HDDL-S installation guide

Hardware: CoffeeLake, Intel(R) Xeon(R) E-2176G CPU @ 3.70GHz

OS: Ubuntu 16.04.3

1. **Install OpenVINO R3**

Download OpenVINO R3:

<https://software.intel.com/en-us/openvino-toolkit/choose-download/free-download-linux>

Install OpenVINO:

<https://software.intel.com/en-us/articles/OpenVINO-Install-Linux>

*#source /opt/intel/computer\_vision\_sdk/bin/setupvars.sh*

1. **Install OpenCL**

*#cd /opt/intel/computer\_vision\_sdk/install\_dependencies*

*#sudo ./install\_NEO\_OCL\_driver.sh*

Add OpenCL users to the video group:

*#sudo usermod -a -G video USERNAME*

e.g. if the user running OpenCL host applications is foo, run: sudo usermod -a -G video foo

Install 4.14 kernel using install\_4\_14\_kernel.sh script and reboot into this kernel

*#sudo ./install\_4\_14\_kernel.sh*

If you use 8th Generation Intel processor, you will need to add:

i915.alpha\_support=1

to the 4.14 kernel command line, in order to enable OpenCL functionality for this platform.

1. **Install OpenCL SDK**

*#sudo apt-get install dkms*

*#tar -xvf intel\_sdk\_for\_opencl\_2017\_7.0.0.2568\_x64.gz*

*#cd intel\_sdk\_for\_opencl\_2017\_7.0.0.2568\_x64*

*#./install\_GUI.sh*

1. **Install HDDL-R**

**4.1 dependency package**

*#sudo apt-get install libelf-dev*

*#sudo apt-get install libusb-1.0-0-dev libudev-dev libssl-dev rpm cmake libboost-program-options1.58-dev libboost-thread1.58 libboost-filesystem1.58 git*

**4.2 Install json-c**

*#git clone https://github.com/json-c/json-c.git*

*#cd json-c*

*#git checkout f8c632f579c71012f9aca81543b880a579f634fc*

*#sudo apt-get install autoconf libtool*

*#sh autogen.sh*

*#./configure*

*#make*

*#sudo make install*

* 1. **install HDDL rpm package**

*#sudo rpm -ivh --nodeps Intel\_Movidius\_MyriadX\_HDDL-R\_Linux-01.02.12.B.36.3.rpm*

*#export HDDL\_INSTALL\_DIR=/usr/local*

*#export LD\_LIBRARY\_PATH=/usr/local/lib*

*#sudo usermod -a -G users,video $USER*

1. **Install gstreamer**

*#sudo apt-get install gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-ugly gstreamer1.0-plugins-bad libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev*

*#export PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/opt/intel/mediasdk/lib64/pkgconfig*

1. **Install uWebSocket**

*#sudo apt-get install libssl-dev*

*#git clone https://github.com/uNetworking/uWebSockets.git*

*#make && sudo make install*

1. **Install OpenCV**

*#sudo apt-get install libgtk2.0-dev pkg-config libgtkglext1-dev*

*#export CPLUS\_INCLUDE\_PATH=/opt/intel/mediasdk/include:$CPLUS\_INCLUDE\_PATH*

*#git clone https://github.com/opencv/opencv.git*

*#cd opencv && git checkout 6ffc48769ac60d53c4bd1913eac15117c9b1c9f7*

*#mkdir build && cd build*

*#cmake -DWITH\_VA\_INTEL=ON -DWITH\_CUDA=OFF ..*

*#make -j8*

*#sudo make install*

Note: OpenVINO has provided OpenCV libraries, but HDDL-S need VA support in OpenCV, so we must rebuild it.

