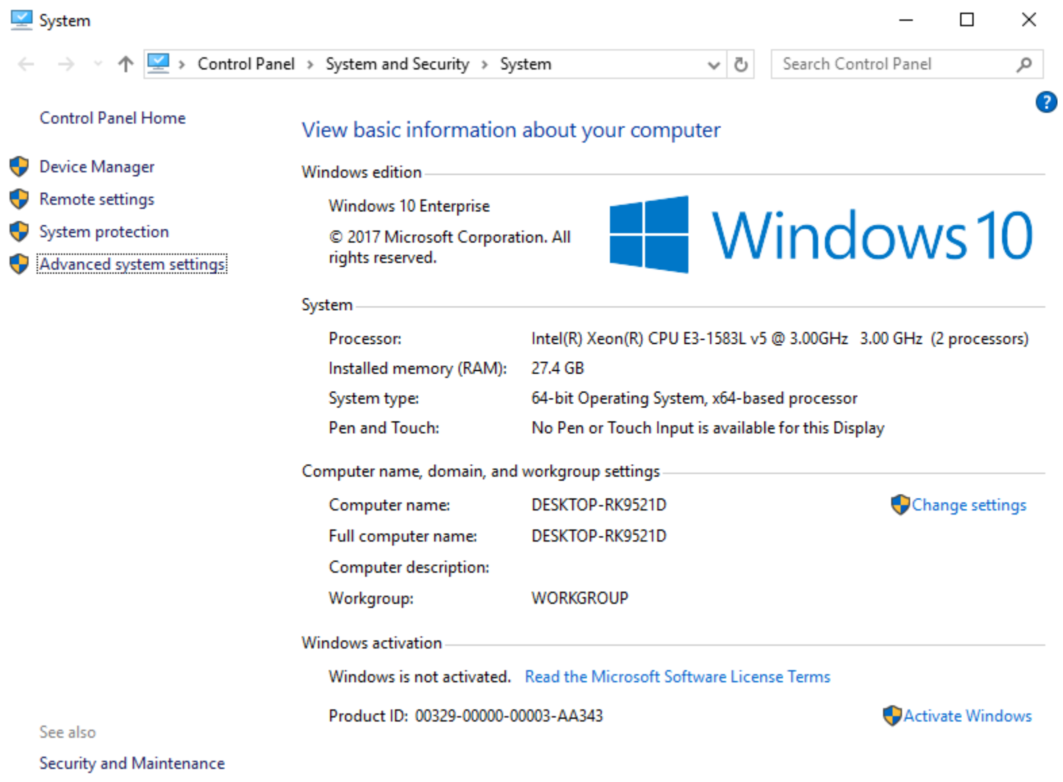
Additional administrative tools

BitLocker settings

Device Manager

System info

Figure 21. System info



View basic information about your computer

Windows edition

Windows 10 Enterprise

© 2017 Microsoft Corporation. All rights reserved.

System

Processor:	Intel(R) Xeon(R) CPU E3-1583L v5 @ 3.00GHz 3.00 GHz (2 processors)
Installed memory (RAM):	27.4 GB
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Computer name, domain, and workgroup settings

Computer name:	DESKTOP-RK9521D
Full computer name:	DESKTOP-RK9521D
Computer description:	
Workgroup:	WORKGROUP

Windows activation

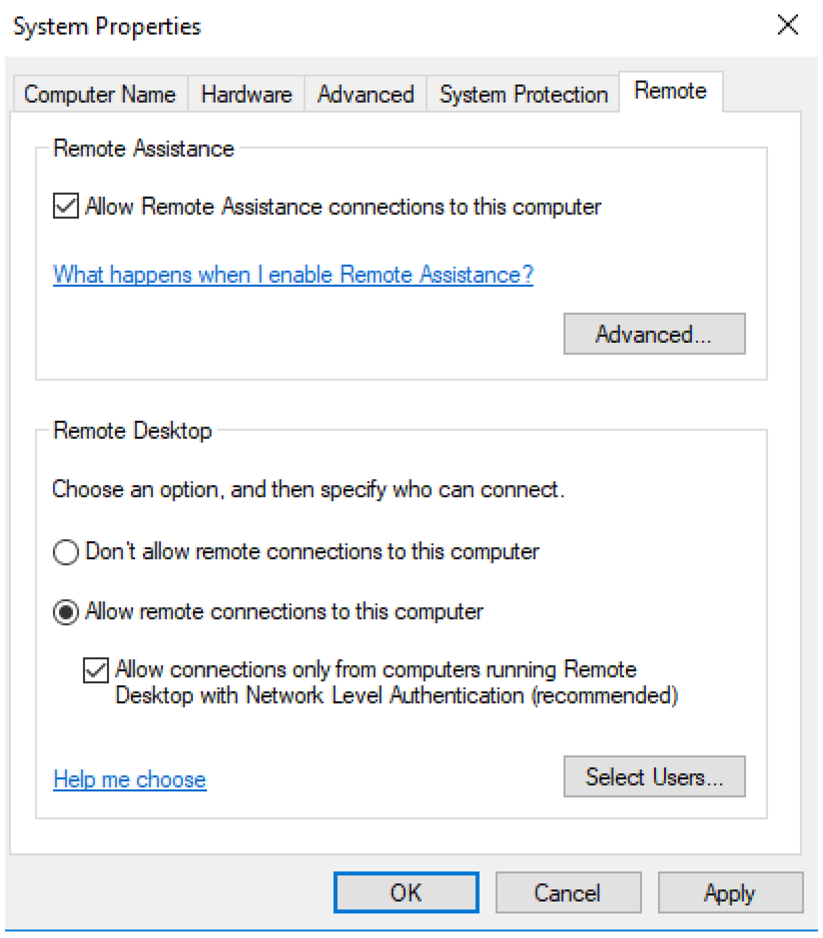
Windows is not activated. Read the Microsoft Software License Terms

Product ID: 00329-00000-00003-AA343

Figure 22. Advanced system settings

14. In the **Remote** tab check **Allow remote connections to this computer** and click **OK** (Figure 23).

Note: The firewall must be also be configured to allow remote desktop connection.



Remote Assistance

☑ Allow Remote Assistance connections to this computer

What happens when I enable Remote Assistance?

Remote Desktop

Choose an option, and then specify who can connect.

○ Don't allow remote connections to this computer

● Allow remote connections to this computer

☑ Allow connections only from computers running Remote Desktop with Network Level Authentication (recommended)

Help me choose

Figure 23. Allow remote connections

15. Shut down the guest. The bootable KVM guest image (`win2k12.img`) is now configured with remote desktop connection (RDP) enabled.

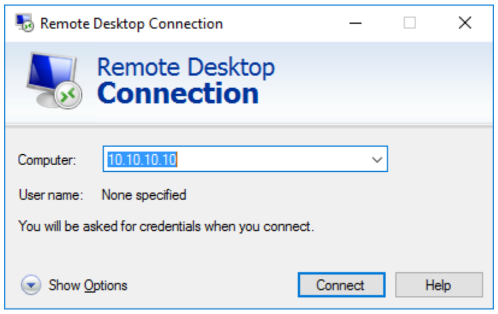
Connect to Guest with Remote Desktop Client

To use RDP, first enable the network bridge so that external computers (other than VCA host) can also access KVM guest. Before performing these steps, follow the instructions in Section 0 to configure the nodes for bridged mode operation.

1. Create KVM guest with `-b` and `--card-mac` options.

```
sudo ./vca_start_card_kvm.sh –b –card-mac -n <node-ID> -c <card-ID> -i
   win10.img --kvm_vnc_port 5900 --vm_rdp_port 5901 -g -p /mnt/kvm
```

2. To find the IP address of the KVM guest, connect to the guest through VNC with address `IP_hypervisor`. In Windows PowerShell, run the command `ipconfig` to find the guest's IP address.
3. Connect to guest with a remote desktop client (Figure 24). If a warning appears select **Don't ask me again for connections to the computer** and click **Yes** (Figure 25).

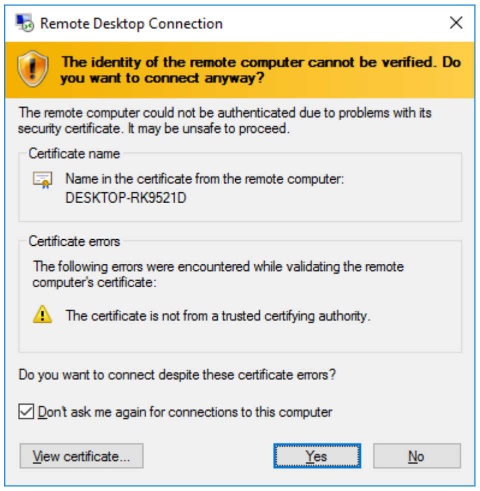


Computer: 10.10.10.10

User name: None specified

You will be asked for credentials when you connect.

Figure 24. Remote desktop connection



The identity of the remote computer cannot be verified. Do you want to connect anyway?

The remote computer could not be authenticated due to problems with its security certificate. It may be unsafe to proceed.

Name in the certificate from the remote computer: DESKTOP-RK9521D

The following errors were encountered while validating the remote computer's certificate:

The certificate is not from a trusted certifying authority.

Do you want to connect despite these certificate errors?

☑ Don't ask me again for connections to this computer

Figure 25. Security certificate warning

Windows 10 is now successfully running as a KVM guest.

There are few small differences between Windows 10 and Windows Server 2016. If there are any problems with Windows installation, visit https://support.microsoft.com/en-us.

Install Graphics Driver

To use hardware accelerated media and graphics processing, the Intel® Graphics Driver (the default display driver of KVM does not support Intel media and graphics hardware) must be installed. The recommend approach to install Intel Graphics Driver is through installation of Intel® Media Server Studio (the community version is free to use), since Intel Graphics Driver is included in Intel Media Server Studio.

After successfully installing the Intel Graphics Driver, Iris® Pro Graphics should appear in the Device Manager.

4.5.7 Running KVM-GT Virtualization on Intel® VCA

IMPORTANT: After setup is completed, and operating system booted on Intel® VCA nodes, it is important that users do not restart the NFS service on the host. This could cause a kernel panic for the nodes booted on the card.

KVM-GT virtualization is supported only on Intel® VCA 2 (VCA1585LMV) card. Maximum number of VMs supported are 7, when aperture size is greater than or equal to 1024MB. If more than 4VMs are required, it is required to have at least 32GB of RAM per node.

4.5.7.1 Running Windows* Baremetal on Node via KVMGT Hypervisor

1. Set up either host or any other server as NFS server, for information on setting up Ubuntu* host as NFS server, see section C.5
2. Setup bridging for the nodes as described in section 4.4.1
3. Create NFS mount points in the node's filesystem.

```
sudo mkdir /mnt/kvmgt
```

Note: Any folder can be created to host the Dom U files. For this document, `/mnt/kvmgt` is used as an example. If a different folder is selected, change the below occurrences of `/mnt/kvmgt`, as well.

4. Create or update `/etc/exports`. This is needed for the nodes to have access.

```
/mnt/kvmgt *(rw,sync,no_root_squash,no_all_squash,no_subtree_check)
```

5. Export the directories in `/etc/exports`.

```
sudo exportfs -a
```

6. Aperture size should be equal or higher than 512MB to use more than 2 VMs. Following command sets the Aperture size for nodes

```
vcactl set-BIOS-cfg gpu-aperture 1024
```

7. Wait until BIOS is up and running on the nodes
8. Boot the nodes with KVMGT image.

```
vcactl boot <card_ID> <node_ID> vca_kvmgt_<version>.img
```

9. Wait until nodes are booted up

```
vcactl wait <card_ID> <node_ID>
```

10. `vca_kvmgtctl` is used to create, manage, and remove the VMs for KVMGT on the host. The following table gives the different command descriptions. The node password needs to be entered during the execution of the included commands.

Table 10. `vca_kvmgtctl` commands

Function	Description	Command
Drivers	Show available drivers	`/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -x drivers`
Help	Print help	`/usr/sbin/vca_kvmgtctl.sh -h`
Info	Show info about actual configuration	`/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -x info`
Remove unused	Remove unused device	`/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -x remove_unused`
Set max GPU freq	Set min GPU freq in MHz (should be lower than 1150)	`/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -x set_max_freq <value>`
Set min GPU freq	Set min GPU freq in MHz (should be higher than 350)	`/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -x set_min_freq <value>`
Start VM	Start vm from selected img	`/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -m <vm_id> -x start <img_path> <img_name> <mount_dir> <host_ip>`
Status	Show VM status with IP, Mac	`/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -x status`

11. Run Windows Baremetal.
    a. Make sure there is a separate image for each VM in `/mnt/kvmgt` (the NFS directory created above).

b. On the host, run following command to start VM on node. The node password needs to be entered during the execution of following command.

```
/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -m <vm_id> -x start <img_path>
   <img_name> <mount_dir> <host_ip>
ex.
/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -m 1 -x start /mnt/sda
   win_10_<version>.img /mnt/disk1 10.91.100.100
```

`/usr/sbin/vca_kvmgtctl.sh -c 0 -n 0 -x status` shows the VM IP address, MAC address and VNC port number. The default VNC port is `vm_id+10`.

c. Turn off the firewall on the node system (on Linux) (necessary if using RDP/VNCviewer).

```
sudo sysctld firewall stop
```

d. Log into Windows (e.g. via VNCviewer/RDP). The default user is `Administrator` and the password is `vista12#`.

e. In `vca_kvmgtctl.sh`, the MAC address is reserved automatically from node MAC address.

```
Ex. Node mac:    12:34:56:  78:                 90:ab
VM1 mac:         12:34:56:  (7+vm_id+1)8:         90:ab
```

4.5.8 Running XEN Virtualization on Intel® VCA

IMPORTANT: After setup is completed, and operating system booted on Intel® VCA nodes, it is important that users do not restart the NFS service on the host. This could cause a kernel panic for the nodes booted on the card.

Intel® VCA can be run in a virtualized environment. Users may want to run in virtualized environment to run applications in an OS other than CentOS* 7.2 or to use virtualization orchestration tools to start their application. A single VM per node on the Intel® VCA card is the only supported configuration at this time. This section describes how to boot the XEN hypervisor and load a guest OS.

4.5.8.1 Configure NFS Server

1. Set up either host or any other server as NFS server, for information on setting up a CentOS* 7 host as NFS server, see section B.5.
2. Create NFS mount points with node's filesystem (setup for two cards).

```
sudo mkdir /mnt/xen
```

Note: Any folder can be created to host the DomU files. For this document, `/mnt/xen` is used as an example. If a different folder is selected, change the below occurrences of `/mnt/xen`, as well.

3. Create or update `/etc/exports`.

```
/mnt/xen *(rw,sync,no_root_squash,no_all_squash)
```

4. Export the directories in `/etc/exports`.

```
sudo exportfs -a
```

5. Copy the DomU image and required scripts to the mount folder.

```
sudo tar -zxvf vca_domu_<version>.tar.gz -C /mnt/xen/
sudo cp /usr/sbin/vca_start_card_domu.sh /mnt/xen/
sudo cp /usr/lib/vca/card_gfx_vm.hvm /mnt/xen
sudo cp /usr/lib/vca/card_vm.hvm /mnt/xen
```

6. Start the NFS server.

```
sudo systemctl enable rpcbind
sudo systemctl enable nfs-server
sudo systemctl start rpcbind
sudo systemctl start nfs
```

7. Make sure the firewall is properly configured.

```
sudo firewall-cmd --permanent --zone=`sudo firewall-cmd --get-default-zone`
   --add-service=nfs
firewall-cmd --permanent --zone=public --add-service vnc-server
sudo firewall-cmd –reload
```

4.5.8.2 Prepare Image

This procedure assumes a bootable DomU image `win10.img`. To create a bootable image from a Windows ISO file, refer to Section 4.5.6. Before performing these steps, follow the instructions in Section 0 to configure the nodes for bridged mode operation.

1. Boot VCA nodes with the Xen image.
   - For Intel® VCA 2:

     ```
     sudo vcactl boot vca_xen_4.4p_<version>.img
     ```

   - For Intel® VCA:

     ```
     sudo vcactl boot vca_xen_<version>.img
     ```

2. Wait for the nodes to be ready. The command `sudo vcactl status` should show the node status as `net_device_ready`.
3. Copy `win10.img` to the shared folder `/mnt/xen`.
4. On the Intel® VCA host platform, boot the Windows VM DomU image with the emulated network.

```
vca_start_card_domu.sh -c <card_ID> -n <node_ID> -p <path to image at E5> -b
   --card-mac -i <guest_image_name> -f card_vm.hvm --legacy-net
```

For example,

```
vca_start_card_domu.sh -c 0 -n 0 -p /mnt/xen -b --card-mac -i win10.img -f
   card_vm.hvm --legacy-net
```

5. Find the IP address of the Xen hypervisor using the command `vcactl network ip` and then log in to the hypervisor using SSH.

   ```
   ssh <IP_of_hypervisor>
   ```

6. Configure the firewall on the node.

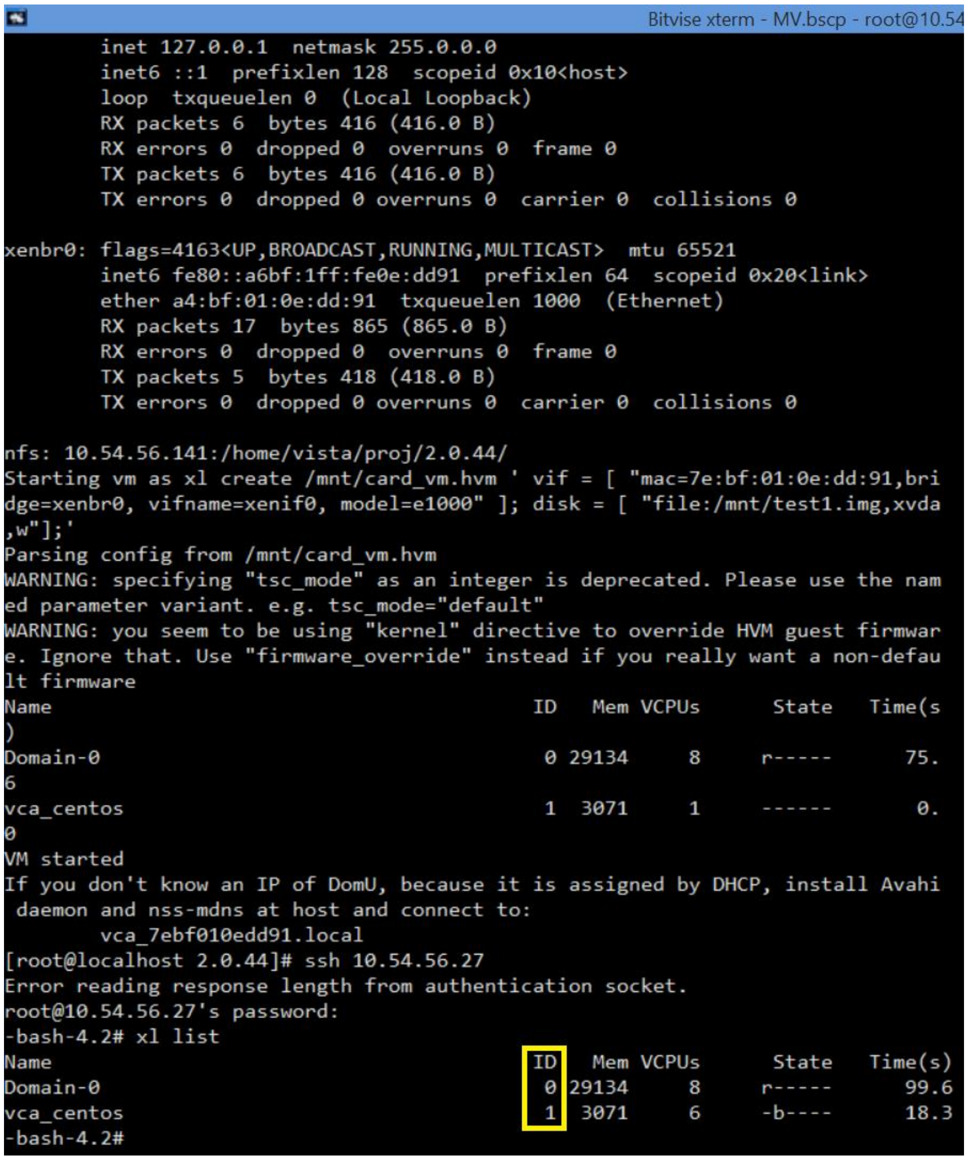
   ```
   firewall-cmd --permanent --zone=public --add-service vnc-server
   firewall-cmd --reload
   ```

4.5.8.3 Install PV Drivers

Install the Xen network PV drivers to replace the emulated network driver for better performance.

1. Download the following drivers from https://www.xenproject.org/downloads/windows-pv-drivers/winpv-drivers-81/winpv-drivers-810.html:
   - Bus driver (`xenbus.tar`)
   - Network class driver (`xenvif.tar`)
   - Network device driver (`xennet.tar`)
2. Untar these driver files to `/mnt/xen`.

3. To find the DomU ID, use the command `xl list`. The ID is listed as shown in Figure 26.

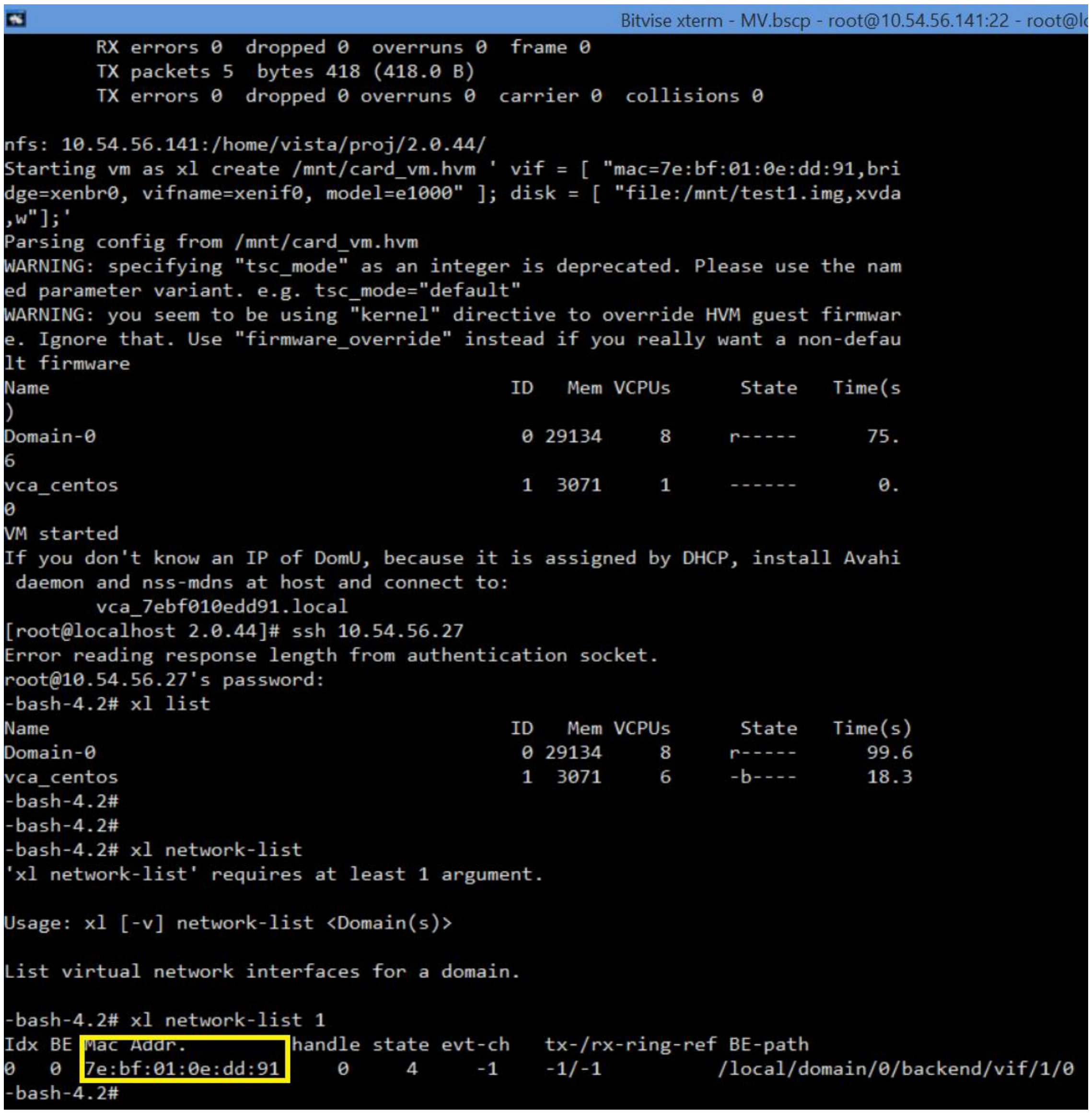


```
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 0  (Local Loopback)
        RX packets 6  bytes 416 (416.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6  bytes 416 (416.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

xenbr0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 65521
        inet6 fe80::a6bf:1ff:fe0e:dd91  prefixlen 64  scopeid 0x20<link>
        ether a4:bf:01:0e:dd:91  txqueuelen 1000  (Ethernet)
        RX packets 17  bytes 865 (865.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5  bytes 418 (418.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

nfs: 10.54.56.141:/home/vista/proj/2.0.44/
Starting vm as xl create /mnt/card_vm.hvm ' vif = [ "mac=7e:bf:01:0e:dd:91,bri
dge=xenbr0, vifname=xenif0, model=e1000" ]; disk = [ "file:/mnt/test1.img,xvda
,w"];'
Parsing config from /mnt/card_vm.hvm
WARNING: specifying "tsc_mode" as an integer is deprecated. Please use the nam
ed parameter variant. e.g. tsc_mode="default"
WARNING: you seem to be using "kernel" directive to override HVM guest firmwar
e. Ignore that. Use "firmware_override" instead if you really want a non-defau
lt firmware
Name                                        ID   Mem VCPUs      State   Time(s
)
Domain-0                                     0 29134     8     r-----      75.
6
vca_centos                                   1  3071     1     ------       0.
0
VM started
If you don't know an IP of DomU, because it is assigned by DHCP, install Avahi
 daemon and nss-mdns at host and connect to:
        vca_7ebf010edd91.local
[root@localhost 2.0.44]# ssh 10.54.56.27
Error reading response length from authentication socket.
root@10.54.56.27's password:
-bash-4.2# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0 29134     8     r-----      99.6
vca_centos                                   1  3071     6     -b----      18.3
-bash-4.2#
```

Figure 26. Find DomU ID using `xl list` command

4. To find the DomU MAC address, run the following command still on the hypervisor (Figure 27).

   ```
   xl network-list <ID of domU>
   ```



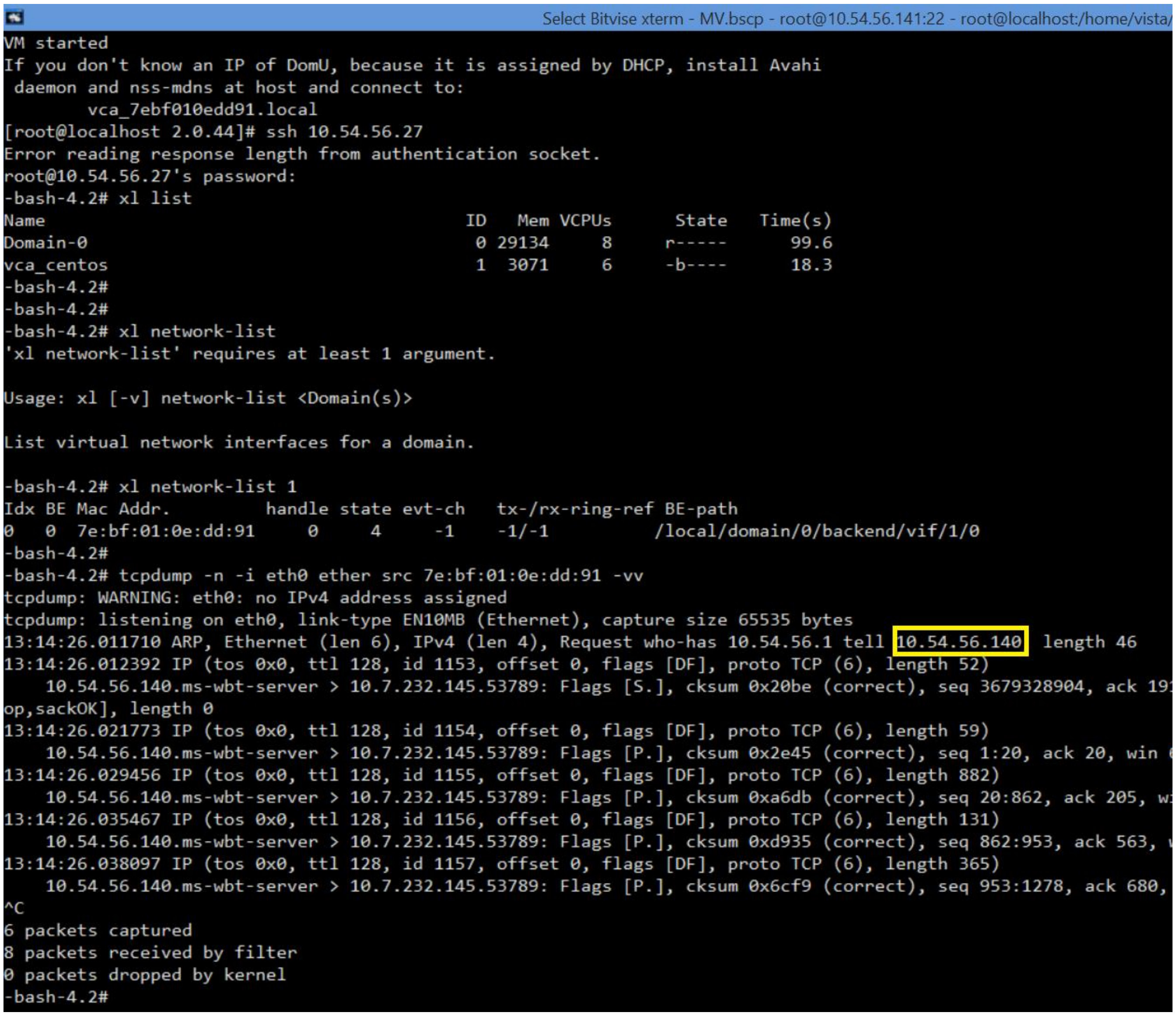
Bitvise xterm - MV.bscp - root@10.54.56.141:22 - root@l

```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5  bytes 418 (418.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

nfs: 10.54.56.141:/home/vista/proj/2.0.44/
Starting vm as xl create /mnt/card_vm.hvm ' vif = [ "mac=7e:bf:01:0e:dd:91,bri
dge=xenbr0, vifname=xenif0, model=e1000" ]; disk = [ "file:/mnt/test1.img,xvda
,w"];'
Parsing config from /mnt/card_vm.hvm
WARNING: specifying "tsc_mode" as an integer is deprecated. Please use the nam
ed parameter variant. e.g. tsc_mode="default"
WARNING: you seem to be using "kernel" directive to override HVM guest firmwar
e. Ignore that. Use "firmware_override" instead if you really want a non-defau
lt firmware
Name                                        ID   Mem VCPUs      State   Time(s
)
Domain-0                                     0 29134     8     r-----      75.
6
vca_centos                                   1  3071     1     ------       0.
0
VM started
If you don't know an IP of DomU, because it is assigned by DHCP, install Avahi
 daemon and nss-mdns at host and connect to:
        vca_7ebf010edd91.local
[root@localhost 2.0.44]# ssh 10.54.56.27
Error reading response length from authentication socket.
root@10.54.56.27's password:
-bash-4.2# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0 29134     8     r-----      99.6
vca_centos                                   1  3071     6     -b----      18.3
-bash-4.2#
-bash-4.2#
-bash-4.2# xl network-list
'xl network-list' requires at least 1 argument.

Usage: xl [-v] network-list <Domain(s)>

List virtual network interfaces for a domain.

-bash-4.2# xl network-list 1
Idx BE Mac Addr.         handle state evt-ch   tx-/rx-ring-ref BE-path
0   0  7e:bf:01:0e:dd:91     0     4     -1    -1/-1          /local/domain/0/backend/vif/1/0
-bash-4.2#
```

