WHITE PAPER

Intel® Quick Sync Video and FFmpeg Installation and Validation Guide

Introduction

Intel® Quick Sync Video technology on Intel® Iris™ Pro Graphics and Intel® HD graphics provides transcode acceleration on Linux* systems in FFmpeg* 2.8 and later editions.
This paper is a detailed step-by-step guide to enabling h264_qsv, mpeg2_qsv, and hevc_qsv hardware accelerated codecs in the FFmpeg framework. For a quicker overview, please see this article.

Performance note: The *_qsv implementations are intended to provide easy access to Intel hardware capabilities for FFmpeg users, but are less efficient than custom applications optimized for Intel® Media Server Studio.

Document note: Monospace type = command line inputs/outputs. Pink = highlights to call special attention to important command line I/O details.

Getting Started

1. **Install Intel Media Server Studio for Linux.** Download from software.intel.com/intel-media-server-studio. This is a prerequisite for the *_qsv codecs as it provides the foundation for encode acceleration. See the next chapter for more info on edition choices. Note: Professional edition install is required for hevc_qsv.
2. **Get the latest FFmpeg source** from https://www.FFmpeg.org/download.html. Intel Quick Sync Video support is available in FFmpeg 2.8 and later editions. The install steps outlined below were verified with **ffmpeg release 3.2.2**
3. **Configure FFmpeg** with "--enable –libmfx –enable-nonfree", build, and install. This requires copying include files to /opt/intel/mediasdk/include/mfx and adding a libmfx.pc file. More details below.
4. **Test transcode** with an accelerated codec such as "-vcodec h264_qsv" on the FFmpeg command line. For more details, see Page 9 of this document.

```
FFmpeg -i in.mp4 -vcodec h264 _ qsv out _ qsv.mp4
```

Installing Intel® Media Server Studio

Intel Media Server Studio is available at: software.intel.com/intel-media-server-studio

- The Community edition is completely free and supports many popular codecs including HEVC, AVC/H.264 and MPEG-2.
- The Essentials edition adds Intel® Premier Support (access to Intel experts).
- The Professional Editions adds HEVC, Audio, and analyzing tools such as Intel® VTuneTM analyzer. An evaluation package is available for Professional which limits HEVC encode to 1000 frames. For more info see the link above.

The 2017 R2 release adds support for 6th Generation Intel® Core™ processors (formerly "Skylake") with integrated graphics. It also supports 5th Generation Intel Core (formerly "Broadwell"). Note: 4th Generation Intel Core and earlier processors are not supported by Intel Media Server Studio 2017.

Please consult this article for more background and details on the processor and OS support matrix: https://software.intel.com/en-us/articles/driver-support-matrix-for-media-sdk-and-opencl.

How to Install:

The steps below cover how to install Intel Media Server Studio 2017 R2 on CentOS* 7.2 (kernel 3.10-327.13.1 and up). Please see the Release Notes and Support Page for more product details, including information on installation for other flavors of Linux. The steps below describe where to find the installation scripts.

```
$ tar -xvzf MediaServerStudio*.tar.gz
$ cd MediaServerStudio*
$ tar -xvzf SDK*.tar.gz
$ cd SDK*
$ tar -xvzf install _ scripts*.tar.gz
```

IMPORTANT NOTE: The installation procedure for Intel® Media Server Studio 2017 R2 is different than previous installations. This product is a combination of driver, library, and graphics stack components requiring specific hardware, Linux* distributions, and kernel versions.

For Intel® Media Server Studio 2017 R2, there is a new Gold OS and a new approach to installing kernel updates. Supported processors also change from Intel® Media Server Studio 2016. Double check the processor on your system with "cat /proc/cpuinfo" before starting.

