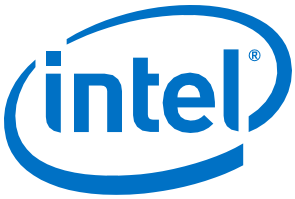
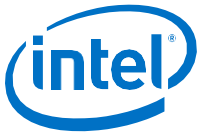


Visual Cloud Accelerator Card - Analytics Software Installation Guide

User Guide

Rev. 6.0

August 2020

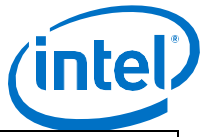


Contents

Revision History

Date	Revision	Description
August 2020	6.0	1.2 VCAC-A Contents of the Release on page 8: Updated contents and links of Release Package. 2.1.4 Build Host Kernel and VCAC-A Driver on Page 14: Updated kernel packages' names. 2.2.2 VCAC-A Software Installation on Host CentOS 8.1 on page 20: Update kernel and module installation files' name. 2.2.4.1 BIOS and EEPROM Version Check on page 21: Update latest BIOS information 2.2.5 VCAC-A Card Boot up with vcad Image on page 23: Update system image name. 2.2.6 VCAC-A Card and Host Network Setting Configuration on page 25: Update Network Manager settings. Appendix A Appendix - Sample Output Message for benchmark_app on page 33: Update the print out message Appendix D Network setting with firewall enabled on page 41: Update Content
May 2020	5.0	1.2 VCAC-A Contents of the Release on page 8: Updated contents and links of Release Package. 2.1.7 DL (Deep Learning) Stream installation opt-in/out on page 17: Added a new chapter Appendix D Network setting with firewall enabled on page 39: Added a new chapter
March 2020	4.0	1.1 Reference Architecture on page 6: Updated Ubuntu version. 1.2 VCAC-A Contents of the Release on page 8: Updated contents and links of Release Package. 2.1.1 System Requirements to Prepare a Build on page 10: Updated Docker information on CentOS8.1 2.1.4 Build Host Kernel and VCAC-A Driver on page 12: Updated links to kernel packages. 2.1.5 Build Host Kernel and VCAC-A Driver for CentOS 7.4 or 7.6 on page 14: Added a new chapter. 2.1.6 Build System Image for VCAC-A on page 14: Updated links and information about new versions. 2.2 Using VCAC-A System Image and Installation Package on page 16: Added new information about MMIO and BAR settings. 2.2.2 VCAC-A Software Installation on Host CentOS 8.1 on page 17: Updated links and commands. 2.2.4.1 BIOS and EEPROM Version Check on page 19: Updated BIOS and hardware information. 2.2.5 VCAC-A Card Boot up with vcad Image on page 21: Updated versions and commands. 2.2.6 VCAC-A Card and Host Network Setting Configuration on page 22: Added more information on Network Manager. 2.2.7 VCAC-A Card Software Installation on page 24: Updated commands and versions. 2.2.8.2 Validate Setup with benchmark_app on page 26: Updated download links for model files. Appendix A Appendix - Sample Output Message for benchmark_app on page 29: Updated sample output message. Appendix C Appendix - VPU Utilization and Debugging Capability on page 31: Added new chapter. C.1 Example Scenario: Check the VPU Device Utilization on page 31: Added new chapter. C.2 Example Scenario: Debugging Capability on page 33: Added new chapter. C.3 Example Scenario: Info Printer Setting on page 34: Added new chapter.

continued...



Date	Revision	Description
November 2019	3.0	1.2 VCAC-A Contents of the Release on page 8: Updated contents of Release Package and added information about Dockerfiles. 2.1.2 Configure Docker Root Dir on CentOS on page 10: Added new chapter. 2.1.4 Build Host Kernel and VCAC-A Driver on page 12: Added information about addressing installation issues of docker; updated links; added build.sh workflow diagram. 2.1.6 Build System Image for VCAC-A on page 14: Added workflow diagram for vcad_build.sh; updated commands and links. 2.1.7 Skip building Docker Image(Optional) on page 15: Updated links and commands. 2.1.8 Customize System Image Size on page 16: Updated command. 2.2 Using VCAC-A System Image and Installation Package on page 16: Reorganized the chapter. 2.2.4 BIOS and EEPROM Update on page 18: Added new chapter. 2.2.4.1 BIOS and EEPROM Version Check on page 19: Added new chapter. 2.2.4.2 Online Update BIOS (optional) on page 19: Added new chapter. 2.2.4.3 Update EEPROM (optional) on page 20: Added new chapter. 2.2.6 VCAC-A Card and Host Network Setting Configuration on page 22: Added additional steps for configuring NAT. 2.2.7 VCAC-A Card Software Installation on page 24: Updated links and reorganized content. 2.2.8 Run Sanity Test on page 25: Reorganized the chapter. 2.2.8.2 Validate Setup with benchmark_app on page 26: Reorganized the chapter to explain how to run benchmark_app sample. 3.0 Run Media Analytics Pipeline with FFmpeg/GStreamer or HOST on page 27: Added new chapter on FFmpeg and GStreamer pipeline description and links. 3.1 Run Media Analytics Pipeline with HOST on page 27: Added new chapter on running pipeline with HOST. 3.2.1 Run Media Analytics Pipeline with FFmpeg on page 27: Added new chapter. 3.2.2 Run Media Analytics Pipeline with GStreamer on page 28: Added new chapter. Appendix B Appendix - Troubleshooting NAT Configuration (Optional) on page 30: Added new chapter on optional NAT configuration.

continued...

Date	Revision	Description
September 2019	2.0	1.2 VCAC-A Contents of the Release on page 8: Updated contents of Initial Release Package; added descriptions of folders. 1.3 VCAC-A Hardware Configuration on page 9: Updated hardware configuration table for hard drive information; added links to the card specifications and hardware customer support. 2.1.1 System Requirements to Prepare a Build on page 10: Added specification of Docker; updated topic for clarity. 2.1.4 Build Host Kernel and VCAC-A Driver on page 12: Added timing information for build process; updated links used for downloading packages. 2.1.6 Build System Image for VCAC-A on page 14: Added timing information for build process; updated links for downloading the binaries. 2.1.7 Skip building Docker Image(Optional) on page 15: Updated topic for clarity. 2.1.8 Customize System Image Size on page 16: Added new chapter. 2.2 Using VCAC-A System Image and Installation Package on page 16: Added commands to check BIOS and EEPROM version. 2.2.1 Setting up Intel® Xeon® Scalable Processor Server Host on page 16: Deleted extra commands which were optional. 2.2.2 VCAC-A Software Installation on Host CentOS 8.1 on page 17: Updated file path of driver. 2.2.3 VCAC-A Software Installation on Host CentOS 7.x on page 18: Updated file path of driver. 2.2.5 VCAC-A Card Boot up with vcad Image on page 21: Added content which indicates health status of the VCAC-A card; added disk space requirement for vcad image; updated commands for clarity. 2.2.6 VCAC-A Card and Host Network Setting Configuration on page 22: Added steps for clarity. 2.2.7 VCAC-A Card Software Installation on page 24: Added information about sample application benchmark_app from OpenVINO; updated section for clarity. 2.2.8 Run Sanity Test on page 25: Added new chapter on running sanity test.
July 2019	1.0	Release 1.0
May 2019	0.1	Initial document version.

1.0 Introduction

Visual Cloud Accelerator Card - Analytics (VCAC-A) is a PCIe add on card comprising of Intel® Core™ i3-7100U Processor with Intel® HD Graphics 620 and 12 Movidius® VPUs. The addition of this card in any system focuses on computer vision, video decoding, and media analytics but is not only limited to them.

This document covers installation of Software release package for VCAC-A. The release package consists of source code, libraries, user mode service/agent components and kernel mode patches.

NOTE

Please get in touch with Intel Field Representative to receive latest Software Release Package.