Figure 27. Find DomU MAC address using `xl network-list` command

5. Find the DomU IP address (Figure 28).

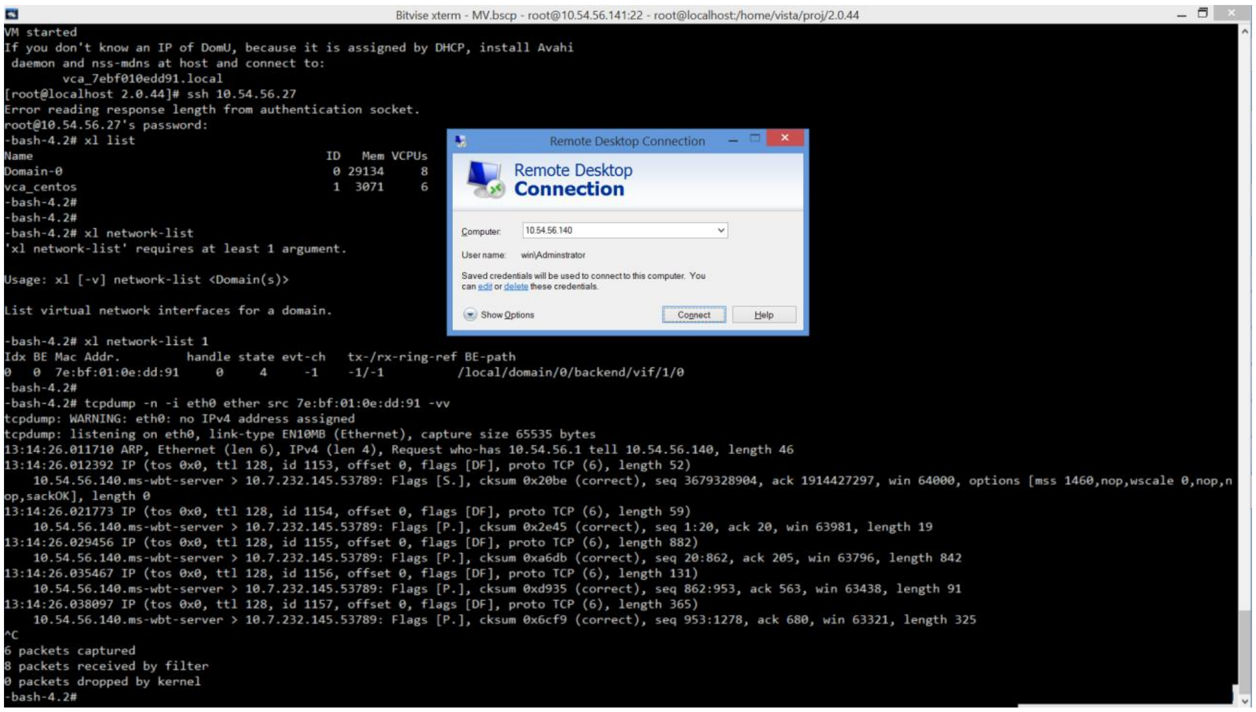
   ```
   tcpdump -n -i eth0 ether src <MAC address> -vv
   ```



```
Select Bitvise xterm - MV.bscp - root@10.54.56.141:22 - root@localhost:/home/vista/
VM started
If you don't know an IP of DomU, because it is assigned by DHCP, install Avahi
 daemon and nss-mdns at host and connect to:
       vca_7ebf010edd91.local
[root@localhost 2.0.44]# ssh 10.54.56.27
Error reading response length from authentication socket.
root@10.54.56.27's password:
-bash-4.2# xl list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0 29134     8     r-----      99.6
vca_centos                                   1  3071     6     -b----      18.3
-bash-4.2#
-bash-4.2#
-bash-4.2# xl network-list
'xl network-list' requires at least 1 argument.

Usage: xl [-v] network-list <Domain(s)>

List virtual network interfaces for a domain.

-bash-4.2# xl network-list 1
Idx BE Mac Addr.         handle state evt-ch   tx-/rx-ring-ref BE-path
0   0  7e:bf:01:0e:dd:91     0     4     -1    -1/-1          /local/domain/0/backend/vif/1/0
-bash-4.2#
-bash-4.2# tcpdump -n -i eth0 ether src 7e:bf:01:0e:dd:91 -vv
tcpdump: WARNING: eth0: no IPv4 address assigned
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
13:14:26.011710 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 10.54.56.1 tell 10.54.56.140  length 46
13:14:26.012392 IP (tos 0x0, ttl 128, id 1153, offset 0, flags [DF], proto TCP (6), length 52)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [S.], cksum 0x20be (correct), seq 3679328904, ack 19
op,sackOK], length 0
13:14:26.021773 IP (tos 0x0, ttl 128, id 1154, offset 0, flags [DF], proto TCP (6), length 59)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [P.], cksum 0x2e45 (correct), seq 1:20, ack 20, win
13:14:26.029456 IP (tos 0x0, ttl 128, id 1155, offset 0, flags [DF], proto TCP (6), length 882)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [P.], cksum 0xa6db (correct), seq 20:862, ack 205, w
13:14:26.035467 IP (tos 0x0, ttl 128, id 1156, offset 0, flags [DF], proto TCP (6), length 131)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [P.], cksum 0xd935 (correct), seq 862:953, ack 563,
13:14:26.038097 IP (tos 0x0, ttl 128, id 1157, offset 0, flags [DF], proto TCP (6), length 365)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [P.], cksum 0x6cf9 (correct), seq 953:1278, ack 680,
^C
6 packets captured
8 packets received by filter
0 packets dropped by kernel
-bash-4.2#
```

Figure 28. Find DomU IP address using `tcpdump` command

6. Start remote desktop connection. Enter the DomU IP address and click **Show options** (do not click **Connect**) (Figure 29).



Bitvise xterm - MV.bscp - root@10.54.56.141:22 - root@localhost:/home/vista/proj/2.0.44

```
VM started
If you don't know an IP of DomU, because it is assigned by DHCP, install Avahi
 daemon and nss-mdns at host and connect to:
        vca_7ebf010edd91.local
[root@localhost 2.0.44]# ssh 10.54.56.27
Error reading response length from authentication socket.
root@10.54.56.27's password:
-bash-4.2# xl list
Name                                        ID   Mem VCPUs
Domain-0                                     0 29134     8
vca_centos                                   1  3071     6
-bash-4.2#
-bash-4.2#
-bash-4.2# xl network-list
'xl network-list' requires at least 1 argument.

Usage: xl [-v] network-list <Domain(s)>

List virtual network interfaces for a domain.

-bash-4.2# xl network-list 1
Idx BE Mac Addr.         handle state evt-ch   tx-/rx-ring-ref BE-path
0   0  7e:bf:01:0e:dd:91     0     4     -1    -1/-1          /local/domain/0/backend/vif/1/0
-bash-4.2#
-bash-4.2# tcpdump -n -i eth0 ether src 7e:bf:01:0e:dd:91 -vv
tcpdump: WARNING: eth0: no IPv4 address assigned
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
13:14:26.011710 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 10.54.56.1 tell 10.54.56.140, length 46
13:14:26.012392 IP (tos 0x0, ttl 128, id 1153, offset 0, flags [DF], proto TCP (6), length 52)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [S.], cksum 0x20be (correct), seq 3679328904, ack 1914427297, win 64000, options [mss 1460,nop,wscale 0,nop,n
op,sackOK], length 0
13:14:26.021773 IP (tos 0x0, ttl 128, id 1154, offset 0, flags [DF], proto TCP (6), length 59)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [P.], cksum 0x2e45 (correct), seq 1:20, ack 20, win 63981, length 19
13:14:26.029456 IP (tos 0x0, ttl 128, id 1155, offset 0, flags [DF], proto TCP (6), length 882)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [P.], cksum 0xa6db (correct), seq 20:862, ack 205, win 63796, length 842
13:14:26.035467 IP (tos 0x0, ttl 128, id 1156, offset 0, flags [DF], proto TCP (6), length 131)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [P.], cksum 0xd935 (correct), seq 862:953, ack 563, win 63438, length 91
13:14:26.038097 IP (tos 0x0, ttl 128, id 1157, offset 0, flags [DF], proto TCP (6), length 365)
    10.54.56.140.ms-wbt-server > 10.7.232.145.53789: Flags [P.], cksum 0x6cf9 (correct), seq 953:1278, ack 680, win 63321, length 325
^C
6 packets captured
8 packets received by filter
0 packets dropped by kernel
-bash-4.2#
```

Remote Desktop Connection

Remote Desktop Connection

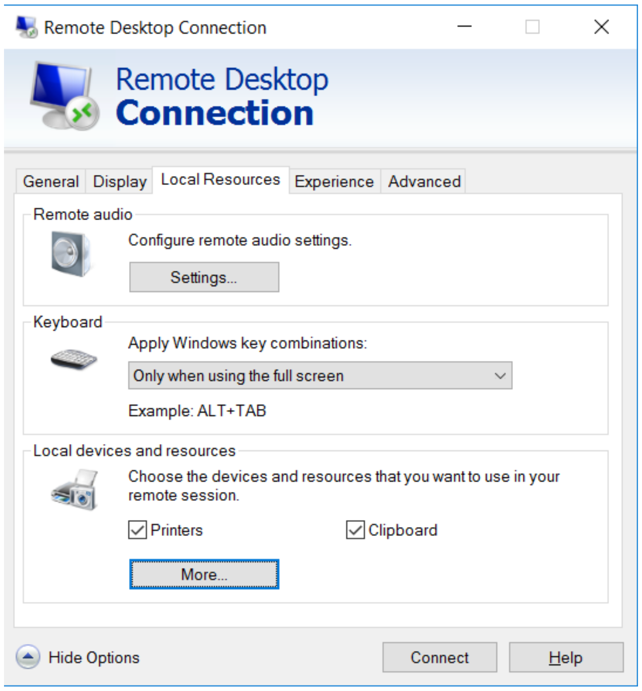
Computer: 10.54.56.140

User name: win\Adminstrator

Saved credentials will be used to connect to this computer. You can edit or delete these credentials.

Figure 29. Start remote desktop connection

7. Go to the **Local Resources** tab and click the **More** button under **Local devices and resources** (Figure 30).



Remote Desktop Connection

Remote Desktop Connection

Remote audio

Configure remote audio settings.

Keyboard

Apply Windows key combinations:

Only when using the full screen

Example: ALT+TAB

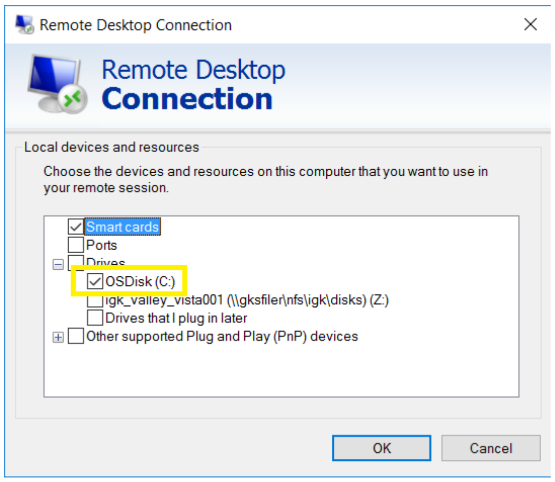
Local devices and resources

Choose the devices and resources that you want to use in your remote session.

☑ Printers ☑ Clipboard

Figure 30. Remote desktop connection local resources options

8. In the newly opened window, check the drive containing the unpacked PV drivers (`xenbus`, `xenvif`, and `xennet`) (Figure 31). Click **OK** to close the window and then **Connect** to start the connection.

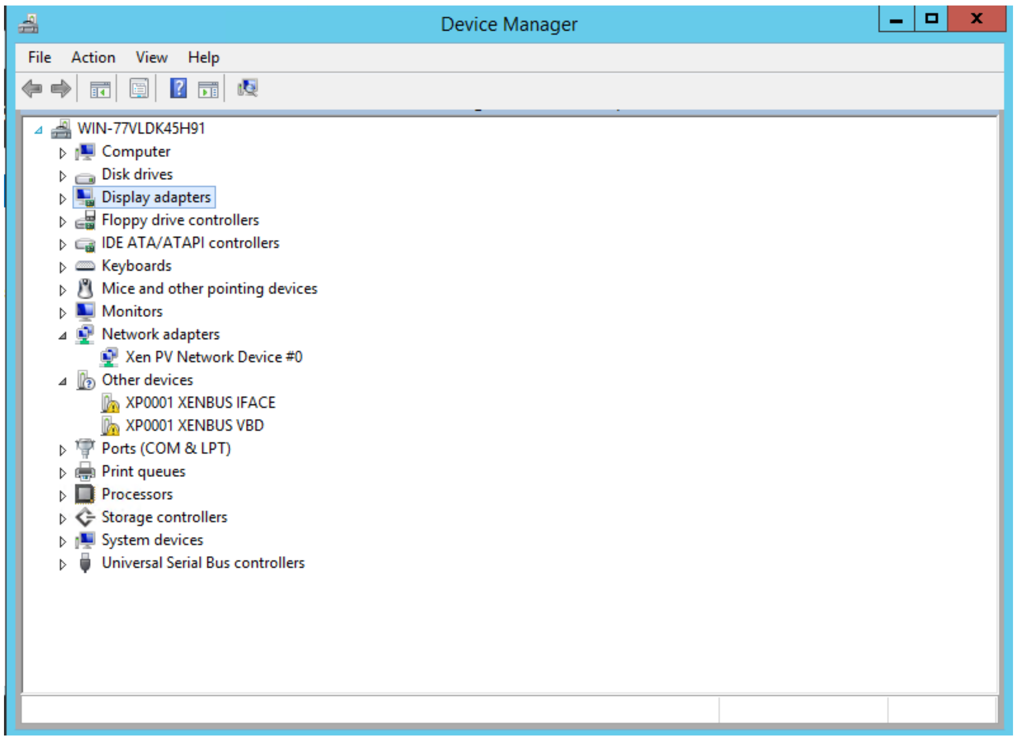

Remote Desktop Connection
Remote Desktop Connection
Local devices and resources

Choose the devices and resources on this computer that you want to use in your remote session.

- [✓] Smart cards
- [ ] Ports
- [ ] Drives
  - [✓] OSDisk (C:)
  - [ ] igk_valley_vista001 (\\gksfiler\nfs\igk\disks) (Z:)
  - [ ] Drives that I plug in later
- [ ] Other supported Plug and Play (PnP) devices

Figure 31. Remote desktop connection drive selection

9. Log in to the Windows guest. In file explorer, locate the drive and directory containing the unpacked PV drivers (`xenbus`, `xenvif`, and `xennet`). For example, the directory for `xenbus` is `C:\Downloads\xenbus\x64`.
10. Double-click `dpinst` to install all three PV drivers and reboot Windows guest.
11. After rebooting, open Device Manager and verify that Xen PV Network Device is used instead of e1000. Ignore missing drivers for XENBUS IFACE and XENBUS VBD.



```
WIN-77VLDK45H91
  Computer
  Disk drives
  Display adapters
  Floppy drive controllers
  IDE ATA/ATAPI controllers
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
    Xen PV Network Device #0
  Other devices
    XP0001 XENBUS IFACE
    XP0001 XENBUS VBD
  Ports (COM & LPT)
  Print queues
  Processors
  Storage controllers
  System devices
  Universal Serial Bus controllers
```

Figure 32. Xen PV network adapter installed

12. Shut down DomU and start it again (from the E5 host) with enabled graphics passthrough and without emulated network adapter (i.e., do not use the `--legacy-net` option).

```
vca_start_card_domu.sh -c <card ID> -n <node ID> -p <path to image at E5> -b
   --card-mac -i <guest_image_file> -f card_gfx_vm.hvm -g
```

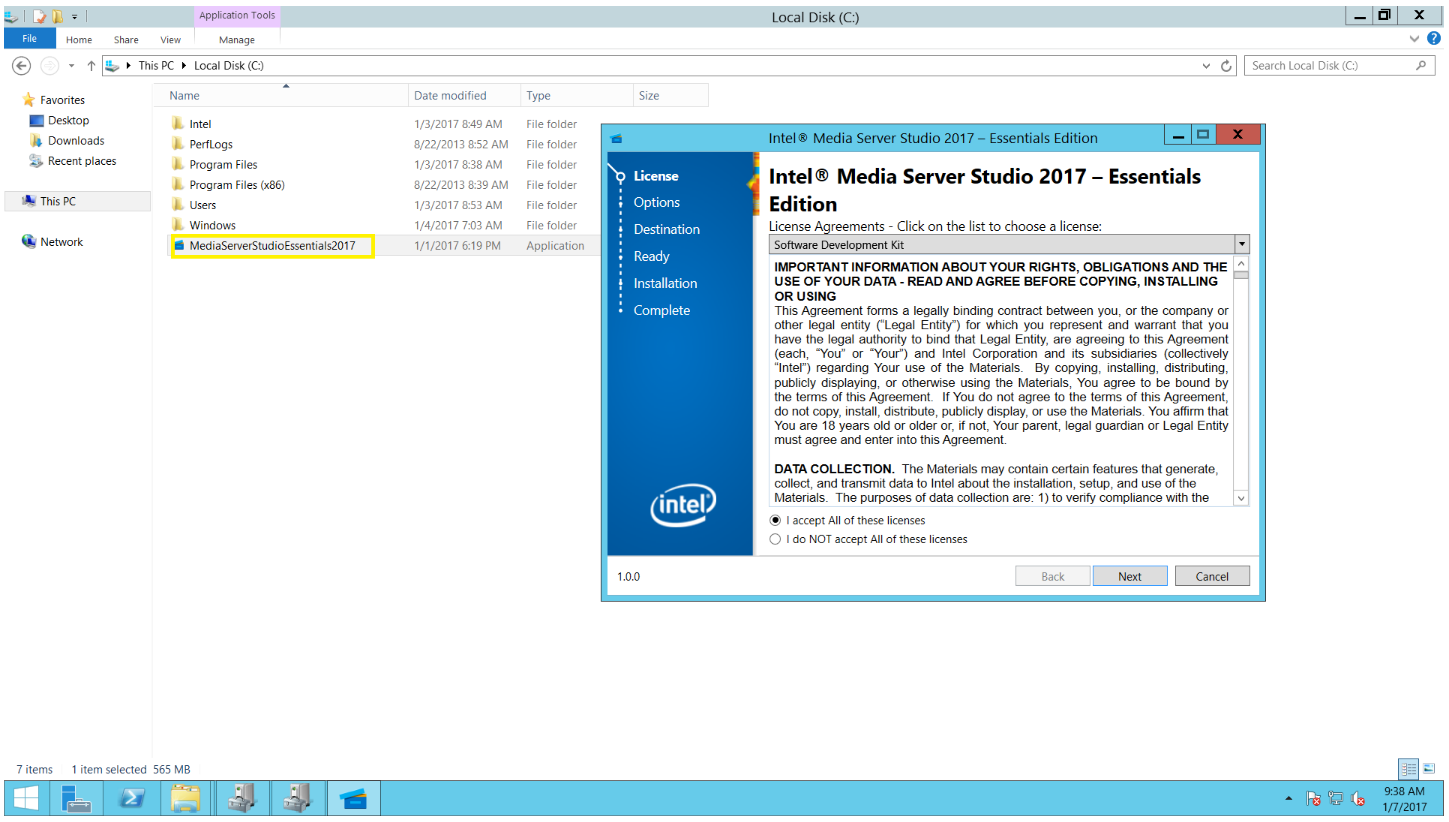
For example,

```
vca_start_card_domu.sh -c 0 -n 0 -p /mnt/xen -b --card-mac -i win2k12.img -f
   card_gfx_vm.hvm -g
```

4.5.8.4 Install Display Drivers

Perform the following steps to replace the Microsoft* basic display driver with the Intel display driver bundled with Intel Media Server Studio.

1. Connect to the Windows guest using a remote desktop connection.
2. Install the Intel display driver by running `MediaServerStudioEssentials2017.exe` (Figure 33).



Name	Date modified	Type	Size
Intel	1/3/2017 8:49 AM	File folder	
PerfLogs	8/22/2013 8:52 AM	File folder	
Program Files	1/3/2017 8:38 AM	File folder	
Program Files (x86)	8/22/2013 8:39 AM	File folder	
Users	1/3/2017 8:53 AM	File folder	
Windows	1/4/2017 7:03 AM	File folder	
MediaServerStudioEssentials2017	1/1/2017 6:19 PM	Application	

7 items   1 item selected 565 MB

Intel® Media Server Studio 2017 – Essentials Edition

Intel® Media Server Studio 2017 – Essentials Edition

License Agreements - Click on the list to choose a license:

Software Development Kit

IMPORTANT INFORMATION ABOUT YOUR RIGHTS, OBLIGATIONS AND THE USE OF YOUR DATA - READ AND AGREE BEFORE COPYING, INSTALLING OR USING

This Agreement forms a legally binding contract between you, or the company or other legal entity ("Legal Entity") for which you represent and warrant that you have the legal authority to bind that Legal Entity, are agreeing to this Agreement (each, "You" or "Your") and Intel Corporation and its subsidiaries (collectively "Intel") regarding Your use of the Materials. By copying, installing, distributing, publicly displaying, or otherwise using the Materials, You agree to be bound by the terms of this Agreement. If You do not agree to the terms of this Agreement, do not copy, install, distribute, publicly display, or use the Materials. You affirm that You are 18 years old or older or, if not, Your parent, legal guardian or Legal Entity must agree and enter into this Agreement.

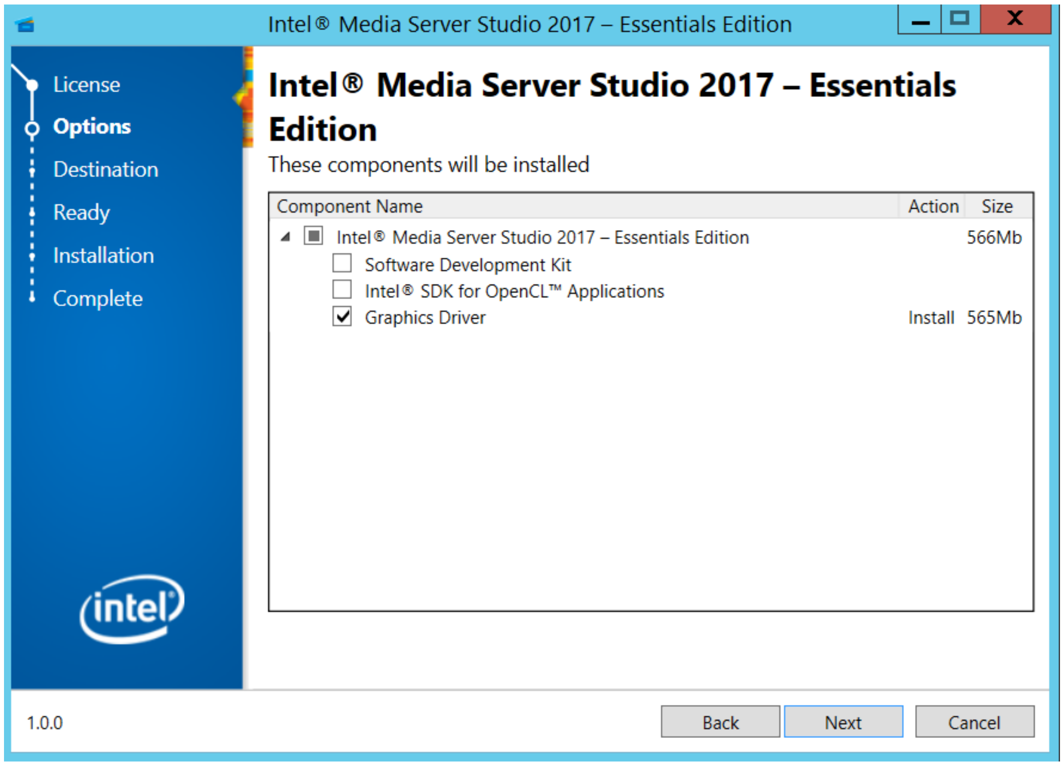
DATA COLLECTION. The Materials may contain certain features that generate, collect, and transmit data to Intel about the installation, setup, and use of the Materials. The purposes of data collection are: 1) to verify compliance with the

[✓ I accept All of these licenses] I do NOT accept All of these licenses

1.0.0

Figure 33. Install Intel® Media Server Studio 2017

3. Select **Graphics Driver** and click **Next** (Figure 34). Click **Next** again to start installation (Figure 35).

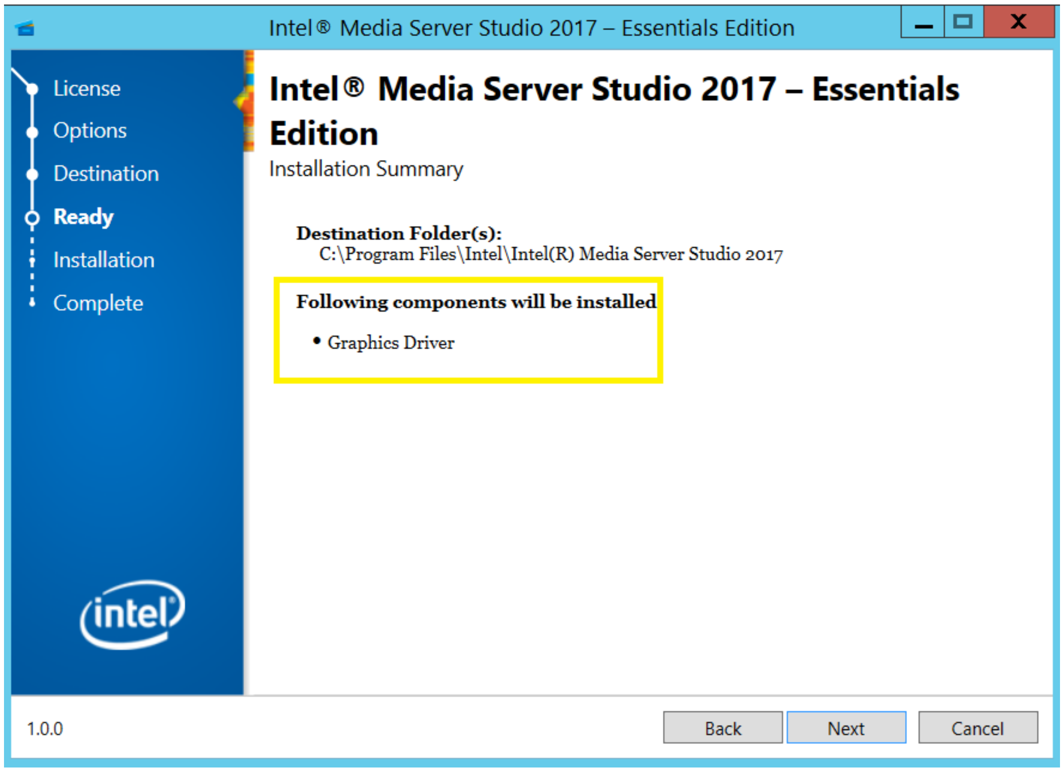


Intel® Media Server Studio 2017 – Essentials Edition

These components will be installed

Component Name	Action	Size
Intel® Media Server Studio 2017 – Essentials Edition		566Mb
☐ Software Development Kit		
☐ Intel® SDK for OpenCL™ Applications		
☑ Graphics Driver	Install	565Mb

Figure 34. Select Intel® graphics driver



Intel® Media Server Studio 2017 – Essentials Edition

Installation Summary

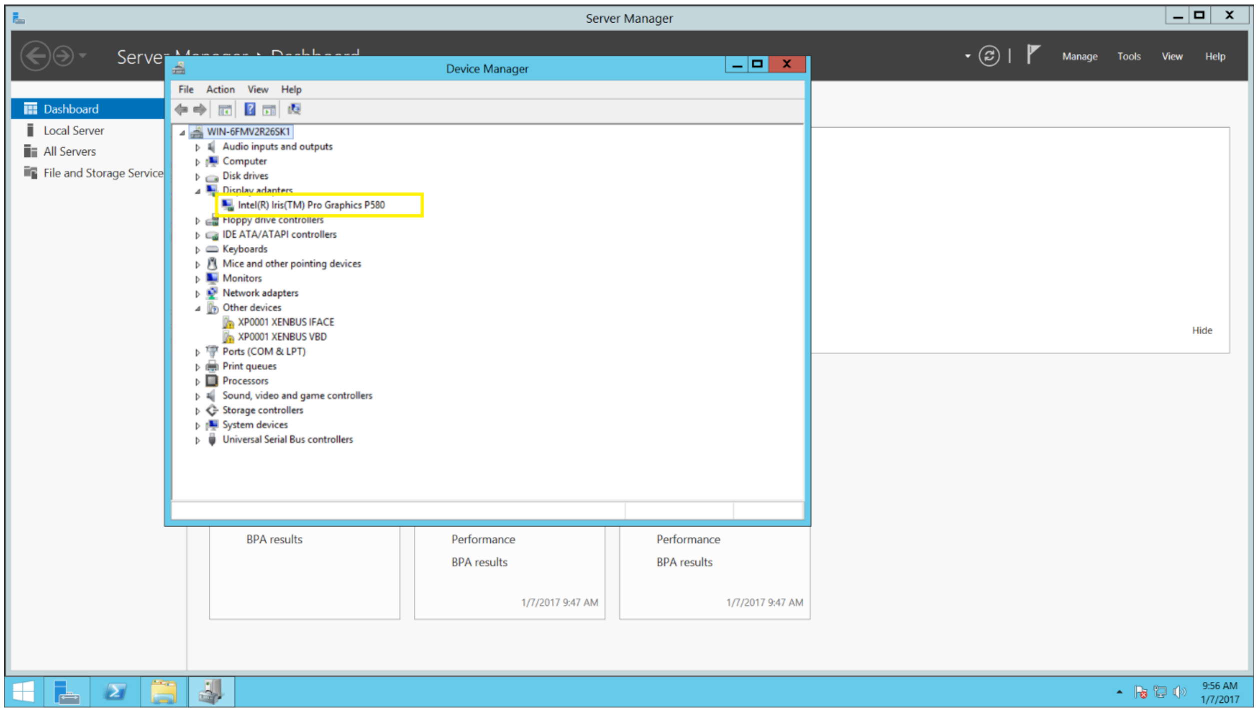
Destination Folder(s):
C:\Program Files\Intel\Intel(R) Media Server Studio 2017

Following components will be installed

- Graphics Driver

Figure 35. Install Intel® graphics driver

If installation is successful, Iris Pro graphics is shown in the Device Manager under Display adapters (Figure 36).



```
WIN-6FMV2R26SK1
    Audio inputs and outputs
    Computer
    Disk drives
    Display adapters
        Intel(R) Iris(TM) Pro Graphics P580
    Floppy drive controllers
    IDE ATA/ATAPI controllers
    Keyboards
    Mice and other pointing devices
    Monitors
    Network adapters
    Other devices
        XP0001 XENBUS IFACE
        XP0001 XENBUS VBD
    Ports (COM & LPT)
    Print queues
    Processors
    Sound, video and game controllers
    Storage controllers
    System devices
    Universal Serial Bus controllers
```

Figure 36. Iris® Pro graphics installed

4.5.8.5 Start DomU for Linux* OS

To start DomU with graphic passthrough enabled on node 0 and card 0:

```
vca_start_card_domu.sh -p <NFS share> -g -f card_gvx_vm.hvm -c 0 -n 0 -i
  vca_domu_1.img
```

To start DomU without graphic passthrough enabled:

```
vca_start_card_domu.sh -p <NFS share> -f card_vm.hvm -c 0 -n 0 -i
  vca_domu_1.img
```

Change `-c`, `-n`, and `-i` parameters for different nodes. The `-c` parameter is the card to start the VM on starting with 0, the `-n` parameter is the CPU within that card to start the VM on starting with 0, and the `-i` parameter is the name of the image used to boot.

