# APPS025-Codensity$^{TM}$ T408 SR-IOV Configuration and Usage Guide

V2.0

# Table of Contents

# 1 Table of Abbreviations

| **Abbreviation:** | **Full Form:** |
|---|---|
| SR-IOV | Single Root – Input Output Virtualization |
| PF | Private Function |
| VF | Virtual Function |
| VM | Virtual Machine |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| AUD | Access Unit Delimiters |
| AV | Audio Video |
| AV1 | Alliance Open Media Video 1 Codec |
| CABAC | Context Adaptive Binary Arithmetic Coding |
| CBR | Constant Bit Rate |
| CPU | Central Processing Unit |
| CRF | Constant Rate Factor |
| DSP | Digital Signal Processing |
| EOF | End Of File |
| FIXQB | Fixed Quantization Parameter |
| FF | Form Factor |
| FPS | Frames Per Second |
| GOP | Group of pictures |
| HDR | High Dynamic Range |
| HHHL | Half Height, Half Length |
| HHFL | Half Height, Full Length |
| HLG | Hybrid Log Gamma |
| HRD | Hypothetical Reference Decoder |
| IE | Inference Engine |
| Mbps | Mega Bit per second |
| MBps | Mega Byte per second |
| NLP | Natural Language Processing |
| NPU | Neural Processing Unit |
| NVMe | Non-Volatile Memory express |
| PCIe | Peripheral Component Interconnect express |
| PPS | Picture Parameter Set |

| | |
|---|---|
| QP | Quoted Printable |
| RGB | Red Green Blue |
| RGBA | Red Green Blue Alpha |
| RDO | Optimization |
| ROI | Region of interest |
| SDK | Software Development Kit |
| SEI | Supplemental Enhancement Information |
| SRIOV | Single Root I/O Virtualization |
| SPS | Sequence Parameter Set |
| VBR | Variable Bit Rat |
| VCL | Video Coding Layer |
| VPS | Video Parameter Set |
| VUI | Video Usability Information |

# 2 References

Technical Notes:

Specifications:

# 3 Background

This document demonstrates the install of KVM and CentOS VM on Linux host and then proceeds to install the T408 libxcoder and ffmpeg environment on the VM. At the end of this install a user shall be able to run ffmpeg transcoding operation from the VM using a T408 VF device connection.

Similar procedure can be used for installing any other Linux OS VM environment on any Linux OS Host. Refer to Notes in this document and the 'Useful Links' section.

## 3.1 Intended Audience

This document is intended to help engineers/technicians/developers operating NETINT transcoders better understand the SR-IOV feature.

## 3.2 Compatibility

**Software Compatibility**

This guide is intended to be used with NETINT Codensity T408 Video Transcoder software Release 2.4.9.

**Hardware Compatibility**

Release 2.4.9 supports NETINT Codensity T408 Video Transcoder hardware.

**Operation system**

All OS supporting T408 Video Transcoder.

# 4 SR-IOV

## 4.1 Intro

The SR-IOV feature in T408 allows users to attach the VFs on a single or multiple T408 controllers to one or multiple VMs running on a Host.

## 4.2 Environment

For the purpose of this document the following hardware and OS was used. Depending on the CPU, BIOS, OS being used on the host, the host may use different settings or commands to enable the same set of features.

The steps in this document use:

Gigabyte Server R181-NA0

Intel Xeon Silver 4110 CPU

CentOS 7

Intel I350 NIC

## 4.3 Check hardware for Virtualization support

Check hardware:

Check and note the host operating system (CentOS or Ubuntu) and if host is using Intel or AMD processors.

```
lsbrelease -a

lspci
```

```
[nvme@nvme-cli129 ~]$ lsb_release -a

LSB Version: :core-4.1-amd64:core-4.1-noarch

Distributor ID: CentOS

Description: CentOS Linux release 7.6.1810 (Core)

Release: 7.6.1810

Codename: Core
```

[nvme@nvme-cli129 ~]$ lspci |grep Intel |more

00:00.0 Host bridge: Intel Corporation Sky Lake-E DMI3 Registers (rev 04)

00:05.4 PIC: Intel Corporation Device 2026 (rev 04)

00:17.0 SATA controller: Intel Corporation C620 Series Chipset Family SATA Controller [AHCI mode] (rev 09)

00:1c.0 PCI bridge: Intel Corporation C620 Series Chipset Family PCI Express Root Port #1 (rev f9)

00:1c.4 PCI bridge: Intel Corporation C620 Series Chipset Family PCI Express Root Port #5 (rev f9)

00:1c.5 PCI bridge: Intel Corporation C620 Series Chipset Family PCI Express Root Port #6 (rev f9)

17:00.0 PCI bridge: Intel Corporation Sky Lake-E PCI Express Root Port A (rev 04)

17:05.0 System peripheral: Intel Corporation Device 2034 (rev 04)

17:05.2 System peripheral: Intel Corporation Sky Lake-E RAS Configuration Registers (rev 04)

17:05.4 PIC: Intel Corporation Device 2036 (rev 04)

Check virtualization support:

```
cat /proc/cpuinfo |grep vmx
lscpu |grep VT-x
```

[nvme@nvme-cli102 ~]$ cat /proc/cpuinfo |grep vmx

flags : fpu …… vmx ……

[nvme@nvme-cli102 ~]$ lscpu |grep VT-x

Virtualization: VT-x

[nvme@nvme-cli102 ~]$

This info is from a PC running Intel CPUs.

## 4.4 Check virtualization is enabled in BIOS

Reboot the PC and keep tapping F2 to get the host to enter Boot or Setup menu.

Based on host make and type, the steps to get the host into a Boot or Setup menu may be different.

Browse through the Boot menus and check for any or all of following and ensure that the setting is 'Enabled':

Virtualization

VMX

VT-d

SRIOV

Different host may have BIOS that shows all or some of the above parameters. Essentially, we want to ensure that BIOS has virtualization enabled.H.265

## 4.5 Install KVM on host machine

KVM stands for Kernel-based Virtual Machine. For this step, we will install KVM on the linux host and then install a CentOS VM.

Run the following commands on the host PC:

```
sudo yum install qemu-kvm libvirt libvirt-python libguestfs-tools
virt-install
```

```
sudo systemctl enable libvirtd
```

```
sudo systemctl start libvirtd
```

```
lsmod | grep -i kvm
```

```
[nvme@nvme-cli129 ~]$ lsmod |grep -i kvm
kvm 586948 0
irqbypass 13503 2 kvm,vfio_pci
[nvme@nvme-cli129 ~]$
```

```
brctl show
```

```
sudo virsh net-list
```

```
[nvme@nvme-cli129 ~]$ sudo virsh net-list
Name State Autostart Persistent
default active yes
[nvme@nvme-cli129 ~]$
```

By default, all VMs have access to VMs on the same server i.e. a private network.