1. **Install HDDL-S software**

**8.1 download source code**

*#git clone* [*git@gitlab-icv.inn.intel.com:hddl/s\_framework.git*](mailto:git@gitlab-icv.inn.intel.com:hddl/s_framework.git)

**8.2 build MSDK gst-plugin**

*#sudo apt-get install libdrm-dev libudev-dev libgstreamer-plugins-bad1.0-dev libx11-xcb-dev libgles2-mesa-dev libgl1-mesa-dev*

*#sudo ln -sf /opt/intel/mediasdk/lib64/libva.so.2 /usr/lib/libva.so*

*#sudo ln -sf /opt/intel/mediasdk/lib64/libva-drm.so.2 /usr/lib/libva-drm.so*

*#cd gstreamer-media-SDK && mkdir build && cd build && cmake ..*

*#make*

*#sudo cp build/lib/release/libgstmfx.so /usr/lib/x86\_64-linux-gnu/gstreamer-1.0/libgstmfx.so*

* 1. **build OpenVINO gst-plugin**

*#cd gstreamer\_plugin\_openVINO*

*#make*

*#sudo make install*

Run below command to check if it has been installed successfully:

*# gst-inspect-1.0 cvdlfilter*

*# gst-inspect-1.0 resconvert*

*# gst-inspect-1.0 wssink*

* 1. **Install hddlspipe**

*#cd gstreamer\_pipeline*

*#make && sudo make install*

1. **Setup HDDL-S Server**

*#sudo apt-get install nodejs-legacy npm*

*#npm config set proxy http://child-prc.intel.com:913*

*#sudo npm install -g n*

*#sudo n stable*

*#npm install ws@6.0.0*

*#npm install child\_process arraybuffer-to-string rimraf commander tree-kill*

*#chmod a+x \*.js*

In client side:

*#echo <server\_ip> > path.txt*

1. **Setup rtsp server as input stream source(option)**

*#wget* [*https://gstreamer.freedesktop.org/src/gst-rtsp-server/gst-rtsp-server-1.8.3.tar.xz*](https://gstreamer.freedesktop.org/src/gst-rtsp-server/gst-rtsp-server-1.8.3.tar.xz)

*#tar -xvf gst-rtsp-server-1.8.3.tar.xz*

*#cd gst-rtsp-server-1.8.3*

*#sudo apt-get install gtk-doc-tools*

*#cp s\_framework/gstreamer\_pipeline/rtsp\_serve/test-launch.c examples/.*

*#./autogen.sh && ./configure && make*

*#cd example*

*#./test-launch --gst-debug=3 "( filesrc location=/home/lijunjie/1600x1200.mp4 ! qtdemux ! rtph264pay name=pay0 pt=96 )"*

Note: in rtsp receiver side, we need set

1). echo 800000 > /proc/sys/net/core/rmem\_max

2). Add udp-buff-size=800000 into rtspsrc

For example:

gst-launch-1.0 -v rtspsrc location=rtsp://10.239.85.64:8554/test udp-buff-size=800000 ! rtph264depay ! h264parse…

1. **Run HDDL-S Software Stack**

**11.1 Set environment variable**

Modify ~/.bashrc and add below command lines into it:

*-----------------------------------------------------------------------------------------------------------------------------*

*source /opt/intel/computer\_vision\_sdk/bin/setupvars.sh*

*export HDDL\_INSTALL\_DIR=/usr/local*

*export LD\_LIBRARY\_PATH=/usr/local/lib*

*export PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/opt/intel/mediasdk/lib64/pkgconfig*

*export LD\_LIBRARY\_PATH=/opt/intel/mediasdk/lib64:/usr/local/lib:/usr/local/lib/ubuntu\_16.04/intel64:$LD\_LIBRARY\_PATH*

*export CVDL\_KERNEL\_DIR=/usr/lib/x86\_64-linux-gnu/libgstcvdl/kernels*

*export CVDL\_CLASSIFICATION\_MODEL\_FULL\_PATH=/usr/lib/x86\_64-linux-gnu/libgstcvdl/models/vehicle\_classify/carmodel\_fine\_tune\_1062\_bn\_iter\_370000.xml*

*export CVDL\_DETECTION\_MODEL\_FULL\_PATH=/usr/lib/x86\_64-linux-gnu/libgstcvdl/models/vehicle\_detect/yolov1-tiny.xml*

*export PATH=$PATH:/opt/intel/mediasdk/bin/*

---------------------------------------------------------------------------------------------

Run .bashrc

*#source ~/.bashrc*

**11.2 Generate certificate**

Please generate all these certificates in one pc!!!