For a set of options for the `vca_start` command, see Table 11. If the DomU IP address is not changed by the parameter below, the default IP is `172.31.X.2` where X is the node starting with 1.

Table 11. Options for `vca_start` command

Option	Operand	Description
-f	filename	VM configuration file. Must be located within NFS shared directory
-g		Enable graphics passthrough
-p	Path	NFS share location (path at host); using /share if not configured
-a	IP	Card's IP address (if other than VCA xml defined is used)
-h	IP	Host's IP address (if other than VCA xml defined is used)
-u	IP	DomU IP address (if other than default is used); this is for routing purposes only
-N	IP	NFS address (if different than host)
-c	ID	Card ID (0, 1, 2, 3, etc.); using 0 if not provided
-n	ID	Node ID (CPU at card; 0, 1, 2); using 0 if not provided; used to generate DomU MAC address (if not provided with -m option) and host's address (if not provided by -a/-h)
-i	filename	VM image filename. Overrides image configured in VCA xml file
-b		Use network bridging instead of routing
-k	filename	Kernel image filename. For booting VMs without bootloader (doesn't support graphics pass-through)
-r	filename	InitRamFs image name. For booting VMs without bootloader (doesn't support graphics pass-through)

4.5.8.6 Add Routing on Dom0 on Each Node

Note: This should not be required for configurations using the network bridging feature of software release 1.2 or later.

To add routing on each node, connect to node:

```
ssh 172.31.x.1
```

where X is the number of the node starting with 1 (i.e., `172.31.1.1` for first node, `172.31.2.1` for second node, etc.)
Add a route to get to the outside network through the host by executing the command:

```
route add 172.31.1.253 gw 172.31.1.254
```

4.5.9 Using Block Devices (via BlockIO)

Block devices can be in four states:

- **Disable** – There is nothing open.
- **Enable** – Something is open but not used yet.
- **Inactive** – Something is open but not working.
- **Active** – Something is open and working.

Devices named `vcablk0` are dedicated to operating system booting. Devices named `vcablk1` through `vcablk7` can be prepared to use as additional drives except for operating system booting.

To prepare devices `vcablk1` through `vcablk7` for Windows OS, perform the following steps:

1. Find and open the drive.

   ```
   vcactl blockio open <card_ID> <node_ID> vcablkN RW|RO <drive_path>
   ```

2. Boot the system on the same node on `vcablk0`.

   ```
   vcactl blockio open <card_ID> <node_ID> vcablk0 RW <image_path>
   vcactl boot <card_ID> <node_ID> vcablk0
   ```

3. Log in to the system (e.g., via Remote Desktop Connection).

4. Right-click the Start icon and choose **Disk Management** (Figure 37).

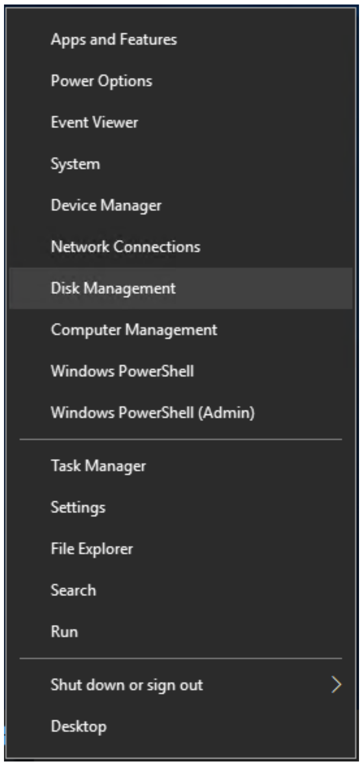
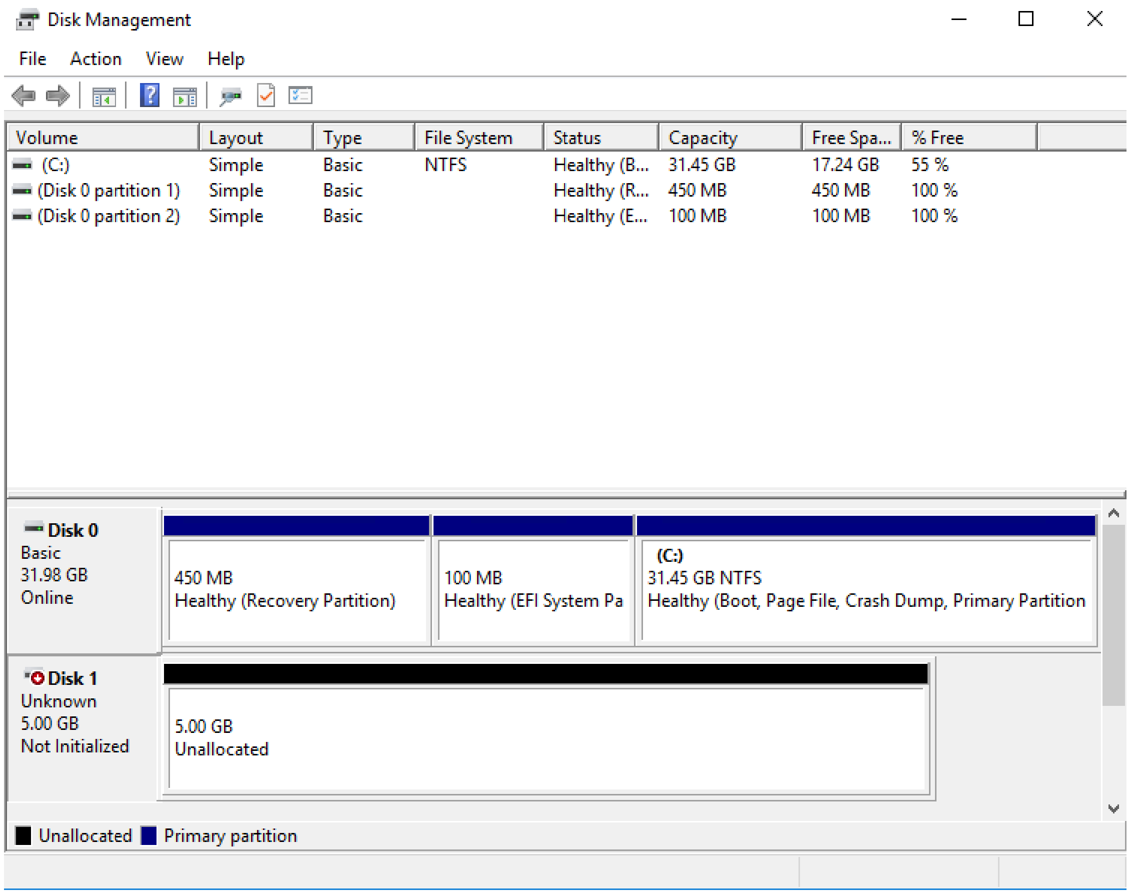


Figure 37. Windows* disk management option

5. In the Disk Management window, the disk should be visible as Unknown and Unallocated as shown in Figure 38 (Disk 1). Right-click the Disk 1 label and select **Initialize Disk**.



Volume	Layout	Type	File System	Status	Capacity	Free Spa...	% Free
(C:)	Simple	Basic	NTFS	Healthy (B...	31.45 GB	17.24 GB	55 %
(Disk 0 partition 1)	Simple	Basic		Healthy (R...	450 MB	450 MB	100 %
(Disk 0 partition 2)	Simple	Basic		Healthy (E...	100 MB	100 MB	100 %

Disk 0
Basic
31.98 GB
Online

450 MB
Healthy (Recovery Partition)

100 MB
Healthy (EFI System Pa

(C:)
31.45 GB NTFS
Healthy (Boot, Page File, Crash Dump, Primary Partition

Disk 1
Unknown
5.00 GB
Not Initialized

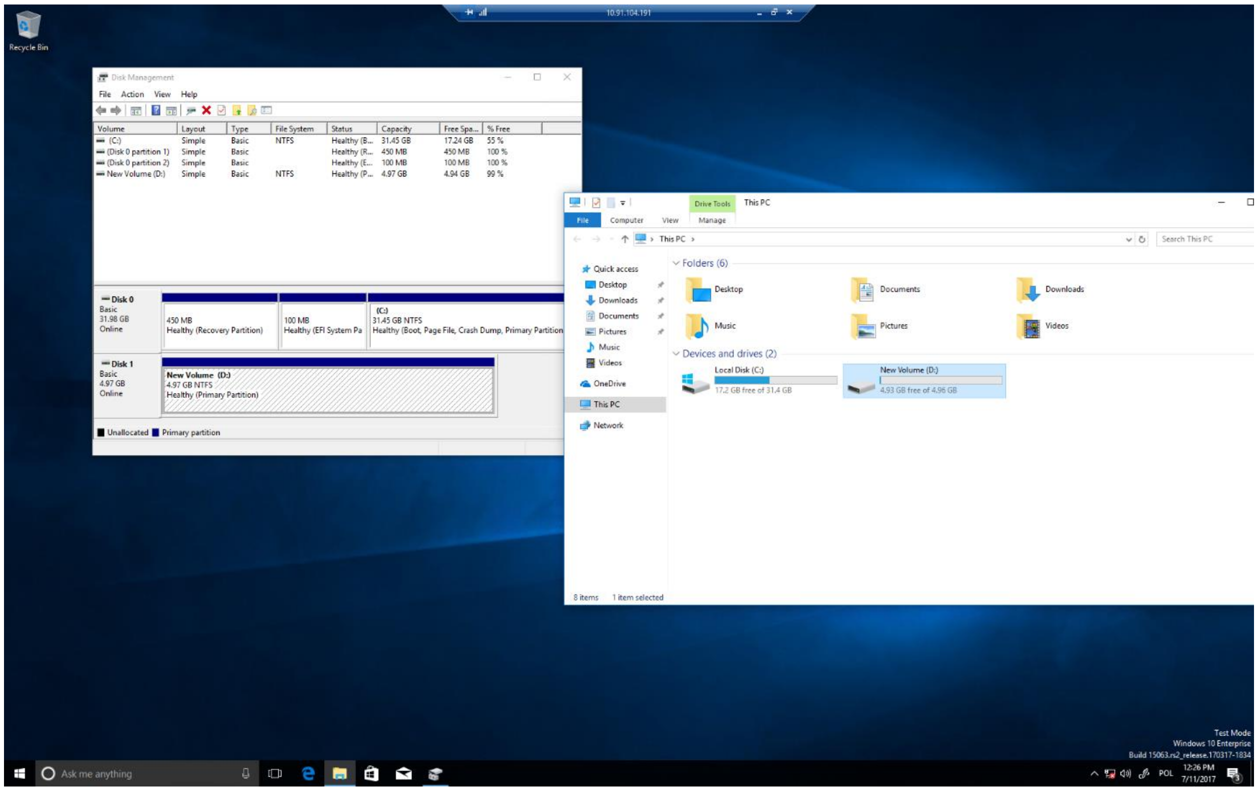
5.00 GB
Unallocated

Figure 38. Disk management window

6. Choose partition style and click **OK**

7. Right-click on the Disk 1 label, select **New Simple Volume**, and follow the on-screen instructions.

8. After successful configuration, the disk should be visible as a normal partition as shown in Figure 39.



Volume	Layout	Type	File System	Status	Capacity	Free Spa...	% Free
(C:)	Simple	Basic	NTFS	Healthy (B...	31.45 GB	17.24 GB	55 %
(Disk 0 partition 1)	Simple	Basic		Healthy (R...	450 MB	450 MB	100 %
(Disk 0 partition 2)	Simple	Basic		Healthy (E...	100 MB	100 MB	100 %
New Volume (D:)	Simple	Basic	NTFS	Healthy (P...	4.97 GB	4.94 GB	99 %

Disk Management
File Action View Help
Disk 0 Basic 31.98 GB Online
450 MB Healthy (Recovery Partition)
100 MB Healthy (EFI System Pa
(C:) 31.45 GB NTFS Healthy (Boot, Page File, Crash Dump, Primary Partition
Disk 1 Basic 4.97 GB Online
New Volume (D:) 4.97 GB NTFS Healthy (Primary Partition)
Unallocated Primary partition
This PC
Folders (6)
Desktop
Documents
Downloads
Music
Pictures
Videos
Devices and drives (2)
Local Disk (C:) 17.2 GB free of 31.4 GB
New Volume (D:) 4.93 GB free of 4.96 GB
8 items 1 item selected
Test Mode
Windows 10 Enterprise
Build 15063.rs2_release.170317-1834
12:26 PM
7/11/2017

Figure 39. Successful drive configuration in Windows*

To prepare devices `vcablk1` through `vcablk7` for Linux CentOS*, perform the following steps:

1. Find the drive.
2. Swap the drive into proper file system using the `mkfs.<file_system> <drive_path>` command.
3. Open the drive.

```
vcactl blockio open <card_ID> <node_ID> vcablkN RW|RO <drive_path>
```

4. Boot the system on the same node on `vcablk0`.

```
vcactl blockio open <card_ID> <node_ID> vcablk0 RW <image_path>
vcactl boot <card_ID> <node_ID> vcablk0
```

5. Log in to the system via SSH.
6. Mount the drive.

```
mount /dev/vcablkN /mnt/
```

5. Using the Intel® VCA Product Family Software

5.1 Using Intel® Media Server Studio

The main software framework for Intel® VCA is Intel® Media Server Studio. Intel® VCA boot reference images in both the volatile and persistent reference packages have pre-installed Intel Media Server Studio 2018 R1 (Community Edition). On Intel® VCA boot production images in both the volatile and persistent packages, Intel® Media Server Studio needs to be installed. To develop applications for video processing, use Intel® Quick Sync Video. OpenCL* is also supported.

To use Intel Media Server Studio on an Intel® VCA card, simply log in Intel® VCA (by SSH or remote desktop/VNC). Then develop or run Intel Media Server Studio programs like on other Intel platforms with Intel Media Server Studio.

More information on Intel Media Server Studio 2018 R1 can be found at https://software.intel.com/en-us/intel-media-server-studio.

Note: Intel® VCA releases may not have the latest Intel Media Server Studio release. If a newer version is available, follow the instructions with the release to perform the upgrade.

Intel® VCA strategy is to test the Intel® VCA software stack against the most recent, publicly released Intel Media Server Studio to provide the reference Intel® VCA software stack. But some customers may want to keep older Intel Media Server Studio releases, which have been well tested in their environment, or may have access to non-public (e.g. obtained under NDA) Intel Media Server Studio releases. To cover such needs, the production VCA software stack does not include Intel Media Server Studio.

Intel Media Server Studio contains two main parts (in terms of driver form):

- An open source Kernel Mode Drivers (KMD).
- A binary User Mode Drivers (UMD) and libraries, to be installed only.

To use Intel Media Server Studio on a production image, first start Intel Media Server Studio at the Intel® VCA node. Perform the following steps after each reset of an Intel® VCA node, unless a persistent filesystem image is in use.

1. Boot a production image. Change the directory to where the Intel® VCA files are placed.

   ```
   sudo vcactl boot vca_baremetal_production_<build version>.img
   ```

2. Copy Intel Media Server Studio components to the Intel® VCA node. Change the directory to `SDK<year>Production<version>` inside the Intel Media Server Studio location. This folder should contain a collection of RPMs and other files. The example below assumes copying files to `/root/` but any folder may be used. Repeat for each node.

   ```
   scp -r CentOS root@<node's ip>:/root/
   ```

3. If devel packages from Intel Media Server Studio need to be installed (to compile other components to work with Intel Media Server Studio, for example, external h265 codec), also copy the kernel-headers package from the build, after changing directory to the location of the kernel-headers package.

   ```
   scp kernel-headers-3.10.0_<kernel version>.x86_64.rpm \ root@<node's ip>:~
   ```

   Change directory to where Intel® VCA software packages are installed. Copy Intel Media Server Studio kernel mode driver to the Intel® VCA node. Repeat for each node.

   ```
   scp -r kmod-ukmd-<version>.rpm root@<node's ip>:/root/
   ```

4. Install Intel Media Server Studio at the Intel® VCA node. Log in to each Intel® VCA node using SSH.

   ```
   ssh root@<Node IP Address>
   ```

Note: Before running commands on the nodes, including `yum`, make sure that the nodes are able to access a valid `yum` repository. An example of setting up a local `yum` repository on the host and enabling the clients to access it may be found in Section B.4.

   To install kernel mode driver, change to directory where `kmod-ukmd-<version>.rpm` is located.

   ```
   rpm –ivh kmod-ukmd-<version>.rpm
   ```

   Install Intel Media Server Studio (only executables, without devel packages). Change directory to where Intel Media Server Studio RPMs were placed above.

   ```
   ls *.rpm | grep -iv devel |grep libdrm|xargs rpm --force -ivh
   ls *rpm | grep -iv samples | grep -iv devel | grep -iv kmod | xargs yum -y
      localinstall
   ```

   Or install Intel Media Server Studio with devel packages.

   ```
   ls *rpm | grep -iv samples | xargs yum -y localinstall
   ```

Note: This requires additional packages not part of the CentOS* core distribution, which must be acquired and installed separately.

To properly set environmental variables, log out from the session and log in again using SSH.

5.2 Running a Sample Transcode

To run a sample transcode, Intel Media Server Studio must be installed.

1. Ensure the file `MediaSamples_Linux_6.0.16043175.175.tar.bz2` is on the host system, and `cd` to the directory containing this file.
2. Unpack the samples tarball with the following command:

   ```
   tar jxf MediaSamples_Linux_6.0.16043175.175.tar.bz2
   ```

3. Verify the nodes are fully operational before continuing (see previous sections of the document).
4. Copy the unpacked samples files to a node. To test on more than one node repeat this command for each subsequent node (e.g. 172.31.2.1, 172.31.3.1, etc.).

   ```
   scp –r MediaSamples_Linux_6.0.16043175.175 \ root@172.31.1.1:/root/
   ```

5. Connect to the node (this, and the following two steps, should be run together if testing on nodes beyond the first; as before, change the third octet as in the previous step).

   ```
   ssh root@172.31.1.1
   ```

6. Change into the correct directory.

   ```
   cd MediaSamples_Linux_6.0.16043175.175/samples/_bin
   ```

7. Run the sample transcode.

   ```
   ./x64/sample_multi_transcode_drm –i::h264 content/test_stream.264 –o::h264
      out.h264
   ```

8. Transcode should run without error and should take less than 0.1 seconds to complete. Verification of successful transcode can be achieved by comparing the output of the command

   ```
   md5sum out.h264
   ```

   with the output of the command

   ```
   24be5fc13bab5bbbc9417842cbdc0e90  out.h264
   ```

6. Intel® VCA Utility (`vcactl`)

6.1 Introduction to `vcactl`

The `vcactl` utility allows the user to interact with the Intel® VCA card and nodes. To execute the utility, the user must be either a member of the special group `vcausers`, which is created during utility installation; a member of the `sudoers` group (that is, they must have sudo privileges on the system); or a user with root privileges. In the first and last cases (a member of `vcausers` or a root user), no extra commands must be typed. In the second case (a member of the `sudoers` group), all commands must be prepended by `sudo` (`sudo` followed by a space character).

To add a user to the `vcausers` group, use the following command as either root or a root-privileged user (e.g., a `sudoers` user using `sudo`):

```
usermod -aG vcausers <username>
```

Note : To add a username to `vcausers`, a relogin or reboot is required.

6.2 `vcactl` Usage

Use the following syntax and structure to use the `vcactl` utility.

```
vcactl [execution_mode] command [subcommand] [card_id [cpu_id]]
   [command_params]
```

The available execution modes are:

```
-v : log verbose level set to DEBUG
-vv : log verbose level set to FULL-INFO
--skip-modprobe-check : turn off checking whether vca device is ready
--force : force command execution (WARNING: you do it at your own risk!)
```

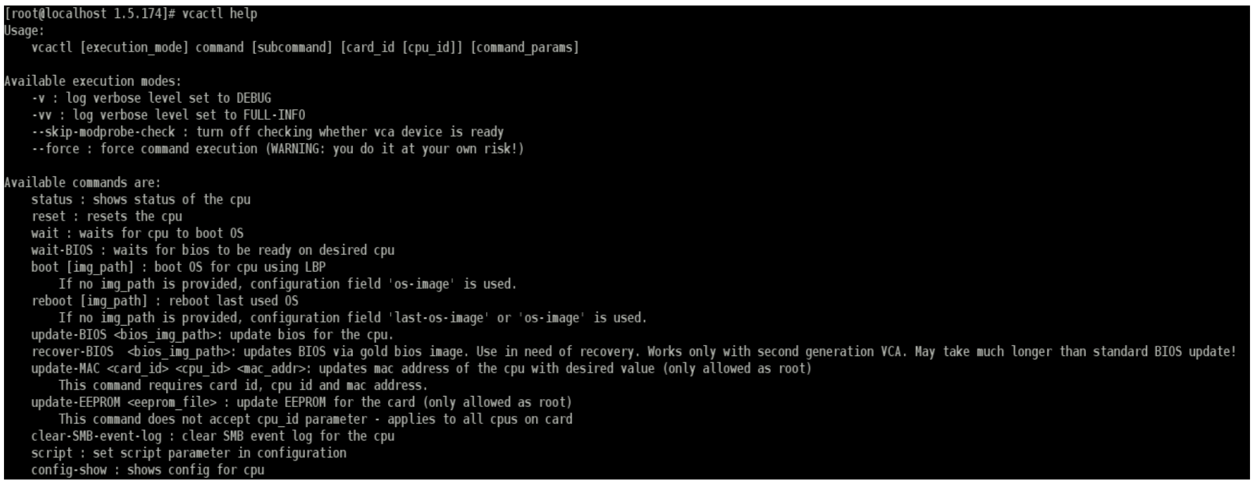
6.3 Command Line Reference

6.3.1 `vcactl help`

Usage

```
vcactl help
```

Description
Prints out help information about how to use `vcactl`.



```
[root@localhost 1.5.174]# vcactl help
Usage:
    vcactl [execution_mode] command [subcommand] [card_id [cpu_id]] [command_params]

Available execution modes:
    -v : log verbose level set to DEBUG
    -vv : log verbose level set to FULL-INFO
    --skip-modprobe-check : turn off checking whether vca device is ready
    --force : force command execution (WARNING: you do it at your own risk!)

Available commands are:
    status : shows status of the cpu
    reset : resets the cpu
    wait : waits for cpu to boot OS
    wait-BIOS : waits for bios to be ready on desired cpu
    boot [img_path] : boot OS for cpu using LBP
        If no img_path is provided, configuration field 'os-image' is used.
    reboot [img_path] : reboot last used OS
        If no img_path is provided, configuration field 'last-os-image' or 'os-image' is used.
    update-BIOS <bios_img_path>: update bios for the cpu.
    recover-BIOS  <bios_img_path>: updates BIOS via gold bios image. Use in need of recovery. Works only with second generation VCA. May take much longer than standard BIOS update!
    update-MAC <card_id> <cpu_id> <mac_addr>: updates mac address of the cpu with desired value (only allowed as root)
        This command requires card id, cpu id and mac address.
    update-EEPROM <eeprom_file> : update EEPROM for the card (only allowed as root)
        This command does not accept cpu_id parameter - applies to all cpus on card
    clear-SMB-event-log : clear SMB event log for the cpu
    script : set script parameter in configuration
    config-show : shows config for cpu
```

Figure 40. `vcactl help` example

6.3.2 `vcactl wait-BIOS`

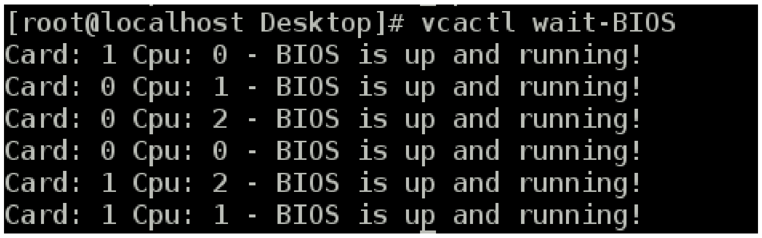
Usage

```
vcactl wait-BIOS [<card_id> <cpu_id>]
```

Description

Waits for CPU `<card_id> <cpu_id>` to enter `BIOS is up and running` mode, typically used after host reboot or `vcactl reset` command.

Reports an error if CPU cannot enter `BIOS is up and running` mode.



```
[root@localhost Desktop]# vcactl wait-BIOS
Card: 1 Cpu: 0 - BIOS is up and running!
Card: 0 Cpu: 1 - BIOS is up and running!
Card: 0 Cpu: 2 - BIOS is up and running!
Card: 0 Cpu: 0 - BIOS is up and running!
Card: 1 Cpu: 2 - BIOS is up and running!
Card: 1 Cpu: 1 - BIOS is up and running!
```

Figure 41. `vcactl wait-BIOS` example

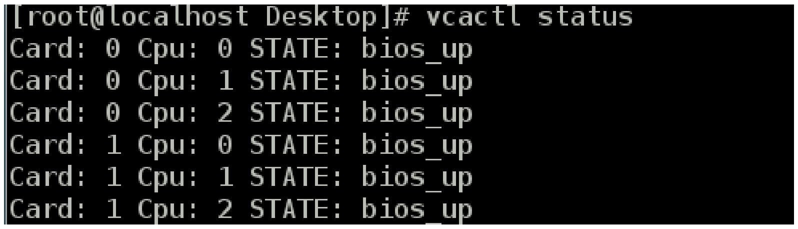
6.3.3 `vcactl status`

Usage

```
vcactl status [<card_id> <cpu_id>]
```

Description

Reads the status of the CPUs in the system. Normally this command should return `bios_up`. Use `<card id>` to read all CPUs on the specified card. Use `<card id> <cpu id>` to read the status of the specified CPU.



```
[root@localhost Desktop]# vcactl status
Card: 0 Cpu: 0 STATE: bios_up
Card: 0 Cpu: 1 STATE: bios_up
Card: 0 Cpu: 2 STATE: bios_up
Card: 1 Cpu: 0 STATE: bios_up
Card: 1 Cpu: 1 STATE: bios_up
Card: 1 Cpu: 2 STATE: bios_up
```

Figure 42. `vcactl status` example

Result

Result	Description
`link_down`	if card CPU is down due to any reason if CPU link to PLX is down (usually means that PLX eprom is invalid)(for VCA1283LVV)
`power_off`	if card CPU is switched off or shutdown
`powering_down`	If card CPU is shutting down
`bios_down`	CPU BIOS failed upon initialization or still booting (usually when memory training fails such state is reported).
`bios_up`	Link is up and BIOS is ready for handshake.
`bios_ready`	Handshake has succeeded with BIOS.
`bios_done`	BIOS is in a temporary state after allocating resources for completing the operation.
`flashing`	BIOS is being updated.
`booting`	OS is being booted.
`drv_probe_done`	Main PCIe* driver has been loaded.
`drv_probe_error`	Main PCIe* driver couldn't be loaded.
`dhcp_in_progress`	Card is obtaining IP address from DHCP.
`dhcp_done`	Card obtained IP address from DHCP.
`dhcp_error`	Card could not obtain IP address from DHCP.
`nfs_mount_done`	Card mounted NFS resources.
`nfs_mount_error`	Card could not mount NFS resources.
`os_ready`	OS boot completed.
`net_device_up`	Intel® VCA network stack is up but OS is not yet loaded.
`net_device_down`	Intel® VCA network stack has been switched off.
`net_device_ready`	Intel® VCA network stack is up, OS is booted, and IP address is assigned.
`net_device_no_ip`	Intel® VCA network stack is up and OS is booted, but IP address is not assigned (for example, DHCP response has been not received).
`resetting`	Intel® VCA CPU is being reset.
`error`	Error occurred different than the specific errors listed above.

For the Intel® VCA card (1283LVV), if the jumper on the card is set for Golden BIOS, `vcactl status` shows this information (only in `bios_up` state). For User BIOS, no additional information is displayed.

6.3.4 `vcactl update-BIOS`

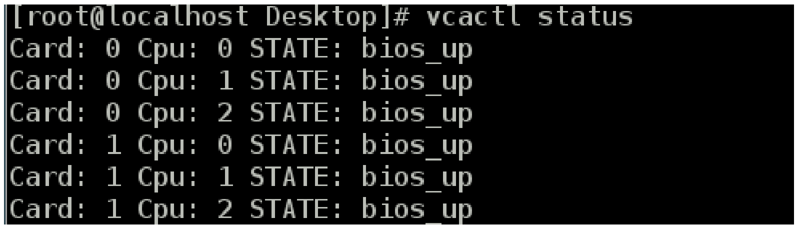
Usage

```
vcactl update-BIOS [<card_id> <cpu_id>] <biosImageFile>
```

Description

Updates BIOS (flash SPI user BIOS partition) on specified CPU(s) in the system. Before running this command, CPUs must be in the `bios_up` or `bios_ready` state. (Use `vcactl reset` and `vcactl wait-BIOS` to force CPUs to enter the `bios_up` state.) Host reboot is not required after `vcactl update-BIOS`.

G3 (mechanical off) is required after `vcactl update-BIOS`. (To perform G3, execute the power off command on the host and then disconnect from the power supply for 15 seconds.)



```
[root@localhost Desktop]# vcactl status
Card: 0 Cpu: 0 STATE: bios_up
Card: 0 Cpu: 1 STATE: bios_up
Card: 0 Cpu: 2 STATE: bios_up
Card: 1 Cpu: 0 STATE: bios_up
Card: 1 Cpu: 1 STATE: bios_up
Card: 1 Cpu: 2 STATE: bios_up
```

Figure 43. `vcactl update-BIOS` example

Result
The CPU status transitions from `bios_up` or `bios_ready` to `flashing` and completes in the `bios_ready` state. A temporary state of `bios_done` may be reported, as well.

For Intel® VCA 1283LVV only, if the jumper on the card is set for Golden BIOS, `vcactl update_BIOS` shows a warning that the next boot will be done on Golden BIOS and not the updated user BIOS.

```
Card: 0 Cpu: 0 - Warning: Jumper selects GOLD BIOS. Next reboot will boot
   it.
```

If there is a failure, an error status is reported (via the `vcactl status` command).