If VMs need to be able to available to other servers on the LAN then we need to setup a network bridge.

Note - the network card on the host PC needs to support bridging. For example, Intel I350-T4.

## 4.6 Activate bridging on host

```
ip address
```

Find the link ID for the active ethernet port. For example, eth0 or eno1 or enp5s0f0 or similar.

On CentOS, the network interfaces are defined in /etc/sysconfig/network-scripts folder.

For this example, the network adapter is enp2s0f0 and the corresponding file is ifcfg-enp2s0f0.

Perform the following steps,

```
modinfo bridge
```

If no output or host reports that module is not loaded, then perform the following step.

```
modprobe --first-time bridge
```

Install bridge-utils for controlling the network adapters.

```
sudo yum install bridge-utils -y
```

Create a network bridge.

```
sudo nmcli conn show --active
```

```
sudo nmcli conn add type bridge con-name br0 ifname br0
```

This step will create a ifcfg-br0 file. Edit the ifcfg-br0 file and edit the contents of the ifcfg-br0 as per sample example below. Ensure the any network IP entries match your lab network environment. Check the network adaptor name for your host by running 'ip address', for example, enp2s0f0 or eth0 or eno1.

Edit the existing adaptor 'enp2s0f0' file ifcfg-enp2s0f0 and ensure the contents are as per sample example below.

```
[nvme@nvme-cli129 ~]$ cd /etc/sysconfig/network-scripts

[nvme@nvme-cli129 network-scripts]$ sudo cat ifcfg-enp2s0f0

TYPE=Ethernet

PROXY_METHOD=none

BROWSER_ONLY=no
```

```
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp2s0f0
UUID=c357bd50-c5ed-483c-b996-7c01dc8be526
DEVICE=enp2s0f0
ONBOOT=no
#IPADDR=10.9.131.111
#NETMASK=255.255.254.0
#GATEWAY=10.9.130.1
#DNS1=10.9.10.10
#DNS2=10.9.10.11
#ZONE=public
[nvme@nvme-cli129 network-scripts]$ sudo cat ifcfg-br0
TYPE=Bridge
STP=yes
BRIDGING_OPTS=priority=32768
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
```

IPV4_FAILURE_FATAL=no

IPV6INIT=yes

IPV6_AUTOCONF=yes

IPV6_DEFROUTE=yes

IPV6_FAILURE_FATAL=no

IPV6_ADDR_GEN_MODE=stable-privacy

NAME=br0

UUID=1bf38e13-960b-406f-b5d5-3c9a552c4369

DEVICE=br0

ONBOOT=yes

IPADDR=10.9.131.111

NETMASK=255.255.254.0

GATEWAY=10.9.130.1

DNS1=10.9.10.10

DNS2=10.9.10.11

ZONE=public

#DELAY=0

[nvme@nvme-cli129 network-scripts]$

Setup br0 as the master.

```
sudo nmcli conn add type ethernet slave-type bridge con-name
bridge-br0 ifname enp2s0f0 master br0
```

Activate the bridge by issuing:

```
sudo nmcli conn up br0
```

or

```
sudo systemctl restart network
```

or

```
sudo reboot now
```

Run 'ip address' and confirm 'br0' link ID is active and shows an IP address

```
sudo nmcli conn show --active
```

```
sudo bridge link show
```

```
ip add
```

[nvme@nvme-cli129 network-scripts]$ ip add

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid_lft forever preferred_lft forever

inet6 ::1/128 scope host

valid_lft forever preferred_lft forever

2: enp2s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master br0 state UP group default qlen 1000

link/ether e0:d5:5e:5d:22:48 brd ff:ff:ff:ff:ff:ff

3: enp2s0f1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000

link/ether e0:d5:5e:5d:22:49 brd ff:ff:ff:ff:ff:ff

4: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000

link/ether e0:d5:5e:5d:22:48 brd ff:ff:ff:ff:ff:ff

inet 10.9.131.111/23 brd 10.9.131.255 scope global br0

valid_lft forever preferred_lft forever

inet6 fe80::e2d5:5eff:fe5d:2248/64 scope link

valid_lft forever preferred_lft forever

5: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000

link/ether 52:54:00:20:39:4a brd ff:ff:ff:ff:ff:ff inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0

valid_lft forever preferred_lft forever

6: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast master virbr0 state DOWN group default qlen 1000

link/ether 52:54:00:20:39:4a brd ff:ff:ff:ff:ff:ff

[nvme@nvme-cli129 network-scripts]$

## 4.7 Enable SSH X11Forwarding on Host

Edit sshd_config file and uncomment the X11Forwarding entry as shown below.

```
sudo vi /etc/ssh/sshd_config
```

#AllowAgentForwarding yes

#AllowTcpForwarding yes

#GatewayPorts no

X11Forwarding yes

#X11DisplayOffset 10

#X11UseLocalhost yes

```
sudo systemctl restart sshd.service
```

## 4.8 Create VM

Download the ISO image for CentOS and the corresponding checksum file.

```
cd /var/lib/libvirt/boot/
```

```
sudo wget
https://mirrors.edge.kernel.org/centos/7.8.2003/isos/x86_64/CentOS-7-x86_64-Minimal-2003.iso
```

```
sudo wget
https://mirrors.edge.kernel.org/centos/7.8.2003/isos/x86_64/sha256sum.txt
```

```
sudo sha256sum -c sha256sum.txt
```

[root@nvme-cli129 boot]# sha256sum -c sha256sum.txt

…

CentOS-7-x86_64-Minimal-2003.iso: OK

…

[root@nvme-cli129 boot]#

The CentOS ISO images are located at https://mirrors.edge.kernel.org/centos/

We are installing a CentOS 7.8.2003 iso x86_64 Minimal image CentOS-7-x86_64-Minimal-2003.iso. The kernel.org gets updated with new supported images. Pick a suitable image for your VM installation.