(1) Prepare certificates

-Generate a Certificate Authority:

*openssl req -new -x509 -days 9999 -keyout ca-key.pem -out ca-crt.pem*

-Insert a CA Password and remember it

-Specify a CA Common Name, like 'root.localhost' or 'ca.localhost'. This MUST be different from both server and client CN.

(2) Server certificate

-Generate Server Key:

*openssl genrsa -out server-key.pem 4096*

-Generate Server certificate signing request:

*openssl req -new -key server-key.pem -out server-csr.pem*

-Specify server Common Name, run cat /etc/hosts to check valid DNS name, please don't name as'localhost'.

-For this example, do not insert the challenge password.

(3) Sign certificate using the CA:

*openssl x509 -req -days 9999 -in server-csr.pem -CA ca-crt.pem -CAkey ca-key.pem -CAcreateserial -out server-crt.pem*

-insert CA Password

(4) Verify server certificate:

*openssl verify -CAfile ca-crt.pem server-crt.pem*

(5) Client certificate

-Generate Client Key:

*openssl genrsa -out client1-key.pem 4096*

-Generate Client certificate signing request:

*openssl req -new -key client1-key.pem -out client1-csr.pem*

-Specify client Common Name, like 'client.localhost'. Server should not verify this, since it should not do reverse-dns lookup.

-For this example, do not insert the challenge password.

(6) Sign certificate using the CA:

*openssl x509 -req -days 9999 -in client1-csr.pem -CA ca-crt.pem -CAkey ca-key.pem -CAcreateserial -out client1-crt.pem*

-insert CA Password

(7) Verify client certificate:

*openssl verify -CAfile ca-crt.pem client1-crt.pem*

(8) After generated, please copy all file start with 'ca' and 'client' into 'cert\_client\_8216', and copy all file start with 'ca' and 'server' into 'cert\_server\_8216'.

**11.3 Run from HDDL side:**

*# cd s\_framework/media\_service*

*#./hddls\_server.js*

**11.3 Run from host side:**

*# cd s\_framework/media\_service*

*#./receiver\_client.js*

*# cd s\_framework/media\_service*

*#./controller\_client.js -s <stream source> -l <loop\_times> -n <pipe\_num>*

Note:

1. stream source can be local file such as: /home/<user>/<file\_name.mp4>

Or rtsp source: rtsp://<rtsp\_addr>%<media\_type>

For example: rtsp://10.239.85.88:8554/video0%H.264

media\_type: H.264 or H.265

1. command to stop and resume pipe task:

stop: <pipe\_id>,p

resume: <pipe\_id>,c,<loop\_times>

1. about pipe\_num

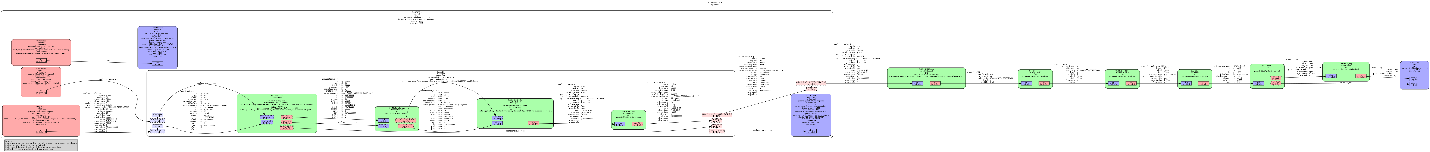
set 1 if rtsp source don’t support multiple client access.

Backup:

1. Test for cvdlfilter and resconvert plugin

*gst-launch-1.0 filesrc location=<file> ! h264parse ! mfxh264dec ! cvdlfilter ! resconvert ! mfxjpegenc ! multifilesink location=/home/lijunjie/hddls\_%d.jpeg*

1. *Gstreamer pipeline graph*

**