The release tested in this document adds support for 6th Generation Intel® Core™ processors (formerly "Skylake") with integrated graphics. It also covers 5th Generation Intel® Core (formerly "Broadwell"). Note: 4th Generation Core™ and earlier processors are not supported by Media Server Studio 2017. This article describes the processor and OS support matrix: https://software.intel.com/en-us/articles/driver-support-matrix-for-media-sdk-andopencl
Please consult this article and the release notes for more background and details.

Prerequisite Steps

1. Add the user(s) who will run Intel® Media Server Studio – SDK applications to the video group
   $ usermod -a -G video [LOGIN]
2. Check that an Intel VGA adapter can be found with lspci:
   $ lspci -nn -s 0:02.0
   00:02.0 VGA compatible controller [0300]: Intel Corporation Broadwell-U
   Integrated Graphics [8086:193b] (rev 09)
   The command output above shows 193b as the graphics device ID. The ID reported by lspci may be different for your machine. The main thing to look for is that an Intel graphics adapter is available. If not, you may need to check your BIOS settings and hardware configuration.

3. Edit /etc/yum.repos.d/CentOS-Base.repo:
   $sudo vi /etc/yum.repos.d/CentOS-Base.repo
   ##Comment out every occurrence of mirrorlist lines and uncomment every occurrence of baseurl lines. Like:
   #mirrorlist=http://mirrorlist.centos.org/?release=\$releasever&arch=\$basearch&repo=os&infra=\$infra
   baseurl=http://mirror.centos.org/centos/\$releasever/os/\$basearch/

4. If the kernel version is earlier than 3.10.327-13.1, you need to update it using the below command:
   $sudo yum --releasever=7.2.1511 update

CentOS installs assume the "Development and Creative Workstation" base environment for included scripts. Other configurations will require additional packages which are not installed by default. Installation scripts are in the install_scripts_centos*.tar.gz bundle. Install is simplified to a single "install_sdk_CentOS.sh" script.

```
$ tar -xzf MediaServerStudio*.tar.gz
$ cd MediaServerStudio*
$ tar -xzf SDK2017*.tar.gz
$ cd SDK2017*/CentOS
$ tar -xzf install_scripts_*.tar.gz
$ su
# ./install_sdk_CentOS.sh
# reboot
```

Verifying correct installation
The /opt/intel/mediasdk directory should be populated

```
$ ls /opt/intel/mediasdk/
builder doc include lib lib64 opensource plugins samples tools
```

The /dev/dri directory should have a renderD interface.

The vainfo utility should show the current driver, Media SDK's iHD (from /opt/intel/mediasdk) and several codec entry points.

```
$ vainfo | grep -v 'unknown'
libva info: VA-API version 0.99.0
libva info: va_getDriverName() returns 0
libva info: User requested driver 'iHD'
libva info: Trying to open /opt/intel/mediasdk/lib64/iHD_drv_video.so
libva info: Found init function __vaDriverInit_0_32
libva info: va_openDriver() returns 0
vainfo: VA-API version: 0.99 (libva 1.67.0.pre1)
vainfo: Driver version: 16.5.53384-ubit
vainfo: Supported profile and entrypoints
 VAProfileH264ConstrainedBaseline: VAEntrypointVLD
 VAProfileH264ConstrainedBaseline: VAEntrypointEncSlice
 VAProfileH264Main : VAEntrypointVLD
 VAProfileH264Main : VAEntrypointEncSlice
 VAProfileH264High : VAEntrypointVLD
 VAProfileH264High : VAEntrypointEncSlice
```

Test Intel® Media Server Studio installation

Taking a few minutes to test the Intel Media Server Studio SDK foundation of the *_qsv codecs outside FFmpeg can help ensure smooth operation. Pre-compiled binaries are included with the install package, as well as short test content in multiple formats. Here is how to find them:

```
$ cd MediaServerStudio*
$ tar -xzf MediaSamples _ Linux _ bins*
$ cd MediaSamples _ Linux _ bins*
```

Run the sample_multi_transcode example using the included test content:

$ **./sample _ multi _ transcode _ drm -i::h264 ../content/test _ stream.264 -o::h264 out.h264 -hw -la**
This provides a quick smoke test for a range of components used in FFmpeg. These examples are also an introduction to the capabilities and parameters of the SDK. Correct output should look like this:

```
libva info: VA-API version 0.99.0
libva info: va_getDriverName() returns 0
libva info: User requested driver 'iHD'
libva info: Trying to open /opt/intel/mediasdk/lib64/iHD_drv_video.so
libva info: Found init function __vaDriverInit_0_32
libva info: va_openDriver() returns 0
Pipeline surfaces number (DecPool): 59
MFX HARDWARE Session 0 API ver 1.19 parameters:
Input  video: AVC
Output video: AVC

Session 0 was NOT joined with other sessions

Transcoding started
..
Transcoding finished

Common transcoding time is  0.15 sec
MFX session 0 transcoding PASSED:
Processing time: 0.15 sec
Number of processed frames: 101

The test PASSED
```

This demonstrates correct operation by:
1. Correct iHD driver name and path are used by libva
2. Transcode starts and fnishes with “The test PASSED” message

The test above performs transcode, decoding an H.264 elementary stream and re-encoding at a different bitrate with lookahead bitrate control. Successful completion indicates the full stack of components necessary for h264_qsv, mpeg2_qsv and vc1_qsv are ready to use. To explore the SDK capabilities further:

- Source for the samples can be found here: https://software.intel.com/en-us/intel-media-server-studio-support/code-samples
- Tutorials (easier to understand than the samples) are here: https://software.intel.com/en-us/intel-media-server-studio-support/code-samples

Passing this smoke test demonstrates that you are ready to proceed to _qsv codec setup in FFmpeg. If errors are seen here, please proceed to the troubleshooting section below.

Installing FFmpeg with Intel® Media Quick Sync Video (Intel® QSV) hardware acceleration support

Where to get FFmpeg with Intel® Quick Sync Video (Intel® QSV) updates

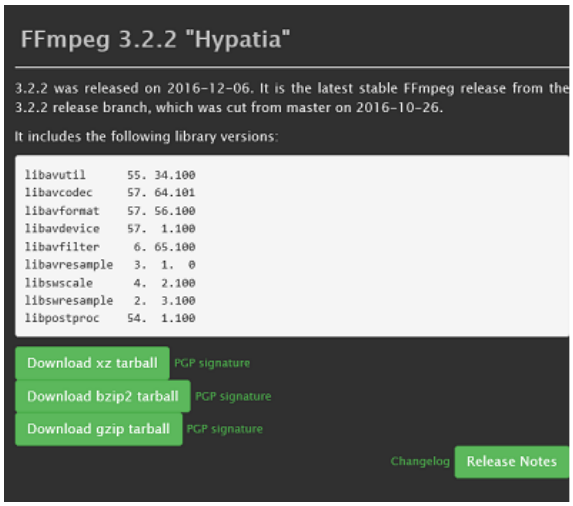
Support for _qsv codecs is included in FFmpeg 2.8 and newer, available from https://www.FFmpeg.org/download.html. Development is active and updates are frequent. We

recommend using the release versions of FFmpeg. The install steps outlined below were verified with **FFmpeg release 3.2.2**

How to install
The *_qsv codecs are enabled when FFmpeg is compiled after the configuration script is run with the "--enable-libmfx –enable-nonfree". A few prerequisite steps are required for configure to find the libmfx/Intel QSV components.

Important license note: The –enable-nonfree flag is required because this specifies that the resulting compile is not GPL. If this configure flag is not added, configure fails with "libmfx is nonfree and –enable-nonfree is not specified." After adding this flag, "nonfree and unredistributable" is printed to the screen as part of configure output, indicating that the executables/libraries compiled are not intended for redistribution.