Creating VM: 1GB RAM with 10GB disk space

On CentOS host:

```
sudo virt-install --virt-type=kvm --name centos71 --ram 1024 --
vcpus=1 --os-variant=centos7.0 --
cdrom=/var/lib/libvirt/boot/CentOS-7-x86_64-Minimal-2003.iso --
noautoconsole --network=bridge=br0,model=virtio --graphics vnc --
disk
path=/var/lib/libvirt/images/centos71.qcow2,size=10,bus=virtio,fo
rmat=qcow2
```

1] The VM name highlighted above should match

2] Update the VM name for each new instance of VM creation

3] Note the above install command is using network bridge 'br0'

Note - For Ubuntu host:

This document focuses on a CentOS host. If installing a VM on an Ubuntu Host, refer to the 'Useful Links' section and follow link to Ubuntu VM install. The VM create command is the same with an extra --hvm option.

```
sudo virt-install --virt-type=kvm --name centos71 --ram 1024 --
vcpus=1 --os-variant=centos7.0 --
cdrom=/var/lib/libvirt/boot/CentOS-7-x86_64-Minimal-2003.iso --
hvm --noautoconsole --network=bridge=br0,model=virtio --graphics
vnc --disk
path=/var/lib/libvirt/images/centos71.qcow2,size=10,bus=virtio,fo
rmat=qcow2
```

Note – For Ampere Server aarch64 Host –

Aarch64 architectures require a different CentOS iso image. Get it from: http://centos.ca-west.mirror.fullhost.io/centos/8.2.2004/isos/aarch64/

```
sudo virt-install --boot uefi --virt-type=kvm --name centos71 --
ram 1024 --vcpus=1 --os-variant=centos7.0 --
cdrom=/var/lib/libvirt/boot/CentOS-8.2.2004-aarch64-minimal.iso -
-hvm --noautoconsole --network=bridge=br0,model=virtio --graphics
vnc --disk
path=/var/lib/libvirt/images/centos71.qcow2,size=10,bus=virtio,fo
rmat=qcow2
```

Note – If installing an Ubuntu VM:

Ubuntu iso images are here:

http://releases.ubuntu.com/

https://releases.ubuntu.com/20.04.1/ubuntu-20.04.1-live-server-amd64.iso

virt-install command for Ubuntu VM is as follows:

```
sudo virt-install --virt-type=kvm --name ubuntu1841 --ram 1024 --
vcpus=1 --cdrom=/var/lib/libvirt/boot/ubuntu-18.04.5-desktop-
amd64.iso --noautoconsole --network=bridge=br0,model=virtio --
graphics vnc --disk
path=/var/lib/libvirt/images/ubuntu1841.qcow2,size=10,bus=virtio,
format=qcow2
```

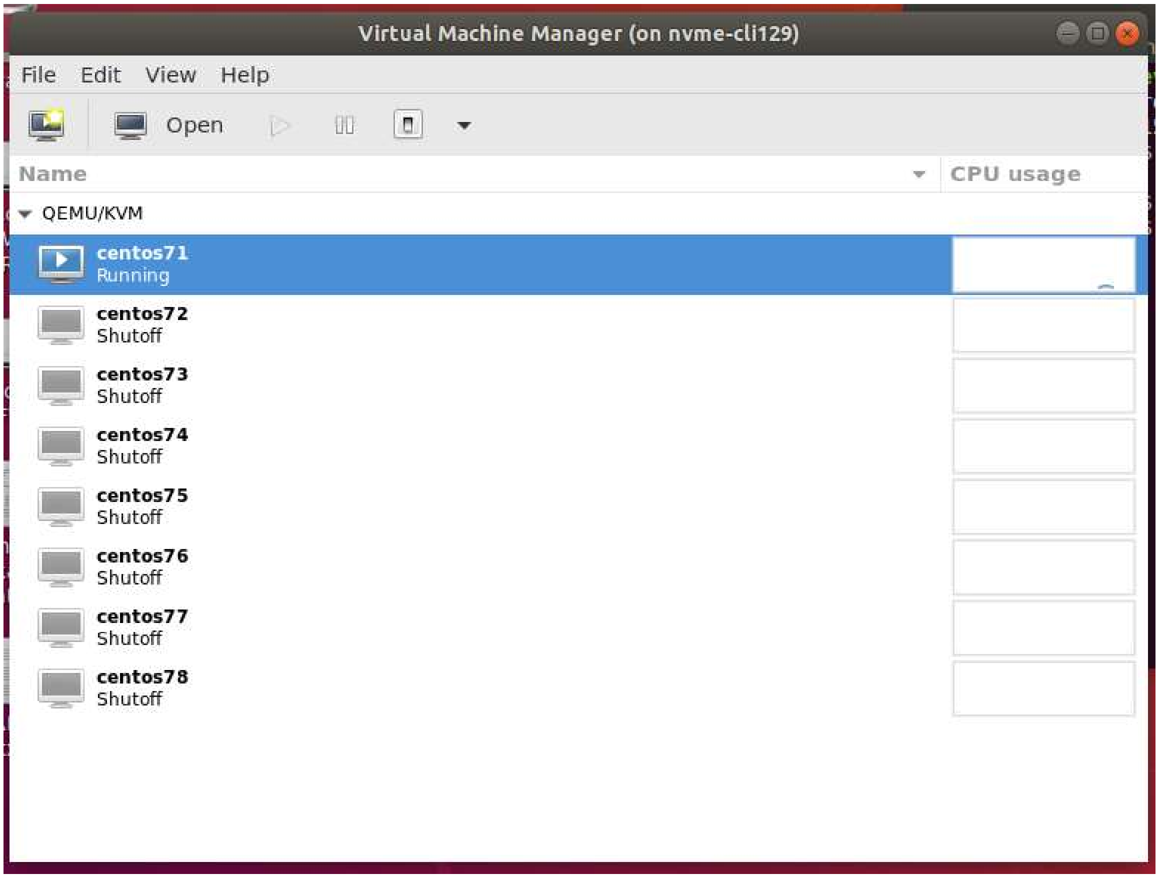
From a Linux terminal (Linux VM running on your Windows laptop, for example), ssh to Host PC and start virt-manager.

```
ssh -X nvme@cli129
```

```
virt-manager
```

In virt-manager gui, select the VM and then select ‘Open’.

virt-manager snapshot pic -




Proceed with completing the VM installation. Setup 'logan' as the root password and a new user 'nvme' and with password 'logan' when prompted. Select the 10G partition.

Once the VM installation shows complete, click on the Reboot icon to reboot the VM.

On host run the following command to set VM for autostart.

```
sudo virsh list

sudo virsh autostart centos71

sudo virsh list

sudo reboot now
```

If you just want to start the VM then run:

```
sudo virsh start centos71
```

VM uses a xml file during launch to mount the devices. Location: /etc/libvirt/qemu/

```
sudo ls /etc/libvirt/qemu/

sudo virsh dumpxml centos71
```

## 4.9 Confirm VM network interface connection

With virt-manager, access the VM terminal.