6.3.5 `vcactl recover-BIOS` (Intel® VCA 2 only)

Usage

```
vcactl recover-BIOS <biosImageFile>
```

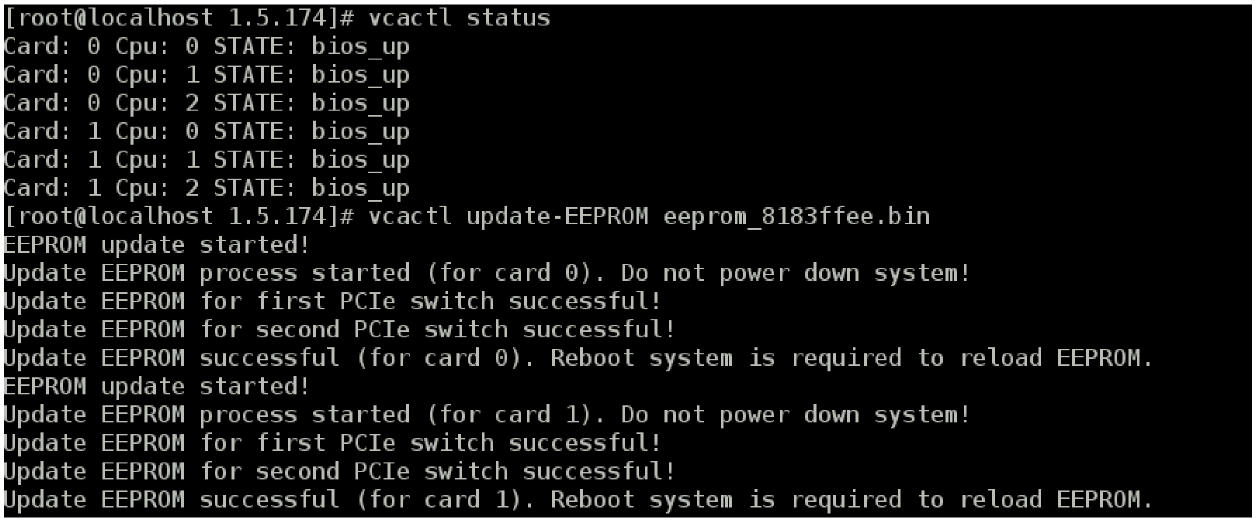
Description
Updates BIOS via the Gold BIOS image. Use when in need of recovery. Before running this command, the Intel® VCA 2 nodes must be in the `bios_up` state. This command may take much longer than the standard BIOS update.

6.3.6 `vcactl update-EEPROM`

Usage

```
vcactl update-EEPROM [<card_id>] <eepromImageFile>
```

Description
Updates PCIe* firmware on all Intel® VCA cards or on the specified card. After successful completion of the command, the host system is rebooted to work with the new firmware on the card. This command requires root privileges. The Intel® VCA nodes must be in the `bios_up` state before running this command.



```
[root@localhost 1.5.174]# vcactl status
Card: 0 Cpu: 0 STATE: bios_up
Card: 0 Cpu: 1 STATE: bios_up
Card: 0 Cpu: 2 STATE: bios_up
Card: 1 Cpu: 0 STATE: bios_up
Card: 1 Cpu: 1 STATE: bios_up
Card: 1 Cpu: 2 STATE: bios_up
[root@localhost 1.5.174]# vcactl update-EEPROM eeprom_8183ffee.bin
EEPROM update started!
Update EEPROM process started (for card 0). Do not power down system!
Update EEPROM for first PCIe switch successful!
Update EEPROM for second PCIe switch successful!
Update EEPROM successful (for card 0). Reboot system is required to reload EEPROM.
EEPROM update started!
Update EEPROM process started (for card 1). Do not power down system!
Update EEPROM for first PCIe switch successful!
Update EEPROM for second PCIe switch successful!
Update EEPROM successful (for card 1). Reboot system is required to reload EEPROM.
```

Figure 44. `vcactl update-EEPROM` example

Result
The EEPROMs on the card(s) are updated upon successful completion, at which point a reboot of the system is required.

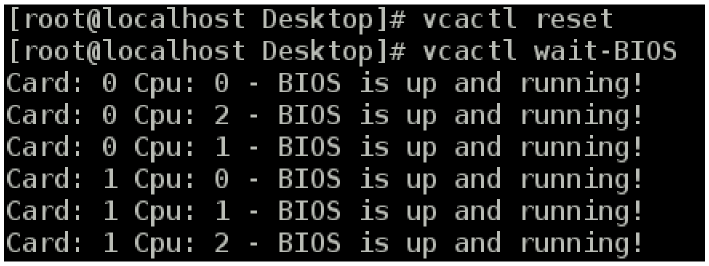
If there is a failure, an error status is reported after completion of the command. In these circumstances, a retry of the update is strongly recommended prior to any reboot; otherwise, the card may be damaged.

6.3.7 `vcactl reset`

Usage

```
vcactl reset [<card_id> <cpu_id>]
```

Description
Resets the specified CPUs to `bios_up` state. Run `vcactl wait-BIOS` after `vcactl reset` to wait for `vcactl reset` to complete. Normally, the CPUs should enter the BIOS is up and running mode.



```
[root@localhost Desktop]# vcactl reset
[root@localhost Desktop]# vcactl wait-BIOS
Card: 0 Cpu: 0 - BIOS is up and running!
Card: 0 Cpu: 2 - BIOS is up and running!
Card: 0 Cpu: 1 - BIOS is up and running!
Card: 1 Cpu: 0 - BIOS is up and running!
Card: 1 Cpu: 1 - BIOS is up and running!
Card: 1 Cpu: 2 - BIOS is up and running!
```

Figure 45. `vcactl reset` example

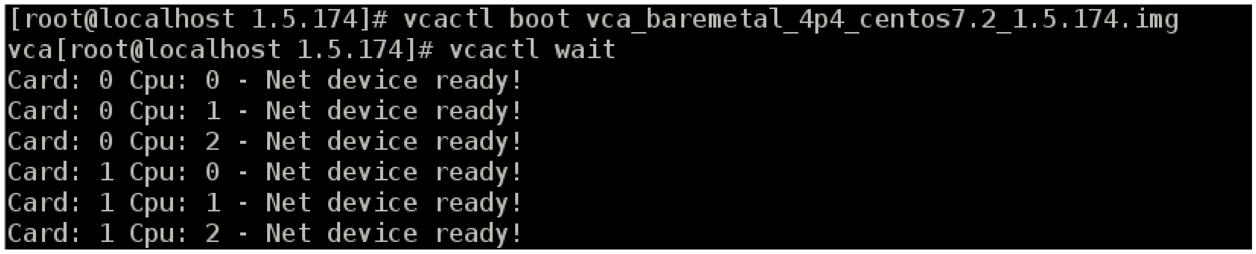
6.3.8 `vcactl boot`

Usage

```
vcactl boot [<card_id> <cpu_id>] [os_image_file.img|vcablk0 (for vca-disk)]
```

Description
Boots the operating system from Leverage Boot Protocol. Image is from the default location (see the `vcactl config os-image` command in section 6.3.20) or from the image file specified in the command. The command requires that the card CPU is in the `bios_up` or `bios_ready` state. Use `vcactl wait` to wait for the completion of the `vcactl boot` command.

It is possible to use the `last-os-image` field as a parameter to boot



```
[root@localhost 1.5.174]# vcactl boot vca_baremetal_4p4_centos7.2_1.5.174.img
vca[root@localhost 1.5.174]# vcactl wait
Card: 0 Cpu: 0 - Net device ready!
Card: 0 Cpu: 1 - Net device ready!
Card: 0 Cpu: 2 - Net device ready!
Card: 1 Cpu: 0 - Net device ready!
Card: 1 Cpu: 1 - Net device ready!
Card: 1 Cpu: 2 - Net device ready!
```

Figure 46. `vcactl boot` example

Result
During the OS boot, the reported state is `booting` (the `bios_done` state may be visible, also). On successful completion, each CPU is in the `net_device_ready` state. If there is a failure, the specific error cause is reported (via the `vcactl status` command) as one of `drv_probe_error`, `dhcp_error`, `nfs_mount_error`, `net_device_no_ip`, or `generic error`.

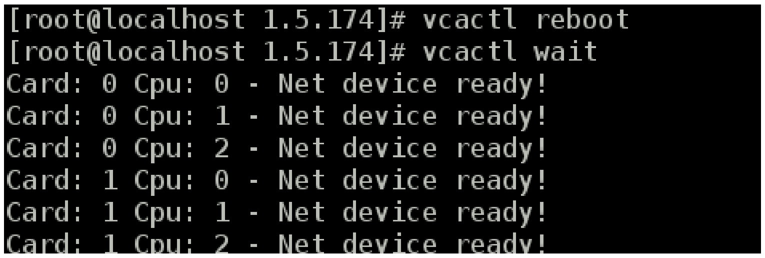
6.3.9 `vcactl reboot`

Usage

```
vcactl reboot [<card_id> <cpu_id>] [os_image_file.img]
```

Description

Reboots the OS (resets the CPU and boots the selected OS) from the host via Leverage Boot Protocol. Image is from the default location (see the `vcactl config` `os-image` command in section 6.3.20) or from the image file specified in the command. The image file should have group access set to `vcausers,` otherwise wait will give TIMEOUT. Use `vcactl wait` to wait for the completion of the `vcactl boot` command.



```
[root@localhost 1.5.174]# vcactl reboot
[root@localhost 1.5.174]# vcactl wait
Card: 0 Cpu: 0 - Net device ready!
Card: 0 Cpu: 1 - Net device ready!
Card: 0 Cpu: 2 - Net device ready!
Card: 1 Cpu: 0 - Net device ready!
Card: 1 Cpu: 1 - Net device ready!
Card: 1 Cpu: 2 - Net device ready!
```

Figure 47. `vcactl reboot` example

Result

During the OS boot, the reported state is `link_down`, `dio_down`, `bios_up`, `bios_ready`, `booting`, `net_device_up`, `dhcp_in_progress`, or `os_ready` (the `bios_done`, `mod_probe_done`, `dhcp_done`, `and nfs_mount_done` states may be visible for a short time, also). On successful completion, each CPU is in the `net_device_ready` state. If there is a failure, the specific error cause is reported (via the `vcactl status` command) as one of `drv_probe_error`, `dhcp_error`, `nfs_mount_error`, `net_device_no_ip`, or `generic error`.

6.3.10 `vcactl wait`

Usage

```
vcactl wait [<card_id> <cpu_id>]
```

Description

Waits until the OS boot is completed on the specified CPUs. This command is typically used after vcactl boot or vcactl reboot to report booting status.

Result

The command completes when the OS boot is done; otherwise, it is blocking. On successful boot, all CPUs finish in the net_device_ready state. If there is a failure, the specific error cause is reported as one of `drv_probe_error`, `dhcp_error`, `nfs_mount_error`, `net_device_no_ip`, or `generic error`.

6.3.11 `vcactl temp`

Usage

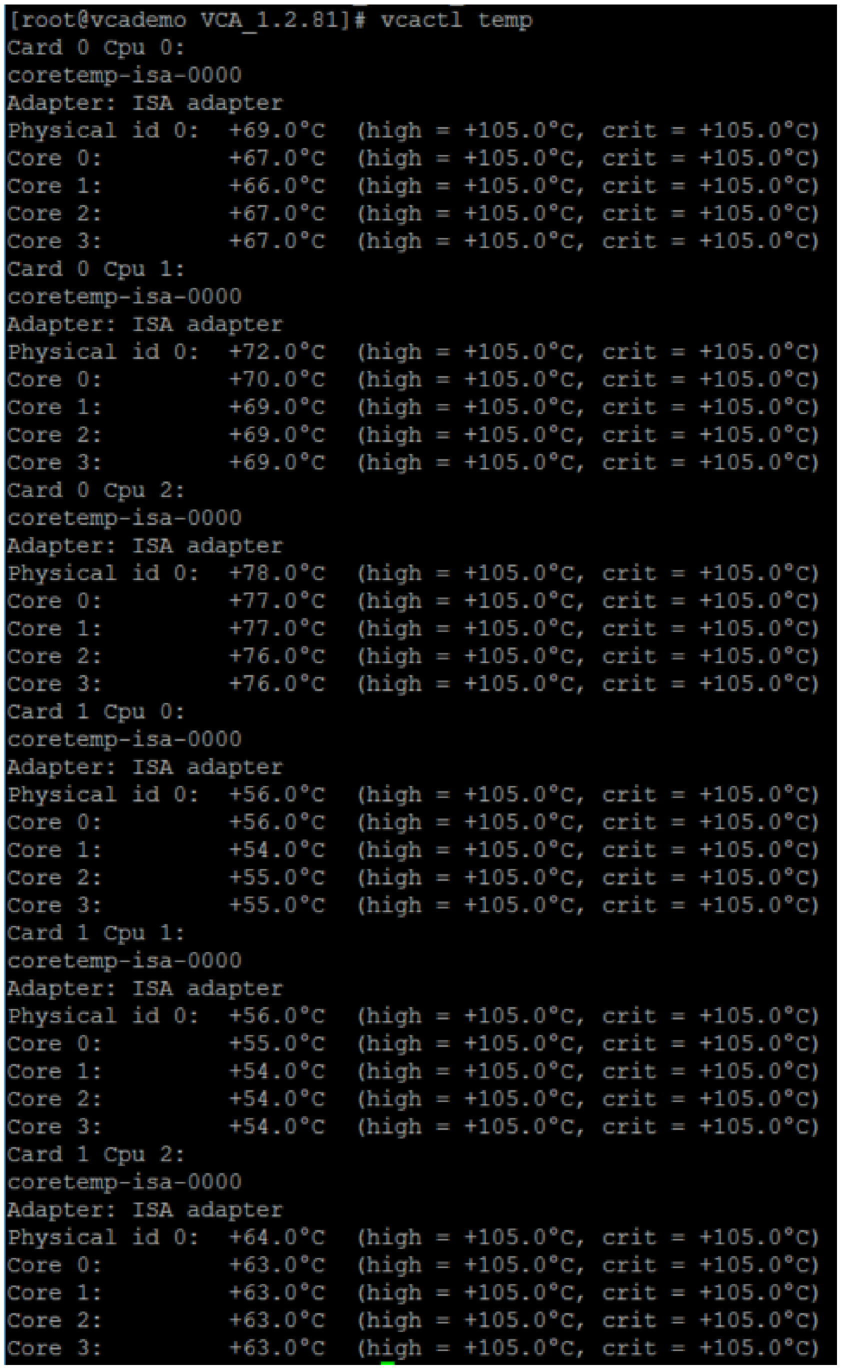
```
vcactl temp [<card_id> <cpu_id>]
```

Description

Retrieves the temperature of one or more CPUs in the system. If no card or CPU is specified, the command returns the temperature of all nodes on all cards.

Result

The current card output temperature and the node processor temperatures are reported for each card.



```
[root@vcademo VCA_1.2.81]# vcactl temp
Card 0 Cpu 0:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +69.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +67.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +66.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +67.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +67.0°C  (high = +105.0°C, crit = +105.0°C)
Card 0 Cpu 1:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +72.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +70.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +69.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +69.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +69.0°C  (high = +105.0°C, crit = +105.0°C)
Card 0 Cpu 2:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +78.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +77.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +77.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +76.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +76.0°C  (high = +105.0°C, crit = +105.0°C)
Card 1 Cpu 0:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +56.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +56.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +54.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +55.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +55.0°C  (high = +105.0°C, crit = +105.0°C)
Card 1 Cpu 1:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +56.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +55.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +54.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +54.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +54.0°C  (high = +105.0°C, crit = +105.0°C)
Card 1 Cpu 2:
coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +64.0°C  (high = +105.0°C, crit = +105.0°C)
Core 0:         +63.0°C  (high = +105.0°C, crit = +105.0°C)
Core 1:         +63.0°C  (high = +105.0°C, crit = +105.0°C)
Core 2:         +63.0°C  (high = +105.0°C, crit = +105.0°C)
Core 3:         +63.0°C  (high = +105.0°C, crit = +105.0°C)
```

Figure 48. `vcactl temp` example

6.3.12 `vcactl blockio open`

Usage

```
vcactl blockio open <card_id> <cpu_id> [vcablk<N>] [[RO|RW
   <file_path>]|[ramdisk <size_mb>]]
```

where 0 ≤ `N` < 8.

Description
Opens block IO device.

Examples

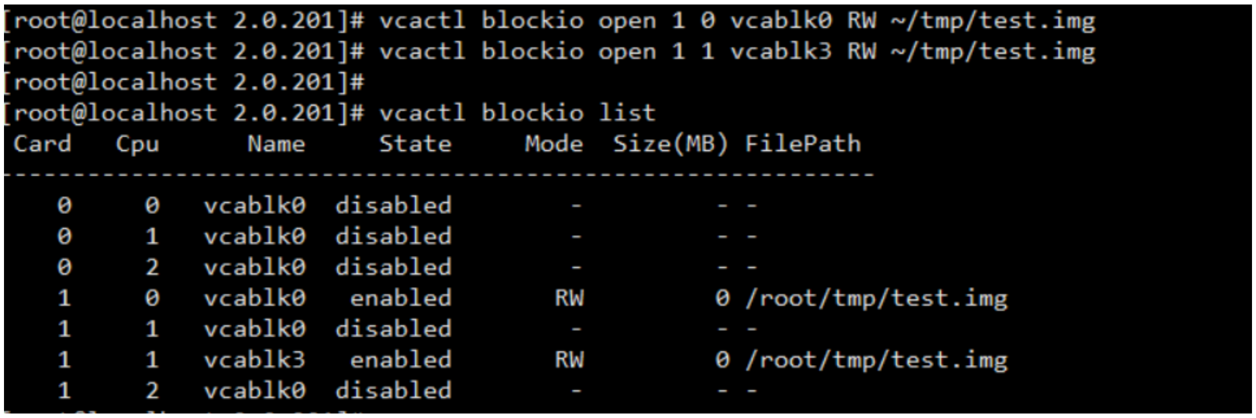
```
vcactl blockio open 0 1 vcablk3 ramdisk 20
vcactl blockio open 0 1 vcablk3 RO ~/disk.img
```

6.3.13 `vcactl blockio list`

Usage

```
vcactl blockio list <card_id> <cpu_id>
vcactl blockio list <blockio_id>
```

Description
Lists block IO devices.



```
[root@localhost 2.0.201]# vcactl blockio open 1 0 vcablk0 RW ~/tmp/test.img
[root@localhost 2.0.201]# vcactl blockio open 1 1 vcablk3 RW ~/tmp/test.img
[root@localhost 2.0.201]#
[root@localhost 2.0.201]# vcactl blockio list
 Card   Cpu      Name     State     Mode  Size(MB) FilePath
-----------------------------------------------------------
   0     0   vcablk0  disabled        -         - -
   0     1   vcablk0  disabled        -         - -
   0     2   vcablk0  disabled        -         - -
   1     0   vcablk0   enabled       RW         0 /root/tmp/test.img
   1     1   vcablk0  disabled        -         - -
   1     1   vcablk3   enabled       RW         0 /root/tmp/test.img
   1     2   vcablk0  disabled        -         - -
```

Figure 49. `vcactl blockio list` example

6.3.14 `vcactl blockio close`

Usage

```
vcactl blockio close <card_id> <cpu_id> [vcablk<N>]
```

where 0 ≤ `N` < 8.

Description
Closes block IO device.

Examples

```
vcactl blockio close 0 1 vcablk3
vcactl blockio close vcablk0
```

6.3.15 `vcactl network`

Usage

```
vcactl network <subcommand> [<card_id> <cpu_id>]
```

Description
Reads current network configuration of the card CPU to learn the IP address assigned by DHCP and the MAC addresses of the network interfaces of the Dom0 OS and DomU OS. The subcommands are the following and work on all or on the specified node:

- `ip` – Displays IP address of the (specified) node's eth0 interface.
- `ip6` – Displays IP v6 address of the (specified) node's eth0 interface.
- `mac` – Displays MAC address of the (specified) node's eth0 interface so that user may configure DHCP server with that information.
- `vm-mac` – Displays MAC address of the DomU virtual network interface of the (specified) node's so that DHCP server can be set accordingly.
- `all` – Displays all information of the (specified) node's eth0 interface.
- `stats` – Displays statistics of the (specified) node's eth0 interface.
- `dhcp-renew` – Forces renewal of DHCP address for (specified) nodes.

6.3.16 `vcactl ICMP-watchdog`

Usage

```
vcactl ICMP-watchdog <1/0> [<card_id> <cpu_id>] [<ip_address>]
```

Description
Enables (1) or disables (0) ICMP watchdog service on the host that pings the specified node. The ICMP ping address is selected according to the current configuration. Optionally, the command may explicitly specify the IP address, so that the command can be used to monitor DomU activity.

6.3.17 `vcactl update-MAC`

Usage

```
vcactl update-MAC <card_id> <cpu_id> <macAddressCanonical>
```

Description
Updates MAC address (flash SPI GBE partition) on the specified CPU in the system. `card_id` and `cpu_id` are mandatory parameters for this command.

Command should be executed only in bios_up state.

Result
Status of the CPU transitions from `bios_up` or `bios_ready` to `flashing` and completes in the `bios_ready` state. If there is a failure, an `error` status is reported (via the `vcactl status` command).

6.3.18 `vcactl script`

Usage

```
vcactl script [<card_id> <cpu_id>] <scriptFile>
```

Description
Configures a bash script for the card CPU to perform Linux user-specific configuration that might be any configuration action such as:

- IP address assignment per virtIO interface or enabling the DHCP client
- Starting DomU guest OS

6.3.19 `vcactl clear-SMB-event-log`

Usage

```
vcactl clear-SMB-event-log <card_id> <smbId>
```

Description

Clears the SMB log that can be accessed by the `vca_elog.py` tool on nodes. The node should be in the `bios_up` or `bios_ready` state.

6.3.20 `vcactl config`

Usage

```
vcactl config [<card_id> <cpu_id>] <parameter-name> <parameter-value>
```

The available options and settings are listed in Table 12.

Table 12. `vcactl config` parameters

Parameter Name	Available Values	Description
Global options		
`auto-boot`	1 – enable, 0 – disable	Enable/disable auto booting. Note that for auto-boot to work, the directory containing the boot image and all its parent directories must either be world-readable or be owned by the `vcausers` group.
`debug-enabled`	1 – enable, 0 – disable	Enable/disable debug commands.
`link-up-timeout-ms`	# of milliseconds	Set timeout for link-up after reset in milliseconds. Default is 2000 milliseconds (2 seconds)
`handshake-irq-timeout-ms`	# of milliseconds	Set timeout for link-up after reset in milliseconds. Default is 30000 milliseconds (30 seconds).
`alloc-timeout-ms`	# of milliseconds	Set timeout for RAMDISK allocation in BIOS. Default 100 is milliseconds.
`cmd-timeout-ms`	# of milliseconds	Set timeout for LBP command reaction in BIOS. Default is 1000 milliseconds.
`mac-write-timeout-ms`	# of milliseconds	Set timeout for MAC update command processing time in BIOS. Default 1000 milliseconds.
`default-daemon-script`	Name and path to daemon script file	Name and path to daemon script file that is executed upon watchdog expiration.
`wait-cmd-timeout-s`	# of seconds	Set timeout for WAIT command in vcactl (time needed to boot OS up). Default is 160 seconds.
`wait-bios-cmd-timeout-s`	# of seconds	Set timeout for WAIT-BIOS command in vcactl for BIOS UP time needed. Default is 300 seconds.
`wait-bios-cmd-flashing-s`	# of seconds	Set timeout for WAIT-BIOS command in vcactl for completion of BIOS update. Default is 1200 seconds.
`ICMP-ping-interval-s`	# of seconds	Set ICMP watchdog period. Default is 1 second.
`ICMP-response-timeout-s`	# of seconds	Set ICMP response timeout. Default is 10 second.
`va-min-free-memory-enabled`	1 – enable, 0 – disable	Enable/disable minimum value (512 MB) of non-cached memory by Linux OS filesystem. This ensures that there will be enough free space to allocate buffers needed by virtual network over PCIe*. Option refers to host system.
Per card node options		
`os-image`	Path to image	Path to bootable OS image
`last-os-image`	Path to image (not user-configurable)	Shows last booted image on node

Parameter Name	Available Values	Description
`script`	Path to script	Path and script file name that is executed on card after OS image has been booted successfully. `/etc/vca_config.d/vca_<slot_id><cpu_id>_default.sh`
`daemon-script`	Path to script	Name and path to daemon script file that is executed upon watchdog expiration.
`ip`	IP address or `dhcp`	Set the node IP address or enable DHCP operation. Default is 172.31.<slot_id * 3 + cpu_id + 1>.1. To enable DHCP (since Intel® VCA Software Release 1.2.) for automatic IP address assignment, set to `dhcp`. Enabling DHCP is allowed only if the node is already added to an existent bridge interface.
`mask`	# of bits	Set mask length in bits. Default is 24 bits.
`gateway`	IP address	Set gateway address. Default is 172.31.<slot_id * 3 + cpu_id + 1>.254.
`host-ip`	IP address	Set host IP address Default is 172.31.<slot_id * 3 + cpu_id + 1>.254.
`host-mask`	# of bits	Set host mask length in bits. Default is 24 bits.
`cpu-max-freq-non-turbo`	# of 100 MHz	Set CPU maximum frequency. Default is 0 (turbo enabled).
`bridge-interface`	Name of the bridge network interface on the host or `none` to disable bridging	To enable bridging to the node, use the name of an existent bridge interface (for example, `vcabr0`). To disable bridging to the node, use `none`. Disabling is allowed if DHCP is disabled (meaning that the IP address already has a static value). The command is available since Intel® VCA Software Release 1.2.
`node-name`	Name of the node	Provide a unique name for each node in the system.
`nfs-server`	IP address or FQDN of NFS host for persistent image boot	Provide a path for the node to access persistent boot filesystem over the network. If using FQDN, name resolution must be operational
`nfs-path`	Path on NFS server where persistent filesystem is stored	Path to share on the NFS server. Use `%s` to indicate "node-name" value in the path.
`block-devs`	N/A	This is a section which describes some additional information about each defined VCA block device. By default there is only one entry: vcablk0. Another devices will appear if you add them using 'vcactl blockio open' command.
`vcablk<N>`	N/A	It is a root entry for each defined block device (where 0 ≤ `N` < 8).
`mode`	RW – read/write, RO – read-only, ramdisk – device will be created into RAM	This is a way (RW or RO) how VCA block device will be created. User can choose if block device may be read-only or with read/write permissions.
`path`	Path to block device.	Path to file used as VCA BlockIO device.
`ramdisk-size-mb`	Size of block device created inside node RAM.	VCA BlockIO device can be also created in node RAM memory (for example as SWAP partition as RAM is very fast memory type).
`enabled`	1 – enable, 0 – disable	Option used to enable/disable block device. All enabled devices will be created/opened during booting node phase.
`va-min-free-memory-enabled-node`	1 – enable, 0 – disable	Enable/disable minimum value (512 MB) of non-cached memory by Linux OS filesystem. This ensures that there will be enough free space to allocate buffers needed by virtual network over PCIe*. Option refers to node system (Linux only).

6.3.21 `vcactl config-use`

Usage

```
vcactl config-use
```

Description

Triggers the `vcactld` daemon to use the last configuration.

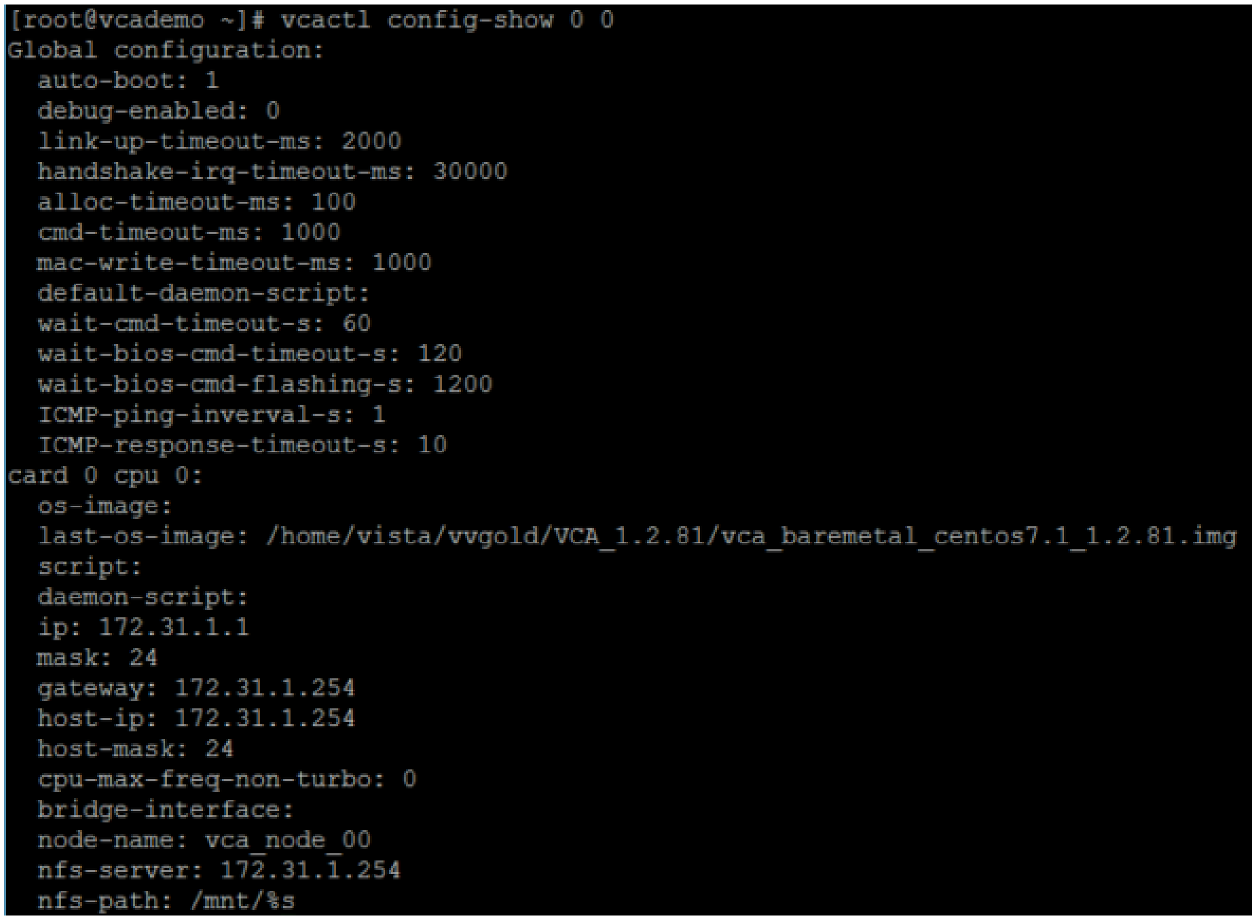
6.3.22 `vcactl config-show`

Usage

```
vcactl config-show [<card_id> <cpu_id>] <parameter-name> <parameter-value>
```

Description

Shows the current configuration of the `vcactld` daemon.



```
[root@vcademo ~]# vcactl config-show 0 0
Global configuration:
  auto-boot: 1
  debug-enabled: 0
  link-up-timeout-ms: 2000
  handshake-irq-timeout-ms: 30000
  alloc-timeout-ms: 100
  cmd-timeout-ms: 1000
  mac-write-timeout-ms: 1000
  default-daemon-script:
  wait-cmd-timeout-s: 60
  wait-bios-cmd-timeout-s: 120
  wait-bios-cmd-flashing-s: 1200
  ICMP-ping-inverval-s: 1
  ICMP-response-timeout-s: 10
card 0 cpu 0:
  os-image:
  last-os-image: /home/vista/vvgold/VCA_1.2.81/vca_baremetal_centos7.1_1.2.81.img
  script:
  daemon-script:
  ip: 172.31.1.1
  mask: 24
  gateway: 172.31.1.254
  host-ip: 172.31.1.254
  host-mask: 24
  cpu-max-freq-non-turbo: 0
  bridge-interface:
  node-name: vca_node_00
  nfs-server: 172.31.1.254
  nfs-path: /mnt/%s
```

Figure 50. `vcactl config-show` example

6.3.23 `vcactl config-default`

Usage

```
vcactl config-default
```

Description

Restores the `vcactld` daemon to default values.