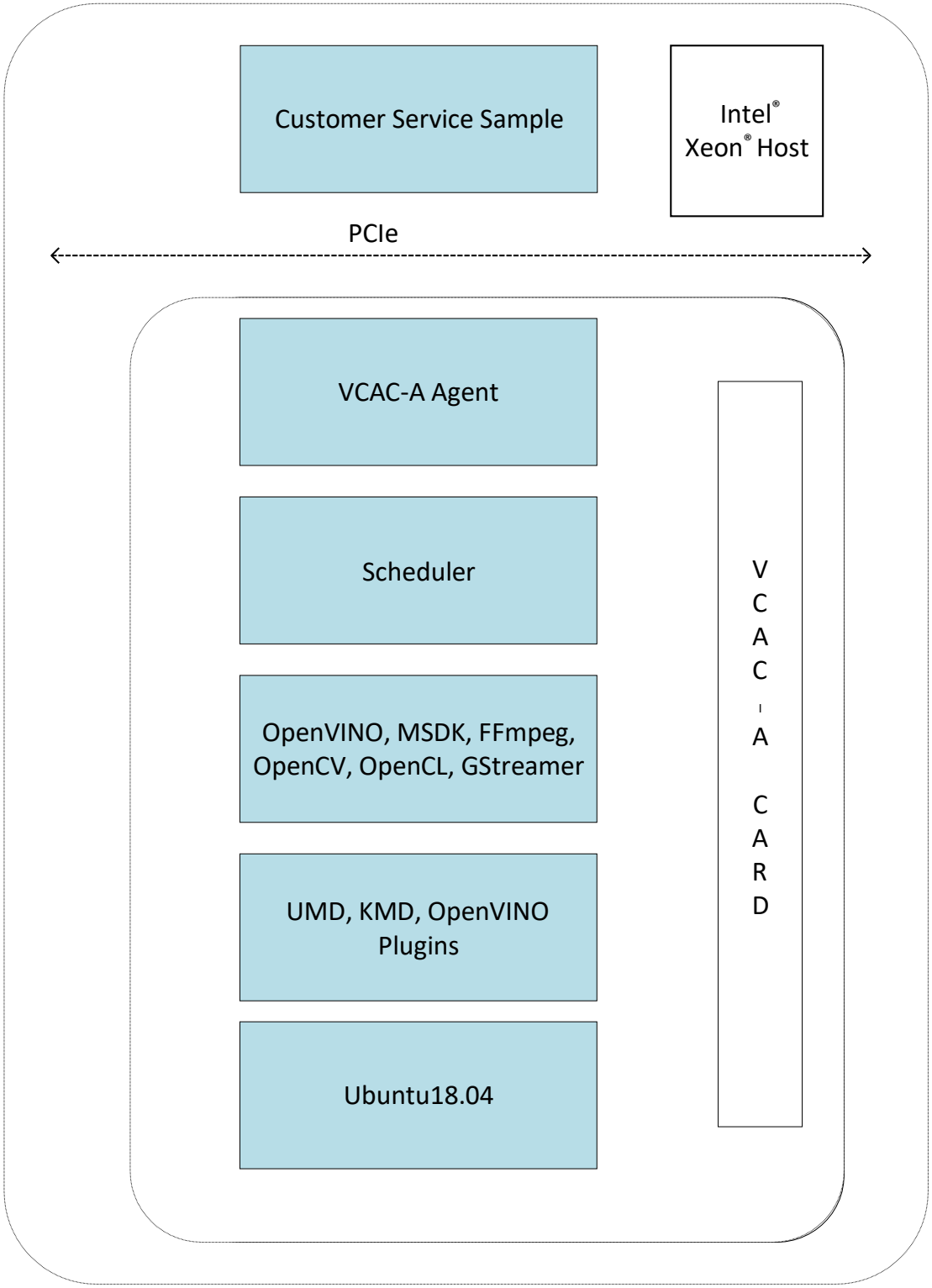
1.1 Reference Architecture

The VCAC-A software runs on the host and target (VCAC-A card) as follows:

1. VCAC-A service on the host
2. VCAC-A agent and all supported software on the VCAC-A card.

The architecture diagram shows the distribution of the software components.

Figure 1. Visual Cloud Analytics Accelerator Diagram



1.2 VCAC-A Contents of the Release

The contents of VCAC- A software release are listed in the table below. Intel drives these ingredients through continuous development and validation to ensure that software updates will perform correctly when integrated into a deployed system.

Download the release content from the link: https://github.com/OpenVisualCloud/VCAC-SW-Analytics/archive/VCAC-A-R6.tar.gz

NOTE

The version of the release software might change. Please refer to the Release Notes (https://github.com/OpenVisualCloud/VCAC-SW-Analytics/blob/release/VCAC-A/R6/VCAC-A/Documents/VCAC-Analytics-releasenotes-rev6-0.pdf) accompanied with the latest Software Releases.

Table 1. VCAC-A Release Contents

Ingredient	Folder Name	Contents
VCAC-A Host Files (Intel_Media_Analytics_Host)	scripts	Scripts to build kernel and VCAC-A PCIe driver module and Dockerfile for creating the docker container of CentOS 8.1, called by the script to build kernel and driver modules
	tar	VCAC-A host supporting packages: kernel patch, VCAC-A PCIe driver patch and utilities
VCAC-A Software Package (Intel_Media_Analytics_Node)	scripts	Script to build the system image to be loaded on VCAC-A card and Dockerfile for creating the docker container of Ubuntu 18.04, called by the script to build system image to be loaded on VCAC-A.
	tar	Patches for kernel on card Installation package of VPU metrics
VCAC-A Documents	Documents	Installation guide (this document) Release Notes

Detailed software contents are listed as below:

VCAC-A host files:

- centos8.1-kernel4.18.0-147-patch.tar.gz
- vcass-modules-R4-patch.tar.gz
- vca_query-1.0_centos8-1.x86_64.rpm
- vca_query-1.0_centos7-1.x86_64.rpm
- centos7.4-kernel3.10.0-patch.tar.gz
- vcass-modules-3.10.0-patch.tar.gz
- build.sh
- Dockerfile-CentOS8
- Dockerfile-CentOS7

VCAC-A card packages:

- ubuntu18.04_kernel5.13.18_patch.tar.gz
- intel-vpu-metric-ubuntu18.04-amd64.deb
- vcad_build.sh
- Dockerfile

VCAC-A documents:

- VCAC-Analytics-releasenotes-rev6-0.pdf
- VCAC-Analytics-software-installation-guide-rev6-0.pdf

1.3 VCAC-A Hardware Configuration

A Server with 2nd Generation Intel® Xeon® processor is recommended as the host to mount the VCAC-A card through PCIe.

NOTE

One of the example system configurations (assumed for all workloads described in this document) is listed in table below:

Table 2. System Hardware Configuration

Component Type	Part Name
KNOCK DOWN KIT	Intel® Server System R2312WFTZS 12x3.5in 2x10GbE
CPU	2nd Generation Intel® Xeon® Scalable Processor ***2**
MEMORY	16GB 2666 Reg ECC 1.2V DDR4 Micron MTA18ASF2G72PDZ-2G6D1 ***12**
ATA HARD DRIVE	480 GB Intel® SSD SATA or Equivalent Boot Drive
Add-in Card	VCAC-A card
NETWORK ADAPTER	On Board
CHASSIS COMPONENT	PCIe 2U Riser Spare Intel® A2UL16RISER2 (2 Slot)
CHASSIS COMPONENT	Passive Airduct Bracket Kit Intel® AWFCOPRODUCTBKT
CHASSIS COMPONENT	Passive Airduct Kit Intel® AWFCOPRODUCTAD
HEATSINK	Included

Table 3. VCAC-A Hardware Configuration

Component Type	Part Name
Form Factor	Full Height (126mm), ¾ Length (254mm) PCIe Adapter
Host Interface	PCIe Gen3 x4
On-board CPU	Intel® Core™ i3-7100U Processor, 2.4 GHz
Memory	2x DDR4 4GB SODIMM
VPU (Inference Acceleration)	12x Myriad X MA2485
Power Consumption	MAX75W
Thermal Cooling	Passive Heat Sink
Operating Temperature	0 ° - 55 ° C @ 15CFM airflow

Hardware specifications datasheet published by Intel is available here: http://intel.com/content/dam/www/public/us/en/documents/datasheets/media-analytics-vcac-a-accelerator-card-by-celestica-datasheet.pdf.

Hardware Customer Support is available from Celestica™ https://hardwaresupport.celestica.com.

NOTE

Customers need to work with their Celestica sales representative to get credentials for the website.

2.0 Installation

2.1 Build Host Kernel, VCAC-A Driver on Host and VCAC-A System Image

Following sections describe how to build host kernel, VCAC-A driver on host and VCAC-A system image with scripts included in the release package.

2.1.1 System Requirements to Prepare a Build

Build the kernel/driver and system image on the Intel® Xeon® host machine which has the VCAC-A card plugged in. They can also be built on any other machine with the following requirements of build environment:

- Linux based OS: the scripts are tested on CentOS 8.1
- Docker installed: Docker is used in the build process. Refer to https://docs.docker.com/install/ for how to install Docker Community version (docker-ce). containerd.io >= 1.2.2-3 is required in CentOS 8.1, Install it via "yum -y install https://download.docker.com/linux/fedora/30/x86_64/stable/Packages/containerd.io-1.2.6-3.3.fc30.x86_64.rpm"
- User privilege: root account (or sudo) is required to run the script.

2.1.2 Configure Docker Root Dir on CentOS

By default, docker runs in folder `/var/lib/docker`. Check the location via the following command:

```
#docker info|grep Dir
Docker Root Dir: /var/lib/docker
```

The folder is under the `/dev/mapper/cl-root` folder and the size of the folder is 50GB in CentOS. The space is not enough for building the vcad and "no space left on device" might occur during compiling. It is recommended to change the Docker Root Dir to `/home` folder to avoid the size limit message:

```
#cd /home
#mkdir docker
```

Open configuration file " `/usr/lib/systemd/system/docker.service`", add "`--graph /home/docker`" after "`ExecStart=/user/bin/dockerd`"

```
[Service]
Type=notify

# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
```

```
# for containers run by docker

ExecStart=/usr/bin/dockerd --graph /home/docker -H fd:// --containerd=/run/
containerd/containerd.sock
ExecReload=/bin/kill -s HUP $MAINPID
```

Restart docker:

```
#systemctl daemon-reload
#systemctl restart docker
```

Check if Docker Root Dir has been changed via:

```
#docker info|grep Dir
Docker Root Dir: /home/docker
```

2.1.3 Setup Proxy for Docker (if needed)

Internet connection is needed to build Docker container, download source code and dependencies. If you are behind proxy, it needs to be setup for Docker.