Open the ifcfg-eth0 file on the VM and change the onboot setting to yes - ‘ONBOOT=yes’

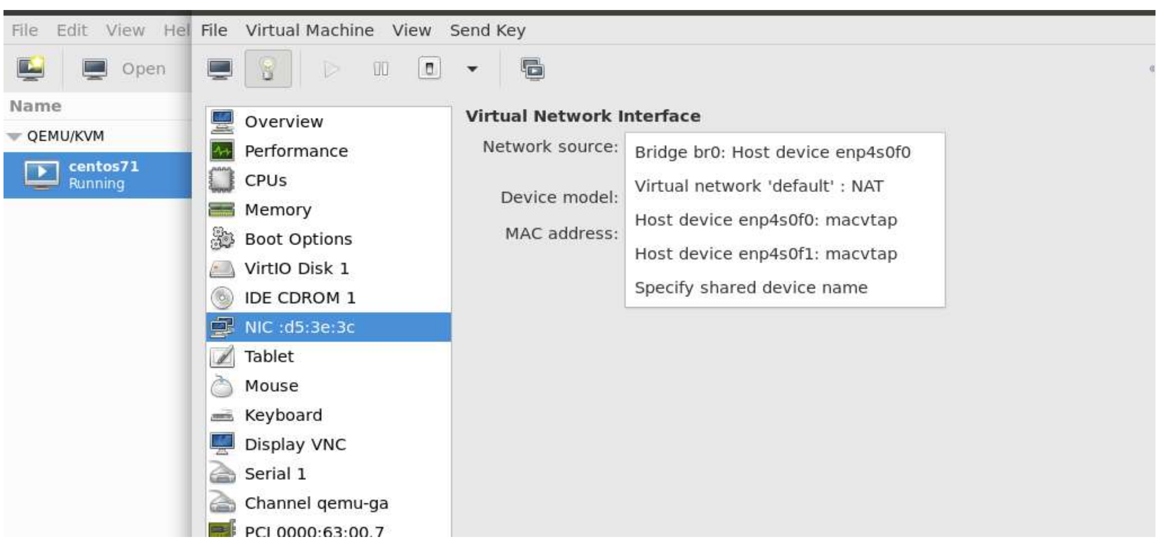
```
sudo vi /etc/sysconfig/network-scripts/ifcfg-eth0

sudo systemctl restart network

ip address
```

Check that the eth0 interface has an IP address assigned. It will be a 10.9.xx.xx IP address. Confirm that the ‘View->Details->NIC-> Network Source’ setting is ‘Bridge br0: Host device enp4s0f0’.

If there is trouble acquiring an IP address then try changing the NIC setting under 'View->Details->NIC' and set 'Network Source' to 'Virtual network 'default': NAT'. and perform a 'sudo reboot now'. This step will acquire a 192.x.x.x. IP address from the host bridge. You should now be able to run 'sudo yum install ...' .



## 4.10 Enabling IOMMU on Host

IOMMU needs to be enabled on the host for PCI passthrough to work. If supported on the host, you will enable it in the grub file.

Check IOMMU status.

```
dmesg |grep -i IOMMU
```

[ 0.000000] DMAR: IOMMU enabled

If iommu, is not enabled then proceed with the following steps to edit the grub file.

```
sudo vi /etc/default/grub
```

GRUB_CMDLINE_LINUX="intel_iommu=on iommu=pt"

the iommu options can be added to any of GRUB_CMDLINE_LINUX or GRUB_CMDLINE_LINUX_DEFAULT

if the GRUB_CMDLINE_LINUX line already exists then just set “intel_iommu=on iommu=pt” (syntax is a space followed by intel_iommu=on iommu=pt )

Refer section 3.10.1 for grub file changes as applicable to AMD hosts.

Update grub using:

```
sudo update grub
```

or

```
sudo update grub2
```

or

```
grub2-mkconfig -o /boot/grub/grub.cfg
```

Reboot host.

```
sudo reboot now
```

Once the host is boot up, on an Intel host the following messages will be seen showing devices being added to iommu groups.

```
dmesg |grep -i IOMMU
```

[ 1.856207] iommu: Adding device 0000:00:00.0 to group 0

[ 1.856522] iommu: Adding device 0000:00:04.0 to group 1

[ 1.856563] iommu: Adding device 0000:00:04.1 to group 1

[ 1.856603] iommu: Adding device 0000:00:04.2 to group 1

[ 1.856663] iommu: Adding device 0000:00:04.3 to group 1

[ 1.856704] iommu: Adding device 0000:00:04.4 to group 1

[ 1.856743] iommu: Adding device 0000:00:04.5 to group 1

[ 1.856783] iommu: Adding device 0000:00:04.6 to group 1

[ 1.856822] iommu: Adding device 0000:00:04.7 to group 1

[ 1.856959] iommu: Adding device 0000:00:05.0 to group 2

[ 1.857000] iommu: Adding device 0000:00:05.2 to group 2

[ 1.857041] iommu: Adding device 0000:00:05.4 to group 2

[ 1.857174] iommu: Adding device 0000:00:08.0 to group 3

[ 1.857215] iommu: Adding device 0000:00:08.1 to group 3

[ 1.857255] iommu: Adding device 0000:00:08.2 to group 3

Note - On AMD hosts the IOMMU messaging is different.

Note - On UEFI boot system,

modify the /boot/efi/EFI/centos/grub.cfg file directly to add ‘iommu=pt intel_iommu=on’

sample from grub.cfg:

BEGIN /etc/grub.d/10_linux

menuentry 'CentOS Linux (3.10.0-957.el7.x86_64) 7 (Core)' --class centos --class gnu-linux --class gnu --class os --unrestricted $menuentry_id_option

'gnulinux-3.10.0-957.el7.x86_64-advanced-d773bb0c-b333-472e-afe4-576192be4ff9' {

load_video

set gfxpayload=keep

insmod gzio

insmod part_gpt

insmod xfs

if [ x$feature_platform_search_hint = xy ]; then

search --no-floppy --fs-uuid --set=root 906568c8-966f-4a76-8ca3-371a7611b597 else

search --no-floppy --fs-uuid --set=root 906568c8-966f-4a76-8ca3-371a7611b597

fi

linuxefi /vmlinuz-3.10.0-957.el7.x86_64 root=/dev/mapper/centos-home ro crashkernel=auto rd.lvm.lv=centos/home rd.lvm.lv=centos/swap rhgb quie

t LANG=en_US.UTF-8 intel_iommu=on iommu=pt

initrdefi /initramfs-3.10.0-957.el7.x86_64.img

}

Note - Better way is to use update-grub similar steps for a UEFI boot device, instead of editing the grub.cfg file directly.