Step 1: Copy the /opt/intel/mediasdk/include header fles to include/mfx. This location is used for historical compatibility
mkdir /opt/intel/mediasdk/include/mfx
cp /opt/intel/mediasdk/include/*.h /opt/intel/mediasdk/include/mfx



Step 2: Provide a libmfx.pc file so pkgconfig can provide path and compile flag settings to FFmpeg configure. Ensure that the libmfx.pc file is in a directory specified by PKG_CONFIG_PATH variable.

```
Example libmfx.pc file:
prefix=/opt/intel/mediasdk
exec _ prefix=${prefix}
libdir=${prefix}/lib/lin _ x64
includedir=${prefix}/include
Name: libmfx
Description: Intel Media Server Studio SDK
Version: 16.5
Libs: -L${libdir} -lmfx -lva -lstdc++ -ldl -lva-drm -ldrm
Cflags: -I${includedir} -I/usr/include/libdrm

Save and close the libmfx.pc file and run the below 2 commands:
#touch libmfx.pc
$pkg-config --modversion libmfx
```

16.5

Step 3: Pre-requisities for buiding FFmpeg
(Covered in https://trac.FFmpeg.org/wiki/CompilationGuide/Centos)

yum install autoconf automake bzip2 cmake freetype-devel gcc gcc-c++ git libtool make mercurial nasm pkgconfig zlib-devel

Install yasm (an assembler used by FFmpeg)

$cd yasm
$autoreconf -fiv
$./configure
$make
$sudo make install

Step 4: Building FFmpeg

$curl -O http://FFmpeg.org/releases/FFmpeg-snapshot.tar.bz2
$tar xjvf FFmpeg-snapshot.tar.bz2
$cd FFmpeg
Run configure. There are many pages of output. Capturing with 'tee' or redirecting to a file for later review and
search is helpful.
$./configure --enable-libmfx --enable-nonfree | tee config.out
Use the captured text output to verify that the _qsv codecs are enabled:
$ grep 'qsv' config.out

adpcm_g726le	cfhd	h264_qsv
adpcm_ima_dat4	cllc	hevc_qsv
mpeg2_qsv	prores	tscc
mvc1	rl2	vc1_qsv
adpcm_ms	mpeg2_qsv	rawvideo
h264_qsv	pcm_u24le	xsub
hevc_qsv	pcm_u8	yuv4
h263_vaapi	hevc_vaapi	vc1_qsv
h264_qsv	mpeg2_qsv	vc1_vaapi
hevc_qsv	mpeg4_vaapi	wmv3_vaapi

License should also be unredistributable:
$ grep 'License:' config.out
License: nonfree and unredistributable
After checking that configure successfully found the libmfx _qsv codec prerequisites on the system, proceed with the make process:
$ make –j 8
$ su
# make install

How to test _qsv codecs in FFmpeg

After compiling and installing FFmpeg, double check that _qsv codecs are available:
$ FFmpeg -codecs | grep 'qsv'

```
DEV.LS h264                 H.264 / AVC / MPEG-4 AVC / MPEG-4 part 10
(decoders: h264 h264_qsv)
(encoders: h264_nvenc h264_qsv h264_vaapi nvenc nvenc_h264)
DEV.L. hevc                 H.265 / HEVC (High Efficiency Video Coding)
(decoders: hevc hevc_qsv)
(encoders: nvenc_hevc hevc_nvenc hevc_qsv hevc_vaapi )
DEV.L. mpeg2video           MPEG-2 video
(decoders: mpeg2video mpegvideo mpeg2_qsv)
(encoders: mpeg2video mpeg2_qsv )
D.V.L. vc1                  SMPTE VC-1 (decoders: vc1 vc1_qsv)
```

This shows that we have mpeg2_qsv, h264_qsv and hevc_qsv encoders & vc1_qsv decoder ready to use. These are intended to coexist as hardware accelerated decode/encode options along with other software implementations. In many cases multiple alternatives exist for the same codec, leaving the choice of codec to application developers via command line or FFmpeg API.