2.1.3.1 Setup for Docker daemon

Take `systemd` for example: Create a `systemd` drop-in directory for the Docker service:

```
# sudo mkdir -p /etc/systemd/system/docker.service.d
```

Create a file called `/etc/systemd/system/docker.service.d/http-proxy.conf` that adds the proxy environment variable:

```
[Service]
Environment="HTTP_PROXY=http://your.http-proxy-server.com:port"
"HTTPS_PROXY=http://your.https-proxy-server.com:port"
"NO_PROXY=localhost,127.0.0.1,*.your-company.com"
```

Flush changes:

```
#sudo systemctl daemon-reload
```

Restart Docker:

```
#sudo systemctl restart docker
```

Refer to https://docs.docker.com/config/daemon/systemd/ for details.

2.1.3.2 Setup for Docker client:

Create or edit the file `~/.docker/config.json`

```
{
  "proxies":
  {
    "default":
    {
      "httpProxy": "http://your.http-proxy-server.com:port",
      "httpsProxy": "http://your.https-proxy-server.com:port",
      "noProxy": "localhost,127.0.0.1,*.your-company.com"
    }
  }
}
```

2.1.3.3 DNS setup in Docker

This is an optional step, in case build still cannot complete due to network issue. Setup the DNS for Docker.

Find your network's DNS server:

```
# nmcli dev show | grep 'IP4.DNS'
  IP4.DNS[1]: your.local.dns.server1
  IP4.DNS[2]: your.local.dns.server2
```

Open or create(if it doesn't exist), `/etc/docker/daemon.json` and add DNS settings:

```
# /etc/docker/daemon.json
  {
  "dns": ["your.local.dns.server1", "your.local.dns.server2"]
  }
```

Restart the Docker daemon:

```
 $ sudo service docker restart
```

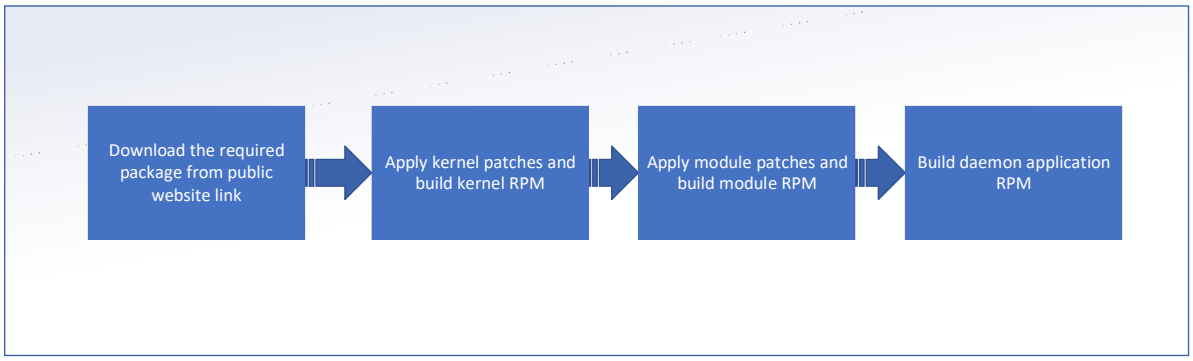
2.1.4 Build Host Kernel and VCAC-A Driver

Build host kernel and VCAC-A driver modules with the script included in the release package:

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Host/scripts/build.sh
```

The diagram below shows each step during the whole build process with "build.sh" to give a general overview of the workflow.

Figure 2. build.sh workflow diagram



If user gets stuck when the container tries to do a yum install even with proxy set, follow Docker BKM https://docs.docker.com/install/linux/linux-postinstall/ to allow running docker without sudo access.

After compilation is completed, the output will be available in `/PATH/TO/PACKAGE/Intel_Media_Analytics_Host/build/host_packages` ("fb4dfe2" in the name of the example output files were randomly generated).

NOTE

The build process takes approximately 1 hour. The duration might vary depending upon the capabilities of the machine and local network speed to download the dependent packages.

- Host kernel packages:
  - `kernel-4.18.0.127fbe8.VCA+-1.x86_64.rpm`
  - `kernel-devel-4.18.0.127fbe8.VCA+-1.x86_64.rpm`
- VCAC-A driver module: `vcass-modules-4.18.0.127fbe8.VCA+-3fa7412-0.x86_64.rpm`
- VCAC-A Utility files: `daemon-vca-2.7.3-centos8-x86_64.rpm`

During the build process, following packages will be downloaded:

- VCAC-A driver modules "`VCAC-A-R6.tar.gz`" : https://github.com/OpenVisualCloud/VCAC-SW/archive/VCAC-A-R6.tar.gz
- Host kernel base "`kernel-4.18.0-147.3.1.el8_1.src.rpm`" : http://vault.centos.org/8.1.1911/BaseOS/Source/SPackages/kernel-4.18.0-147.3.1.el8_1.src.rpm

If you want to skip downloading source code and dependencies, put the files under `/PATH/TO/PACKAGE/cache`, then pass "-c" flag to the build script (this could save the time used for downloading the packages.), e.g.

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Host/scripts/build.sh -c
```

2.1.5 Build Host Kernel and VCAC-A Driver for CentOS 7.4 or 7.6

CentOS 7.4 and CentOS 7.6 for Xeon host is also supported. Build kernel and VCAC-A drivers for CentOS 7.4/7.6 via option "`-o centos7`"

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Host/scripts/build.sh -o centos7
```

If you want to skip downloading source code and dependencies, put the following files under /PATH/TO/PACKAGE/cache :

- VCAC-A driver modules "VCAC-SW-VCAC-A_R2.tar.gz" : https://github.com/OpenVisualCloud/VCAC-SW/archive/VCAC-A_R2.tar.gz
- Host kernel base "kernel-3.10.0-693.17.1.el7.src.rpm" : http://vault.centos.org/7.4.1708/updates/Source/SPackages/kernel-3.10.0-693.17.1.el7.src.rpm

Then pass "-c" flag to the build script:

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Host/scripts/build.sh -c -o centos7
```

After compilation is completed, the output will be available in `/PATH/TO/PACKAGE/Intel_Media_Analytics_Host/build/host_packages` ("fb4dfe2" in the name of the example output files were randomly generated).

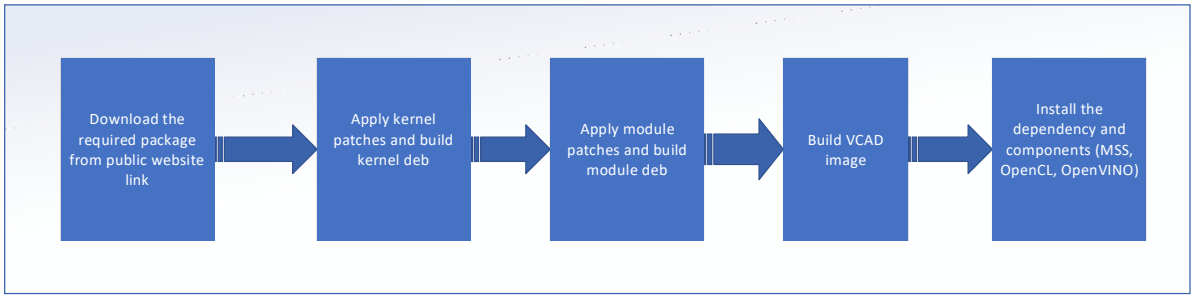
- Host kernel packages:
  - kernel-3.10.0_1.fb4dfe2.VCA+-1.x86_64.rpm
  - kernel-devel-3.10.0_1.fb4dfe2.VCA+-1.x86_64.rpm
- VCAC-A driver module:
  - vcass-modules-3.10.0_1.fb4dfe2.VCA +-1.690990a-0.x86_64.rpm
- VCAC-A Utility files:
  - daemon-vca-2.7.3-centos7-x86_64.rpm

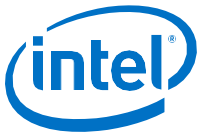
2.1.6 Build System Image for VCAC-A

Build VCAC-A system image with script included in the release package:

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/vcad_build.sh -o FULL
```

Figure 3. vcad_build.sh workflow diagram





During the build process, following packages will be downloaded:

- VCAC-A on card kernel base "`linux_5.3.0.orig.tar.gz`": https://launchpad.net/ubuntu/+archive/primary/+sourcefiles/linux/5.3.0-53.47/linux_5.3.0.orig.tar.gz

  On card kernel diff patch "`linux_5.3.0-53.47.diff.gz`": https://launchpad.net/ubuntu/+archive/primary/+sourcefiles/linux/5.3.0-53.47/linux_5.3.0-53.47.diff.gz
- Intel GPU firmware binary "`kbl_dmc_ver1_04.bin`": https://cgit.freedesktop.org/drm/drm-firmware/plain/i915/kbl_dmc_ver1_04.bin
- VCAC-A driver modules "VCAC-A-R6.tar.gz": https://github.com/OpenVisualCloud/VCAC-SW/archive/VCAC-A-R6.tar.gz
- OpenVINO package "l_openvino_toolkit_p_2020.4.287.tgz": http://registrationcenter-download.intel.com/akdlm/irc_nas/16803/l_openvino_toolkit_p_2020.4.287.tgz
- MSS package "MediaServerStudioEssentials2019R1HF3_16.9_10020.tar.gz": https://github.com/Intel-Media-SDK/MediaSDK/releases/download/MSS-KBL-2019-R1-HF1/MediaServerStudioEssentials2019R1HF3_16.9_10020.tar.gz
- numpy files: https://files.pythonhosted.org/packages/62/20/4d43e141b5bc426ba38274933ef8e76e85c7adea2c321ecf9ebf7421cedf/numpy-1.18.1-cp36-cp36m-manylinux1_x86_64.whl
- OpenCV files: https://files.pythonhosted.org/packages/c0/a9/9828dfaf93f40e190ebfb292141df6b7ea1a2d57b46263e757f52be8589f/opencv_python-4.1.2.30-cp36-cp36m-manylinux1_x86_64.whl

NOTE

The build process takes approximately 1 hour. The duration might vary depending upon the capabilities of the machine and local network speed to download the dependent packages.

The build script will always download the packages listed above for each time of execution. If the packages were already downloaded and you want to skip downloading source code and dependencies, put the files under `/PATH/TO/PACKAGE/cache`, then pass "-c" flag to the build script, e.g.

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/vcad_build.sh -c -o FULL
```

