# Render + Encode Pipeline User Guide

# 1. Setup the environment

## 1.1.Install SG1 release package

\$./install-sg1.sh

Enter "y" when asked "Do you want to install Mesa? 'y/n' default is n:" during the process.

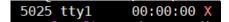
## 1.2.Stop gdm

\$ systemctl stop gdm

# 1.3. Check whether X service is stopped

\$ ps -e | grep X

If you see something like the image below, it means X service is still on.



Please kill it:

\$ kill -9 pid\_of\_X

## 1.4.Export environmental variables

### Make sure X service is stopped before this step.

### For CentOS:

\$ export LD\_LIBRARY\_PATH=/opt/intel/mediasdk/lib64/:/usr/local/lib:/usr/local/lib64/:/usr/local/lib64/:/usr/local/lib64/:/usr/lib64/dri/:/opt/intel/mediasdk/share/mfx/samples/\_bin/x64/:\$LD\_LIBRARY\_PATH

### For **Debian & Ubuntu**:

\$ export LD\_LIBRARY\_PATH=/opt/intel/mediasdk/lib64/:/usr/local/lib:/usr/local/lib/x86 \_64-linux-gnu/:/usr/local/lib/x86\_64-linux-gnu/dri/:/usr/lib64/:/usr/lib64/dri/:/opt/intel/mediasdk/share/mfx/samples/\_bin/x64/:\$LD\_LIBRARY\_PATH

#### For all OS:

\$ export DISPLAY=:0.0

\$ export MESA\_LOADER\_DRIVER\_OVERRIDE=iris

\$ export INTEL\_DEBUG=norbc

```
$ export LIBVA DRIVER NAME=iHD
```

- \$ export LIBVA DRIVERS PATH=/opt/intel/mediasdk/lib64
- \$ export PKG CONFIG PATH=/opt/intel/mediasdk/lib64/pkgconfig:\$PKG CONFIG PATH

### 1.5. Restart X service and check the mesa version

\$ xinit&

\$ glxinfo | grep Mesa

You should see the information below. Otherwise, the environment is not set properly, and you need to check the above steps again.

```
client glx vendor string: Mesa Project and SGI
    Device: Mesa Intel(R) Graphics (DG1 GT2) (0x4905)
OpenGL renderer string: Mesa Intel(R) Graphics (DG1 GT2)
OpenGL core profile version string: 4.6 (Core Profile) Mesa 20.3.0-devel (git-b00c3a03e4)
OpenGL version string: 4.6 (Compatibility Profile) Mesa 20.3.0-devel (git-b00c3a03e4)
OpenGL ES profile version string: OpenGL ES 3.2 Mesa 20.3.0-devel (git-b00c3a03e4)
```

Please be aware that X service is only required for "glxinfo". You can turn is off again after checking the mesa version because the pipeline has no dependency on it.

## 2. Build MediaSDK

#### 2.1.Check CMake version

CMake 3.6 or higher is required to build MediaSDK. You can check it with:

\$ cmake --version

If your CMake doesn't meet the requirement, please remove the old version then install a new one (version 3.8.2 is recommended).

\$ wget <a href="https://github.com/Kitware/CMake/releases/download/v3.8.2/cmake-3.8.2.tar.gz">https://github.com/Kitware/CMake/releases/download/v3.8.2/cmake-3.8.2.tar.gz</a>

\$ tar -xzvf cmake-3.8.2.tar.gz

\$ cd cmake-3.8.2

\$./bootstrap

\$ make -j\$(nproc)

\$ make install

Check cmake version after installation. Assume your cmake is installed in "/usr/local/bin":

\$ /usr/local/bin/cmake --version

## 2.2.Build MediaSDK source code with OpenGL enabled

\$ cd /opt/intel/mediasdk/share/mfx/samples

\$ mkdir build && cd build

\$ /usr/local/bin/cmake .. -DENABLE OPENGL=ON

\$ make -j\$(nproc)

\$ cd bin/release

## 2.3. (optional) Solve X11 header files issue

Should you encounter the error: "/usr/local/include/EGL/eglplatform.h:134:10: fatal error: X11/Xlib.h: No such file or directory" during the building process, it happens because the "USE\_X11" macro in eglplatform.h is defined for some reason, but X11 header files are not found.

#### For CentOS:

\$ yum install libX11-devel.x86 64

For **Debian & Ubuntu**:

\$ apt install libx11-dev

# 3. Run the pipeline

Both **h264** and **h265** are supported, the command line to run the pipeline is:

\$ ./sample\_encode h264|h265 -o output -w 1920 -h 1080 -hw -vaapi -opengl - lowpower:on -n 1000 -device /dev/dri/renderD128

Parameters marked in bold can't be changed in order to run the pipeline properly.

If "-device" is not specified, it will run on device "/dev/dri/renderD128" as default.

If "-o" is not specified, no output file will be produced.

If "-n" is not specified, it will continue running until manually stopped.

# 4. Check the output file

The output file can be opened with tools like VQAnalyzer. Currently, the result is the rotation of a rectangle divided into 4 parts, as the image below shows.