To select a hardware accelerated _qsv codec use "-vcodec" from the command line. Where software implementations of a codec are available they usually are frst in the list, so the Intel QSV hardware accelerated version must be directly selected by name. While a large range of parameters are available, here is a minimal command line to test h264_qsv operation:

$ FFmpeg -y –i test.mp4 -vcodec h264 _ qsv -acodec copy -b:v 8000K out.mp4

Expected output, part 1:

```
FFmpeg version 3.2.2 Copyright (c) 2000-2016 the FFmpeg developers
....
libva info: VA-API version 0.99.0
libva info: va_getDriverName() returns 0
libva info: User requested driver 'iHD'
libva info: Trying to open /opt/intel/mediasdk/lib64/iHD_drv_video.so
libva info: Found init function __vaDriverInit_0_32
libva info: va_openDriver() returns 0
```

Initialization output for libva should be visible, with correct library name and driver path. More initialization details are available if "-v verbose" is added to the command line.
As the FFmpeg pipeline setup proceeds, the output summary should show h264_qsv is used as the output codec, with expected resolution, bitrate, frame rate, etc.

```
Output #0, mp4, to 'out.mp4':
  Metadata:
    major_brand     : isom
    minor_version   : 512
    compatible_brands: isomiso2mp41
    encoder         : Lavf57.56.100
    Stream #0:0(und): Video: h264 (h264_qsv) ([33][0][0][0] / 0x0021), nv12, 176x96 [SAR 1:1
DAR 11:6], q=2-31, 1000 kb/s, 30 fps, 15360 tbn, 30 tbc (default)
    Metadata:
      handler_name    : VideoHandler
      encoder         : Lavc57.64.101 h264_qsv
    Side data:
      cpb: bitrate max/min/avg: 0/0/1000000 buffer size: 0 vbv_delay: -1
Stream mapping:
  Stream #0:0 -> #0:0 (mpeg4 (native) -> h264 (h264_qsv))
```

Success is indicated by h264_qsv as the encoder.

Sample command lines for testing mpeg2 and h264 encodes:
$ FFmpeg –y –s <widthxheight> -i input.yuv –vcodec h264_qsv output.h264
$ FFmpeg –y -s <widthxheight> -i input.yuv –vcodec mpeg2_qsv output.mpg

The next section shows how to verify that the _qsv codecs use the GPU.

A quick note on hevc_qsv: HW accelerated HEVC is available only in the professional package . FFmpeg needs the GUID for HEVC which can be found in /opt/intel/mediasdk/plugins/plugin.cfg.

Here is an example of how the plugins.cfg would look like:

```
[H264LA_Encoder_588f1185d47b42968dea377bb5d0dcb4]
GUID = 588f1185d47b42968dea377bb5d0dcb4
PluginVersion = 1
APIVersion = 275
Path = /opt/intel/mediasdk/plugins/libmfx_h264la_hw64.so
Type = 4
CodecID = AVC
Default = 0
[VP8_Decoder_f622394d8d87452f878c51f2fc9b4131]
GUID =  f622394d8d87452f878c51f2fc9b4131
PluginVersion = 1
APIVersion = 275
Path = /opt/intel/mediasdk/plugins/libmfx_vp8d_hw64.so
Type = 1
CodecID = VP8
Default = 0
[HEVC_Decoder_33a61c0b4c27454ca8d85dde757c6f8e]
GUID = 33a61c0b4c27454ca8d85dde757c6f8e
PluginVersion = 1
APIVersion = 275
Path = /opt/intel/mediasdk/plugins/libmfx_hevcd_hw64.so
```

Type = 1
CodecID = HEVC
Default = 0
[HEVC_Encoder_6fadc791a0c2eb479ab6dcd5ea9da347]
GUID = 6fadc791a0c2eb479ab6dcd5ea9da347
PluginVersion = 1
APIVersion = 275
Path = /opt/intel/mediasdk/plugins/libmfx_hevce_hw64.so
Type = 2
CodecID = HEVC
Default = 0

