WHITE PAPER

Intel® QuickSync Video and FFmpeg

Installation and Validation

Introduction

Intel® Quick Sync Video technology on Intel® Iris™ Graphics and Intel® HD graphics provides transcode acceleration on Linux* systems in FFmpeg* 2.8 and forward. 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: https://software.intel.com/en-us/articles/accessing-intel-media-server-studio-for-linux-codecs-with-ffmpeg.

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.

To get started:

1. **Install Intel Media Server Studio for Linux.** Download from https://software.intel.com/en-us/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 newer for those who prefer a stable release. Development is active so anyone needing latest updates and fixes should check the git repository tip.
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. Performance boost increases with resolution.

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

Intel Corporation
October 2015

Table of Contents

Installing Intel® Media Server Studio

Where to get it

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

- The Community edition is completely free and supports many popular codecs including 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® VTune™ analyzer. An evaluation package is available for Professional which limits HEVC encode to 1000 frames. For more info see the link above.

How to install

The steps below cover how to install Intel Media Server Studio 2015 releases on CentOS* 7.1. Please see the Release Notes, Getting Started Guide for more product details, including information on installation for other flavors of Linux.

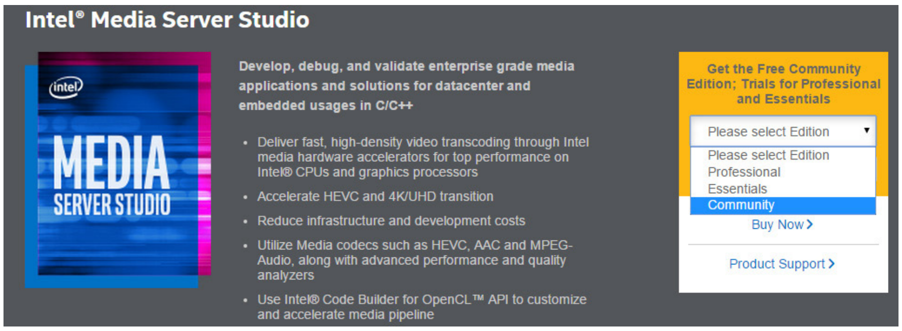
The steps below describe where to find the installation scripts.

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

Install has three stages:

1. To unpack and install the user mode components, run this script as root: **install_sdk_UMD_CentOS.sh**
2. Apply patches to the i915 driver as well as to the rest of the kernel. Rebuild and package as RPMs with **build_kernel_rpm_CentOS.sh**. Note: this step is run as a regular user, not as root.

Please note: the build_kernel_rpm script must be run from a very short path. Kernel RPM preparation may fail if a longer build directory path is used.



Intel® Media Server Studio

Develop, debug, and validate enterprise grade media applications and solutions for datacenter and embedded usages in C/C++

- Deliver fast, high-density video transcoding through Intel media hardware accelerators for top performance on Intel® CPUs and graphics processors
- Accelerate HEVC and 4K/UHD transition
- Reduce infrastructure and development costs
- Utilize Media codecs such as HEVC, AAC and MPEG-Audio, along with advanced performance and quality analyzers
- Use Intel® Code Builder for OpenCL™ API to customize and accelerate media pipeline

Intel® Media Server Studio

3. Install the kernel RPMs built by step 2. (If installing on multiple machines you can copy the RPMs to new systems instead of rebuilding.)

```
(as root)
# ./install_sdk_UMD_CentOS.sh
# mkdir /MSS
# chown {regular user}:{regular group} /MSS

(as regular user)
$ cp build_kernel_rpm_CentOS.sh /MSS
$ cd /MSS
$ ./build_kernel_rpm*.sh

(as root)
# cd /MSS/rpmbuild/RPMS/x86_64
# rpm -Uvh kernel-3.10.*.rpm
# reboot
```

These steps install multiple libraries and update the kernel. Your GRUB bootloader screen should have entries like this:

```
    CentOS Linux (3.10.0…buildnum.MSSr…) 7
    CentOS Linux …
```

Please make sure to boot to the new kernel, which has Intel Media Server Studio in the name.

If any difficulties are encountered, please see the troubleshooting section.