#### 4.10.1 Enabling IOMMU on AMD Host

The steps to enable IOMMU described above apply to AMD servers also. Some changes that are different are mentioned below:

1] Updating /etc/default/grub

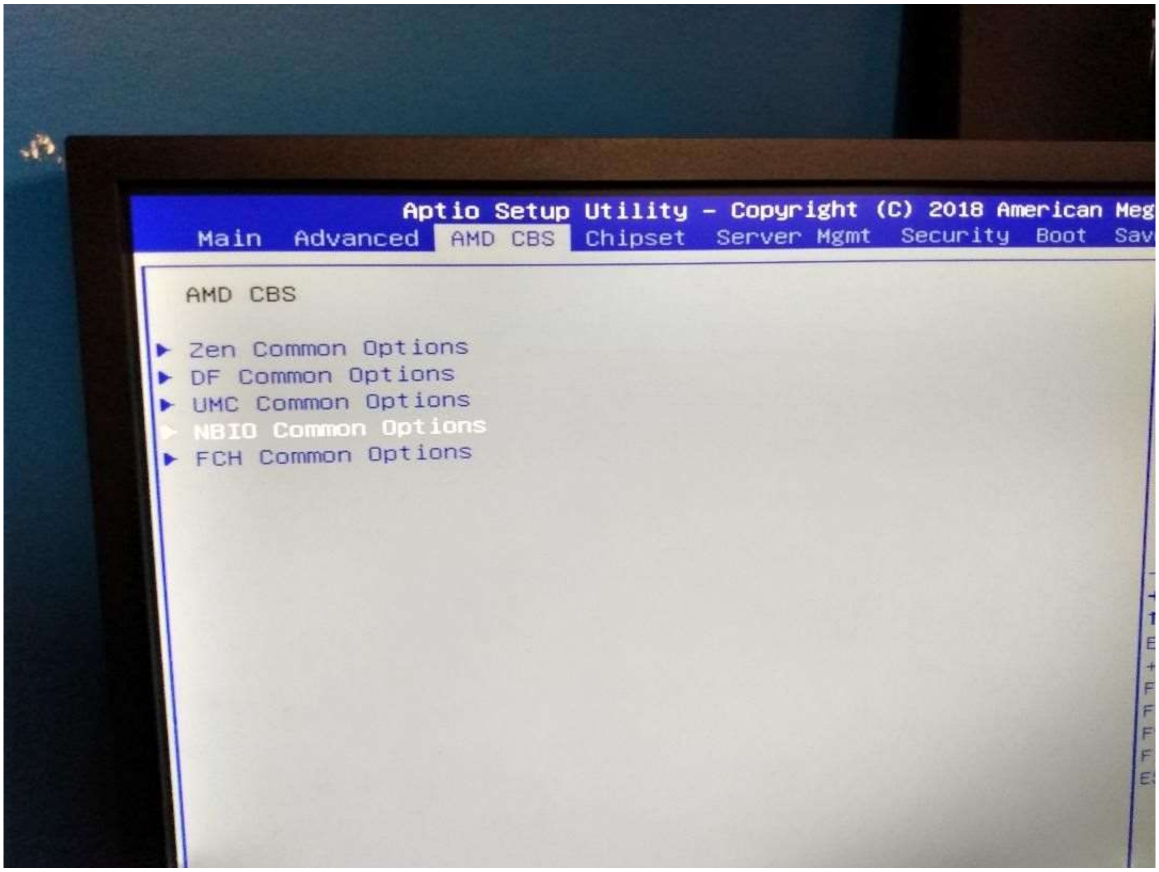
```
sudo vi /etc/default/grub
```

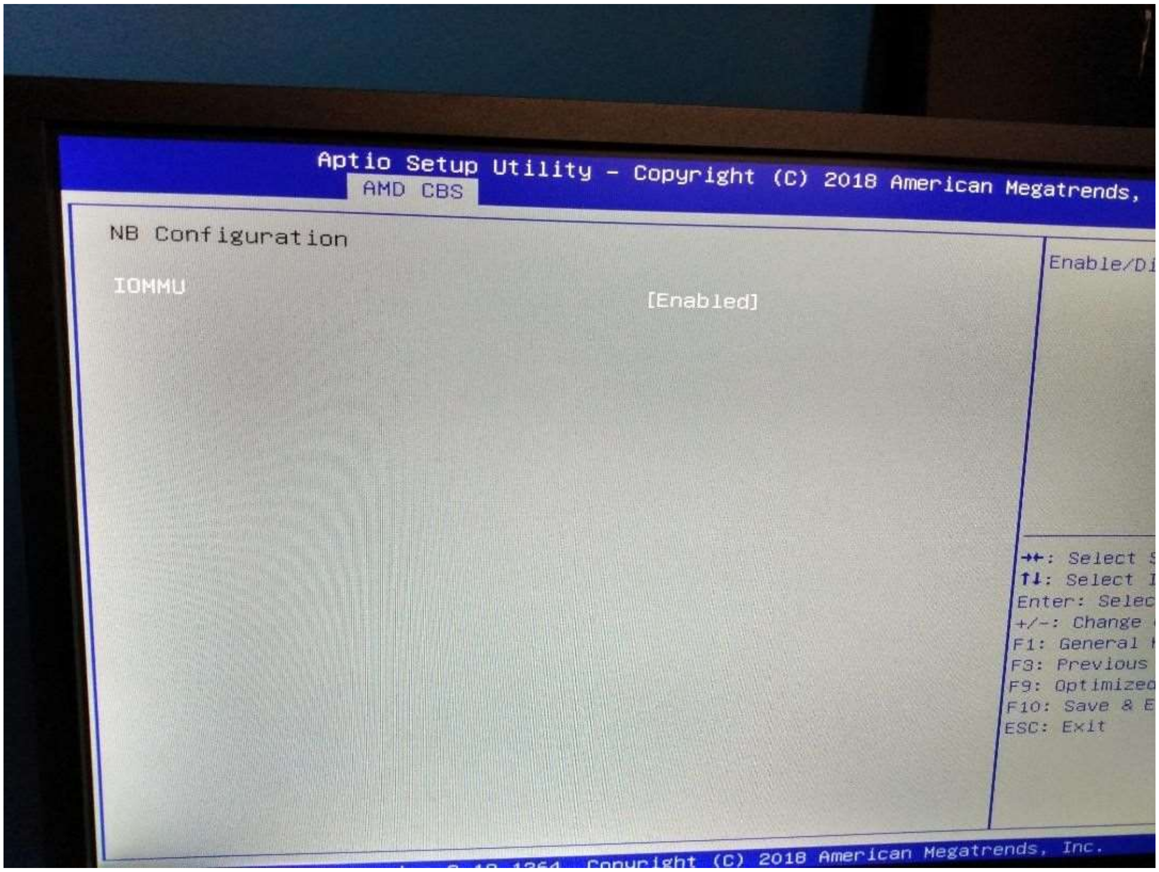
GRUB_CMDLINE_LINUX="amd_iommu=on iommu=pt"

Rest of the steps are the same as in section 3.10.