After compilation is completed, the output will be available in `/PATH/TO/PACKAGE/Intel_Media_Analytics_Node/build/vcad/INSTALL/`:

- System image loaded on VCAC-A card: `vca_disk48_k5.3_ubuntu18.04_1.0.1.gz`

2.1.7 DL (Deep Learning) Streamer installation opt-in/out

DL (Deep Learning) Streamer (GStreamer* Video Analytics Plugin, GVA) is included in OpenVINO 2020.2. And it will take about 300M disk space, and installed to "/opt/intel/openvino/data_processing/dl_streamer" and "/opt/intel/openvino/data_processing/gstreamer"

```
DL (Deep Learning) Streamer (GStreamer* Video Analytics Plugin, GVA) is included in
OpenVINO 2020.2. DL Streamer will take about 300M disk space.
Choose yes/y or no/n to select to install DL Streamer or not:
```

After start vcad_build.sh with option "-o FULL" or "-o EXTENDED", below message will pop up to opt-in/out DL Streamer while installing OpenVINO:

Silent mode is also provided with option "--silent silent.cfg", e.g.

The DL streamer opt-in/out is controlled by "GVA_INSTALL" parameter in silent.cfg under (/PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/). Set "GVA_INSTALL=yes" to opt-in, and "GVA_INSTALL=no" to opt-out.

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/vcad_build.sh -o FULL --
silent /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/silent.cfg
```

2.1.8 Skip building Docker Image (Optional)

Every time this build script is executed, docker images of the build environment will be generated. These images are - docker image of CentOS for building driver/kernel on host, ubuntu image for building kernel on the VCAC-A card and vcad system image to be loaded on the card.

If the docker images have been generated in an earlier execution, user can save time required to build them again by the following executions:

```
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Host/scripts/build.sh -c -s
#sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/vcad_build.sh -c -s -o
FULL
```

2.1.9 Customize System Image Size

The system image will be mapped from host disk via blockIO to VCAC-A card. The size of the vcad system image is set to 48GB by default. And the system image size is configurable through passing flag "-e" followed by the image size measured in GB, e.g.

```
 #sudo /PATH/TO/PACKAGE/Intel_Media_Analytics_Node/scripts/vcad_build.sh -e 24 -o
FULL
```

NOTE

The default image size of 48GB is fully validated. User is recommended to keep the default image size.

2.2 Using VCAC-A System Image and Installation Package

This method requires that VCAC-A card is plugged in to the PCIe slot of the Intel® Xeon® Scalable Processor.

The host system BIOS must be configured to enable large Memory-Mapped Input/ Output (MMIO) and allow for large per-device BAR (Base Address Register) 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
```

2.2.1 Setting up Intel® Xeon® Scalable Processor Server Host

- Install CentOS8.1 in host Intel® Xeon® Scalable Processor server
- Set proxy in host (Optional, depends on local network environment):

```
# vim /etc/yum.conf
export https_proxy="local network https proxy"
export http_proxy="local network http proxy"
export ftp_proxy="local network ftp proxy"
```

- Before installing the VCAC-A software package on host, check PCIe device availability through the following:

```
# lspci | grep PLX
af:00.0 PCI bridge: PLX Technology, Inc. PEX 8717 16-lane, 8-Port PCI Express
Gen 3 (8.0 GT/s) Switch with DMA (rev ca)
b0:01.0 PCI bridge: PLX Technology, Inc. PEX 8717 16-lane, 8-Port PCI Express
Gen 3 (8.0 GT/s) Switch with DMA (rev ca)
b0:02.0 PCI bridge: PLX Technology, Inc. PEX 8717 16-lane, 8-Port PCI Express
Gen 3 (8.0 GT/s) Switch with DMA (rev ca)
b0:03.0 PCI bridge: PLX Technology, Inc. PEX 8717 16-lane, 8-Port PCI Express
Gen 3 (8.0 GT/s) Switch with DMA (rev ca)


# lspci | grep 295
18:00.1 System peripheral: Intel Corporation Device 2952 (rev ca)
1a:00.0 Bridge: Intel Corporation Device 2954 (rev ca)
af:00.1 System peripheral: Intel Corporation Device 2952 (rev ca)
b1:00.0 Bridge: Intel Corporation Device 2954 (rev ca)
```

2.2.2 VCAC-A Software Installation on Host CentOS 8.1

Follow the steps below to install the VCAC-A software on host CentOS 8.1:

- Copy the kernel and VCAC-A card driver package compiled in Build Host Kernel and VCAC-A Driver on page 13 to a temporary folder on the host server.
- Install the kernel packages:

```
#sudo yum -y localinstall kernel-4.18.0.127fbe8.VCA+-1.x86_64.rpm
#sudo yum -y localinstall kernel-devel-4.18.0.127fbe8.VCA+-1.x86_64.rpm
```

- Configure the new kernel to be the default at boot.

NOTE

This step is very important to boot the operating system with proper kernel.

```
While booting, in Grub Menu, select the installed VCA kernel "CentOS Linux
(4.18.0.127fbe8.VCA+)" manually
```

- Install the VCAC-A driver and utility. (must connected to internet, as dependencies will be installed during the installation process)

```
#sudo yum -y localinstall vcass-modules-4.18.0.127fbe8.VCA+-3fa7412-
0.x86_64.rpm
#sudo yum -y localinstall daemon-vca-2.7.3-centos8-x86_64.rpm
#sudo yum -y localinstall ../../tar/Centos8/vca_query-1.0_centos8-1.x86_64.rpm
```

- Reboot the system to enable the new kernel.

```
#sudo reboot
```

- After reboot, confirm that the expected kernel on the host is being used.

```
#uname -r
4.18.0.127fbe8.VCA+
```

- (Optional) If updating from a previous VCAC-A version, remove the older RPMs with the following command:

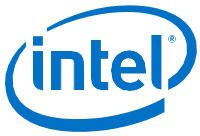
```
#rpm -qa | grep -e daemon-vca -e vcass-modules | xargs yum -y erase
```

2.2.3 VCAC-A Software Installation on Host CentOS 7.x

Follow the steps below to install the VCAC-A software on host:

- Copy the kernel and VCAC-A card driver package compiled in Build Host Kernel and VCAC-A Driver on page 15 to a temporary folder on the host server.
- Install the kernel packages:

```
#sudo yum -y localinstall kernel-3.10.0_1.fb4dfe2.VCA+-1.x86_64.rpm
#sudo yum -y localinstall kernel-devel-3.10.0_1.fb4dfe2.VCA+-1.x86_64.rpm
```



- Configure the new kernel to be the default at boot.

  NOTE

  This step is very important to boot the operating system with proper kernel.

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

- Install the VCAC-A driver and utility. (must connected to internet, as dependencies will be installed during the installation process)

```
#sudo yum -y localinstall vcass-modules-3.10.0_1.fb4dfe2.VCA
+-1.690990a-0.x86_64.rpm
#sudo yum -y localinstall daemon-vca-2.7.3-centos7-x86_64.rpm
#sudo yum -y localinstall ../../tar/Centos7/vca_query-1.0_centos7-1.x86_64.rpm
```

- Reboot the system to enable the new kernel.

```
#sudo reboot
```

- After reboot, confirm that the expected kernel on the host is being used.

```
#uname -r
3.10.0_1.fb4dfe2.VCA+
```

- (Optional) If updating from a previous VCAC-A version, remove the older RPMs with the following command:

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

2.2.4 BIOS and EEPROM Update

Latest BIOS is available at https://hardwaresupport.celestica.com, and EEPROM is available at https://01.org/openvisualcloud/download (in section "Visual Cloud Accelerator Cards - Github Repos & Binary").

If following error is encountered, the EEPROM and BIOS need to be updated:

```
vcactl status
Card: 0 Cpu: 0  STATE: link_down, caterr
```

2.2.4.1 BIOS and EEPROM Version Check

Check the BIOS and EEPROM version via `vcactl info` command, sample output as below:

```
#vcactl info BIOS 0 0
Version: VCAA.4
Release Date: 2020.03.19 11:21:54
```

```
#vcactl info hw 0 0
Card 0: VCAC-A,
EEPROM version: VCAC-A 1.3 (CRC:6cc483ef),
Serial Number: F0001CSS207CL9AT017
```

2.2.4.2 Online Update BIOS (optional)

Follow instructions below to update BIOS through `vcactl` commands via host in case the BIOS needs to be updated:

1. Check VCAC-A status via "`vcactl status 0 0`"

   If the card is not in "`bios_up`" state. Do step 2 and 3 to make sure the status is changed to "`bios_up`".