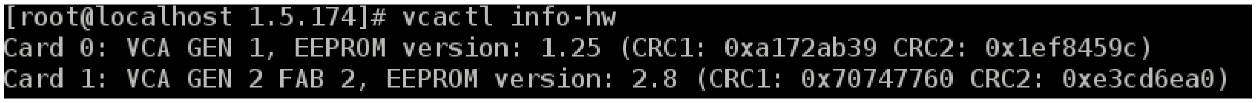
6.3.24 `vcactl info-hw`

Usage

```
vcactl info-hw
```

Description

Displays hardware information for the Intel® VCA card including if the card is GEN 1 or GEN 2 and its EEPROM version.



```
[root@localhost 1.5.174]# vcactl info-hw
Card 0: VCA GEN 1, EEPROM version: 1.25 (CRC1: 0xa172ab39 CRC2: 0x1ef8459c)
Card 1: VCA GEN 2 FAB 2, EEPROM version: 2.8 (CRC1: 0x70747760 CRC2: 0xe3cd6ea0)
```

Figure 51. `vcactl info-hw` example

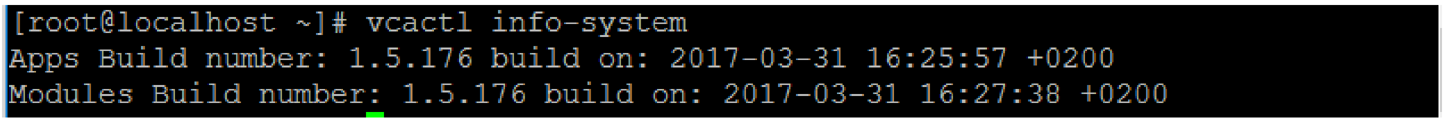
6.3.25 `vcactl info-system`

Usage

```
vcactl info-system
```

Description

Displays information of the software installed on the Intel® VCA card including version and build date.



```
[root@localhost ~]# vcactl info-system
Apps Build number: 1.5.176 build on: 2017-03-31 16:25:57 +0200
Modules Build number: 1.5.176 build on: 2017-03-31 16:27:38 +0200
```

Figure 52. `vcactl info-system` example

6.3.26 `vcactl pwrbtn-short` (only for Intel® VCA 2)

Usage

```
vcactl pwrbtn-short <card_id> <cpu_id>
```

Description

- Power button toggle of CPU `<cpu_id>`
- Shutdown active OS on the specified CPUs or turn CPU on when it's off. Command imitates short press the power button.

Result

- Command completes when CPU is off or on depending on previous state. All CPUs finish in `power_off` state in case of shutting down or in `bios_up` or `bios_ready` state when CPUs were off.

6.3.27 `vcactl pwrbtn-long` (only for Intel® VCA 2)

Usage

```
vcactl pwrbtn-long <card_id> <cpu_id>
```

Description

- Power button override five seconds on CPU `<cpu_id>` to cut off Intel® VCA power supply immediately; the OS can be damaged. For comparison, `vcactl pwrbtn-short` first shuts down the OS gracefully, then cuts off Intel® VCA power supply.
- Normally, `vcactl pwrbtn-short` should be used to power off Intel® VCA. The `vcactl pwrbtn-long` command is only used when Intel® VCA is in strange mode and `vcactl pwrbtn-short` cannott power off the Intel® VCA card.
- CPU `<cpu_id>` is in `power_off` status until next `vcactl pwrbtn-short`.

Result

- Command completes when all CPUs finish in `power_off` state.

6.3.28 `vcactl os-shutdown`

Usage

```
vcactl os-shutdown <card_id> <cpu_id>
```

Description

Shuts down the running operating system on the specified CPU.

It is not recommended to use the `os-shutdown` command in combination with `pwrbtn-short`. (After using `os-shutdown`, it is necessary to wait some time before executing the `pwrbtn-short` command again to bring up CPU power.)

6.3.29 `vcactl get-BIOS-cfg`

Usage

```
vcactl get-BIOS-cfg <card_id> <cpu_id> [bios_cfg_name]
```

Description

Read BIOS configuration of the specified CPU. Available configurations to be read are:

- `sgx`
- `gpu-aperture`
- `tdp`.

Example

```
vcactl get-BIOS-cfg 0 1 sgx
```

6.3.30 `vcactl set-BIOS-cfg`

Usage

```
vcactl set-BIOS-cfg <card_id> <cpu_id> <bios_cfg_name> <bios_cfg_value>
```

Description

Set/change BIOS configuration of specified CPU. Available configurations to be read are:

- `sgx`: Enable/disable Intel® Software Guard Extensions. Set to `enable` or `disable`.
- `gpu-aperture`: Megabytes of graphics memory. Set to 128, 256, 512, 1024, 2048, or 4096.
- `tdp`: Set to 0 to 11, where zero means base value, 1 means base value + 1, etc.

The following restrictions apply:

- `sgx` works only with 512 `gpu-aperture` size or less.
- `sgx` and `tdp` are supported on Intel® VCA 2 only.
- Maximum `gpu-aperture` for Intel® VCA (VCA1283LVV) is 1024.

Examples

```
vcactl set-BIOS-cfg 1 0 sgx enable
vcactl set-BIOS-cfg gpu-aperture 1024
```

6.3.31 `vcactl info`

Usage

```
vcactl info <subcmd> [<card_id> <cpu_id>]
```

Description

Command allows to read specific information about either Intel® VCA SW/FW or Node OS. The subcommands are the following and work on all or on the specified node:

- `BIOS`: Displays BIOS version of the Intel® VCA2 nodes.
- `hw`: Displays hardware information for the Intel® VCA card including if the card is GEN 1 or GEN 2 and its EEPROM version.
- `system`: Displays information of the software installed on the Intel® VCA card including version and build date.
- `node-os`: Displays information of node OS on the Intel® VCA card.

Examples

```
vcactl info hw
vcactl info BIOS
```

7. IP Multicast with Intel® VCA Nodes

An Intel® VCA network stack implements standard network device and thus is capable of sending and receiving packets to multicast addresses as well as Internet Group Management Protocol (IGMP) packets.

Default Intel® VCA topology consists of separate sub-networks for each node. In such a setup, the host system must act as a multicast router for the nodes to be able to reach each other via multicast.

7.1 Multicast Routing with mrouted

This section contains example configurations for multicast routing using an mrouted daemon.

7.1.1 Notes on mrouted

mrouted is a daemon used to manage dynamic multicast routing, using Distance Vector Multicast Routing Protocol (DVMRP). It is an open source tool, released on Stanford's license, available at http://troglobit.github.io/mrouted.html.

Several limitations of the mrouted daemon have been discovered.

- Relies only on network addressing and masks and does not analyze routing tables.
- Does not support interfaces with 32-bit masks.
- Ignores network interface if its IP range is a sub-range or other interface address range. For example, interface `eth0 172.31.1.1/32` would not be considered for routing if `xenif0 172.31.1.2/24` is present.
- Requires network addressing to be changed when DomU is used on host or VCA nodes so that all networks are separated (see section 7.1.2).
- Scans for network interfaces only during daemon start.
- Must be manually started by user, when all VCA nodes intended for multicast using are already up (can connect to their unicast addresses); because of this, mrouted cannot be a service auto started within host OS boot.
- Does not detect interface down and up.
- Needs to be restarted after bringing up the node when any VCA node using multicast is reset.
- Clears multicast routing table at exit.
- Requires all clients to re-register into multicast group when restarted; otherwise they are not able to receive multicast messages.
- Results in multicast messages sent by the host that cannot be received by the card's nodes. (This limitation is under investigation, as network packet analysis shows that multicast packets arrive to the nodes.)

7.1.2 Example Configuration for Xen Domain U

If DomU needs to be used, network addresses of host and card interfaces need to be changed, as well as mask width. This is the implication of the mrouted limitation that confines operation only on networks with exclusive address ranges. Suggested network addressing is shown in Figure 53.

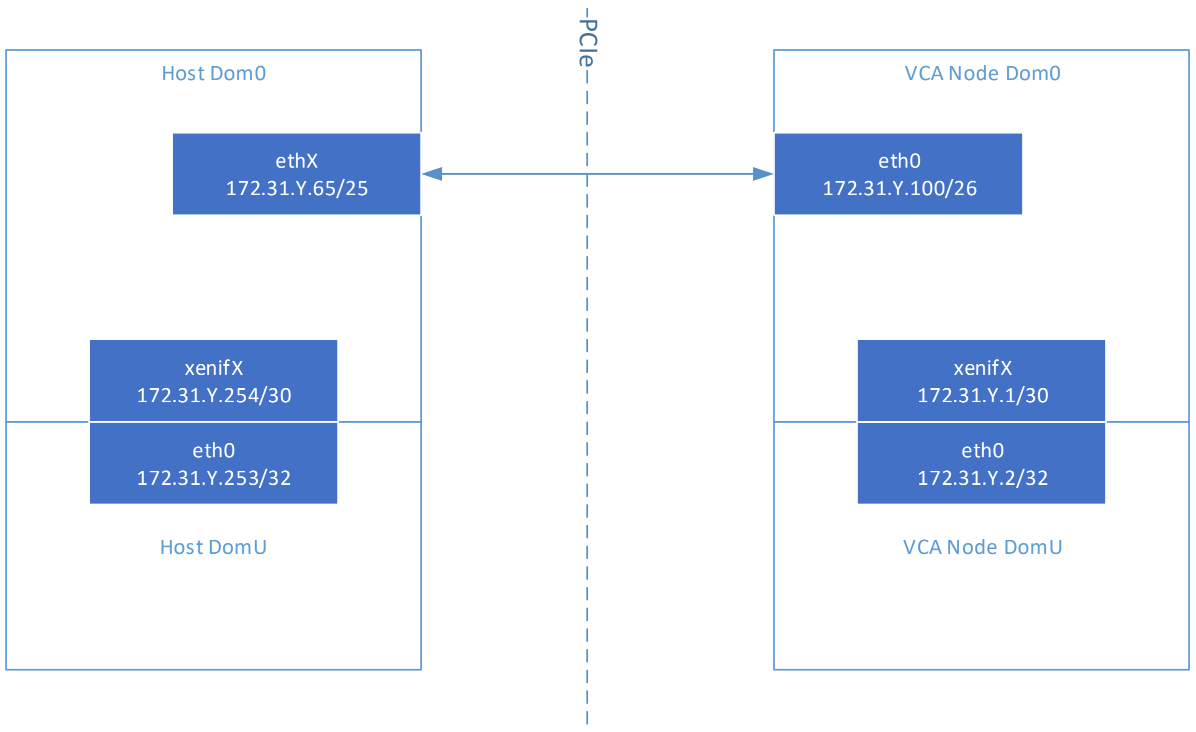


Figure 53. Suggested network addressing for DomU usage

The settings must be manually applied. This can be done by simple copying `/etc/vca_config.d/vca_xen_multicast_config.xml` to `/etc/vca_config.d/vca_config.xml` prior to the card nodes booting.

7.1.3 Preparing for Multicasts

1. Before using multicasts for the first time, user must install mrouted daemon on the host. The RPM package can be found within the Intel® VCA release in `INSTALL\Binaries\mrouted\` directory.
2. Boot an Intel® VCA node using the `vcactl boot` command (baremetal or Xen image).
3. Start the domain.
   - If Xen domain U is required on the Intel® VCA node, start it with the `--multicast` option.

     ```
     vca_start_card_domu.sh -p <nfs path> -c <card id> -n <CPU ID> -i
        <image name> -f <config file name> --multicast
     ```

   - To start DomU on the host, no additional parameters are needed for multicast support.

     ```
     vca_start_host_domu.sh -c <card id> -n <CPU ID> -i <image name> -f
        <config file name>
     ```

4. Start the mrouted daemon:

   ```
   mrouted
   ```

Note: After restarting DomU on the host, the mrouted daemon on the host needs to be restarted. There is no such need when restarting DomU on the card (assuming that the Dom0 on card has not been restarted).

7.1.4 Restarting mrouted Daemon

There is no option to restart the mrouted daemon. Instead, kill the existing instance with system tools and start the daemon manually once again.

```
killall mrouted
mrouted
```

7.1.4.1 Checking Multicast Routes

To check registered multicast groups known to mrouted on available network adapters, use the following command.

```
mrouted -r
```

A sample output looks like the following.

```
vifs_with_neighbors = 1
[This host is a leaf]

Virtual Interface Table
Vif  Name  Local-Address                               M  Thr  Rate   Flags
 0   eth0  172.31.1.100    subnet: 172.31.1.64/26      1   1      0
                            peers: 172.31.1.65 (3.255) [0] have-genid up
   1:34:37
           group host (time left): 224.94.1.1      172.31.1.100    (
   0:03:40)
                                   224.0.0.2       172.31.1.65     (
   0:03:39)
                                   224.0.0.4       172.31.1.65     (
   0:03:40)
                      IGMP querier: 172.31.1.65        up  1:34:32 last heard
   0:00:45 ago
                       Nbr bitmaps: 0x0000000000000001
                    pkts/bytes in : 1913/2831956
```

```
                      pkts/bytes out: 0/0
  1 xenif0  172.31.1.1      subnet: 172.31.1.0/30       1   1      0
    querier leaf
            group host (time left): 224.94.1.2      172.31.1.2      (
    0:03:39)
                                    224.0.0.4       172.31.1.1      (
    0:03:37)
                                    224.0.0.2       172.31.1.1      (
    0:03:37)
                        IGMP querier: 172.31.1.1         (this system)
                         Nbr bitmaps: 0x0000000000000000
                      pkts/bytes in : 0/0
                      pkts/bytes out: 540/808920

Multicast Routing Table (9 entries)
 Origin-Subnet      From-Gateway    Metric Tmr Fl In-Vif  Out-Vifs
 172.31.1.0/30                          1    75 ..   1    0*
 172.31.1.64/26                         1    75 ..   0    1*
 192.168.122/24     172.31.1.65         2    25 ..   0    1*
 172.31.6/24        172.31.1.65         2    25 ..   0    1*
 172.31.5/24        172.31.1.65         2    25 ..   0    1*
 172.31.4/24        172.31.1.65         2    25 ..   0    1*
 172.31.3/24        172.31.1.65         2    25 ..   0    1*
 172.31.2/24        172.31.1.65         2    25 ..   0    1*
 10.102.108/23      172.31.1.65         2    25 ..   0    1*
```

The virtual interface table shows a list of all managed interfaces (here: `eth0` and `xenif0`) with their addresses and subnet masks. For each interface, registered multicast groups are shown (`eth0`: `224.94.1.1`, `224.0.0.2` and `224.0.0.4`; `xenif0: 224.94.1.2`, `224.0.0.4` and `224.0.0.2`)

To check all active routings use the following command.

```
ip mroute
```

The following is a sample output.

```
(172.31.1.100, 224.94.1.1)        Iif: eth0
(10.102.109.202, 224.94.1.2)      Iif: eth0       Oifs: xenif0
```

This means that only a single stream is active, transmitting data to multicast group `224.94.1.2`, originated by an external host with address `10.102.109.202`, received on `eth0` interface and passed to `xenif0` interface (because `xenif0` interface has registered to `224.94.1.2` group).

7.2 Sample Usage

A sample application supporting multicasts is the `iperf` tool. (Please note that `iperf3` does not support multicast, only `iperf` does.) All images delivered with the Intel® VCA release already contain `iperf`. On the host it might be required to install it manually.

To start multicast listener invoke:

```
iperf -s -u -B <group address> -i 1
```

for example:

```
iperf -s -u -B 224.94.1.1 -i 1
```

To start multicast transmitter invoke:

```
      iperf –c <group address> -u -T 32 -t 3 -i 1
```

for example:

```
      iperf -c 224.94.1.1 -u -T 32 -t 3 -i 1
```

Order of operations is the following:

1. Boot host.
2. Boot card's nodes (and DomUs, if needed).
3. Start mrouted daemon.
4. On all nodes and on server, stop `firewalld` service (or provide proper firewall exceptions for IGMP and `iperf` 5001 port).
5. Start `iperf` multicast listeners on selected nodes.
6. Start `iperf` multicast transmitters.

8. `vca_config.xml` File

The `vca_config.xml` file, located at `/etc/vca_config.d/vca_config.xml`, contains the configuration information for interaction with the Intel® Visual Compute Accelerator and its nodes from the host. This section serves as a brief overview of the sections and most important parts of the file.

The file is broken into two primary sections: global configuration, and card (node) configuration.

Note that most, if not all, of the configuration options stored in this file are accessible via the command line utility `vcactl config`, discussed in Section 6.3.20.

8.1 Global Section

The global section of the XML file consists of the following lines:

```
<global>
  <auto-boot>1</auto-boot>
  <debug-enabled>0</debug-enabled>
  <link-up-timeout-ms>2000</link-up-timeout-ms>
  <handshake-irq-timeout-ms>30000</handshake-irq-timeout-ms>
  <alloc-timeout-ms>100</alloc-timeout-ms>
  <cmd-timeout-ms>1000</cmd-timeout-ms>
  <mac-write-timeout-ms>1000</mac-write-timeout-ms>
  <default-daemon-script/>
  <wait-cmd-timeout-s>60</wait-cmd-timeout-s>
  <wait-bios-cmd-timeout-s>120</wait-bios-cmd-timeout-s>
  <wait-bios-cmd-flashing-s>1200</wait-bios-cmd-flashing-s>
  <ICMP-ping-inverval-s>1</ICMP-ping-inverval-s>
  <ICMP-response-timeout-s>10</ICMP-response-timeout-s>
</global>
```

A description of each line is below:

- `auto-boot`
  - `1`: Attempts to boot the `last-os-image` (see next section) on each node at operating system boot time.
  - `0`: Brings the nodes to `bios_ready`, but does not attempt to boot the nodes automatically.
- `debug-enabled`:
  - `0`: Show no additional debug information.
  - `1`: Allows additional debug information to be made available through a debug daughter card, primarily used for issue investigation.
- `link-up-timeout-ms`: Sets the delay, in milliseconds, that the Intel® VCA software waits for the node's network link to become responsive before reporting a timeout.
- `handshake-irq-timeout-ms`: Sets the delay, in milliseconds, that the Intel® VCA software waits for the node to complete handshake negotiation before reporting a timeout.
- `alloc-timeout-ms`: Sets the delay, in milliseconds, that the Intel® VCA software waits for the node allocation before reporting a timeout.
- `cmd-timeout-ms`: Sets the delay, in milliseconds, that the Intel® VCA software waits for the node to respond to a command before reporting a timeout.
- `mac-write-timeout-ms`: Sets the delay, in milliseconds, that the Intel® VCA software waits for a MAC reprogramming process to complete before reporting a timeout.
- `default-daemon-script`: Allows for a custom script to be run at daemon invocation.

- `wait-cmd-timeout-s`: Sets the delay, in seconds, that the Intel® VCA software waits for the node to report status OK during the `vcactl wait` command invocation before reporting a timeout.
- `wait-bios-cmd-timeout-s`: Sets the delay, in seconds, that the Intel® VCA software waits for the node to report status OK during the `vcactl wait-BIOS` command invocation before reporting a timeout.
- `wait-bios-cmd-flashing-s`: Sets the delay, in seconds, that the Intel® VCA software waits for the node BIOS update to report status OK before reporting a timeout.
- `ICMP-ping-interval-s`: Sets the delay, in seconds, that the Intel® VCA software waits between each attempt to send a ping packet to the node.
- `ICMP-response-timeout-s`: Sets the delay, in seconds, that the Intel® VCA software waits for node to report status OK to each ping request.

8.2 Card Section

The card section of the XML file consists of multiple sections, each starting with `<card id="X">` and ending with `</card>` (where `X` increments from 0 for the first card in the system). Within each card section are three node sections, which begin with `<cpu id="Y">` and end with `</cpu>` (where `Y` increments from 0 for the first node in the first card, and continues iterating through all card sections in the file). A node section contains the following lines:

```
<cpu id="0">
  <os-image/>
  <last-os-image>/media/vca/vca_baremetal_centos7.1_1.2.img</last-os-image>
  <script/>
  <daemon-script/>
  <ip>172.31.1.1</ip>
  <mask>24</mask>
  <gateway>172.31.1.254</gateway>
  <host-ip>172.31.1.254</host-ip>
  <host-mask>24</host-mask>
  <cpu-max-freq-non-turbo>0</cpu-max-freq-non-turbo>
  <bridge-interface/>
  <node-name>vca_node_00</node-name>
  <nfs-server>172.31.1.254</nfs-server>
  <nfs-path>/mnt/%s</nfs-path>
</cpu>
```

A description of each line is below:

- `last-os-image`: Provides the path and file name of the image most recently booted to the node.
- `ip`: The IP address to be assigned to the node, or `dhcp`

Note: Only set to DHCP if bridging is enabled to an external network with a DHCP server.

- `mask`: The network mask bit value if the IP address is statically assigned; ignored if `ip` is `dhcp`.
- `gateway`: The host IP address; ignored if `ip` is `dhcp`.
- `host-ip`: The host IP address for the connection to this node; ignored if `ip` is `dhcp`.
- `host-mask`: The network mask bit value for the host IP address if IP address is statically assigned (should be the same as the `mask` field); ignored if `ip` is `dhcp`.
- `cpu-max-freq-non-turbo`:
  - o `0`: Does not limit CPU maximum (non-turbo) frequency.

  - Non-zero value: Limits node's CPU maximum (non-turbo) frequency. Setting this to a low value could cause the node to become functionally useless as the processor is moving too slowly.
- `bridge-interface`: The bridge interface on the host (e.g., `virbr0`) to enable bridging from the node to the outside of the host. Leave unset (as shown above) to run in host-only networking.
- `node-name`: The hostname value for the node.
- `nfs-server`: The IP address or DNS name of the NFS server used for remote file storage.
- `nfs-path`: The path on the NFS server where the node accesses files. `%s` is an alias to `node-name`.

9. Troubleshooting

9.1 Intel® VCA Debug Logs

From Intel® VCA software version 2.1, a new package is provided to collect different logs from host server. For any issue, install this package, collect logs, and attach them to issue ticket. The following command installs the `vcaDebug` RPM.

- For CentOS*

  ```
  sudo yum -y localinstall vcadebug*rpm
  ```

- For Ubuntu*

  ```
  sudo dpkg -i <vcaDebug_package>.deb
  vcadebug packlogs
  ```

`packlogs` creates a ZIP archive with various logs and system information. The following files (if they exist) and process outputs are registered in the ZIP archive.

```
o  /var/log/vca_ifup_log
o  /var/log/vcactld
o  /var/log/vca/vcactld.log
o  /var/log/vca/vca_ifup.log
o  /var/log/vca/vcactl.log
o  /var/log/vca/vcactl.log.old
o  /etc/vca_config.d/vca_config.xml
o  /etc/exports exports.log
o  dmesg
o  ifconfig
o  systemctl status vcactl
o  systemctl status NetworkManager
o  vcactl info system
o  vcactl info hw
o  vcactl info BIOS
o  rpm -qa --pipe "grep -i vca"
o  uname -a
```

9.2 Reading Intel® VCA Events with vca_elog.py Script

The `vca_elog` Python script prints SMBIOS Type 15 events. These events are logged by BIOS in its NVRAM area. They can be viewed in the BIOS event log or by this script.

By default, this tool displays only error events.

9.2.1 Prerequisites

A Python interpreter must be installed on nodes on which this script is to be run. This has already been performed on all Intel® VCA Reference OS images.

The script is primarily designed to run on Intel® VCA nodes, although it may be also run on other computers to offline interpret previous uploaded NVRAM area (use the `-i` option).

To install a Python interpreter on a CentOS* computer, invoke the following command:

```
sudo yum install python
```

9.2.2 Script Usage

When the script is installed on the image, use the following command to execute the script.

```
vca_elog.py [options]
```

The command with all arguments is given below.

```
vca_elog.py [-h] [-e | -a | -t TYPE [TYPE ...] | -p] [-s] [-d DOWNLOAD_LOG |
   -i INPUT_FILE] [--debug] [-v]
```

The optional arguments are defined below.

```
-h, --help               show this help message and exit
-e, --errors             print error events only
-a, --all                print all events
-t TYPE [TYPE ...], --types TYPE [TYPE ...]
                         print specified events only
-p, --print_types        print event types with associated numbers for use
   with --types option
-s, --statistics         print events statistics
-d DOWNLOAD_LOG, --download_log DOWNLOAD_LOG
                         download log data from NVRAM to binary file
-i INPUT_FILE, --input_file INPUT_FILE
                         parse file downloaded with download_log option
--debug                  print additional debug information
-v, --version            print version and exit; ignores other options
```

To print this help, use the following command:

```
vca_elog.py -h
```

9.2.3 Displaying Events

By default the script displays only error events, so the outputs of following two commands are identical.

```
vca_elog.py
vca_elog.py –e
```

The following is an example output showing errors only.

```
Single Bit ECC Memory Error
   Date: 2015-07-29 11:35:49
   S1:C1:D0
Single Bit ECC Memory Error
   Date: 2015-07-29 11:36:08
   S1:C0:D1
Single Bit ECC Memory Error
   Date: 2015-07-29 11:36:14
   S1:C1:D1
...
```

Use the parameter `–t TYPE [TYPE…]` to print specified events only.

```
vca_elog.py –t 1 2 0x17
```

The following is an example output from the preceding command.

```
System Boot
   Date: 2015-07-29 10:07:49
OEM0
```

```
      Date: 2015-07-29 10:07:49
   OEM0
      Date: 2015-07-29 10:07:49
   System Boot
      Date: 2015-07-29 11:30:41
   Single Bit ECC Memory Error
      Date: 2015-07-29 11:35:49
      S1:C1:D0
   Single Bit ECC Memory Error
      Date: 2015-07-29 11:36:08
      S1:C0:D1
   Single Bit ECC Memory Error
      Date: 2015-07-29 11:36:14
      S1:C1:D1
   ...
```

Each event report has two or three lines and are generally formatted in the following way.

```
Event name
Date: YYYY-MM-DD HH:MM:SS/no valid date
[data]
```

The first line is the name of the event and may be one of the following strings.

- `Single Bit ECC Memory Error`
- `Multi Bit ECC Memory Error`
- `Parity Memory Error`
- `Bus Time Out`
- `I/O Channel Check`
- `Software NMI`
- `POST Memory Resize`
- `POST Errors`
- `PCI Parity Error`
- `PCI System Error`
- `CPU Failure`
- `EISA Failsafe Timer Timeout`
- `Correctable Memory Log Disabled`
- `Logging Disabled for Event Type`
- `System Limit Exceeded`
- `Asyn HW Timer Expired`
- `System Configuration Information`
- `Hard Disk Information`
- `System Reconfigured`
- `Uncorrectable CPU Complex Error`
- `Log Area Reset`
- `System Boot`
- `OEM0`
- `OEM1`
- `OEM2`

The second line is the date of the event. The date is printed in format `YYYY-MM-DD HH:MM:SS` or `no valid date` is printed when the date is not set properly.

The last line is the custom data of the event. This version of software (1.5) only prints data for event types Single Bit ECC Memory Error and Multi Bit ECC Memory Error.

9.2.4 Event Types

To identify which IDs can be used with `-t` options, `-p` may be used. Output for the `-p` option is the following.

```
                                   Type's ID
 Name                              dec   hex     Class
---------------------------------------------------------
 Single Bit ECC Memory Error         1  0x01     error
 Multi Bit ECC Memory Error          2  0x02     error
 Parity Memory Error                 3  0x03     error
 Bus Time Out                        4  0x04     error
 I/O Channel Check                   5  0x05     status
 Software NMI                        6  0x06     status
 POST Memory Resize                  7  0x07     status
 POST Errors                         8  0x08     error
 PCI Parity Error                    9  0x09     error
 PCI System Error                   10  0x0a     error
 CPU Failure                        11  0x0b     error
 EISA Failsafe Timer Timeout        12  0x0c     error
 Correctable Memory Log Disabled    13  0x0d     status
 Logging Disabled for Event Type    14  0x0e     status
 System Limit Exceeded              16  0x10     status
 Asyn HW Timer Expired              17  0x11     status
 System Configuration Information   18  0x12     status
 Hard Disk Information              19  0x13     status
 System Reconfigured                20  0x14     status
 Uncorrectable CPU Complex Error    21  0x15     error
 Log Area Reset                     22  0x16     status
 System Boot                        23  0x17     status
 OEM0                              224  0xe0     oem
 OEM1                              225  0xe1     oem
 OEM2                              226  0xe2     oem
```

9.2.5 Displaying Statistics

Use the `-s` option to display statistics. The following is an example output.

```
Nb of log events: 279
Events statistics:
                                   Type's ID
 Name                              dec   hex     Class          Nb
-------------------------------------------------------------------
 Single Bit ECC Memory Error         1  0x01     error         266
 Multi Bit ECC Memory Error          2  0x02     error           0
 Parity Memory Error                 3  0x03     error           0
 Bus Time Out                        4  0x04     error           0
 I/O Channel Check                   5  0x05     status          0
 Software NMI                        6  0x06     status          0
 POST Memory Resize                  7  0x07     status          0
 POST Errors                         8  0x08     error           0
 PCI Parity Error                    9  0x09     error           0
 PCI System Error                   10  0x0a     error           0
```

```
CPU Failure                          11  0x0b     error          0
EISA Failsafe Timer Timeout          12  0x0c     error          0
Correctable Memory Log Disabled      13  0x0d     status         0
Logging Disabled for Event Type      14  0x0e     status         0
System Limit Exceeded                16  0x10     status         0
Asyn HW Timer Expired                17  0x11     status         0
System Configuration Information     18  0x12     status         0
Hard Disk Information                19  0x13     status         0
System Reconfigured                  20  0x14     status         0
Uncorrectable CPU Complex Error      21  0x15     error          0
Log Area Reset                       22  0x16     status         0
System Boot                          23  0x17     status         3
OEM0                                224  0xe0     oem           10
OEM1                                225  0xe1     oem            0
OEM2                                226  0xe2     oem            0
```

To print event report with statistics, `-e`, `-a`, and `-t` options may be used with `-s`.

9.2.6 Saving a File and Parsing a Saved FIle

It may be useful for reporting issues to a support team to save an event area from the NVRAM to a file. To do this, use the following command.

```
vca_elog.py -d nvram_log15.bin
```

The script confirms a properly completed dump with the following information.

```
Log successfully downloaded to file nvram_log15.bin
```

When the tool cannot save the area to a file, the following information is displayed.

```
Cannot open output binary file /invalid/path/filename/data.extension
```

To parse a previously stored file, use the following command.

```
vca_elog.py –i nvram_log15.bin
```

By default, only error are printed. To print all events or specified types, use options `–a` or `–t` with `–i`.