2] Updating BIOS

On AMD servers, the IOMMU setting may also need to be enabled in the BIOS. Refer picture snapshots below, for example.


Aptio Setup Utility – Copyright (C) 2018 American Meg
Main
Advanced
AMD CBS
Chipset
Server Mgmt
Security
Boot
AMD CBS
Zen Common Options
DF Common Options
UMC Common Options
NBIO Common Options
FCH Common Options




## 4.11 Adding T408 Device to vfio.conf

Check if /etc/modprobe.d/vfio.conf file exists. If not, the create one. If it exists then modify the contents to match as follows:

```
sudo vi /etc/modprobe.d/vfio.conf
```

and add the following line

options vfio-pci ids=1d82:0202,1d82:0600

```
[nvme@nvme-cli129 ~]$ sudo cat /etc/modprobe.d/vfio.conf
options vfio-pci ids=1d82:0202,1d82:0600
[nvme@nvme-cli129 ~]$
```

Check if vfio-pci.conf file exists. Create one or edit the existing file.

```
sudo vi /etc/modules-load.d/vfio-pci.conf
```

vfio-pci

[nvme@nvme-cli129 ~]$ sudo cat /etc/modules-load.d/vfio-pci.conf

vfio-pci

[nvme@nvme-cli129 ~]$

Confirm no error messages when running the following two commands. This is to ensure vfio-pci module is correctly installed and supported on the Linux host.

```
sudo modprobe vfio-pci
```

```
sudo modinfo vfio-pci
```

## 4.12 Creating VFs for T408 Device

Load a SR-IOV enabled firmware to the T408 device. For example, release 2.2.4.

Perform the following steps to create the VFs (virtual functions) on the T408 device.

to identify the 'pci device id' of the T408 device,

```
lspci -nnk
```

identify the pci device id and use it for the next step

example,

db:00.0 Non-Volatile memory controller [0108]: NETINT Technologies Inc. Device [1d82:0202]

Subsystem: NETINT Technologies Inc. Device [1d82:0000]

Kernel modules: nvme

```
sudo sh -c "echo 7 > /sys/bus/pci/devices/<pci device
id>/sriov_numvfs"
```

```
sudo cat /sys/bus/pci/devices/<pci device id>/sriov_numvfs
```

[nvme@nvme-cli129 ~]$ sudo cat /sys/bus/pci/devices/0000:da:00.0/sriov_numvfs

7

[nvme@nvme-cli129 ~]$

in this example, we have created 7 VFs. For your purpose, you can create 1 - 7 VFs

lspci -nnk

The output should show 8 instances of the T408 i.e. 1 PF (physical functions) and 7 VFs (virtual functions).

```
sudo nvme list
```

The output should show the 8 instances of the T408 device.

Note – Some Linux host OS assign new VFs to use vfio-pci driver by default. In latter, case the VFs will not show up in 'sudo nvme list' output. To confirm the VFs, run the following command:

```
lspci -nnk -d 1d82:
```

## 4.13 Attaching VF to VM

Perform the following steps. For this example, we use Virtual Function instance 7.

```
sudo sh -c "echo 0000:da:00.7 > /sys/bus/pci/drivers/nvme/unbind"
```

```
sudo sh -c "echo 0000:da:00.7 > /sys/bus/pci/drivers/vfio-pci/bind"
```

```
sudo virsh nodedev-dettach pci_0000_da_00_7
```

```
lspci -nnk -d 1d82:
```

In the lspci output, check and confirm that the driver associated to da:00.7 pci device is vfio-pci

63:00.7 Non-Volatile memory controller [0108]: NETINT Technologies Inc. Device [1d82:0600]

Subsystem: NETINT Technologies Inc. Device [1d82:0000]

Kernel driver in use: vfio-pci

Kernel modules: nvme

## 4.14 Starting a VM

From a Linux terminal (Linux VM running on your Windows laptop, for example), ssh to Host PC and start virt-manager.

```
ssh -X nvme@cli129
```

```
virt-manager
```

In virt-manager gui, select the VM, for example, centos71 and then select 'Open' and click 'View → Details'.

Do not start the VM yet i.e. the VM needs to be in shutoff state for the next step.

In 'View → Details', select 'Add Hardware' and add the 'PCI device'. Select the PCI device id corresponding to the VF, for example da:00.7, and click 'Finish' to close the window.

Start the VM, and open the command display window.

The VM will start up and provide a command prompt.

```
sudo cd /dev
```

The /dev folder should show a 'nvme0' and 'nvme?n?' device created.

```
lspci -nnk
```

The output of lspci should also show the T408 device. This means that the T408 VF has now successfully attached to the VM and is ready for use.

We can't run the 'sudo nvme list' since the VM does not have the package installed by default. This will be done in the next step.

## 4.15 Installing Build Environment on VM

On VM run the following steps:

```
sudo yum install nvme-cli gcc gcc-c++ python3-devel nasm pciutils
pkgconfig
```

Edit the /etc/bashrc and / etc/profile files to add highlighted text below.

```
sudo vi /etc/bashrc
```

export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig

export LD_LIBRARY_PATH=/usr/local/lib/

```
sudo vi / etc/profile
```

export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig

```
source /etc/bashrc
```

Edit the /etc/ld.so.conf file to add highlighted text below.

```
sudo vi /etc/ld.so.conf
```

/usr/local/lib

Update the default PATH variable for sudo user.

```
sudo visudo
```

Defaults secure_path = "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/usr/local/lib"

Note – If installing and using a Ubuntu VM, the build environment will need some different steps

```
sudo apt-get install make
sudo apt-get install make-guile
sudo apt-get install gcc  cmake  libboost-all-dev  pkg-config
sudo apt-get install --reinstall g++
sudo apt-get install nasm
sudo apt-get install yasm
sudo apt-get install pciutils
sudo apt-get install nvme-cli
sudo apt-get install ssh
```

Note - some equivalent files in Ubuntu VM are /etc/bash.bashrc , /etc/profile, ~/.bashrc, ~/.profile , /etc/ld.so.conf, visudo

## 4.16 Installing Libxcoder and FFmpeg Environment on VM

Tar a libxcoder folder and a FFmpeg-n4.2.1 folder from host and scp to the VM.