Use the GUID entry that you see that corresponds to HEVC encode (highlighted above)

$FFmpeg -y -s <widthxheight> -i <input.yuv> -vcodec hevc_qsv -b:v <bitrate> -maxrate <maxrate> -load_plugins 6fadc791a0c2eb479ab6dcd5ea9da347 output.h265

Checking GPU utilization

This final step shows how to use the Intel Media Server Studio Metrics Monitor to access detailed information about GPU utilization. This tool is distributed with all Intel Media Server Studio editions. The Metrics Monitor Reference Manual has more details. It can be useful for single machine optimizations, as well as for load balancing across multiple machines. As an installation step, it can verify that hardware acceleration is activated.

The Metrics Monitor is placed in /opt/intel/mediasdk/tools during Intel Media Server Studio install. It must be run as root. An API can be used to construct custom GPU load queries in applications, but the sample is a great starting point to show GPU utilization.

First compile the sample with build.sh, then run with run.sh:
$ $ cd /opt/intel/mediasdk/tools/metrics _ monitor/sample/
$ ./build.sh
$ su
# ./run.sh

Lines of output like these should start appearing:
RENDER usage: 55.00, VIDEO usage: 22.00, VIDEO _ E usage: 0.00 VIDEO2 usage: 85.00
RENDER usage: 63.00, VIDEO usage: 26.00, VIDEO _ E usage: 0.00 VIDEO2 usage: 86.00

Here is how to interpret the output:

SOLUTION STACK

COMPONENT	NOTES
FFmpeg _qsv codecs	Glue code bridging FFmpeg APIs to Intel® Media Server Studio SDK, requires everything below.
Intel® Media Server Studio plugins	HEVC plugin is required for hevc_qsv
Intel® Media Server Studio libraries	Ensure all components installed
Graphics stack	Check libva initialization
Driver	Is i915 loaded?
OS/kernel	Is supported Linux* distro used? Is modified kernel loaded?
Hardware/BIOS	Can OS see the Intel® Processor Graphics device?

• RENDER shows % EU (GPGPU) utilization
• VIDEO and VIDEO2 show fixed function hardware utilization. Note: 4th Generation Intel® Core™ processors and Intel® Xeon® processor E3 v3 family have only one VIDEO metric

If running software codecs, CPU load should be high and GPU load should be zero or low in all of these categories. As GPU work is started, especially for mpeg2_qsv and h264_qsv, values for RENDER and VIDEO should go up showing that the GPU is being utilized. Multiple concurrent transcodes are more efficient, and will increase utilization closer to 100 percent than a single transcode.

The steps described so far outline end to end install with basic smoke test validation. Running _qsv codecs and
demonstrating GPU utilization indicates that they are ready to use.

Troubleshooting

These details are included to help diagnose and fix problems that may come up.
When troubleshooting, it is often helpful to consider the solution stack to guide investigating each component.
The rest of this section will walk through the stack from bottom up, starting with hardware.

Hardware test details
Intel processor graphics must be visible to the OS for any of the components above to work.
Check with lspci:

$ lspci -vv -nn -s 00:02.0

> 00:02.0 VGA compatible controller [0300]: Intel Corporation Broadwell-U Integrated Graphics [8086:162a] (rev 0a) (prog-if 00 [VGA controller])
> Subsystem: Intel Corporation Device [8086:2010]
> Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERRFastB2B- DisINTx+
> Status: Cap+ 66MHz- UDF- FastB2B+ ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR-
> <PERR- INTxLatency: 0

Interrupt: pin A routed to IRQ 48
Region 0: Memory at 90000000 (64-bit, non-prefetchable) [size=16M]
Region 2: Memory at a0000000 (64-bit, prefetchable) [size=512M]
Region 4: I/O ports at 5000 [size=64]
Expansion ROM at <unassigned> [disabled]
Capabilities: <access denied>
Kernel driver in use: i915

In the output above, note that you should see an Intel device 00:02.0 (/sys/bus/pci/devices/0000:00:02.0). It should report a valid PCI id, which can be found in linux-3.10.0-229.1.2.el7/include/drm/i915_pciids.h.