   If the card is in "`bios_up`" state, jump to step 4.

2. Power cycle the VCAC-A card:

```
#vcactl pwrbtn-long 0 0
#vcactl pwrbtn-short 0 0
```

3. Check VCAC-A status continuously with the following command till the status changes to "`bios_up`"

```
# vcactl status 0 0
Card: 0 Cpu: 0  STATE: bios_up
```

4. Update BIOS, this command takes about 2 minutes to be completed

```
#vcactl update-BIOS 0 0 /PATH/TO/BIOS/your_bios.img
Card: 0 Cpu: 0 - BIOS UPDATE STARTED. DO NOT POWERDOWN SYSTEM
Card: 0 Cpu: 0 - UPDATE BIOS SUCCESSFUL
Card: 0 Cpu: 0 - Node will power down and up automatically to make the change
active.
Please wait for 'bios_up' to start working with the node.
```

5. Check VCAC-A status continuously till the status is changed to "`bios_up`"

```
# vcactl status 0 0
Card: 0 Cpu: 0  STATE: bios_up
```

6. After status is changed to "`bios_up`", check BIOS version again to confirm if the bios was updated successfully

```
#vcactl info BIOS 0 0
```

2.2.4.3 Update EEPROM (optional)

Follow instructions below to update EEPROM through `vcactl` commands via host in case the EEPROM needs to be updated:

1. Check VCAC-A status via "`vcactl status 0 0`"

   If the card is not in "`bios_up`" state. Do step 2 and 3 to make sure the status is changed to "`bios_up`".

   If the card is in "`bios_up`" state, jump to step 4.

2. Power cycle the VCAC-A card:

```
#vcactl pwrbtn-long 0 0
#vcactl pwrbtn-short 0 0
```

3. Check VCAC-A status continuously with the following command till the status changes to "bios_up"

```
# vcactl status 0 0
Card: 0 Cpu: 0  STATE: bios_up
```

4. Update EEPROM with the following command.

NOTE

Please note that vcactl update-EEPROM command does not take "card id", so if there are multiple cards connected to one host, the EEPROM of all the cards will be updated.

```
#vcactl update-EEPROM /PATH/TO/EEPROM/your_eeprom.bin
update-EEPROM  eeprom_VCAA_v1.3_2.6.219.bin
Update EEPROM process started (for card 0). Do not power down system!
Update EEPROM for first PCIe switch successful!
Update EEPROM successful (for card 0). Reboot system is required to reload
EEPROM.
Update EEPROM process started (for card 1). Do not power down system!
Update EEPROM for first PCIe switch successful!
Update EEPROM successful (for card 1). Reboot system is required to reload
EEPROM.
```

If you see error like below :

```
ERROR: Found 0 matching eeprom binary entries for card 0 in input file, but
exactly 1 is required.
WARNING: Consider using `--skip-card-type-check' option
```

Then use command:

```
 #vcactl update-EEPROM /PATH/TO/EEPROM/your_eeprom.bin --force --skip-card-
type-check
```

5. Reboot host

```
reboot
```

6. Power cycle the card

```
#vcactl pwrbtn-long
#vcactl pwrbtn-short
```

7. Check VCAC-A status continuously with following command till the status is changed to "bios_up"

```
#vcactl status 0 0
```

8. After Card is up, check EEPROM version again to confirm if the bios was updated successfully

```
#vcactl info hw
```

2.2.5 VCAC-A Card Boot up with vcad Image

The system image will be mapped from host disk via blockIO to VCAC-A card. The size of the vcad system image is set to 48GB, so the host should have 48GB free space for the vcad image. This size can be customized as suggested here Customize System Image Size on page 17. Boot up the VCAC-A card with the vcad image with the steps listed below:

NOTE

LED lights on the VCAC-A card indicate the health status of the card:

- LED marked as "PW-LED" indicates the power status of the card.
- LED marked as "ERR" indicates that the card is in error state.

- Untar and load VCAC-A vcad image. The image was the output of steps done in Build System Image for VCAC-A on page 15 .

- Boot VCAC-A card

```
gzip -d vca_disk48_k5.3_ubuntu18.04_1.0.1.vcad.gz
#vcactl blockio open vcablk0 RW $PATH/vca_disk48_k5.3_ubuntu18.04_1.0.1.vcad


Note: If you have 2 VCAC-A cards on one host, use "0 0" and "1 0" in the
command to distinguish the cards:
#vcactl blockio close 0 0 vcablk0
#vcactl blockio open 0 0 vcablk0 RW $PATH/
vca_disk48_k5.3_ubuntu18.04_1.0.1.vcad

Note: One system image file cannot be loaded on two cards. Make a copy of the
system file, then load the copy to the 2nd card:

#cp vca_disk48_k5.3_ubuntu18.04_1.0.1.vcad
vca_disk48_k5.3_ubuntu18.04_1.0.1-copy.vcad
#vcactl blockio close 1 0 vcablk0
#vcactl blockio open 1 0 vcablk0 RW $PATH/vca_disk48_k5.3_ubuntu18.04_1.0.1-
copy.vcad
```

Check VCAC-A status:

```
#vcactl status
```

If status is anything other than "bios_up", for example "link_down" as in the following:

```
Card: 0 Cpu: 0  STATE:link_down,
```

Then try the steps below:

```
#vcactl pwrbtn-long 0 0
#vcactl pwrbtn-short 0 0
```

NOTE

If there are more than 1 cards mounted on host machine, then the commands `pwrbtn-long/pwrbtn-short` will power cycle all the cards. It is important to pass the parameter "`0 0`" like above to power cycle only `CARD 0`.

Check the status again via

```
#vcactl status
```

If status is "`bios_up`".

```
Card: 0 Cpu: 0  STATE:bios_up
```

Then boot the card via the following commands:

```
#vcactl reset 0 0 --force
#vcactl boot 0 0 vcablk0 --force

Wait for about 30 seconds, then check the status, the STATE should turn into
"net_device_ready"

#vcactl status
Card: 0 Cpu: 0  STATE: net_device_ready
```

- If the device status is ready, you can login into the VCAC-A card using its IP address.

  Check IP address of the VCAC-A. Following is an example of one host connected with two VCAC-A cards:

```
#vcactl network ip
Card 0 Cpu 0:
172.32.1.1
Card 1 Cpu 0:
172.32.2.1
```

- Login to VCAC-A Card 0 through `ssh`:

```
#ssh root@172.32.1.1
(password:vista1)
```

2.2.6 VCAC-A Card and Host Network Setting Configuration

Network settings need to be configured in VCAC-A card and host. Following steps are required to achieve them:

Log in to the host machine and configure NAT:

- Create the proxy file to add necessary proxy settings if proxy is required to connect to external network.

```
#touch /etc/yum.repos.d/10proxy
#vim 10proxy
Acquire::http::Proxy " local network http proxy ";
#scp /etc/yum.repos.d/10proxy root@172.32.1.1:/etc/apt/apt.conf.d/10proxy
```

- Setup DNS server on host and copy the configuration file to VCAC-A card

NOTE

In Ubuntu 18.04, the resolv.conf will be cleared after reboot. User will need to copy the file from host after each reboot.

```
#vim /etc/resolv.conf
nameserver name server 1 ip
nameserver name server 2 ip
nameserver name server 3 ip
search sh.intel.com
#scp /etc/resolv.conf root@172.32.1.1:/etc
```

- Disable the firewall

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

NOTE
If firewall needs to be enabled in Xeon host, please refer to Appendix D Network setting with firewall enabled on page 39 for how to setup network.

- Stop the Network Manager

```
#systemctl stop NetworkManager
```

NOTE

Network Manager needs to be "on" if Xeon host has a DHCP Ethernet to the network infrastructure. Otherwise IP loss will happen due to the IP lease expiration.

```
#systemctl start NetworkManager
```

As NetworkManager will try to manage VCAC-A card virtual interfaces (eth0/eth1) by default.

On CentOS 7.x User will need to setup a rule to avoid this, on CentOS 8 there is no need to set the rule manually as it was handled by the vca daemon utility:

```
#echo "ACTION==\"add\", SUBSYSTEM==\"net\", KERNEL==\"eth0\",
ENV{NM_UNMANAGED}=\"1\"" >> /etc/udev/rules.d/00-net.rules
```

If there are more than 1 VCAC-A cards connected to the Xeon host, add the same rule for the virtual interface (eth1, eth2 ...) for each of the VCAC-A cards.

```
#echo "ACTION==\"add\", SUBSYSTEM==\"net\", KERNEL==\"eth1\",
ENV{NM_UNMANAGED}=\"1\"" >> /etc/udev/rules.d/00-net.rules
#echo "ACTION==\"add\", SUBSYSTEM==\"net\", KERNEL==\"eth2\",
ENV{NM_UNMANAGED}=\"1\"" >> /etc/udev/rules.d/00-net.rules
```

Restart udev:

```
#systemctl restart systemd-udevd
```

- Enable forwarding in kernel

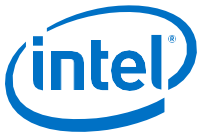
```
#echo 1 > /proc/sys/net/ipv4/ip_forward
```

- Add rules to NAT (**run every time after reboot**)

```
# iptables -t nat -A POSTROUTING -s 172.32.1.1 -d 0/0  -j MASQUERADE
```

(Optional) If ip forwarding does not work with the above command, try adding FORWARD rules, for example:

```
iptables -A FORWARD -i eno1 -o eth0 -m state --state RELATED,ESTABLISHED -j
ACCEPT
iptables -A FORWARD -i eth0 -o eno1 -j ACCEPT
```



Refer to Appendix - Troubleshooting NAT Configuration (Optional) on page 32 for further details.