On a host run the following steps.

```
cd ~/FFmpegXcoder
tar -zcvf libxcoder.tar.gz libxcoder
tar -zcvf FFmpeg-n4.2.1.tar.gz FFmpeg-n4.2.1
scp FFmpeg-n4.2.1.tar.gz nvme@10.9.131.237:~
scp libxcoder.tar.gz nvme@10.9.131.237:~
```

Note - replace 10.9.131.237 with the IP address of the VM

On the VM run the following steps.

```
mkdir ~/FFmpegXcoder
cp ~/FFmpeg-n4.2.1.tar.gz ~/FFmpegXcoder
cp ~/libxcoder.tar.gz ~/FFmpegXcoder
cd
```

```
cd ~/FFmpegXcoder

tar -zxvf FFmpeg-n4.2.1.tar.gz

tar -zxvf libxcoder.tar.gz

cd libxcoder

make clean

sh build.sh

cd ../FFmpeg-n4.2.1

make clean

sh build_ffmpeg.sh

sudo make install
```

## 4.17 Running a Transcoding session on VM

Perform a transcode operation as follows.

```
cp ~/FFmpegXcoder/libxcoder/test/1920x1080_ParkScene.264 ~

cd

sudo ldconfig

sudo ffmpeg -y -nostdin -hide_banner -vsync 0 -c:v h264_ni_dec -
dec 0 -i 1920x1080p_ParkScene.264 -vf scale=1920:1080 -xcoder-
params intraQP=27:intraPeriod=120 -c:v h265_ni_enc -enc 0 -
xcoder-params intraQP=27:intraPeriod=120 1920x1080p_Parkscene.265
```

The ffmpeg transcoding session should successfully complete

## 4.18 Useful Links

1] CentOS KVM install - https://www.cyberciti.biz/faq/how-to-install-kvm-on-centos-7-rhel-7-headless-server/

2] Ubuntu KVM install - https://www.cyberciti.biz/faq/installing-kvm-on-ubuntu-16-04-lts-server/

3] CentOS releases – https://mirrors.edge.kernel.org/centos/

4] Ubuntu releases - http://releases.ubuntu.com/

5] CentOS images for aarch64 - http://centos.ca-west.mirror.fullhost.io/centos/8.2.2004/isos/aarch64/

# 5 Appendix-A – Windows VM

## 5.1 Windows VM install notes

The KVM install and configuration steps as detailed in this document remain the same if goal is to install Windows VM on the host. This appendix provides notes on steps that may differ for Windows VM install per section 3.8 and 3.9.

1] Refer to section 4.2 Item 1] for steps to install windows 10 VM on linux host.

2] The virt-install command:

sudo virt-install --virt-type=kvm --name Windows101 --ram 1024 --vcpus=1 --cdrom=/var/lib/libvirt/boot/Windows10_English_x64_Oct16_2020.iso --noautoconsole --network=bridge=br0,model=virtio --graphics vnc --disk path=/var/lib/libvirt/images/windows101.qcow2,size=40,bus=virtio,format=qcow2

Note that the 'size=40'. The disk size needed for Windows VM is bigger than needed for linux VM. The Windows OS itself takes up atleast 20GB disk space.

2] If the Windows VM captures the mouse pointer, then use 'CTL+ALT' to toggle the mouse point out to Primary OS.

3] windows drivers – download and install virtio-win

wget https://fedorapeople.org/groups/virt/virtio-win/virtio-win.repo -O /etc/yum.repos.d/virtio-win.repo

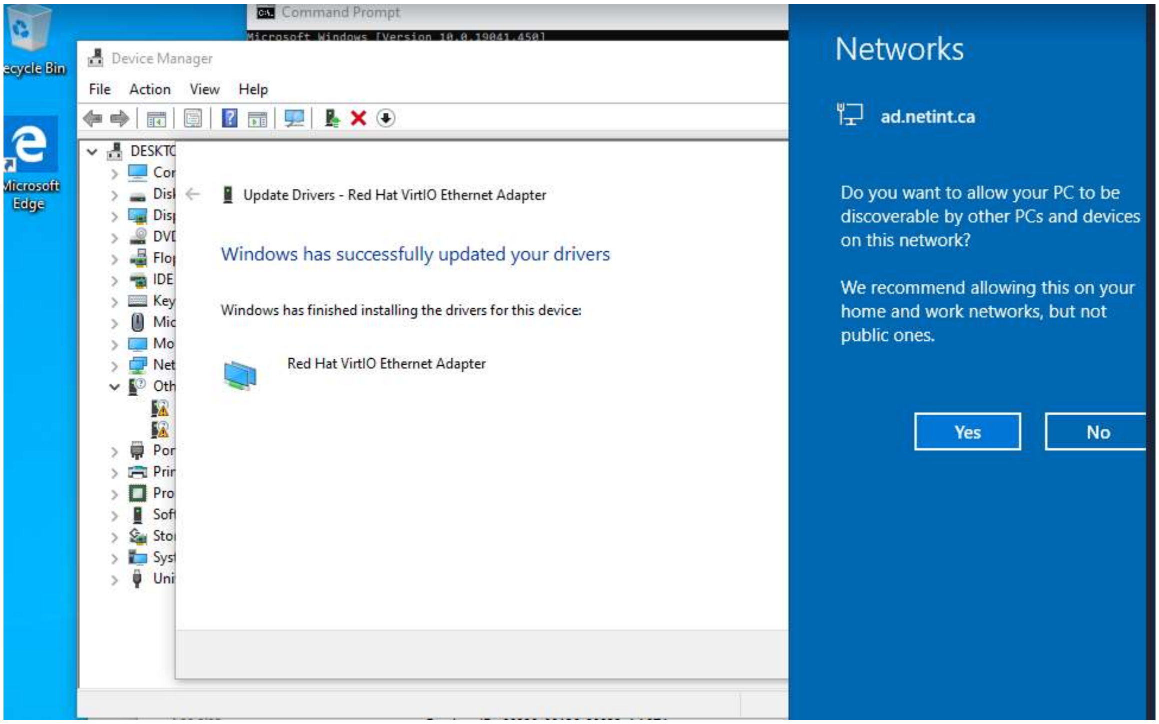
which dnf

sudo yum install virtio-win

sudo yum --enablerepo=virtio-win-latest upgrade virtio-win

4] Adding Windows 10 storage and hardware drivers for KVM – Refer Step 5 on section 4.2 item 1].