9.3 Using Kernel Crash Dump on Intel® VCA

In case of OS crash, it is possible to get back logs. Logs are useful for finding the cause of crash.

9.3.1 Crash Dump on Linux*

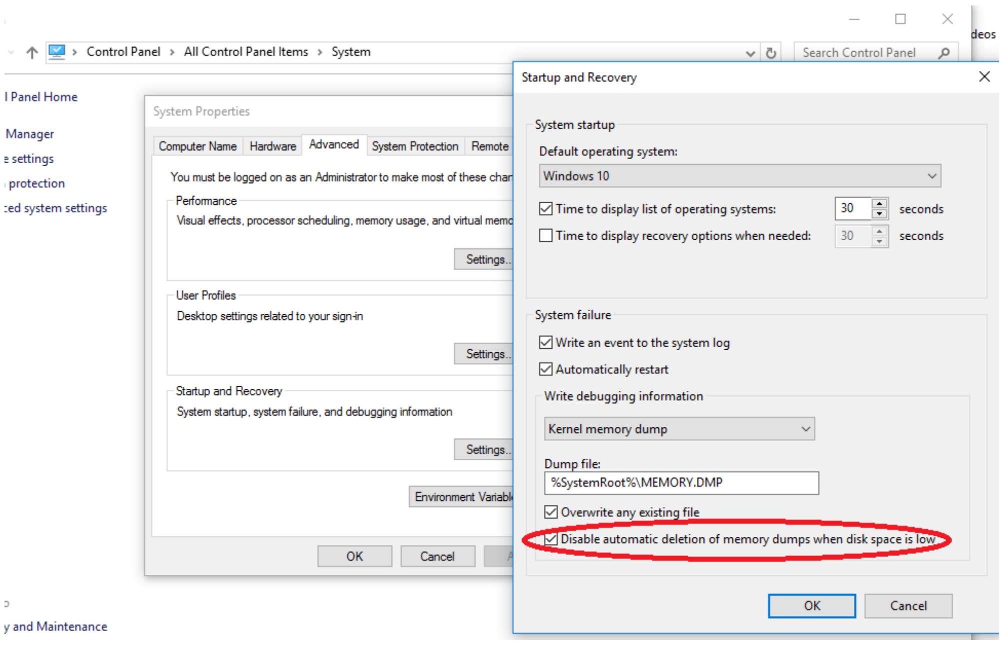
Kernel crash dump on Linux is supported by kdump. Logs after a crash are placed by default in `/var/crash`. (It is possible to change the directory.) Sample path:

```
/var/crash/127.0.0.1-2017-07-06-10\:16\:05/
```

9.3.2 Crash Dump on Windows*

To enable kernel crash dump on Windows, it is necessary to configure the environment.

1. In the **Control Panel**, open **System** settings.
2. Select **Advanced system settings.**
3. Click **Settings** under **Startup and Recovery**.
4. Select **Kernel memory dump** and check the option shown in Figure 54.



Startup and Recovery

System startup

Default operating system: Windows 10

☑ Time to display list of operating systems: 30 seconds

☐ Time to display recovery options when needed: 30 seconds

System failure

☑ Write an event to the system log

☑ Automatically restart

Write debugging information: Kernel memory dump

Dump file: %SystemRoot%\MEMORY.DMP

☑ Overwrite any existing file

☑ Disable automatic deletion of memory dumps when disk space is low

Figure 54. Enable kernel crash dump on Windows*

To read the crash dump logs, download the NotMyFault tool from https://technet.microsoft.com/en-us/sysinternals/notmyfault.aspx.

After a crash, copy the file from `C:\windows\mempory.dmp` to the folder where WinDbg is placed (e.g. `C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Windows Kits\Debugging Tools for Windows (X64)`). Then turn on WinDbg and select **File** > **Open Crash Dump**.

10. Quick Validation of VCA SW

To quickly validate the server setup and Intel® VCA SW, Intel® Visual Compute Accelerator AVC benchmark script is available to download. Please go through the readme.txt file for the requirements and usage once downloaded.

For this section, the following example configuration is assumed:

- Host
  - System: 2U x 2 GPU server
  - CPU: 2x Intel® Xeon® E5-2630 v4
  - Memory: 16 GB per socket (2x DDR4 8 GB 2400 MHz DIMMs)
  - Fans set to meet CFM requirements defined in section 3.3 of VCA2 HW Guide at card inlet temperature 35 °C.
- Intel® VCA
  - System: Intel VCA 2
  - Memory: 32 GB per node (2x DDR4 16 GB 2133 MHz DIMMs)
  - VCA SW version: `VCA SW 2.1.292`

10.1 AVC Transcoding Benchmark for Simultaneous AVC-AVC Transcodes

This test attempts to extract the maximum number of simultaneous AVC transcodes while maintaining a minimum of 30 FPS using Intel® Media Server Studio and a web-available video sample.

- **Target**: Most number of simultaneous AVC transcodes achieved while keeping 30 FPS (real-time)
- **Test configuration**: 1x Intel VCA 2 with all 3 nodes running simultaneous transcoding
- **Environment**: Bare metal, non-persistent image
- **Card OS**: CentOS and Ubuntu
- **Transcoding software**: Intel Media Server Studio 2018 R1 (available at https://software.intel.com/en-us/intel-media-server-studio)
- **Script files**: Script files to execute benchmarking is available at https://downloadcenter.intel.com/search?keyword=VCA and select Intel® Visual Compute Accelerator AVC benchmark script
- **PAR file details**:

```
-async 3 -hw -u 7 -cqp -qpi 24 -qpp 26 -qpb 28 -gop_size 61 -dist 3 -i::h264
   /input/big_buck_bunny_1080p.h264 -o::h264 /dev/null
```

- **Results**: Results are printed on the screen and can be obtained from log file in tools\h264\logs directory.

Table 13. Transcoding benchmark results

```
Card: 0 Node: 0 IP Address: w.x.y.z achieved_fps: >=30 Streams: 17
Card: 0 Node: 1 IP Address: w.x.y1.z1 achieved_fps: >=30 Streams: 17
Card: 0 Node: 2 IP Address: w.x.y2.z2 achieved_fps: >=30 Streams: 17
```

10.2 Temperature Profile

The temperature while booted in idle mode is ~35 °C. Temperature during workload is expected to reach ~60 °C. During transcoding workload, the script runs the load more times to ensure frame rate achieved is correct. Also longer input file is used during temperature profile. Temperature values from the VCA nodes are captured for every 10ms during the avc_benchmark script in a log file (avc_benchmark_temp_log.txt) and sorted along the Card and CPU in log file (avc_benchmark_sorted_temp.txt). Each VCA node has 4 cores and each core temperature is captured. If the average temperature is captured is higher than ~60 °C, the possible cause of temperature is CFM requirements may not be met. Check the fan settings in BIOS. For example, on Intel® Server Board S2600WT based systems, this can be enabled in BIOS setup by configuring the following two options on the PCI Configuration screen.

- Set **Advanced->System Acoustic and Performance Configuration->Set Fan Profile** as **Performance**
- Set **Advanced->System Acoustic and Performance Configuration->Set PWM Offset** as **80**

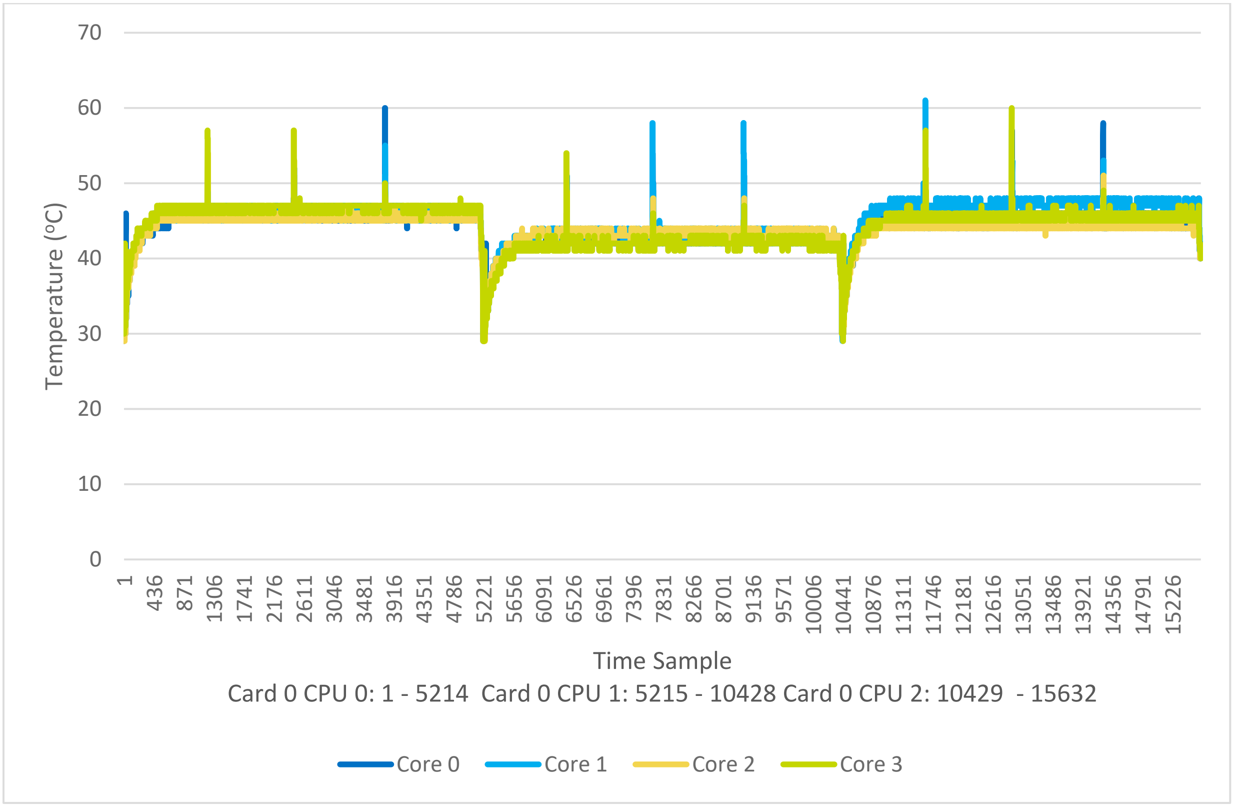


Figure 55. Temperature profile while transcoding

Appendix A. Creating a Custom OS Image

A.1. Overview

The Intel® VCA card provisions the OS to the card CPU over the PCIe* bus. The Intel® VCA's BIOS exposes a memory buffer to which the host copies the selected OS image. The following components are copied to each node when boot is requested:

- EFI bootloader
- Kernel image
- Initial RAM File System (InitRAMFS) image

InitRAMFS is responsible for early system configuration, mounting the main Root File System (RootFS) and root pivoting to this mounted RootFS. In the Intel® VCA environment, there are two strategies for provisioning RootFS:

- **Volatile OS file system** – The RootFS image is embedded in the InitRamFS image. InitRamFS is responsible for extracting the RootFS and mounting it onto a RAM disk.
- **Persistent OS file system** – The RootFS image is provisioned over NFS. InitRamFS establishes virtual network over PCIe*, mounts a remote NFS share, and switches to that remote root directory.

This chapter covers the process required to create a customized operating system image appropriate for booting the Intel® VCA nodes. Creating the custom images requires:

- Getting additional tools and build scripts.
- Setting up software archives or local repositories.
- Building images.

A.2. Getting Tools and Build Scripts

A.2.1. Ubuntu* Tools

To create a custom image, the following software packages must be added to a server installation of Ubuntu* 16.04:

```
autoconf bc bridge-utils build-essential checkinstall cmake debootstrap
   dracut fakeroot gdisk git kernel-wedge libboost-all-dev libc6-dev
   libc6:i386 libfdt-dev liblzma-dev libncurses5:i386 libpixman-1-dev
   libsdl1.2-dev libsdl1.2-dev libspice-server-dev libssl-dev libssl-dev
   libstdc++6:i386 libtool openssl python-dev socat tightvncserver uml-
   utilities uuid uuid-runtime vim x11vnc xtightvncviewer zip
[sudo] apt-get install <package_name> [<package_name> …]
```

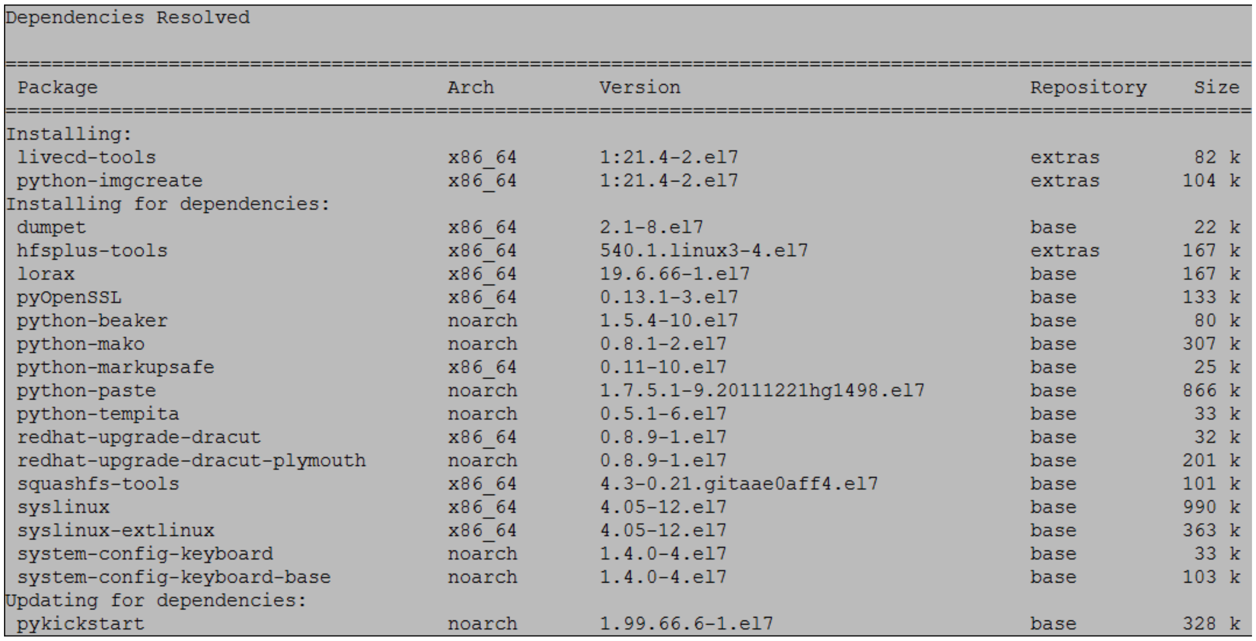
A.2.2. CentOS* Tools

To create a custom image, the following software packages must be added to the "Development and Creative Workstation" software environment of CentOS* 7.4 (1708):

```
acpica-tools asciidoc audit-libs-devel binutils-devel bison boost-devel
   bzip2-devel cmake createrepo dev86 discount e2fsprogs-devel edk2.git
   edk2.git-ovmf-x64 elfutils-devel fakeroot gdisk glibc-devel glibc-
   devel.i686 gnutls-devel gtk2-devel hfsplus-tools hmaccalc ipxe-roms-qemu
   java-1.8.0-openjdk-devel libaio-devel libcurl-devel libidn-devel libnl3-
   devel libuuid-devel libX11-devel livecd-tools ncurses-devel net-tools
   newt-devel nodejs numactl-devel openssl-devel pciutils-devel perl-
   ExtUtils-Embed perl-generators pesign python-devel python-imgcreate qemu-
```

```
    firmware-jenkins SDL-devel seabios-bin systemd-devel texi2html texinfo
    texlive-latex transfig xmlto yajl-devel zlib-devel
  [sudo] yum -y install <package_name> [<package_name> …]
```

Using `yum` for package installation automatically resolves and installs any dependencies needed for the tools to function properly as shown in Figure 56.



```
Dependencies Resolved

================================================================================================================
 Package                              Arch          Version                               Repository     Size
================================================================================================================
Installing:
 livecd-tools                         x86_64        1:21.4-2.el7                          extras         82 k
 python-imgcreate                     x86_64        1:21.4-2.el7                          extras        104 k
Installing for dependencies:
 dumpet                               x86_64        2.1-8.el7                             base           22 k
 hfsplus-tools                        x86_64        540.1.linux3-4.el7                    extras        167 k
 lorax                                x86_64        19.6.66-1.el7                         base          167 k
 pyOpenSSL                            x86_64        0.13.1-3.el7                          base          133 k
 python-beaker                        noarch        1.5.4-10.el7                          base           80 k
 python-mako                          noarch        0.8.1-2.el7                           base          307 k
 python-markupsafe                    x86_64        0.11-10.el7                           base           25 k
 python-paste                         noarch        1.7.5.1-9.20111221hg1498.el7          base          866 k
 python-tempita                       noarch        0.5.1-6.el7                           base           33 k
 redhat-upgrade-dracut                x86_64        0.8.9-1.el7                           base           32 k
 redhat-upgrade-dracut-plymouth       noarch        0.8.9-1.el7                           base          201 k
 squashfs-tools                       x86_64        4.3-0.21.gitaae0aff4.el7              base          101 k
 syslinux                             x86_64        4.05-12.el7                           base          990 k
 syslinux-extlinux                    x86_64        4.05-12.el7                           base          363 k
 system-config-keyboard               noarch        1.4.0-4.el7                           base           33 k
 system-config-keyboard-base          noarch        1.4.0-4.el7                           base          103 k
Updating for dependencies:
 pykickstart                          noarch        1.99.66.6-1.el7                       base          328 k
```

Figure 56. Example of dependencies resolved using `yum` for package installation

A.2.3 Windows Tools

The following hardware and software resources are required to create Windows images.

- Windows installation ISO (for example, `en_windows_10_enterprise_version_1703_updated_march_2017_x64_dvd_10189290.iso`).
- Windows Server 2012/2016 system to perform build. System should be at least as up-to-date as the installation ISOs used to create the images and all Microsoft* updates that are available should be installed.
- Oracle VM VirtualBox software (available at https://www.virtualbox.org/wiki/Downloads).
- Hyper V Server Role installed.
- Windows Assessment and Deployment Kit (Windows ADK) version 10 (available at https://developer.microsoft.com/en-us/windows/hardware/windows-assessment-deployment-kit).
- 45 GB of free space. (This is the default amount; the actual amount depends on container size.)
- Intel® graphics driver (for example, from https://downloadcenter.intel.com/product/88352/Intel-Iris-Pro-Graphics-P580).
- OpenSSH, NetKVM drivers, or Xen drivers if required by intended use.
- Powershell script for image creation (`vca_create_vhd_windows.ps1`).

A.2.4. Build Scripts

Build scripts may be downloaded from Intel's public website under the Intel® VCA downloads section. The file name is `build_scripts-<version>.tar.gz`. Copy this file to the host system, then uncompress it with the following command.

```
tar xf build_scripts-<version>.tar.gz
```

This command creates a directory matching the filename with the `.tar.gz` extensions. Change to this directory. Within the directory are the following scripts.

- `create_image.sh` – Main script, generates an image based on the provided set of parameters and configuration files.
- `generate_vca_production_image_<OS version>_<kernal version>.sh` – Wrapper script that invokes `create_image.sh` with parameters to generate a set of production images (currently, only a volatile bare metal image without Intel® Media Sever Studio).
- `generate_vca_reference_image_<OS version>_<kernel version>.sh` – Wrapper script that invokes `create_image.sh` with parameters to generate a set of reference images (currently, a volatile bare metal with Intel Media Server Studio, a persistent bare metal with Intel Media Server Studio, a XEN Dom0 without Intel Media Server Studio, and a Xen DomU with Intel Media Server Studio).
- `create_local_repo.sh` – Helper script, used to quickly generate repository containing Intel® VCA rpms.
- `dracut_module` – Directory containing internal scripts used in the creation of initramfs. Do not modify.
- `grub_cfgs` – Directory containing config files for the GRUB bootloader to be installed in image.
    - `grub_production_baremetal.cfg` – Config file for production volatile bare metal image (without Intel Media Server Studio).
    - `grub_production_kvm.cfg` – Config file for production volatile KVM ("Dom0") image (without Intel Media Server Studio).
    - `grub_reference_baremetal.cfg` – Config file for reference volatile bare metal image (with Intel Media Server Studio).
    - `grub_reference_dom0.cfg` – Config file for reference XEN Dom0 image (without Intel Media Server Studio).
    - `grub_reference_domu.cfg` – Config file for reference XEN DomU image (with Intel Media Server Studio).
    - `grub_reference_kvm.cfg` – Config file for production volatile KVM (Dom0) image (with Intel Media Server Studio).
    - `grub_reference_persistent.cfg` – Config file for reference persistent bare metal image (with Intel Media Server Studio).
    - `grub_vca_disk.cfg` – config file for persistent blockIO image
- `ks_files` – Directory containing kickstart configuration files, which contain all the configuration details for a complete installation (e.g., list of packages to be installed, admin password, post installation custom scripts, etc.).
    - `vv_production_baremetal.ks` – Kickstart file for production volatile bare metal image (without Intel Media Server Studio).
    - `vv_production_kvm.ks` – Kickstart file for production volatile KVM (Dom0) image (without Intel Media Server Studio).
    - `vv_reference_baremetal.ks` – Kickstart file for reference volatile bare metal image (with Intel Media Server Studio).
    - `vv_reference_dom0.ks` – Kickstart file for reference XEN Dom0 image (without Intel Media Server Studio).
    - `vv_reference_domu.ks` – Kickstart file for reference XEN DomU image (with Intel Media Server Studio).
    - `vv_reference_kvm.ks` – Kickstart file for production volatile KVM ("Dom0") image (with Intel Media Server Studio).
    - `vv_reference_persistent.ks` – Kickstart file for reference persistent bare metal image (with Intel Media Server Studio).
    - `mv_production_baremetal.ks` – Kickstart file for production volatile bare metal image (without Intel Media Server Studio).

  - `mv_production_kvm.ks` – Kickstart file for production volatile KVM (“Dom0”) image (without Intel Media Server Studio).
  - `mv_reference_baremetal.ks` – Kickstart file for reference volatile bare metal image (without Intel Media Server Studio).
  - `mv_reference_dom0.ks` – Kickstart file for reference XEN Dom0 image (withoutout Intel Media Server Studio).
  - `mv_reference_domu.ks` – Kickstart file for reference XEN DomU image (without Intel Media Server Studio).
  - `mv_reference_kvm.ks` – Kickstart file for production volatile KVM (“Dom0”) image (without Intel Media Server Studio).
  - `mv_reference_persistent.ks` – Kickstart file for reference persistent bare metal image (without Intel Media Server Studio).
  - `vca_disk_<kernel version>_reference.ks` – Kickstart file for reference persistent blockIO image.
  - `vca_disk_<kernel version>.ks` – Kickstart file for persistent blockIO image.
- `persistent_files` – Directory containing internal scripts used during preparation of persistent images. Do not modify.
- `windows` – Directory containing scripts used during preparation of Windows images.

A.2.4.1 Main Script (`create_image.sh`) Invocation

The script `create_image.sh` is the primary script used to build a custom image. Before invocation, the repositories (see Section REF _Ref512604889 \r \h 1.1A.3.2) must be created and made available to the host system. There are three levels of priority for parameters used by this script, which are, in order of precedence:

1. Parameters passed to the script on the command line. These override all lower priority parameters.
2. Parameters passed to the script as an environment variable. These override all lower priority parameters, but do not override command line parameters.
3. Parameters passed as defaults from within the script. These override no other parameters, but are present to allow the user to not be required to enter all the parameters on every invocation of the utility.

The `–image-type` parameter is required and does not have a default value, so attempting to invoke the script without this command-line provided parameter results in an error message.

Table 14 provides a list of all available parameters and their related environment variables, default values, and descriptions.

Table 14. `create_image.sh` parameters

Parameter	Environment Variable	Default Value	Description
`--image-type <type>`	`VCA_IMG_TYPE`	Undefined	Selects image creation strategy, which may be one of the following <types>: • `volatile-bm` • `persistent-bm` • `volatile-dom0` • `domu` This option must be provided on the command line or by a global variable.

Parameter	Environment Variable	Default Value	Description
`-h` `--help`	Undefined	Undefined	Prints help screen with options list.
`--kernel <version>`	`VCA_KERNEL_VERSION`	Automatically obtained from kernel in `vca_build` repo	Kernel version, used to generate a valid GRUB config; if not defined, automatically obtained from a kernel located in vca build repo.
`--version <version>`	`VCA_VERSION_NAME`	Undefined	Version appended to image name.
`--grub-cfg <filename>`	`VCA_GRUB_CFG`	One of the existing files from the `grub_cfgs` directory, selected based on value of `VCA_IMAGE_TYPE`	GRUB config file to be embedded in custom OS image.
`--ks-file`	`VCA_KS_FILE`	One of the existing files from the `ks_files` directory, selected based on value of `VCA_IMAGE_TYPE`	Kickstart file used to create OS image.
`--compressor <name>`	`VCA_DRACUT_COMPRESSOR`	Undefined	Optional external compression engine to be passed to `dracut` as `–compress`.
`--image-name <name>`	`VCA_ISO_IMG`	depending on `VCA_IMG_TYPE`: `vca_baremetal`, `vca_xen`, `vca_domu`, `vca_persistent`	Name of the output file.
`--out-dir <path>`	`VCA_OUT_DIR`	`./out`	Directory where created images are stored.
`--tmp-dir <path>`	`VCA_TMP_DIR`	`./tmp`	Directory where files temporarily required by the build process are stored.
`--os-repo <path>`	`VCA_OS_REPO`	`file:/usr/lib/vv_repos_ CentOS7.1/local_repo`	Repository containing the base CentOS* 7.2 files.
`--extras-repo <path>`	`VCA_EXTRAS_REPO`	`file:/usr/lib/vv_repos_ CentOS7.1/extras_repo`	Optional repository containing extra rpms (e.g., validation tools, utilities, applications, etc.).
`--build-repo <path>`	`VCA_BUILD_REPO`	`file:/usr/lib/vv_repos_ CentOS7.1/vv_repo`	Repository containing rpms from the build.

Note: If utilizing environment variables, but invoking the `create_image.sh` script via `sudo`, use the `–E` option of `sudo` (for example, `sudo –E ./create_image.sh <options>`) to properly pass the environment variables to the `sudo` instance.

A.3. Software Archives or Repositories

For purposes of automated image building, it is recommended that software archives or local repositories be used, as remote repositories:

- Increase build time (several hundred megabytes must be downloaded for each image from repositories; invocation of the reference build script would invoke this download five times),
- Lead to inconsistent package versions between image builds, as the external repositories may update at any time without notice, and
- Require a machine with consistent and unlimited access to Internet.

A.3.1. Ubuntu* Software Archives

The process of creating Ubuntu* images to be booted on the Intel® VCA card nodes is based on applying consecutive layers of software archives (SAs). An SA contains software in controlled, well-defined version to be installed. The SA also contains installation scripts used to customize the installation of the software.

A software archive can consist of software packages, apt-get dependencies database, custom files, information file, and custom pre-installation and post-installation scripts. The notion of SA has been introduced to guarantee that the dependencies database is up-to-date with the version of the included packages, and that software and the corresponding installation scripts are kept together.

The first SA to apply is called ***bootstrap*** and consists of the basic OS files necessary for a minimal booting Ubuntu* system. The next SA to apply is called ***base***. It extends this minimal OS with tools necessary for the VCA software installation. The next SA to apply is called ***vca*** and contains the VCA software packages. The other SAs are optional, and include:

- ***mss*** – for installing Intel MSS SDK software
- ***kvmgt*** – for installing Intel GVT-G enabled KVM software.

The software archives have dependencies, which means that they must be applied strictly in the order specified above. Software archives must also be created in the same order, in which they are applied.

Software archives are created with ***create_sw_archives.sh*** script. The script requires root privileges to run.

Table 15. `create_sw_archives.sh` parameters

Parameter	Example Values	Description

`-d, --descr <descrptn>`	`-d "UBUNTU 16.04 "`
	`--descr "UBUNTU 16.04.3 "`
		OS version description, for which the SA is prepared.

`-h` `--help`	`Undefined`	Prints help screen with options list.

`-i, --in-dir <dir>`	`--in-dir .`
	`--in-dir tmp/vca_sas`
	`-i /home/vca/sa`
		The input directory containing existing predecessor SAs which will be used during creation the requested SA.

`-k, --kernel <version>`	`--kernel 4.4.98-1.2.1.217.vca`
	`-k 4.4.0-1.2.1.182.vca`
		Kernel version, for which the SA is created, as returned by 'uname –r' in the final image after booting

`-m, --mss <dir>`	`--mss tmp/mss`
	`--mss /home/vca/mss`
	`-m .`
		Name of directory containing the MSS installation files. Only needed when requesting the creation of the mss SA, or any of its predecessors in the process of creation of a VCA bootable image.

`-o, --out-dir <dir>`	`--out-dir ./out`
	`--out_dir /tmp/out`
	`-o /tmp`
		The name of the destination directory for the SAs created. By default the current directory is assumed.

`-t, --target <name>`	`--target bootstrap`
	`--target vca`
	`-t kvmgt`
		The name of the SA to be created. Additionally, all the SAs will be created which follow the indicated SA in the process of creation of a VCA bootable image. Supported names:
		bootstrap, base, vca, mss, kvmgt.

`-v, --vca <dir>`	`--vca tmp/vca_pkgs`
	`--vca /home/vca/pkgs`
	`-v .`
		Name of directory containing the VCA *.deb files. Only needed when requesting the creation of the vca SA, or any of its predecessors in the process of creation of a VCA bootable image.

Usage examples:

To create the following SAs: bootstrap, base, vca, mss, and kvmgt in /tmp/b in one shot, i.e. not using any existing SAs:

```
create_sw_archives.sh -d "UBUNTU 16.04" -o /tmp/b -k 4.4.0-1.2.1.182.vca -t
   bootstrap -v /tmp/vca_pkgs_dir -m /tmp/mss_instal_dir
```

To create in /tmp/o the base SA (using existing bootstrap SA from /tmp/b), and thus to create vca, mss, and kvmgt SAa as well:

```
create_sw_archives.sh -d "UBUNTU 16.04" -o /tmp/o -k 4.4.98-1.2.1.217.vca -t
   base -v /tmp/v -m /tmp/m/custom -i /tmp/b
```

To create only the kvmgt SA in /tmp/o , using existing bootstrap, base, and vca SAs from /tmp/b:

```
create_sw_archives.sh -d "UBUNTU 16.04" -o /tmp/o -k 4.4.98-1.2.1.217.vca -t
   kvmgt -v /tmp/v -m /tmp/m/custom -i /tmp/b
```

A.3.2. CentOS* Repositories

This chapter assumes the location of the repositories (as noted in the default column of Table 14) is:

```
/usr/lib/vv_repos_CentOS7.x/
```

where x is the supported version of CentOS*.

If a different directory structure is used (e.g., `/var/www/html/repos`), substitute that directory throughout the following instructions.