Log into the VCAC-A card, and set up proxy for it:

- Set up proxy for VCAC-A card

```
#vim ~/.bashrc
export https_proxy=" local network https proxy"
export http_proxy=" local network http proxy"
export ftp_proxy=" local network ftp proxy"
```

2.2.7 VCAC-A Card Software Installation

As the VCAC-A card is boot up with vcad image, login to VCAC-A card through `ssh`, and install the VCAC-A software as below:

- Install dependencies

```
#apt update
apt-get install -y libjson-c3 libboost-program-options1.65-dev libboost-
thread1.65 libboost-filesystem1.65 libusb-dev cron python3-pip build-
essential curl wget libssl-dev ca-certificates git libboost-all-dev gcc-
multilib g++-multilib libgtk2.0-dev pkg-config libpng-dev libcairo2-dev
libpango1.0-dev libglib2.0-dev libusb-1.0-0-dev i2c-tools libgstreamer-
plugins-base1.0-dev libavformat-dev libavcodec-dev libswscale-dev
libgstreamer1.0-dev libusb-1.0-0-dev i2c-tools libjson-c-dev usbutils ocl-
icd-libopencl* ocl-icd-opencl-dev libsm-dev libxrender-dev libavfilter-dev
tzdata
```

- Load SMBus driver

```
# modprobe i2c-i801
# modprobe i2c-dev
```

- Check SMBus status. User should be able to see `smbus` device

```
#i2cdetect -l

i2c-3   i2c             i915 gmbus dpb                          I2C adapter
i2c-1   i2c             Synopsys DesignWare I2C adapter         I2C adapter
i2c-6   smbus           SMBus I801 adapter at f040              SMBus adapter
i2c-4   i2c             i915 gmbus dpd                          I2C adapter
i2c-2   i2c             i915 gmbus dpc                          I2C adapter
i2c-0   i2c             Synopsys DesignWare I2C adapter         I2C adapter
```

- Load HDDL driver

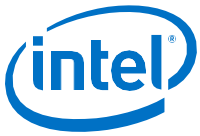
  Check driver status:

```
#lsmod | grep myd

Correct output should be like below:
myd_vsc                24576  0
myd_ion                61440  0
```

  If not, do the following steps:

```
#insmod /lib/modules/4.19.97-1.29bfe52.vca+/kernel/drivers/ion/myd_ion.ko
#insmod /lib/modules/4.19.97-1.29bfe52.vca+/kernel/drivers/usb/myd/myd_vsc.ko
(Note: "29bfe52" is a randomly generated number during compilation)
```



- Make sure 12 usb (Movidius VPU) devices can be found (device 2485)

```
# lsusb
Bus 012 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 011 Device 023: ID 03e7:2485
Bus 011 Device 022: ID 03e7:2485
Bus 011 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 009 Device 023: ID 03e7:2485
Bus 009 Device 022: ID 03e7:2485
Bus 009 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 010 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 008 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 007 Device 021: ID 03e7:2485
Bus 007 Device 022: ID 03e7:2485
Bus 007 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 005 Device 022: ID 03e7:2485
Bus 005 Device 023: ID 03e7:2485
Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 023: ID 03e7:2485
Bus 003 Device 022: ID 03e7:2485
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 023: ID 03e7:2485
Bus 001 Device 022: ID 03e7:2485
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

- Run script from OpenVINO to set environment for OpenVINO

```
#source /opt/intel/openvino/bin/setupvars.sh
```

2.2.8 Run Sanity Test

Test for VCAC-A software setup before proceeding further. To do this, try to run an OpenVINO sample application named `benchmark_app`.

2.2.8.1 Build OpenVINO Sample

After setup of software ingredients for VCAC-A is done, sample from OpenVINO can be built by following the steps below:

- Install dependencies

```
#cd /opt/intel/openvino/install_dependencies
# ./install_openvino_dependencies.sh
```

- Build Samples

```
#cd /opt/intel/openvino/inference_engine/samples/cpp

# apt-get install cmake
# ./build_samples.sh
```

After Build is completed, binaries for OpenVINO samples are in folder `/root/inference_engine_samples_build/intel64/Release`.

2.2.8.2 Validate Setup with benchmark_app

After sample application build is complete, `benchmark_app` is available in `/root/inference_engine_samples_build/intel64/Release`.

Follow instructions below to run `benchmark_app` for sanity test:

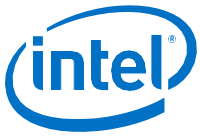
Download the model files:

```
# cd /root/inference_engine_samples_build/intel64/Release
#wget
https://download.01.org/opencv/2020/openvinotoolkit/2020.4/open_model_zoo/models_bin/3/vehicle-detection-adas-0002/FP16/vehicle-detection-adas-0002.bin
#wget
https://download.01.org/opencv/2020/openvinotoolkit/2020.4/open_model_zoo/models_bin/3/vehicle-detection-adas-0002/FP16/vehicle-detection-adas-0002.xml
```

Run commands as below ("`car_1.bmp`" is a sample image included as part of OpenVINO release):

```
# source /opt/intel/openvino/bin/setupvars.sh
# cd /root/inference_engine_samples_build/intel64/Release
#./benchmark_app -i /opt/intel/openvino/deployment_tools/demo/car_1.bmp -m vehicle-detection-adas-0002.xml -d HDDL -niter 1000 -nireq 128
```

The output of this command can be found here in Appendix - Sample Output Message for benchmark_app on page 31.



3.0 Run Media Analytics Pipeline with FFmpeg/ GStreamer or HOST

Video is the fastest growing data and to support real-time analytics on video streams, it is essential to accelerate complex and intensive operations like media decode/ encode and inference. VCAC-A card is designed to accelerate these operations and provide dense solution.

To construct an end to end pipeline, Intel supports popular multimedia industry frameworks - FFmpeg and GStreamer. These frameworks allow to build complex applications combining variety of multimedia blocks using vast libraries to process video, audio and inference.

3.1 Run Media Analytics Pipeline with HOST

HOST (Heterogeneous Optimized Scheduling Tool), which is part of the release, is an optimization tool to build efficient and flexible pipelines for a variety of workloads (like decoding, inferencing, etc) based on API/SDKs (MSDK, OpenVINO). It provides heterogeneous scheduler to deploy routines in task to different hardware components (GPU/CPU/VPU).

Media Analytics end to end pipeline could be constructed with HOST. The contents and guide are available in IPS, contact Intel representative if interested with this solution.

3.2 Run Media Analytics Pipeline

There are two options to install FFmpeg or GStreamer Media Analytic plugin:

- Install in Docker image.
- Install directly in system.

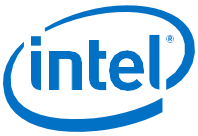
If chosen to install the FFmpeg/GStreamer Video Analytics in docker image, refer to the following link for the extra steps to setup VCAC-A to run docker containers:

https://github.com/OpenVisualCloud/Dockerfiles/blob/master/VCAC-A/README.md

3.2.1 Run Media Analytics Pipeline with FFmpeg

Intel provides FFmpeg video analytics plugin which is built on Intel OpenVINO's Inference Engine. It brings deep learning capabilities like object detection, classification and recognition to open-source framework FFmpeg and it helps developers and customers to build highly efficient and scalable video analytics applications.

For the detailed instructions, please refer to the Getting Started guide at https://github.com/VCDP/FFmpeg-patch/wiki/Getting-Started-Guide. It contains the steps to build FFmpeg with the analytics plugin and the examples to run analytics pipelines.



3.2.2 Run Media Analytics Pipeline with GStreamer

GStreamer is an open source framework for processing of video/audio streaming data. It provides tools and APIs for pipeline management and many plug-ins (over 250 plug-ins with more than 1000 elements) for building extensible media pipeline. Intel provides GStreamer Video Analytics (GVA) plugin, that contains multiple GStreamer elements to construct video analytics pipelines. GVA plugin has two types of elements:

1. The elements that provide inference operations such as detection, classification, identification using OpenVINO Inference Engine.

2. The elements that provide handling of inference output.

All the GStreamer elements are highly optimized for Intel Hardware and it is an open source project. The repository link below provides Wiki where you can find `Getting Started Guide, README, API reference and Performance Optimization technique` for Intel Platforms. The repository also contains code samples for object detection, face detection etc.

gst-video-analytics: https://github.com/opencv/gst-video-analytics

Appendix A Appendix - Sample Output Message for benchmark_app

Sample output message after running benchmark_app:

```
----------<Message for step 1 and 2 are truncated>-------------------
[Step 3/11] Setting device configuration
[Step 4/11] Reading the Intermediate Representation network
[ INFO ] Loading network files
[ INFO ] Read network took 49.47 ms
[Step 5/11] Resizing network to match image sizes and given batch
[ INFO ] Network batch size: 1
[Step 6/11] Configuring input of the model
[Step 7/11] Loading the model to the device
[21:48:57.1314][8197]I[main.cpp:246] ## HDDL_INSTALL_DIR:
/opt/intel/vcaa/hddl
[21:48:57.1315][8197]I[main.cpp:248] Config file
'/opt/intel/vcaa/hddl/config/hddl_service.config' has been loaded
----------------------------<Truncated Here>-------------------------
[21:49:20.6991][8197]I[DeviceSchedulerFactory.cpp:56] Info: ##
DeviceSchedulerFacotry ## Created Squeeze Device-Scheduler2.
[21:49:20.6991][8197]I[DeviceManager.cpp:551] ## SqueezeScheduler
created ##
[21:49:20.6991][8197]I[DeviceManager.cpp:649] times 0: try to create
worker on device(12.2)
[21:49:22.7016][8197]I[DeviceManager.cpp:670] [SUCCESS] times 0:
create worker on device(12.2)
----------------------------<Truncated Here>-------------------------
[21:49:44.7364][8197]I[DeviceManager.cpp:145] DEVICE FOUND : 12
[21:49:44.7365][8197]I[DeviceManager.cpp:146] DEVICE OPENED : 12
----------------------------<Truncated Here>-------------------------
[21:49:44.7396][8197]I[main.cpp:106] SERVICE IS READY ...
[ INFO ] Load network took 49970.84 ms
[Step 8/11] Setting optimal runtime parameters
[ WARNING ] Number of iterations was aligned by request number from
1000 to 1024 using number of requests 128
[Step 9/11] Creating infer requests and filling input blobs with
images
[ INFO ] Network input 'data' precision U8, dimensions (NCHW): 1 3
384 672
[ WARNING ] Some image input files will be duplicated: 128 files are
required but only 1 are provided
[ INFO ] Infer Request 0 filling
[ INFO ] Prepare image car_1.bmp
[ WARNING ] Image is resized from (749, 637) to (672, 384)
----------------------------<Truncated Here>-------------------------
[Step 10/11] Measuring performance (Start inference asyncronously, 128 inference
requests, limits: 1024 iterations)

[Step 11/11] Dumping statistics report
Count:      1024 iterations
Duration:   3630.58 ms
Latency:    429.19 ms
Throughput: 282.05 FPS
```

Appendix B Appendix - Troubleshooting NAT Configuration (Optional)

Under normal conditions, following command run after every reboot should set NAT configuration:

```
#iptables -t nat -A POSTROUTING -s 172.32.1.1 -d 0/0 -j
MASQUERADE
```

Following will be optional steps in case NAT configuration does not work. Please check relevant settings of the local network environment or customer's own platform network setup.

1. echo 1 > /proc/sys/net/ipv4/ip_forward
2. ip route show

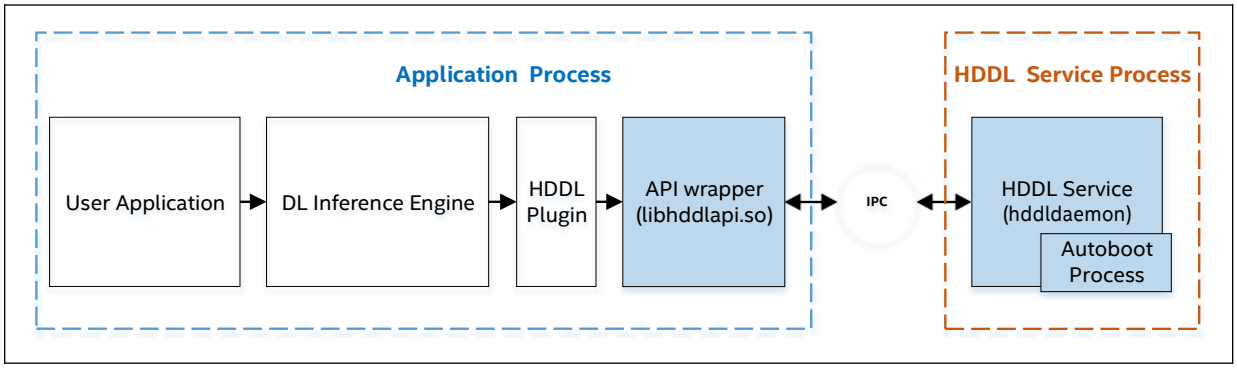
   This step helps to check the network interface where the ip traffic is routed on the host. For example, across eno1 or eno2 and so on.
3. /sbin/iptables -t nat -A POSTROUTING -o <eno1 or eno2> -j MASQUERADE
4. /sbin/iptables -A FORWARD -i <eno1 or eno2 > -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
5. /sbin/iptables -A FORWARD -i eth0 -o <eno1 or eno2> -j ACCEPT

Appendix C Appendix - VPU Utilization and Debugging Capability

The Inference Engine API that is available in the Intel® Distribution of OpenVINO™ toolkit provides the HAL(Hardware Abstract Layer) application interface, and an Inference Engine plugin (HDDLPlugin) access to the HAL.

The HAL consists of two components: the HAL wrapper API library (libhddlapi.so), and the high density deep learning daemon process (hddldaemon). The wrapper API is called inside the Inference Engine plugin and communicates with the daemon.

Figure 4. HDDL HAL High-level Architecture Diagram



The HAL uses configuration files to let you configure HAL behavior. These configuration files are in $HDDL_INSTALL_DIR/config/ (See following table for details), and VPU utilization and debugging logs are available by properly setting the parameters in the configuration files.

Table 4. Configuration Files

Configuration File	Target	Capability
hddl_service.config	hddldaemon	1. Log level/Debugging settings 2. InfoPrinter settings for runtime statistics dumping
hddl_api.config	libhddlapi.so	Log level settings.

Refer to HAL Configuration Guide for the detailed description of each parameter in the configurations files. This chapter is a summary description for how to get VPU utilization and debugging logs.

C.1 Example Scenario: Check the VPU Device Utilization

This scenario prints status information which could be controlled through the hddl_service.config. This scenario is often used to check the status of each connected Intel® Movidius™ Myriad™ X VPU.

For a sample VPU utilization, the information could be printed out with parameter set below:

- log_info: "on" (in "log_level" section)
- debug_service: true (in "debug_settings" section)
- device_utilization: "on" (in "debug_settings"->"info_printer" section)

Sample output message as below:

NOTE

Each line for the usage of each VPU, 12 lines in total for the 12 VPUs in VCAC-A

```
[06:51:00.9748][24683]I[DeviceManager.cpp:713] DeviceUtilization(count=12):
[12.2@model] = 44.48
[10.1@model] = 46.03
[8.2@model] = 46.40
[10.2@model] = 3.02
[8.1@model] = 44.73
[6.1@model] = 46.26
[4.2@model] = 46.34
[4.1@model] = 46.57
[2.2@model] = 44.25
[2.1@model] = 45.86
[6.2@model] = 44.62
[12.1@model] = 44.37
```

With a parameter set shown below, the device status including VPU utilization will be printed out in table format:

- log_info: "on" (in "log_level" section)
- debug_service: true (in "debug_settings" section)
- device_snapshot_mode'': ''full''/"base" (in "debug_settings"->"info_printer" section, less items will be listed with "base" mode)

The output is a similar table in device_snapshot_mode:

NOTE

One column for one VPU, VCAC-A have 12 VPUs. 7 columns for other 7 VPUs were removed from the snapshot picture to make the content more readable.

```
[21:51:35.5660][1862]I[DeviceManager.cpp:916] DeviceSnapshot(subclass=0):
| deviceId     | 6(0x6)              | 7(0x7)              | 0(00)               | 11(0xb)             | 10(0xa)             |
| device       | 10.1                | 10.2                | 4.1                 | 2.2                 | 2.1                 |
| util%        | 100.00 %            | 100.00 %            | 100.00 %            | 100.00 %            | 100.00 %            |
| thermal      | 45.55(0)            | 46.45(0)            | 46.23(0)            | 46.90(0)            | 45.09(0)            |
| scheduler    | squeeze             | squeeze             | squeeze             | squeeze             | squeeze             |
| comment      |                     |                     |                     |                     |                     |
| resetTimes   | 0                   | 0                   | 0                   | 0                   | 0                   |
| cacheNum     | 1                   | 1                   | 1                   | 1                   | 1                   |
| cacheGraph0  | Function_4          | Function_4          | Function_4          | Function_4          | Function_4          |
| cacheGraph1  |                     |                     |                     |                     |                     |
| cacheGraph2  |                     |                     |                     |                     |                     |
| cacheGraph3  |                     |                     |                     |                     |                     |
+--------------+---------------------+---------------------+---------------------+---------------------+---------------------+
| status       | RUNNING             | RUNNING             | RUNNING             | RUNNING             | RUNNING             |
| fps          | 25.34               | 25.35               | 25.39               | 25.36               | 25.37               |
| curGraph     | Function_4          | Function_4          | Function_4          | Function_4          | Function_4          |
| rPriority    | 0                   | 0                   | 0                   | 0                   | 0                   |
| loadTime     | 20200513 21:51:26   | 20200513 21:51:26   | 20200513 21:51:26   | 20200513 21:51:26   | 20200513 21:51:26   |
| runTime      | 00:00:08            | 00:00:08            | 00:00:08            | 00:00:08            | 00:00:08            |
| inference    | 206                 | 206                 | 206                 | 206                 | 206                 |
| prevGraph    |                     |                     |                     |                     |                     |
| loadTime     |                     |                     |                     |                     |                     |
| unloadTime   |                     |                     |                     |                     |                     |
| runTime      |                     |                     |                     |                     |                     |
| inference    |                     |                     |                     |                     |                     |
+--------------+---------------------+---------------------+---------------------+---------------------+---------------------+
```

C.2 Example Scenario: Debugging Capability

For HDDL HAL daemon, *hddldaemon*, "hddl_service.config" controls its behavior. "device_service" is the master switch of all *log_xxx* messages (display service log information), if set as true.

Log Level Setting

These settings allow users to get different level log messages to understand the runtime status of HAL service.

Table 5. Log Level Settings

log_level	Function	Options
log_frequent	Define whether to log messages frequently or infrequently.	"on", "off" (Default)
log_debug	Define whether to log debug messages.	"on", "off" (Default)
log_process	Define whether to log process messages.	"on", "off" (Default)
log_info	Define whether to log informational messages.	"on" (Default), "off"

continued...

log_level	Function	Options
log_warn	Define whether to log warning messages.	"on" (Default), "off"
log_error	Define whether to log error messages.	"on" (Default), "off"
log_fatal	Define whether to log fatal error messages.	"on" (Default), "off"

Sample output log from *hddldaemon* is listed as below. "Error", "I" (info) and "D" (debug) stand for different levels of log messages:

```
[15:33:10.9070][31217]ERROR[hddl_protocol.cpp:153] read socket(idx=13) failed,
user=HDDLPlugin leftBytes=4
[15:33:10.9070][31217]I[MessageDispatcher.cpp:209] failed to receive hddl message
from socket (13)
[15:33:10.9071][31209]D[HddlClient.cpp:35] socket 13 closed
[15:33:10.9072][31217]D[hddl_protocol.cpp:96] --> Receiving Msg (fd = 11)
```

C.3 Example Scenario: Info Printer Setting

Info Printer is a high-density deep learning service component that collects information like the FPS of different components, device utilization, device manager snapshot, task manager snapshot, and so on. The collected information helps user understand the status and running mode of the HAL Service.

NOTE

User must enable *log_info* before using these settings.

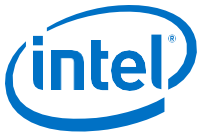
Log Level Setting

These settings allow users to get different level log messages to understand the runtime status of HAL service.

Table 6. Info Printer Settings

Info Printer Setting	Function	Options
enable	Determine whether to enable the information printer.	"false","true" (Default)
print_interval	Set the print interval in milliseconds.	user-defined interval, "5000" (Default)
client_fps	Determine whether to print each client frame rate.	"on", "off" (Default)
device_fps	Print the device working rate.	"on", "off" (Default)
service_fps	Determine whether to print the HAL service's serving rate.	"on", "off" (Default)

continued...



Info Printer Setting	Function	Options
graph_fps	Determine whether to print each graph's process rate.	"on", "off" (Default)
device_utilization	Determine whether to print each device's service utilization.	"on", "off" (Default)
memory_usage	Determine whether to print the service's memory usage.	"on", "off" (Default)
device_snapshot_mode	Set the display mode for device snapshot.	"base", "full", "none" (Default)
device_snapshot_style	Set the style for device snapshot.	"tape", "table" (Default)
client_snapshot_mode	Set the display mode for client snapshots.	"base","none" (Default)
client_snapshot_style	Set the style for client snapshots.	"base","table" (Default)
graph_snapshot_mode	Set the display mode for graph snapshots.	"base","none" (Default)
graph_snapshot_style	Set the display style for a graph snapshots.	"base","table" (Default)
task_snapshot_mode	Set the display mode for task snapshots.	"base","none" (Default)
task_snapshot_style	Set the display style for task snapshots.	"list" (Default)

The HDDL API Configuration configures the behavior in *libhddlapi.so*. The high-density deep learning service and API provide detailed log printing options for the debug levels described in this section. Use the log level settings in this section in the hddl_api.config file to control the logged information.

Table 7. HDDL API Log Level Settings

Log Level Setting	Function	Options
log_frequent	Define whether to log messages frequently or infrequently.	"on", "off" (Default)
log_debug	Define whether to print debug messages.	"on", "off" (Default)
log_process	Define whether to print process messages.	"on", "off" (Default)
log_info	Define whether to print informational messages.	"on", "off" (Default)

continued...

Log Level Setting	Function	Options
log_warn	Define whether to print warning messages.	"on" (Default), "off"
log_error	Define whether to print non-fatal error information.	"on" (Default), "off"
log_fatal	Define whether to print fatal error messages.	"on" (Default), "off"

Sample output log from *libhddlapi.so* are listed as below, starting with "[HDDLPlugin]". "I" (info), "P" (process), and "D" (debug) stand for different level of log messages:

```
[HDDLPlugin] [15:55:37.2264][4204]I[HddlClient.cpp:289] Info: RegisterClient
HDDLPlugin.
[HDDLPlugin] [15:55:37.2264][4204]P[HddlClient.cpp:1103] [Client] Emit Request
HDDL_MESSAGE_REGISTER_REQ (ReqSeqNo: 0).
[HDDLPlugin] [15:55:37.2264][4204]P[Dispatcher2.cpp:137] to push request
HDDL_MESSAGE_REGISTER_REQ (ReqSeqNo: 0)to reqToSendList.
[HDDLPlugin] [15:55:37.2264][4204]P[Dispatcher2.cpp:140] to signal sender
[HDDLPlugin] [15:55:37.2264][4204]P[HddlClient.cpp:1110] [Client] Wait for
response for HDDL_MESSAGE_REGISTER_REQ (ReqSeqNo: 0).
[HDDLPlugin] [15:55:37.2264][4204]P[HddlRequest.cpp:62] [Request
HDDL_MESSAGE_REGISTER_REQ (0)] Wait response for 60 seconds.
[HDDLPlugin] [15:55:37.2264][4285]P[Dispatcher2.cpp:173] [Sender] Request 0 comes.
[HDDLPlugin] [15:55:37.2264][4285]P[Dispatcher2.cpp:176] [Sender] Insert request
0 to SentList.
[HDDLPlugin] [15:55:37.2264][4285]P[Dispatcher2.cpp:189] [Sender] Sending request
0 ...
[HDDLPlugin] [15:55:37.2265][4285]D[hddl_protocol.cpp:187] <-- Send Msg (fd = 7):
HDDL_MESSAGE_REGISTER_REQ [ReqSeq = 0]
[HDDLPlugin] [15:55:37.2265][4285]D[hddl_protocol.cpp:224] <-- Send Msg (fd = 7):
HDDL_MESSAGE_REGISTER_REQ [ReqSeq = 0] DONE
```

Appendix D Appendix - Network setting with firewall enabled

If the firewall MUST be enabled in Xeon host, follow below instructions to setup network on Xeon host:

1. Turn on firewall:

```
#systemctl start firewalld.service
#systemctl enable firewalld.service
```

2. Enable ip forwarding

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

3. Change the network card interface (eg: eno1) of the Xeon host to zone external

```
#firewall-cmd --permanent --zone=public --change-interface=eno1
```

4. Add masquerade

```
#firewall-cmd --zone=public --add-masquerade --permanent
```

5. Add ssh service to zone

```
#firewall-cmd --permanent --zone=public --add-service=ssh
```

6. Reload firewall to make the settings take effect:

```
#firewall-cmd --reload
```

On Xeon host there will be a network interface with internal IP to connect to each VCAC-A card, e.g.:

Card #1: eth0, IP 172.32.1.1

Card #2: eth1, IP 172.32.2.1

Same for the other cards connected. Apply below settings for each card, take card #1 with IP 172.32.1.1 as example:

7. Set NAT rule

```
#firewall-cmd --permanent --direct --passthrough ipv4 -t nat
POSTROUTING -o eno1 -j MASQUERADE -s 172.32.1.1
```

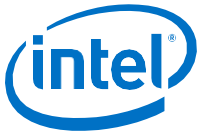
8. Configure port forwarding, besides below regular ports, any specific ports in your local environment could also be added.

```
#firewall-cmd --zone=public --add-forward-
port=port=5000:proto=tcp:toport=5000:toaddr=172.32.1.1 --permanent
#firewall-cmd --zone=public --add-forward-
port=port=8080:proto=tcp:toport=8080:toaddr=172.32.1.1 --permanent
#firewall-cmd --zone=public --add-forward-
port=port=6443:proto=tcp:toport=6443:toaddr=172.32.1.1 --permanent
#firewall-cmd --zone=public --add-forward-port=port=2379-
2380:proto=tcp:toport=2379-2380:toaddr=172.32.1.1 --permanent
#firewall-cmd --zone=public --add-forward-
port=port=10250:proto=tcp:toport=10250:toaddr=172.32.1.1 --permanent
#firewall-cmd --zone=public --add-forward-
port=port=10251:proto=tcp:toport=10251:toaddr=172.32.1.1 --permanent
#firewall-cmd --zone=public --add-forward-
port=port=10252:proto=tcp:toport=10252:toaddr=172.32.1.1 --permanent
#firewall-cmd --zone=public --add-forward-
port=port=10255:proto=tcp:toport=10255:toaddr=172.32.1.1 --permanent
```

Repeat step 6&7 for the other VCAC-A Cards connected to the Xeon host.

9. Reload firewall to make the settings take effect and check the settings

```
#firewall-cmd --reload
#firewall-cmd --zone=public --list-all
```



Glossary

HDDL
High Density Deep Learning

NAT
Network Address Translation

PCIe
Peripheral Component Interconnect Express

SMBus
System Management Bus

VCAC-A
Visual Cloud Accelerator Card - Analytics

VPU
Visual Process Unit