If output from your machine differs:

- Check processor SKU at http://ark.intel.com.
- Check system specs. Chipset and hardware block diagrams should anticipate driving a display from processor graphics. Please contact your hardware vendor for more information.
    - Chipset: most Intel Core processor based systems have a chipset suitable for processor graphics use, as Core processors are designed for interactive graphics use cases. Since many Intel Xeon processor based systems operate headlessly (no display), not all chipsets enable graphics.
    - Hardware: some servers assume the only GPU is a small VGA controller on the motherboard, and do not enable processor graphics.
- BIOS: On some systems processor graphics is not enabled by default. Boot to BIOS and search for a processor graphics option. In some cases, the BIOS shipped with a system does not include this option but the vendor may have a newer version which does. Please check with your hardware vendor.

OS/Kernel tests

The recommended “gold” OS for Intel Media Server Studio 2017 R2 is CentOS 7.2 (kernel 3.10-327.13.1)

The /dev/dri directory should have renderDN+128 and cardN interfaces

```
$ ls -l /dev/dri
total 0
crw-rw----+ 1 root video 226, 0 Sep 18 09:29 card0
crw-rw----+ 1 root video 226, 128 Sep 18 09:29 renderD128
```

Note: As permissions indicate, the user running Intel® Media Server Studio applications and FFmpeg _qsv codecs can be a regular user but must belong to the video group.

Using the render nodes interface is recommended, especially for applications which must run as a system service. FFmpeg initialization automatically searches render nodes first, and initialization should be sufficient for most cases. However, if there are initialization problems add “-v verbose” to the command line. This will print extra debug output, including initialization feedback. Checking which interface is used can be useful when multiple graphics adapters are available.

```
libva info: VA-API version 0.99.0
libva info: va_getDriverName() returns 0
libva info: User requested driver 'iHD'
libva info: Trying to open /opt/intel/mediasdk/lib64/iHD_drv_video.so
```

```
libva info: Found init function __vaDriverInit_0_32
libva info: va_openDriver() returns 0
Pipeline surfaces number (DecPool): 20
MFX HARDWARE Session 0 API ver 1.19 parameters:
Input  video: AVC
Output video: AVC

Session 0 was NOT joined with other sessions

Transcoding started
..
Transcoding finished
```

Driver tests

The i915 Intel graphics driver must start, and lsmod should show use count >1.

$ lsmod | grep 'i915'

```
i915                   938175  3
drm _ kms _ helper      98226  1  i915
drm                    311336  5  i915,drm _ kms _ helper
i2c _ algo _ bit        13413  2  igb,i915
i2c _ core              40325  6  drm,igb,i915,i2c _ i801,drm _ kms _
helper,i2c _ algo _ bit
video                   9263   1  i915
```

Check dmesg for any warning/error messages from i915 or drm.

VAAPI tests

The vainfo utility shows more details about libva initialization. Note: X server is not required.