The current kickstart files use the following local repositories:

Table 16. Local repositories

Name	Location	Description
`vca_os_repo`	`/usr/lib/vca/repos/CentOS7.x/os_repo`	Local version of CentOS* repository, with official packages only.
`vca_extras`	`/usr/lib/vca/repos/CentOS7.x/extras_repo`	Additional packages, not from CentOS* repository, not created by development team or changing very rarely.
`vca`	`/usr/lib/vca/repos/build_repo`	Often (each build) changing packages, delivered by development team.

Such repositories may be physically located on a local hard disk or be a link to an NFS repository at some other location.

A.3.2.1 Creating a Local `vca_os_repo` Repository

If the directions in section B.4 of this document for setting up a local yum repository have been followed, please run the following commands to create the required directory structure and mount the DVD ISO to the `local_repo` directory:

```
sudo mkdir -p \
   /usr/lib/vca/repos/CentOS7.x/{local_repo,extras_repo,vca_build_repo}
sudo mount --bind /var/www/html/centos7.x \
   /usr/lib/ vca/repos/CentOS7.x/os_repo
```

If the directions in section B.4 have not yet been executed, the following instructions may be followed to create a local repository:

1. Download the "Everything" CentOS* 7.x ISO, `CentOS-7-x86_64-Everything-yyyy.iso`. where yyyy is based on x version. This can be found on a mirror download site listed at: http://vault.centos.org/notonvault.html.
2. Run the following command to create the repository directories:

```
sudo mkdir -p \
/usr/lib/vca/repos/CentOS7.x/{local_repo,extras_repo,vca_build_repo}
```

3. Run the following command to mount the ISO as the OS repository:

```
sudo mount CentOS-7-x86_64_Everything-yyyy-01.iso \
/usr/lib/vca/repos/CentOS7.x/os_repo
```

If using a non-default directory location, the environment variable `VCA_OS_REPO`, or the command-line parameter `--os-repo` must be set to the correct location. For example,

- `file:/var/www/html/cent7.x/base` for a local file path, or
- `http://localhost/cent7.x/base` for a web address.

Note: This OS image creation process supports only the pre-build addition of packages in .rpm format. It does not support automatic inclusion of packages in any other format, nor of "tarballs" or similar archives.

A.3.2.2 Creating a Local `vca_build_repo` Repository

This repository contains essential Intel® VCA packages, changing with almost every build. Therefore the repository needs to be updated very often. It contains at a minimum:

- Intel® VCA kernel (`kernel` RPM file)
- Intel® VCA software stack (`vcass` RPM file)

Please see the `PackageList.txt` file available at the Intel® VCA download page to see which rpms are used in each type of image build; rpm files other than the two listed above should be placed in the `extras_repo` repository, discussed later in this document. Also note that there are two distinct kernel packages (reference and production), and only one type of kernel shall be placed in the repositories before attempting the build of the custom OS image.

If the directory already exists and contains some older packages, remove them all.

```
sudo rm -rf /usr/lib/vca/repos/build_repo/*
```

Copy all desired rpm files (packages) to the directory `/usr/lib/vca/repos/build_repo`. Once all desired rpm files are in the directory, create the repository with the following command:

```
sudo createrepo /usr/lib/vca/repos/build_repo
```

If using a non-default directory location, the environment variable `VCA_BUILD_REPO`, or the command-line parameter `--build-repo` must be set to the correct location. For example,

- `file:/var/www/html/cent7.x/vca` for a local file path, or
- `http://localhost/cent7.x/vca` for a web address.

Automated Creation of the `vca_build_repo`
The `build_scripts` package includes a script `create_local_repo.sh` that may be used to automatically create this repository.

Note: This script deletes all files from the `/usr/lib/vca/repos/build_repo` directory to avoid accidental presence of multiple kernel versions. Use this script with caution. Additionally, this script currently does not support command line parameter `--build-repo`, so if it is desired that a custom location for this repository be used, the environment variable `VCA_BUILD_REPO` must be set.

The following are example uses of the script.

Example 1:

```
[sudo] ./create_local_repo.sh -r <package_directory>
```

Example 2:

```
[sudo] ./create_local_repo.sh -r <package_diretory> \
   -r <binaries_directory> -r <xen_directory>
```

A.3.2.3 Creating a Local Extras Repository

The following packages are copied to a final repository named `extras_repo` and is intended for packages with a slow change cycle (including packages produced by both Intel and third parties). Sample packages for the repo may be:

- Validation tools:
  - `iperf`
  - `iperf3`
  - `fio`
- Intel® Media SDK (required for reference images.):
  - `intel-linux-media`
  - `intel-opencl-1.2`
  - `intel-opencl-1.2-devel`
- Xen packages:
  - `xen`
  - `xen-hypervisor`
  - `xen-runtime`

Copy all desired `.rpm` files (packages) to the directory:

```
/usr/lib/vca/repos/CentOS7.x/extras_repo
```

Once all desired `.rpm` files are in the directory, create the repository with the following command:

```
sudo createrepo/usr/lib/vca/repos/CentOS7.x/extras_repo
```

If using a non-default directory location, the environment variable `VCA_EXTRAS_REPO`, or the command-line parameter `--extras-repo` must be set to the correct location. For example,

- `file:/var/www/html/cent7.x/vca_extras` for a local file path, or
- `http://localhost/cent7.x/vca_extras` for a web address.

Once the packages are correctly stored in the repository directory, and the `createrepo` command has successfully created the required repository indexes, any custom packages added to the `extras_repo` repository must be added to the appropriate kickstart file, under the `%packages` header. Modify each of each relevant kickstart files as follows:

1. Open the kickstart file (in the `ks_files` directory) for editing.
2. Locate the line that begins with `%packages`
3. And add a line below that line with the name of the package to be installed from the `extras_repo`. (Provide only one package name per line.)
4. Save the file, exit the editor, and modify any additional kickstart config files, as needed.

Skipping the Local Extras Repository
If there is no requirement for extra packages to be installed, the `extras_repo` can be commented out of each relevant kickstart files as follows:

1. Open the kickstart file (in the `ks_files` directory) for editing.
2. Locate the line that begins with `repo-name=extras_repo`.
3. Insert a hash character (`#`) as the first character of the line, so it begins instead with `#repo-name=extras_repo`.
4. Save the file, exit the editor, and modify any additional kickstart config files, as needed.

A.4. Building Images

To create an OS image, the script `generate_vv_image.sh` must be called with a proper set of parameters. The only mandatory parameter is `--image-type <type>`. All other parameters may remain at default values. However, defaults for volatile bare metal are configured for a production set of RPMs; for all other image types, defaults are configured for reference sets of RPMs. So for full control over the process, it is advised to provide at least:

- kickstart file to be used (e.g., `--ks-file ks_files/vv_reference_baremetal.ks`)
- grub config file to be used (e.g., `--grub-cfg grub_cfgs/grub_reference_baremetal.cfg`)

A.4.1 Building a Single Image

The following sections cover the specific command line required to build each of the single images supported by the `generate_vv_image.sh` script. Each section contains a single command to be typed on a single line. It assumes that the repositories have been correctly set up, that the current directory is the `build_scripts-<version>` directory, and that the user has `sudo` privileges if not root.

- Creating Production Volatile Baremetal Image

```
[sudo -E] ./create_image.sh --image-type volatile-bm --grub-cfg
  grub_cfgs/grub_production_baremetal.cfg --ks-file
  ks_files/<vv,mv>_production_baremetal.ks
```

- Creating Production Volatile KVM (DOM0) Image

```
[sudo -E] ./create_image.sh --image-type volatile-bm --grub-cfg
  grub_cfgs/grub_production_kvm.cfg --ks-file
  ks_files/<vv,mv>_production_kvm.ks
```

- Creating Production vca-disk Over BlockIO Image

```
[sudo -E] ./create_image.sh --add-rootfs --image-type vca-disk --grub-cfg
  grub_cfgs/grub_vca_disk.cfg --ks-file ks_files/vv_production_vca_disk.ks
```

- Creating Reference Volatile Baremetal Image

```
[sudo -E] ./create_image.sh --image-type volatile-bm --grub-cfg
  grub_cfgs/grub_reference_baremetal.cfg --ks-file
  ks_files/<vv,mv>_reference_baremetal.ks
```

- Creating Reference Persistent Baremetal Image

```
[sudo -E] ./create_image.sh --image-type persistent-bm --grub-cfg
  grub_cfgs/grub_reference_persistent.cfg --ks-file
  ks_files/<vv,mv>_reference_persistent.ks
```

- Creating Reference Xen Dom0 Volatile Image

```
[sudo -E] ./create_image.sh --image-type volatile-dom0 --grub-cfg
  grub_cfgs/grub_reference_dom0.cfg --ks-file
  ks_files/<vv,mv>_reference_dom0.ks
```

- Creating Reference Xen DomU Persistent Image

```
[sudo -E] ./create_image.sh --image-type domu --grub-cfg
  grub_cfgs/grub_reference_domu.cfg --ks-file
  ks_files/<vv,mv>_reference_domu.ks
```

- Creating Reference vca-disk Over BlockIO Image

```
[sudo -E] ./create_image.sh --add-rootfs --image-type vca-disk --grub-cfg
   grub_cfgs/grub_vca_disk.cfg --ks-file ks_files/mv_reference_vca_disk.ks
```

- Creating Reference volatile image on Ubuntu*

  Example of calling the scripts to build an Ubuntu* volatile image with Intel Media Server Studio:

```
[sudo -E] ./create_image.sh
   --bootstrap deb.tar --archive base.tar --archive vca.tar --archive
   mss.tar
   --image-type volatile-bm
   --grub-cfg grub_cfgs/grub_baremetal.cfg
   --image-name "vca_baremetal_k4.4.98_ubuntu16.04.3_2.1.3.img
   --out-dir ./out
   --descr "UBUNTU 16.04.3"
   --kernel 4.4.98-1.2.1.3.vca
```

A.4.2 Building a Complete Set of Production or Reference Images

There are two wrapper scripts – `generate_vv_production_image.sh`/`generate_mv_production_image.sh` and `generate_vv_reference_image.sh`/`generate_mv_reference_image.sh` – that may be used to create a complete set of images by a single script invocation. Each script accepts only a single optional parameter: the build version to append to the image names.

Note: Each script, when invoked, removes all files from the `./out` (or user-specified output) directory; do not invoke this script before verifying desired built images have been moved from this location to a safe location.

To use a custom repository location (or custom set of repository locations), define the appropriate environment variables `VCA_OS_REPO`, `VCA_EXTRAS_REPO`, and `VCA_BUILD_REPO`.

A.4.2.1 Creating production images

Currently, there is only a single production image: the `production_volatile_baremetal` image. If future releases of the software support additional production images, the build scripts released with that version of the software will allow for the automated building of those additional production images.

```
[sudo -E] ./generate_vca_production_image.sh [version]
```

A.4.2.2 Creating reference images

Currently, the reference image set is comprised of the following images:

- Volatile baremetal
- Persistent baremetal
- Volatile Dom0
- Volatile DomU

It requires (at least) the packages: mrouted, the rpm files from the latest version of Intel Media Server Studio SDK, and the rpm files from Xen Intel® VCA package.

```
[sudo -E] ./generate_vca_reference_image.sh [version]
```

A.4.2.3 Creating Windows* Baremetal Images

Windows baremetal images are dedicated for BlockIO/KVM or Xen depending on the `boot_part_type` parameter set. GUID Partition Table (GPT) can be used for BlockIO or KVM, Master Boot Record (MBR) for KVM or Xen. For instructions how to use this image, refer to Sections 4.5.5.2 and 4.5.6.1.

Image Creation Script Parameters

- **`-iso`** – Path to Windows installation iso file.
- **`-output_dir`** – Directory which will contain compressed resulting img for VCA (if not specified, default directory name is out).
- **`-mount_dir`** – Directory to which wim file from windows installation iso will be mounted during image creation (if not specified, default directory name is `mount_dir`).
- **`-tmp_dir`** – Directory used for temporary files (if not specified, default directory name is `tmp`).
- **`-driver_dir`** – Directory containing VCA drivers (available with build release). This directory needs to contain the `Win8.1Release-x64` folder, as the folder name is hardcoded.
- **`-vcagent_dir`** – Directory containing VCAgent (available with build release). This directory should contain `VCAgent.*` files.
- **`-answer_file`** – Path to Windows answerfile (same as unattend file).
- **`-dism_path`** – Path to dism file (from Windows ADK).
- **`-vca_image_version`** – Added to resulting file name (default is 0.0.0).
- **`-gfx_drv_dir`** – Path to directory containing Intel graphics driver for Intel Iris.
- **`-win_edition`** – Windows edition to be incorporated into img file (for example ServerStandard or Enterprise).
- **`-netkvm_drv_dir`** – If image is going to be used on KVM, it specifies path to NetKVM driver directory (optional).
- **`-xen_driver_dir`** – If image is going to be used on Xen, it specifies path to Xen driver directory (containing XenBus, XenNet, and Xenvif) (optional).
- **`-openssh_dir`** – Path to OpenSSH installation files directory, if it is to be installed (optional). Hardcoded in `AutoUnattend.xml` file for `setupssh-7.4p1-1.exe`.
- **`-virtualbox`** –Path to `VBoxManage.exe` file.
- -**`vhd_size`** – Resulting image size in GB. Recommended minimum size is 11 GB.
- **`-boot_part_type`** – Type of created image (GPT for BlockIO or KVM and MBR for KVM/Xen).
- **`-zip_img`** – Name of zip file containing resulting Windows image file (optional). Default is `vca_windows_baremetal.zip`.

Installation of Xen Drivers During Image Creation

If there is such requirement, Xen drivers can be added to the Windows image file. However, the installation process can only be run during the first boot of Windows, through the use of answerfile (`AutoUnattend.xml`). There is no available option to silently install those drivers and, as such, installation requires approval of the Xen driver certificate. To overcome this obstacle, answerfile assumes that, in the directory in which the Xen drivers reside, there is a Xen driver certificate (hardcoded as `xen_certificate.cer`) that can be automatically installed and approved; the drivers then install silently, without user interaction. Said certificate can be obtained through installation of any of three required Xen drivers (`xenbus`, `xennet`, or `xenvif`) and then copied from the Microsoft Management Console into a `.cer` file. The Xen driver directory should look like this:

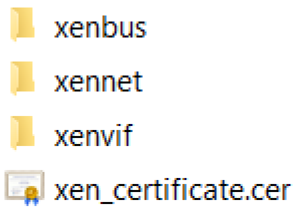


Figure 57. Xen driver directory with driver certificate file

Creation of BlockIO/KVM Windows Images

Once all required software and files are present, image creation is quite straightforward. The script has to be executed with parameters specified above. It results in a zipped `.img` file created in the output directory. This file can be booted (depending on parameters provided) on BlockIO/KVM/Xen.

Directories `tmp_dir`, `out_dir`, and `mount_dir` can be specified on a different drives (SSD or M.2) to speed up image building process. If not defined, they are created in the same directory in which the PowerShell script is executed.

An example use of the image building script is below:

```
.\vca_create_windows_image.ps1 -iso
   .\en_windows_server_2016_x64_dvd_9718492.iso -driver_dir
   .\VcaKmdWin\Win8.1Release-x64 -vcagent_dir .\VCAgent -answer_file
   .\answer_files\WS2016\AutoUnattend.xml -gfx_drv_dir
   .\GFX_driver_20.19.15.4549_BDW_SKL -dism_path 'C:\Program Files
   (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment
   Tools\x86\DISM\dism.exe' -win_edition ServerStandard -virtualbox
   'C:\Program Files\Oracle\VirtualBox\VBoxManage.exe' -vhd_size 11GB -
   boot_part_type GPT  -xen_driver_dir .\Xen  -openssh_dir .\OpenSSH -
   netkvm_drv_dir .\NetKVM\2k16 -tmp_dir d:\tmp -mount_dir d:\mount_dir  -
   zip_img vca_windows_server_2016_baremetal_MBR.zip
```

Windows Image Resizing

If required, images or containers can be resized simply by running following command:

```
dd if=/dev/zero bs=1M count=4000 >> ./binary.img
```

The `count` parameter depends on the amount of space to be added. After resizing the file, the image should be booted and the disk resized normally in Windows disk manager using unallocated space.

Appendix B. CentOS Tips and Tricks*

This section goes through several useful CentOS* configurations to assist the user in setting up optional components on the Intel® VCA. Scripts have been written below to save time. These scripts are not supported nor guaranteed to work and may stop working on future versions. Intel will make a best effort attempt to keep this section up to date.

B.1. Installing CentOS*

1. Connect CentOS* (Required version) install media to server.
2. Boot server. If necessary, select install media as boot device.
3. At the CentOS* 7 boot menu, select **Install CentOS 7** (up-arrow key, or press **<I>**), then press **<Enter>** to load the installer, which takes a couple of moments.
4. At the WELCOME TO CENTOS 7 menu, select the language (and dialect, if appropriate), then click the **Continue** button in the lower right corner of the screen.
5. At the INSTALLATION SUMMARY menu, click **INSTALLATION DESTINATION**.
6. At the INSTALLATION DESTINATION screen:
   - Below Device Selection, click the <size>240GB (e.g. 240GB) **host OS drive**.
   - Below Other Storage Options > Partitioning, select **I will configure partitioning**.
   - Click the **Done** button in the upper left corner of the screen.
7. At the MANUAL PARTITIONING screen:
   - Click the **Click here to create them automatically** link.
   - Click the **/home** partition at the top of the left side.
   - Click the **–** button at the bottom of the left side.
   - Click the **/** partition.
   - On the right side, delete the contents of the **Desired Capacity** text box.
   - Click the **Update Settings** button in the middle of the right side of the screen.
   - Click the **Done** button in the upper left corner of the screen.
   - At the SUMMARY OF CHANGES dialog box, click the **Accept Changes** button in the lower right corner of the dialog box.
8. At the INSTALLATION SUMMARY screen, click **Software Selection**.
   - From the left pane, Base Environment, select **Development and Creative Workstation**.
   - From the right pane, Add-Ons for Selected Environment, select **File and Storage Server**, and **Development Tools**.
   - Click the **Done** button in the upper left corner of the screen.
9. At the INSTALLATION SUMMARY screen, click **NETWORK & HOSTNAME**.
   - Select the connected network device (e.g., "Ethernet (enp3s0f0)"). Note that disconnected network devices are marked as unplugged.
   - Click the slider button so that it is **On**.
   - Enter a system name in the **Hostname** text box in the lower left corner of the screen.
   - Click the **Done** button in the upper left corner of the screen
10. At the INSTALLATION SUMMARY screen, click the **Begin Installation** button in the lower right corner of the screen.
11. At the CONFIGURATION screen, click **ROOT PASSWORD**.
    - Enter a secure password in both the **Root Password** and **Confirm** text boxes. (Note that the pre-loaded test image uses the insecure password `vista1`.)
    - Click the **Done** button in the upper left corner of the screen. (Note that insecure passwords require the **Done** button be clicked twice.)
12. At the CONFIGURATION screen, click **USER CREATION**.
    - Enter the user's **Full name** and **Username** in the appropriate text boxes.

- If the user should be configured with sudo access at creation, check the **Make this user administrator** checkbox.
- Enter a secure password in both the **Password** and **Confirm password** text boxes. (Note that the pre-loaded test image uses the username `vista` and the insecure password `vista1` and the user is configured for sudo use.)
- Click the **Done** button in the upper left corner of the screen. (Note that insecure passwords require the **Done** button be clicked twice.)

13. Once the installation completes, a **Reboot** button is displayed in the lower right corner of the screen. When that is displayed, click the **Reboot** button, then disconnect the installation media from the server when the POST screen is displayed.
14. When the OS loads, an INITIAL SETUP screen is displayed.
15. At the INITIAL SETUP screen, click **LICENSE INFORMATION** (below **LOCALIZATION**).
    - At the License Agreement screen, click the checkbox next to **I accept the license agreement**.
    - Click the **Done** button in the upper left corner of the screen.
16. At the INITIAL SETUP screen, click the **FINISH CONFIGURATION** button in the lower right corner of the screen.
17. At the Kdump screen, click the **Forward** button in the lower right corner of the screen.
18. Setup network proxy if it is required.
    a. Add following to the end of `/etc/profile`.

```
export http_proxy=http://serveraddrs:port
export https_proxy=https://serveraddrs:port
export ftp_proxy=ftp://servraddrs:port
export socks_proxy=http://servraddrs:port
export HTTP_PROXY=http://servraddrs:port
export HTTPS_PROXY=https://servraddrs:port
export FTP_PROXY=ftp://servraddrs:port
export SOCKS_PROXY=http://servraddrs:port
```

    b. Run the following command to set the proxy.

```
source /etc/profile
```

    c. Edit `/etc/yum.conf` with proxy settings.

```
proxy=http://proxy_address:port/
```

    d. Edit `/etc/wgetrc` with proxy settings.

```
http_proxy=http://serveraddrs:port
https_proxy=https://serveraddrs:port
ftp_proxy=ftp://servraddrs:port
```

B.2. SSH Passwordless Configuration

First, create the SSH password-less files by issuing the following commands once for each host.

```
cd
sudo ssh-keygen
```

Press **<Enter>** at each prompt until the system returns to the command prompt.

```
sudo cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys
sudo chmod 600 /root/.ssh/authorized_keys
sudo cp -af /root/.ssh ~/
sudo chown -R `whoami`:`whoami` ~/.ssh
```

Next, copy the `.ssh` directory to each Intel® VCA node. Complete this process each time volatile nodes boot if not using a persistent filesystem through NFS. At the **Are you sure you want to continue connecting** prompt, type `yes` and then press **<Enter>**. At the password prompt, enter the password `vista1`. (This is the password for the root user on the Intel® VCA node). The following command copies the directory to all three nodes. Enter the password at each prompt.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do scp -r .ssh
   root@172.31.${i}.1:/root/ ; done
```

Note: The quote before `ip enumeration` and after `-l` is a black tick. On an en_US keyboard, it is the key left of **<1>** on the number row (an unshifted ~).

The following steps complete the same function without using a “for” loop. Run each line as appropriate for the number of nodes in the system.

```
scp -r .ssh root@172.31.1.1:/root/
scp -r .ssh root@172.31.2.1:/root/
scp -r .ssh root@172.31.3.1:/root/
scp -r .ssh root@172.31.4.1:/root/
scp -r .ssh root@172.31.5.1:/root/
scp -r .ssh root@172.31.6.1:/root/
```

B.3. Setting Up Hostnames

To set the hostnames, issue the following command for each Intel® VCA node.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
   root@172.31.${i}.1 "hostname node${i}.localdomain ; echo
   node${i}.localdomain > /etc/hostname" ; done
```

The following steps complete the same function without using a “for” loop. Run each line as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.2.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.3.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.4.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.5.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.6.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
```

B.4. Creating a Local `yum` Repository

1. Copy the CentOS* 7 ISO to the system.
2. Issue the following command to mount the ISO to `/media`. Change the name `CentOS-7.iso` to the ISO filename on the system.

   ```
   sudo mount -o loop CentOS-7.iso /media
   ```

3. Issue the following commands to install the web server. If prompted for password, enter host password setup as user. Start the service and open the firewall.

   ```
   ssh root@localhost 'echo [localbase] > /etc/yum.repos.d/base.repo'
   ssh root@localhost 'echo name=localbase >> /etc/yum.repos.d/base.repo'
   ssh root@localhost 'echo baseurl=file:///media >>
      /etc/yum.repos.d/base.repo'
   ssh root@localhost 'echo gpgcheck=0 >> /etc/yum.repos.d/base.repo'
   sudo yum -y install --disablerepo=* --enablerepo=localbase httpd
   sudo systemctl enable httpd
   sudo systemctl start httpd
   for i in public external dmz work home internal trusted; do sudo firewall-
      cmd --zone=$i --add-service=http --permanent ; done
   sudo firewall-cmd --reload
   ```

4. Issue the following commands to remount the ISO to the web server file structure, and mount it on reboots. Note that the last command in this step currently causes the system to fail boot; please see the next step for workaround.

   ```
   sudo umount /media
   sudo mkdir /var/www/html/centos70
   sudo mount -o loop CentOS-7.iso /var/www/html/centos70
   ssh root@localhost "echo mount -o loop /home/vista/vvgold/CentOS-7.iso
      /var/www/html/centos70" >> /etc/rc.local
   ```

5. Modify the local repo file to point to the new location under `/var/www/html/centos70`.

   ```
   sudo sed -i -e s=media=var/www/html/centos70=   /etc/yum.repos.d/base.repo
   ```

6. Make sure any repos previously loaded on the nodes are moved somewhere safe.

   ```
   for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
      root@172.31.${i}.1 "mv /etc/yum.repos.d/* /home" ; done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   ssh root@172.31.1.1 "mv /etc/yum.repos.d/* /home"
   ssh root@172.31.2.1 "mv /etc/yum.repos.d/* /home"
   ssh root@172.31.3.1 "mv /etc/yum.repos.d/* /home"
   ssh root@172.31.4.1 "mv /etc/yum.repos.d/* /home"
   ssh root@172.31.5.1 "mv /etc/yum.repos.d/* /home"
   ssh root@172.31.6.1 "mv /etc/yum.repos.d/* /home"
   ```

7. Create a repository file to copy to the Intel® VCA nodes.

   ```
   echo [host] > vv.repo
   echo name=host >> vv.repo
   echo baseurl=http://172.31.1.254/centos70/ >> vv.repo
   echo gpgcheck=0 >> vv.repo
   ```

8. Copy it to the nodes and fix up nodes to point to the right IP address.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do scp vv.repo
   root@172.31.${i}.1:/etc/yum.repos.d/ ; done
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
   root@172.31.${i}.1 "sed -i -e s/1.1.254/1.${i}.254/
   /etc/yum.repos.d/vv.repo" ; done
```

The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

```
scp vv.repo root@172.31.1.1:/etc/yum.repos.d/
ssh root@172.31.1.1 "sed -i -e s/1.1.254/1.1.254/ /etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.2.1:/etc/yum.repos.d/
ssh root@172.31.2.1 "sed -i -e s/1.1.254/1.2.254/ /etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.3.1:/etc/yum.repos.d/
ssh root@172.31.3.1 "sed -i -e s/1.1.254/1.3.254/ /etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.4.1:/etc/yum.repos.d/
ssh root@172.31.4.1 "sed -i -e s/1.1.254/1.4.254/ /etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.5.1:/etc/yum.repos.d/
ssh root@172.31.5.1 "sed -i -e s/1.1.254/1.5.254/ /etc/yum.repos.d/vv.repo"
scp vv.repo root@172.31.6.1:/etc/yum.repos.d/
ssh root@172.31.6.1 "sed -i -e s/1.1.254/1.6.254/ /etc/yum.repos.d/vv.repo"
```

9. (Optional) Verify each node is able to see the `yum` repository.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
   root@172.31.${i}.1 "yum repolist" ; done
```

The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "yum repolist"
ssh root@172.31.2.1 "yum repolist"
ssh root@172.31.3.1 "yum repolist"
ssh root@172.31.4.1 "yum repolist"
ssh root@172.31.5.1 "yum repolist"
ssh root@172.31.6.1 "yum repolist"
```

B.5. Creating NFS Share

1. Install NFS on the host system.

```
sudo yum -y install --disablerepo=* --enablerepo=localbase nfs-utils
```

2. Open up the firewall to allow NFS access.

```
for i in public external dmz work home internal trusted; do sudo firewall-
   cmd --zone=$i --add-service=nfs --permanent ; done

       sudo firewall-cmd --reload
```

3. Share a directory in `/etc/exports`. This example uses `/tmp`; replace with selected share directory in these steps.

```
ssh root@localhost "echo '/tmp     *(no_root_squash,rw)' >> /etc/exports"
```

4. Reset NFS service (fully stop and disable, then enable and start) to avoid possible issues.

```
sudo systemctl stop nfs-server
sudo systemctl disable nfs-server
sudo systemctl enable rpcbind
sudo systemctl enable nfs-server
sudo systemctl start rpcbind
sudo systemctl start nfs-server
```

Note that restarting NSF using presented commands or by:

```
service nfs stop
service nfs restart
```

can hang booted Dom0 DomU images. It is recommended to prepare NFS before booting the OS or to use a safer command:

```
exportsfs --a
```

5. Install NFS on the Intel® VCA nodes.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
   root@172.31.${i}.1 "yum -y install nfs-utils" ; done
```

The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "yum -y install nfs-utils"
ssh root@172.31.2.1 "yum -y install nfs-utils"
ssh root@172.31.3.1 "yum -y install nfs-utils"
ssh root@172.31.4.1 "yum -y install nfs-utils"
ssh root@172.31.5.1 "yum -y install nfs-utils"
ssh root@172.31.6.1 "yum -y install nfs-utils"
```

6. Mount the share on the Intel® VCA nodes. This command mounts to the `/media` directory on the nodes; change `/tmp` in the command to the directory shared in step 4, and `/media` in the command to the directory to which the NFS share should be mounted.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
   root@172.31.${i}.1 "mount 172.31.${i}.254:/tmp  /media" ; done
```

The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "mount 172.31.1.254:/tmp  /media"
ssh root@172.31.2.1 "mount 172.31.2.254:/tmp  /media"
ssh root@172.31.3.1 "mount 172.31.3.254:/tmp  /media"
ssh root@172.31.4.1 "mount 172.31.4.254:/tmp  /media"
ssh root@172.31.5.1 "mount 172.31.5.254:/tmp  /media"
ssh root@172.31.6.1 "mount 172.31.6.254:/tmp  /media"
```

B.6. Setting System Time

1. Set host system time zone. . To find the correct time zone name, run the command timedatectl list-timezones. As an example, for the Pacific Time Zone, set to `America/Los_Angeles`.

   ```
   sudo timedatectl set-timezone America/Los_Angeles
   ```

2. Set host system time. Replace `MM` with the current two-digit month; `DD` with the current two-digit date; `HH` with the current two-digit, 24-hour format hour; `MM` with the current two-digit minute; and `YYYY` with the current four-digit year. For example, `032316522015` sets the time to 4:52 pm on 23 March 2015.