5] Once windows VM installation is complete, load windows drivers:



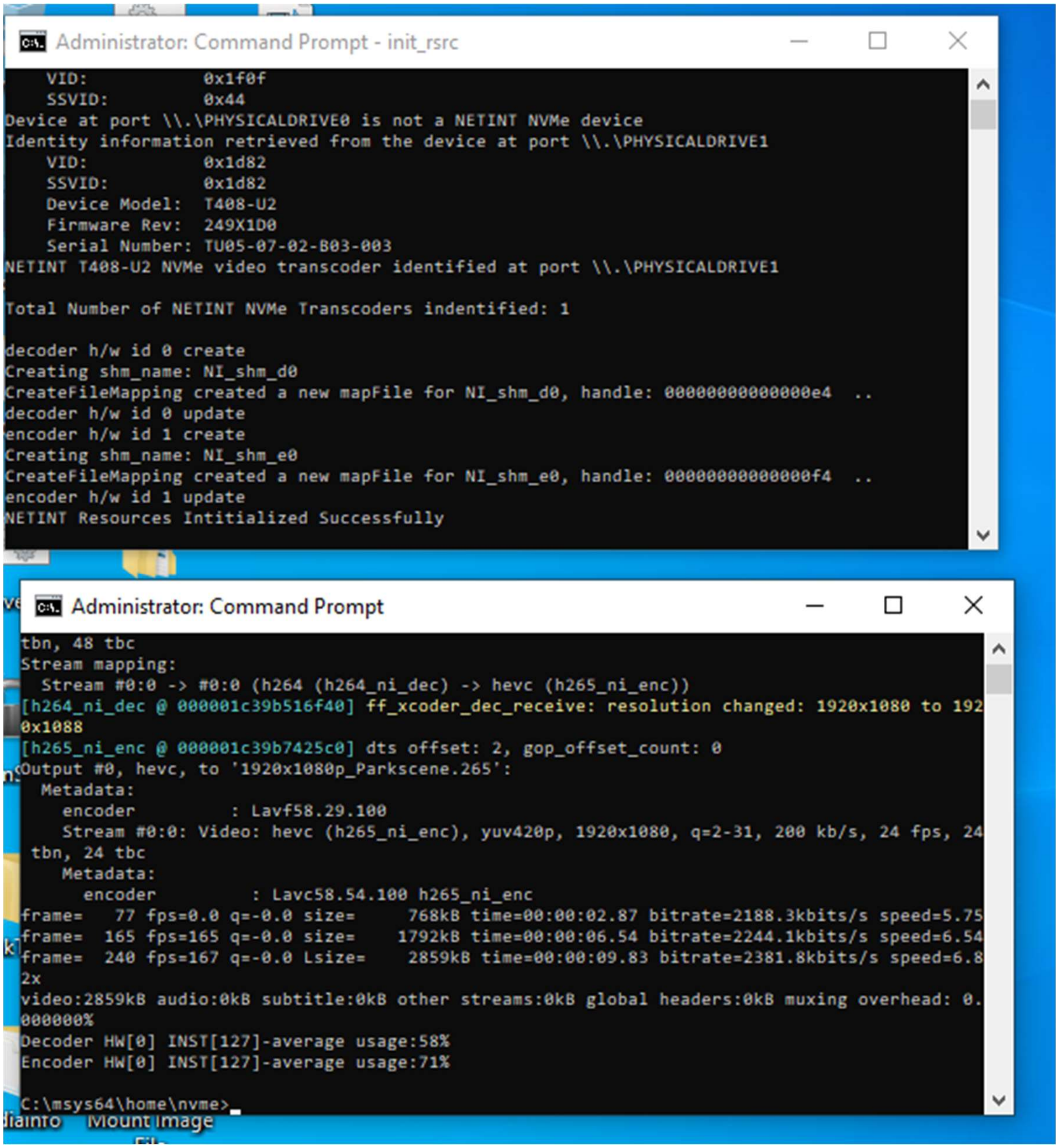

6] Firewall settings - Enable 'File and Printer Sharing' in Firewall settings. This will allow for host to be able to ping the windows VM.

7] Install msys2 in Windows VM. Refer section 4.2 item 2] link.

8] Use 'pacman' to install packages in msys2 to help compile FFmpegXcoder and libxcoder in windows VM. Always launch msys2 in admin mode.

9] ffmpeg can be run from windows command prompt also.

10] To perform transcoding using T408, start cmd window with admin privilege and leave init_rsrc running in one window. Run ffmpeg transcoding in a second window.




## 5.2 Useful Links

1] https://getlabsdone.com/10-easy-steps-to-install-windows-10-on-linux-kvm/

2] https://www.msys2.org/

# 6 Revision History

| Version | Change | Create Date | Author |
|---|---|---|---|
| 1.0 | Document Created | December 19, 2020 | Balwinder Sidhu |
| 1.1 | Add Appendix A – Windows VM | January 5, 2021 | Balwinder Sidhu |
| 1.2 | Add section 3.10.1 | September 14, 2021 | Balwinder Sidhu |
| 2.0 | Updated as per new app note template | December 7, 2021 | Balwinder Sidhu |

# 7 Legal Notice

Information in this document is provided in connection with NETINT products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in NETINT's terms and conditions of sale for such products, NETINT assumes no liability whatsoever and NETINT disclaims any express or implied warranty, relating to sale and/or use of NETINT products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right.

A "Mission Critical Application" is any application in which failure of the NETINT Product could result, directly or indirectly, in personal injury or death. Should you purchase or use NETINT's products for any such mission critical application, you shall indemnify and hold NETINT and its subsidiaries, subcontractors and affiliates, and the directors, officers, and employees of each, harmless against all claims costs, damages, and expenses and reasonable attorney's fees arising out of, directly or indirectly, any claim of product liability, personal injury, or death arising in any way out of such mission critical application, whether or not NETINT or its subcontractor was negligent in the design, manufacture, or warning of the NETINT product or any of its parts.

NETINT may make changes to specifications, technical documentation, and product descriptions at any time, without notice. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications.

NETINT, Codensity, and NETINT Logo are trademarks of NETINT Technologies Inc. All other trademarks or registered trademarks are the property of their respective owners.