$ vainfo

```
error: can't connect to X server!
libva info: VA-API version 0.99.0
libva info: va_getDriverName() returns 0
libva info: User requested driver 'iHD'
libva info: Trying to open /opt/intel/mediasdk/lib64/iHD_drv_video.so
libva info: Found init function __vaDriverInit_0_32
libva info: va_openDriver() returns 0
vainfo: VA-API version: 0.99 (libva 1.67.0.pre1)
vainfo: Driver version: 16.5.55964-ubit
vainfo: Supported profile and entrypoints
    VAProfileH264ConstrainedBaseline: VAEntrypointVLD
    VAProfileH264ConstrainedBaseline: VAEntrypointEncSlice
    VAProfileH264ConstrainedBaseline: <unknown entrypoint>
    VAProfileH264ConstrainedBaseline: <unknown entrypoint>
    VAProfileH264Main             : VAEntrypointVLD
```

```
VAProfileH264Main          : VAEntrypointEncSlice
VAProfileH264Main          : <unknown entrypoint>
VAProfileH264Main          : <unknown entrypoint>
VAProfileH264High          : VAEntrypointVLD
VAProfileH264High          : VAEntrypointEncSlice
VAProfileH264High          : <unknown entrypoint>
VAProfileH264High          : <unknown entrypoint>
VAProfileMPEG2Simple         : VAEntrypointEncSlice
VAProfileMPEG2Simple         : VAEntrypointVLD
VAProfileMPEG2Main          : VAEntrypointEncSlice
VAProfileMPEG2Main          : VAEntrypointVLD
VAProfileVC1Advanced         : VAEntrypointVLD
VAProfileVC1Main           : VAEntrypointVLD
VAProfileVC1Simple          : VAEntrypointVLD
VAProfileJPEGBaseline        : VAEntrypointVLD
VAProfileJPEGBaseline        : VAEntrypointEncPicture
VAProfileVP8Version0_3       : VAEntrypointEncSlice
VAProfileVP8Version0_3       : VAEntrypointVLD
VAProfileVP8Version0_3       : <unknown entrypoint>
VAProfileHEVCMain          : VAEntrypointVLD
VAProfileHEVCMain          : VAEntrypointEncSlice
VAProfileVP9Profile0        : <unknown entrypoint>
<unknown profile>          : VAEntrypointVideoProc
VAProfileNone            : VAEntrypointVideoProc
VAProfileNone            : <unknown entrypoint>
```

- Driver name should be iHD, with /opt/intel/mediasdk path
- Version should be 1.67.0.pre1 and driver version should match Intel Media Server SDK install
- Entry points should exist for MPEG2, VC1. VP8, JPEG, HEVC, H264

Since Intel Media Server Studio's stack uses non-standard names and locations, two variables should be set during installation:

```
$ env | grep 'LIBVA'
LIBVA _ DRIVERS _ PATH=/opt/intel/mediasdk/lib64
LIBVA _ DRIVER _ NAME=iHD
```

Intel® Media Server Studio libraries/plugins

In addition to the kernel and graphics stack changes described so far, Intel Media Server Studio install puts several libraries in /opt/intel/mediasdk. The directory should look like this:

```
$ ls /opt/intel/mediasdk
bin doc include lib lib64 opensource plugins tools uninstall uninstall.sh
```

FFmpeg troubleshooting

The FFmpeg _qsv codecs are the top of the stack described so far. If all Intel Media Server Studio diagnostics pass and samples run but the FFmpeg _qsv codecs do not start, look at:
• Was FFmpeg confgure successful?

• Do simpler FFmpeg command lines, such as those used in this paper, execute successfully?
• For hevc_qsv, is HEVC from Intel Media Server Studio Professional installed?

Conclusion

Enabling Intel Media Server Studio codecs in FFmpeg 2.8 provides the benefits of hardware acceleration for media encoding workflows.

Once compiled in, just change codec name to include acceleration on supported hardware. For example, switch libx264 to h264_qsv. These codecs provide a new range of choices and performance/quality options to add access to the hardware video processing capabilities on Intel processors.

The wide variety of software codecs enabled in FFmpeg and the _qsv codecs enable you to decrease your time to encode or increase density when compared to software only encoding when used on supported Intel hardware.

Legal Disclaimers

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer, or learn more at http://www.intel.com/content/www/us/en/architecture-and-technology/quick-sync-video/quick-sync-video-general.html

Copyright © 2017 Intel Corporation. All rights reserved. Intel, the Intel logo, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.
* Other names and brands may be claimed as the property of others.