   ```
   sudo date MMDDHHMMYYYY
   sudo hwclock --systohc
   ```

3. Set node time zones

   ```
   for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )); do ssh
      root@172.31.${i}.1 "timedatectl set-timezone America/Los_Angeles" ; done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   ssh root@172.31.1.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.2.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.3.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.4.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.5.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.6.1 "timedatectl set-timezone America/Los_Angeles"
   ```

4. Get the current host system time and push it to the nodes.

   ```
   NOW=`date +%m%d%H%M%Y` ; for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++
      )) ; do ssh root@172.31.${i}.1 "date $NOW" & done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.1.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.2.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.3.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.4.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.5.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.6.1 "date $NOW"
   ```

B.7. Configuring Network Time Protocol (NTP)

1. Install NTP package on host system. Each command starts with `sudo` or `ssh`; enter on a single line.

   ```
   sudo yum -y install --disablerepo=* --enablerepo=localbase ntp
   ```

2. Configure NTP server on host system. Each command starts with `sudo` or `ssh`; enter on a single line.

   ```
   sudo sed -i -e 's/^server/#server/' /etc/ntp.conf
   ssh root@localhost 'echo "server 127.0.0.1" >> /etc/ntp.conf'
   ssh root@localhost 'echo "restrict 172.31.0.0 mask 255.255.0.0 nomodify
      notrap" >> /etc/ntp.conf'
   sudo systemctl restart ntpd
   sudo systemctl enable ntpd
   ```

3. Install and configure NTP package on Intel® VCA nodes.

   ```
   for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
      root@172.31.${i}.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.${i}.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd" ; done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   ssh root@172.31.1.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.1.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.2.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.2.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.3.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.3.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.4.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.4.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.5.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.5.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.6.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.6.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ```

B.8. Turn On Affinity

Affinity setting can be changed on the host to balance simultaneous transfers between the host and all nodes. It is recommended to turn it on.

1. Open the file `/etc/sysconfig/irqbalance`.
2. Find the line:

   ```
   #IRQBALANCE_ARGS=
   ```

3. Change this line to (do not forget to remove `#`):

   ```
   IRQBALANCE_ARGS="-h exact"
   ```

4. Save the file and reboot the host.

B.9. Host BIOS Additional Configuration

It is recommended to set BIOS parameters (if available) to following settings:

- Memory Mapped IO above 4GB: Enabled
- Memory Mapped IO size: 256GB

B.10. Configuring Microsoft Windows* Client Access to VNCViewer

Before performing the following steps, install Xming (http://sourceforge.net/projects/xming/) and PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) on the Windows client.

1. Launch XMing.
2. Launch PuTTY.
3. Enter the IP address of the Intel® VCA host system in the **Host Name (or IP address)** text box.
4. Enter a name for the system in the **Saved Sessions** text box.
5. In the left-side navigation pane, open **Expand Connection** -> **SSH**, then click **X11** below **SSH**.
6. In the right-side settings pane, check the **Enable X11 forwarding** checkbox, then enter `:0` in the **X display location** text box.
7. In the left-side navigation pane, click **Window**.
8. In the right-side settings pane, enter `10000` in the **Lines of scrollback** text box (replacing whatever value is there; standard default is 200).
9. In the left-side navigation pane, scroll to the top, and click **Session**.
10. In the right-side settings pane, click the **Save** button next to the **Saved Sessions** list.
11. At any future point, the connection to the Intel® VCA system can be opened by launching PuTTY then either:
    a. Double-clicking the name from the **Saved Sessions** list, or
    b. Clicking the name from the **Saved Sessions** list, then clicking **Load**, and then clicking **Open**.

B.11. Install and Configure VNC

1. Install and configure the VNC package on Intel® VCA nodes.

   ```
   for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
      root@172.31.${i}.1 "yum -y install tigervnc-server xauth; cp
      /lib/systemd/system/vncserver@.service
      /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
      /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
      vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd" ; done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   ssh root@172.31.1.1 "yum -y install tigervnc-server xauth; cp
      /lib/systemd/system/vncserver@.service
      /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
      /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
      vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
   ssh root@172.31.2.1 "yum -y install tigervnc-server xauth; cp
      /lib/systemd/system/vncserver@.service
      /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
      /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
      vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
   ssh root@172.31.3.1 "yum -y install tigervnc-server xauth; cp
      /lib/systemd/system/vncserver@.service
      /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
      /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
      vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
   ssh root@172.31.4.1 "yum -y install tigervnc-server xauth; cp
      /lib/systemd/system/vncserver@.service
      /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
      /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
      vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
   ssh root@172.31.5.1 "yum -y install tigervnc-server xauth; cp
      /lib/systemd/system/vncserver@.service
      /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
      /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
      vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
   ssh root@172.31.6.1 "yum -y install tigervnc-server xauth; cp
      /lib/systemd/system/vncserver@.service
      /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
      /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
      vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
   ```

2. Install the VNC client package on the host.

   ```
   sudo yum -y install --disablerepo=* --enablerepo=localbase tigervnc
   ```

3. Configure the firewall on the nodes.

   ```
   for i in 1 2 3 4 5 6 ; do ssh root@172.31.${i}.1 \
   "firewall-cmd --permanent --add-port=5901/tcp ; \
   firewall-cmd –reload"; done
   ```

The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.2.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.3.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.4.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.5.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.6.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
```

4. Configure the firewall on the host.

```
sudo firewall-cmd --permanent --add-port=5901/tcp
sudo firewall-cmd –reload
```

B.12. Launching VNC

1. Launch the VNC server on Intel® VCA nodes.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
   root@172.31.${i}.1 "vncserver &" ; done
```

The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "vncserver &"
ssh root@172.31.2.1 "vncserver &"
ssh root@172.31.3.1 "vncserver &"
ssh root@172.31.4.1 "vncserver &"
ssh root@172.31.5.1 "vncserver &"
ssh root@172.31.6.1 "vncserver &"
```

2. Connect to the host with PuTTY from a Windows client, and launch VNCViewer.
   a. Launch PuTTY on the Windows client.
   b. Double-click the session saved in the previous section.
   c. Log in as `root`, with the root user's password.
   d. Issue the command.

```
vncviewer 172.31.X.1:5901
```

where X is replaced by the node number to which a connection is desired. Note that `5901` may be `5902` or `5903`.

Note: This command fails if Xming is not running on the Windows client before the VNCViewer command is issued.

Appendix C. Ubuntu* Tips and Tricks.

C.1. Installing Ubuntu*

1. Connect Ubuntu* OS (Required version) install media to server.
2. Boot server. If necessary, select install media as boot device.
3. At the Ubuntu* boot menu, select **Install Ubuntu** (down-arrow key), then press **<Enter>** to load the installer, which takes a couple of moments.
4. At the **WELCOME** menu, select the language (and dialect, if appropriate), then click the **Continue** button in the lower right corner of the screen.
5. At the **Preparing to Install Ubuntu** menu, select **Download updates while installing Ubuntu** and **Install third-party software for graphics and Wi-Fi hardware, Flash, MP3 and other media,** then click the **Continue** button.
6. At the **Installation Type** screen select:
    - Erase disk and install Ubuntu.
    - Use LVM with the new Ubuntu installation.
    - Click the **Install Now** button.
7. At the **Write the Changes to Disks?** screen, click **Continue**.
8. At the **Where are you?** screen select your location and click **Continue**.
9. At the **Keyboard layout** screen select your keyboard layout and click **Continue**.
10. At the **Who are you?** screen:
    - Enter the user's **Full name** and **Username** in the appropriate text boxes.
    - Enter a secure password in both the **Password** and **Confirm password** text boxes.
    - Click the **Continue** button.
11. Once the installation completes, a **Restart Now** button is displayed. When that is displayed, click the **Restart Now** button, then disconnect the installation media from the server when the POST screen is displayed.
12. When the operating system loads, login into the system with the credentials created during install.
13. Ensure that the host can access the external network. Add necessary proxy settings in case the host requires a proxy to connect to external network.
    a. Add following lines with proper `serveraddrs:port` to `/etc/environment` file (use `sudo`).

    ```
    http_proxy=http://serveraddrs:port
    https_proxy=https://serveraddrs:port
    ftp_proxy=ftp://servraddrs:port
    socks_proxy=http://servraddrs:port
    no_proxy=localhost,127.0.0.0/8
    HTTP_PROXY=http://servraddrs:port
    HTTPS_PROXY=https://servraddrs:port
    FTP_PROXY=ftp://servraddrs:port
    SOCKS_PROXY=http://servraddrs:port
    NO_PROXY=localhost,127.0.0.0/8
    Create or edit /etc/apt/apt.conf with proxy settings.
    Acquire::http::Proxy "http://proxy_address:8080/";
    Acquire::ftp::Proxy "ftp://proxy_address:8080/";
    ```

    b. For some programs that support higher level APIs that enable autoproxy use the following commands if `gsettings` exists. Please use proper `autoproxy_url:port`

    ```
    sudo gsettings set org.gnome.system.proxy mode 'auto'
    sudo gsettings set org.gnome.system.proxy autoconfig-url
       'http://autoproxy_url:port'
    ```

```
        sudo gsettings set org.gnome.system.proxy ignore-hosts "['localhost',
           '127.0.0.0/8', '*.local']"
        gsettings set org.gnome.system.proxy mode 'auto'
        gsettings set org.gnome.system.proxy autoconfig-url
           'http://autoproxy_url:port'
        gsettings set org.gnome.system.proxy ignore-hosts "['localhost',
           '127.0.0.0/8', '*.local']"
```

14. Reboot the system and check whether external network can be accessed.
15. Update the operating system, if necessary.

```
sudo apt-get update
```

TIP: Use `sudo -i` to log in as root so `sudo` is not required.

16. Install additional software required for VCA: `ipcalc`, `bridge-utils`, **`libboost-thread-dev`**, **`libboost-filesystem1.58.0`**, `openssh-server` and `sshpass`.

```
sudo apt-get install ipcalc
sudo apt-get install bridge-utils
sudo apt-get install openssh-server
sudo apt-get install libboost-thread-dev
sudo apt-get install libboost-filesystem1.58.0
sudo apt-get install sshpass
```

C.2. Enable root login on Ubuntu*

Install `openssh-server` with following command, if it is not already installed.

```
sudo apt-get install openssh-server
```

After install complete, edit `/etc/ssh/sshd_config`.

```
PermitRootLogin yes
```

Execute following commands to restart the ssh

```
sudo systemctl restart ssh
sudo systemctl restart sshd
```

Set root password

```
sudo passwd
```

Login to the operating system as root.

C.3. SSH Passwordless Configuration

First, create the SSH password-less files by issuing the following commands once for each host.

```
cd
sudo ssh-keygen
```

Press **<Enter>** at each prompt until the system returns to the command prompt.

```
sudo cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys
sudo chmod 600 /root/.ssh/authorized_keys
sudo cp -af /root/.ssh ~/
sudo chown -R `whoami`:`whoami` ~/.ssh
```

Next, copy the `.ssh` directory to each Intel® VCA node. Complete this process each time volatile nodes boot if not using a persistent filesystem through NFS. At the **Are you sure you want to continue connecting** prompt, type `yes` and then press **<Enter>**. At the password prompt, enter the password `vista1`. (This is the password for the root user on the Intel® VCA node). The following command copies the directory to all three nodes. Enter the password at each prompt.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do scp -r .ssh
   root@172.31.${i}.1:/root/ ; done
```

Note: The character before `sudo` and after `-l` is a black tick. On an en_US keyboard, it is the key left of **<1>** on the number row (an unshifted ~).

The following steps complete the same function without using a "for" loop. Run each line as appropriate for the number of nodes in the system.

```
scp -r .ssh root@172.31.1.1:/root/
scp -r .ssh root@172.31.2.1:/root/
scp -r .ssh root@172.31.3.1:/root/
scp -r .ssh root@172.31.4.1:/root/
scp -r .ssh root@172.31.5.1:/root/
scp -r .ssh root@172.31.6.1:/root/
```

C.4. Setting Up Hostnames

To set the hostnames, issue the following command for each Intel® VCA node.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
   root@172.31.${i}.1 "hostname node${i}.localdomain ; echo
   node${i}.localdomain > /etc/hostname" ; done
```

The following steps complete the same function without using a "for" loop. Run each line as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.2.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.3.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.4.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.5.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
ssh root@172.31.6.1 "hostname node1.localdomain ; echo node1.localdomain >
   /etc/hostname"
```

C.5. Creating NFS Share

1. Install NFS on the host system.

   ```
   sudo apt-get install rpcbind nfs-kernel-server
   ```

2. Share a directory in `/etc/exports`. This example uses `/mnt`; replace with selected share directory in these steps.

   ```
   ssh root@localhost "echo '/mnt     *(rw,sync,no_root_squash,
      no_all_squash,no_subtree_check)' >> /etc/exports"
   ```

3. Reset NFS service (fully stop and disable, then enable and start) to avoid possible issues.

   ```
   sudo systemctl stop nfs-server
   sudo systemctl disable nfs-server
   sudo systemctl enable rpcbind
   sudo systemctl enable nfs-server
   sudo systemctl start rpcbind
   sudo systemctl start nfs-server
   ```

   Note that restarting NSF using presented commands or by:

   ```
   service nfs stop
   service nfs restart
   ```

   can hang booted Dom0 DomU images. It is recommended to prepare NFS before booting the operating system or to use a safer command:

   ```
   exportsfs –ra
   ```

4. Install NFS on the Intel® VCA nodes. (Make sure nodes can access external network)

   ```
   for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
      root@172.31.${i}.1 "apt-get install nfs-kernel-server" ; done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   ssh root@172.31.1.1 "apt-get install nfs-kernel-server"
   ssh root@172.31.2.1 "apt-get install nfs-kernel-server"
   ssh root@172.31.3.1 "apt-get install nfs-kernel-server"
   ssh root@172.31.4.1 "apt-get install nfs-kernel-server"
   ssh root@172.31.5.1 "apt-get install nfs-kernel-server"
   ssh root@172.31.6.1 "apt-get install nfs-kernel-server"
   ```

5. Mount the share on the Intel® VCA nodes. This command mounts to the `/media` directory on the nodes; change `/tmp` in the command to the directory shared in step 4, and `/media` in the command to the directory to which the NFS share should be mounted. 172.31.${i}.254, is the NFS server address (host or others)

   ```
   for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
      root@172.31.${i}.1 "mount 172.31.${i}.254:/tmp  /media" ; done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   ssh root@172.31.1.1 "mount 172.31.1.254:/tmp  /media"
   ssh root@172.31.2.1 "mount 172.31.2.254:/tmp  /media"
   ssh root@172.31.3.1 "mount 172.31.3.254:/tmp  /media"
   ssh root@172.31.4.1 "mount 172.31.4.254:/tmp  /media"
   ssh root@172.31.5.1 "mount 172.31.5.254:/tmp  /media"
   ssh root@172.31.6.1 "mount 172.31.6.254:/tmp  /media"
   ```

C.6. Setting System Time

1. Set host system time zone. . To find the correct time zone name, run the command timedatectl list-timezones. As an example, for the Pacific Time Zone, set to `America/Los_Angeles`.

   ```
   sudo timedatectl set-timezone America/Los_Angeles
   ```

2. Set host system time. Replace `MM` with the current two-digit month; `DD` with the current two-digit date; `HH` with the current two-digit, 24-hour format hour; `MM` with the current two-digit minute; and `YYYY` with the current four-digit year. For example, `032316522015` sets the time to 4:52 pm on 23 March 2015.

   ```
   sudo date MMDDHHMMYYYY
   sudo hwclock –systohc
   ```

3. Set node time zones

   ```
   for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )); do ssh
      root@172.31.${i}.1 "timedatectl set-timezone America/Los_Angeles" ; done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   ssh root@172.31.1.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.2.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.3.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.4.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.5.1 "timedatectl set-timezone America/Los_Angeles"
   ssh root@172.31.6.1 "timedatectl set-timezone America/Los_Angeles"
   ```

4. Get the current host system time and push it to the nodes.

   ```
   NOW=`date +%m%d%H%M%Y` ; for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++
      )) ; do ssh root@172.31.${i}.1 "date $NOW" & done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.1.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.2.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.3.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.4.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.5.1 "date $NOW"
   NOW=`date +%m%d%H%M%Y` ; ssh root@172.31.6.1 "date $NOW"
   ```

C.7. Configuring Network Time Protocol (NTP)

1. Install NTP package on host system. Each command starts with `sudo` or `ssh`; enter on a single line.

   ```
   sudo yum -y install --disablerepo=* --enablerepo=localbase ntp
   ```

2. Configure NTP server on host system. Each command starts with `sudo` or `ssh`; enter on a single line.

   ```
   sudo sed -i -e 's/^server/#server/' /etc/ntp.conf
   ssh root@localhost 'echo "server 127.0.0.1" >> /etc/ntp.conf'
   ssh root@localhost 'echo "restrict 172.31.0.0 mask 255.255.0.0 nomodify
      notrap" >> /etc/ntp.conf'
   sudo systemctl restart ntpd
   sudo systemctl enable ntpd
   ```

3. Install and configure NTP package on Intel® VCA nodes.

   ```
   for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
      root@172.31.${i}.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.${i}.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd" ; done
   ```

   The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

   ```
   ssh root@172.31.1.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.1.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.2.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.2.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.3.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.3.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.4.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.4.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.5.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.5.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ssh root@172.31.6.1 "yum -y install ntp; sed -i -e 's/^server/#server/'
      /etc/ntp.conf ; echo server 172.31.6.254 >> /etc/ntp.conf ; systemctl
      enable ntpd; systemctl restart ntpd"
   ```

C.8. Turn On Affinity

Affinity setting can be changed on the host to balance simultaneous transfers between the host and all nodes. It is recommended to turn it on.

1. Open the file `/etc/sysconfig/irqbalance`.
2. Find the line:

   ```
   #IRQBALANCE_ARGS=
   ```

3. Change this line to (do not forget to remove `#`):

   ```
   IRQBALANCE_ARGS="-h exact"
   ```

4. Save the file and reboot the host.

C.9. Configuring Microsoft Windows* Client Access to VNCViewer

Before performing the following steps, install Xming (http://sourceforge.net/projects/xming/) and PuTTY (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) on the Windows client.

1. Launch XMing.
2. Launch PuTTY.
3. Enter the IP address of the Intel® VCA host system in the **Host Name (or IP address)** text box.
4. Enter a name for the system in the **Saved Sessions** text box.
5. In the left-side navigation pane, open **Expand Connection** -> **SSH**, then click **X11** below **SSH**.
6. In the right-side settings pane, check the **Enable X11 forwarding** checkbox, then enter `:0` in the **X display location** text box.
7. In the left-side navigation pane, click **Window**.
8. In the right-side settings pane, enter `10000` in the **Lines of scrollback** text box (replacing whatever value is there; standard default is 200).
9. In the left-side navigation pane, scroll to the top, and click **Session**.
10. In the right-side settings pane, click the **Save** button next to the **Saved Sessions** list.
11. At any future point, the connection to the Intel® VCA system can be opened by launching PuTTY then either:
    a. Double-clicking the name from the **Saved Sessions** list, or
    b. Clicking the name from the **Saved Sessions** list, then clicking **Load**, and then clicking **Open**.

C.10. Install and Configure VNC

1. Install and configure the VNC package on Intel® VCA nodes.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
   root@172.31.${i}.1 "yum -y install tigervnc-server xauth; cp
   /lib/systemd/system/vncserver@.service
   /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
   /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
   vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd" ; done
```

The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "yum -y install tigervnc-server xauth; cp
   /lib/systemd/system/vncserver@.service
   /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
   /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
   vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.2.1 "yum -y install tigervnc-server xauth; cp
   /lib/systemd/system/vncserver@.service
   /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
   /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
   vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.3.1 "yum -y install tigervnc-server xauth; cp
   /lib/systemd/system/vncserver@.service
   /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
   /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
   vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.4.1 "yum -y install tigervnc-server xauth; cp
   /lib/systemd/system/vncserver@.service
   /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
   /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
   vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.5.1 "yum -y install tigervnc-server xauth; cp
   /lib/systemd/system/vncserver@.service
   /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
   /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
   vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
ssh root@172.31.6.1 "yum -y install tigervnc-server xauth; cp
   /lib/systemd/system/vncserver@.service
   /etc/systemd/system/vncserver@:1.service; sed -i -e s='<USER>'=root=
   /etc/systemd/system/vncserver@:1.service; echo vista1 > /tmp/passwd;
   vncpasswd -f < /tmp/passwd > .vnc/passwd; chmod 600 .vnc/passwd"
```

2. Install the VNC client package on the host.

```
sudo yum -y install --disablerepo=* --enablerepo=localbase tigervnc
```

3. Configure the firewall on the nodes.

```
for i in 1 2 3 4 5 6 ; do ssh root@172.31.${i}.1 \
"firewall-cmd --permanent --add-port=5901/tcp ; \
firewall-cmd –reload"; done
```

The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.2.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.3.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.4.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.5.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
ssh root@172.31.6.1 "firewall-cmd --permanent --add-port=5901/tcp ;
   firewall-cmd –reload"
```

4. Configure the firewall on the host.

```
sudo firewall-cmd --permanent --add-port=5901/tcp
sudo firewall-cmd –reload
```

C.11. Launching VNC

1. Launch the VNC server on Intel® VCA nodes.

```
for (( i=1 ; i<=`sudo vcactl status | wc -l` ; i++ )) ; do ssh
   root@172.31.${i}.1 "vncserver &" ; done
```

The following steps complete the same function without the "for" loop; run each line, as appropriate for the number of nodes in the system.

```
ssh root@172.31.1.1 "vncserver &"
ssh root@172.31.2.1 "vncserver &"
ssh root@172.31.3.1 "vncserver &"
ssh root@172.31.4.1 "vncserver &"
ssh root@172.31.5.1 "vncserver &"
ssh root@172.31.6.1 "vncserver &"
```

2. Connect to the host with PuTTY from a Windows client, and launch VNCViewer.
   a. Launch PuTTY on the Windows client.
   b. Double-click the session saved in the previous section.
   c. Log in as `root`, with the root user's password.
   d. Issue the command.

      ```
      vncviewer 172.31.X.1:5901
      ```

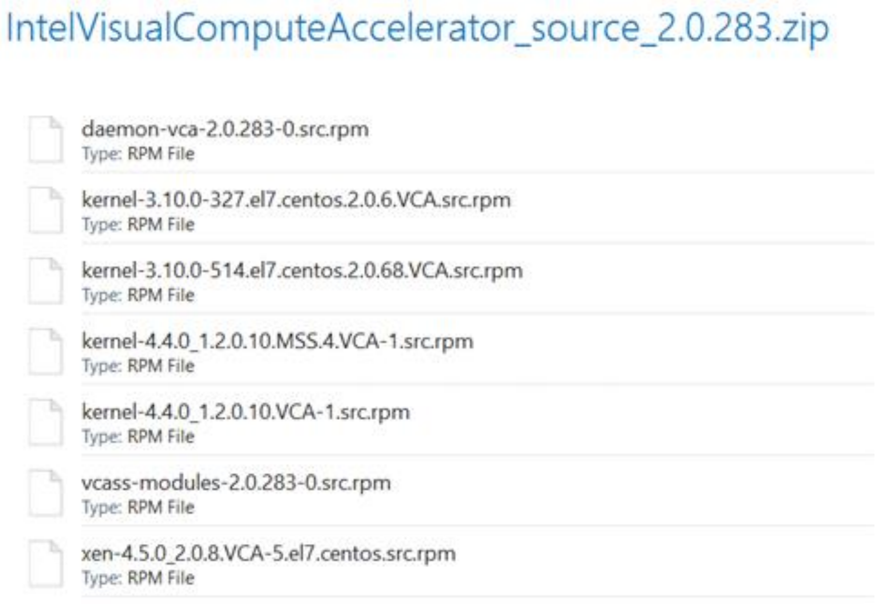
      where X is replaced by the node number to which a connection is desired. Note that `5901` may be `5902` or `5903`.

Note: This command fails if Xming is not running on the Windows client before the VNCViewer command is issued.

Appendix D. Build Intel® VCA Modules in Custom Kernel

Intel® VCA modules (`vcass module` and `daemon vca` (app)) can be built for a custom non-supported kernels by following the procedure given in this appendix.

First, download Intel® VCA source files from https://downloadcenter.intel.com. Multiple versions of these files are available; select the required version. Download the `.zip` file that contains the source of the Intel® VCA modules. For example, version 2.0.283 contains the files shown in Figure 58.


IntelVisualComputeAccelerator_source_2.0.283.zip

daemon-vca-2.0.283-0.src.rpm
Type: RPM File

kernel-3.10.0-327.el7.centos.2.0.6.VCA.src.rpm
Type: RPM File

kernel-3.10.0-514.el7.centos.2.0.68.VCA.src.rpm
Type: RPM File

kernel-4.4.0_1.2.0.10.MSS.4.VCA-1.src.rpm
Type: RPM File

kernel-4.4.0_1.2.0.10.VCA-1.src.rpm
Type: RPM File

vcass-modules-2.0.283-0.src.rpm
Type: RPM File

xen-4.5.0_2.0.8.VCA-5.el7.centos.src.rpm
Type: RPM File

Figure 58. Intel® VCA source files

Unzip and install the `vcass-modules` and `daemon-vca` source `.rpm` files and follow the steps depending on the operating system.

For Ubuntu*:

1. Install package named `linux-headers*.deb` which comes with the kernel `.deb` package.
2. Install usual compilation tools: `cmake`, `gcc`, and `checkinstall`.
3. Compile modules. Replace `<vcamodules-dir>` and `<buildscripts-dir>` with appropriate paths.

```
Cd <vcamodules-dir>
KVER=$(uname -r) # works if you are building for currently booted kernel,
   otherwise set accordingly
OS=UBUNTU PKG_VER=2.1.999 KERNEL_NAME=$KVER
   KERNEL_PATH=/lib/modules/$KVER/build
   <buildscripts-dir>/quickbuild/generate_modules.sh # build modules deb
```

4. At the end of the building process, the `.deb` file name is printed.

For CentOS*:

1. Install following tools: `rpm2cpio`, `make`, and `gcc`.
2. Install `kernel-devel.rpm` from custom kernel provider
3. Compile modules. Replace `<vcamodules-dir>` and `<buildscripts-dir>` with appropriate paths.

```
cd <vcamodules-dir>
DEVEL_RPM=<kernel-devel-rpm-file-full-path> OS=CENTOS PKG_VER=2.1.999
   <buildscripts-dir>/quickbuild/generate_modules.sh
```

Appendix E. Resizing Persistent VCA Boot image

Intel® VCA persistent boot images can be resized using the script `vca_image_resize.sh` provided as part of daemon-vca rpm package and is in `/usr/sbin/`.

E.1. Introduction

`vca_image_resize.sh` script resizes block-io based Intel® VCA persistent bootable image by resizing its root partition. Intel® VCA persistent image contains two partitions (Boot partition and Root partition) as shown in Figure 59.

Boot Partition: a static UEFI boot partition (EFI System Partition, ESP) with a fixed size of 100MB. The size of 100MB this the minimum dictated by the UEAFI standard and is also entirely sufficient for booting VCA nodes

Root Partition: a resizable Linux partition hosting the root file system of the Linux operating system used on VCA nodes.

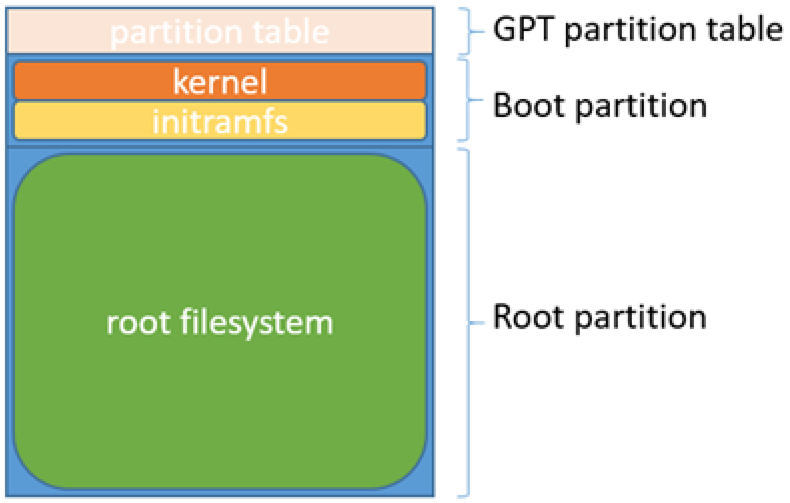


Figure 59. Partitions in Intel® VCA 2 persistent image

Root partition is the only partition accessible to Intel® VCA node after booting the nodes, so it should not be made too small. Other file systems can be added to Intel® VCA nodes by NFS mounting remote directories and/or additional block-io devices described in previous sections. The root file system should be enough to hold all installed binaries, configuration files, user data and system logs.

Current Intel® VCA persistent image can't be made smaller than the boot partition, some organizational data, storage required for the Linux partition of the image, and the maximum space required for application usage. Intel® VCA persistent boot image resides on the host file system or storage connected to host server. Increasing the size of any file can be done in two ways: by completely allocating the required space in disk blocks from underlying storage or by creating a sparse file. For a sparse file, the disk space is not allocated until the blocks are actually written to, but the requested blocks of file data are linked to a single fictitious storage block filled with zeroes. All the blocks of file data remain linked to this fictitious storage block as long as these data blocks are not actually written to. Thus for some time after creation, the file does NOT require more storage, while its internal structure foresees the exact storage space for future needs. Only during later use of the file, when actually witting to the file data blocks, they gradually get assigned to real disk storage blocks.

When requesting the script *`vca_image_resize.sh`* to grow the Intel® VCA boot image as a sparse file, the resulting file will require less storage from the underlying device. Also, the image will be grown significantly quicker than without requesting to use the sparse file feature. Specific options applied to tools used during local copying may also spare copy time during administrative tasks (e.g. "cp --sparse=always"). This may often be advantageous at the initial stages of preparing, multiplying and transferring the image. But as the image will typically grow during usage to its defined size, enough storage space must be provided for it in the underlying device to prevent Intel® VCA node failures. When growing Intel® VCA boot image as a non-sparse file, the resize will take more time and more space, and may fail due to insufficient disk space on the underlying device. If the disk storage at the beginning of growing Intel® VCA boot image is not sufficient, the script issues a warning and continues in the hope that the caller will make additional disk space available before imminent failure.

The final size of the Intel® VCA persistent boot image may not be exactly the same as requested size due to boot partition size, partition structures and alignments (rounded up or down).

E.2. Commandline Options

Note: The script `vca_image_resize.sh` is based on the assumption that the image has been created with the VCA tools.

The image is resized in-place, so user should consider making a backup copy of the image first.

The image should not be mounted, nor booted, nor used in any other way during resize.

This script has been tested on CentOS. For running on CentOS the following software packages are required: `gdisk, gnucoreutils, findutils, e2fsprogs, kpartx, gawk, util-linux`.

-d <number> **Debug level**. The number must be integer. Value 0 means no debug, higher values increase debug verbosity. The default debug level is ${DEBUG_LEVEL}

-i <path> **Image path**. The image will be resized in-place. The image cannot be used during resize.

-n <number> The new size of the root partition in the image. Can be an integer or a real number. The default size unit is GB.

-s Create a sparse output file. Sparse files require less space initially as they are allowed to contain holes. With writes to the holes of the file during its latter usage, the file gradually grows to eventually occupy its size defined with option -n.

-u <letter> A (lowercase or uppercase) letter describing the units of the new size. Supported values are: B for bytes, K for 1024 bytes, M for 1024 Kbytes, or G for 1024 Mbytes. This parameter is optional. The default size unit is GB.

-y Spare the usual safety warnings about avoiding to modify a file concurrently

E.3. Examples

Resize the VCA boot image named image10g.vcad to 15 GB, in place, creating a sparse file:

```
vca_image_resize.sh -n 15 -s -i image10g.vcad
```

Resize the VCA boot image named image10g.vcad to 2000MB, in place, creating a non-sparse file, with no debug messages and no safety warnings:

```
vca_image_resize.sh -d 0 -n 2000 -u M -y -i image10g.vcad
```

Appendix F. Glossary

Term	Definition
AVC	Advanced Video Coding
BIOS	Basic Input/Output System
DHCP	Dynamic Host Configuration Protocol
ECC	Error Correction Code
EFI	Extensible Firmware Interface
GPT	GUID Partition Table
HEVC	High Efficiency Video Coding
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
ISV	Independent Software Vendor
MBR	Master Boot Record
NFS	Network File System
NTP	Network Time Protocol
NVRAM	Non-Volatile RAM
OpenCL*	Open Computing Language
PCH	Platform Controller Hub
PCIe*	PCI Express*
SDK	Software Development Kit
SSH	Secure Shell
Intel® VCA	Intel® Visual Compute Accelerator
VNC	Virtual Network Computing