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 0.35.0
    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: 60
    MFX HARDWARE Session 0 API ver 1.16 parameters:
```

```
Input  video: AVC
Output video: AVC

Session 0 was NOT joined with other sessions
Transcoding started
..
Transcoding finished

Common transcoding time is  0.16 sec
MFX session 0 transcoding PASSED:
Processing time: 0.16 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 finishes 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. (The hevc_qsv codec requires the additional step of installing the HEVC plugin from the professional package. More info in the next section.) 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/training

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.

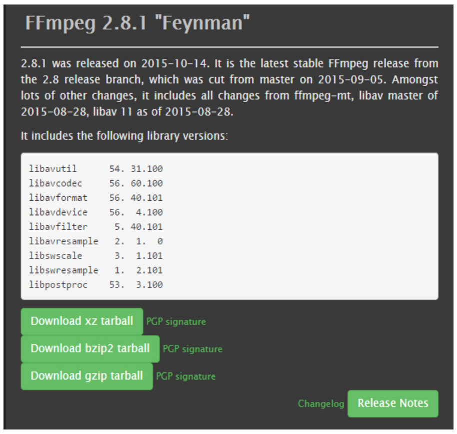
Development is active and updates are frequent. The FFmpeg repository tip is an option for evaluating the latest improvements.

How to install

The *_qsv codecs are enabled when FFmpeg is compiled. When running the configure script add "`--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 files to include/mfx. This location is used for historical compatibility.

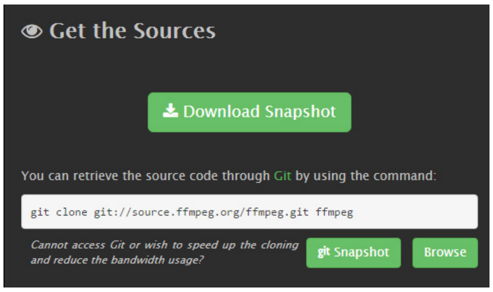


FFmpeg 2.8.1 "Feynman"

2.8.1 was released on 2015-10-14. It is the latest stable FFmpeg release from the 2.8 release branch, which was cut from master on 2015-09-05. Amongst lots of other changes, it includes all changes from ffmpeg-mt, libav master of 2015-08-28, libav 11 as of 2015-08-28.

It includes the following library versions:

```
libavutil      54. 31.100
libavcodec     56. 60.100
libavformat    56. 40.101
libavdevice    56.  4.100
libavfilter     5. 40.101
libavresample   2.  1.  0
libswscale      3.  1.101
libswresample   1.  2.101
libpostproc    53.  3.100
```



Get the Sources

You can retrieve the source code through Git by using the command:

```
git clone git://source.ffmpeg.org/ffmpeg.git ffmpeg
```

Cannot access Git or wish to speed up the cloning and reduce the bandwidth usage?

```
# 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. Same search rules apply as for other pkg-config configurations. This can be placed in the existing search path, such as /usr/lib64/pkgconfig for CentOS 7.1. The PKG_CONFIG_PATH environment variable can be used to customize the search path.

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: 16.4.2

Libs: -L${libdir} -lmfx -lva -lstdc++ -ldl -lva-drm -ldrm

Cflags: -I${includedir} -I/usr/include/libdrm
```

Step 3: 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_ea_r1         ccaption        h264_qsv
adpcm_ea_xas        cinepak         hevc_qsv
mpeg2_qsv           ppm             tscc
mvc1                realtext        vc1_qsv
```

License should also be unredistributable:

```
$ grep 'License:' config.out

License: nonfree and unredistributable
```

Step 4: 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_qsv )
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 two decoder/encoders are ready to use (h264_qsv, mpeg2_qsv) and one decoder (vc1_qsv). 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 first 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 N-75909-gf226c25 Copyright (c) 2000-2015 the FFmpeg developers
…
libva info: VA-API version 0.35.0
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     : M4V
    minor_version   : 1
    compatible_brands: M4V mp42isom
    encoder         : Lavf57.3.100
    Stream #0:0(eng): Video: h264 (h264_qsv) ([33][0][0][0] / 0x0021), nv12, 1280x720 [SAR 1:1 DAR
    16:9], q=2-31, 8000 kb/s, 29.97 fps, 30k tbn, 29.97 tbc (default)
    Metadata:
      creation_time   : 2014-10-01 16:47:05
      handler_name    : Mainconcept MP4 Video Media Handler
      encoder         : Lavc57.5.100 h264_qsv
    Stream #0:1(eng): Audio: aac ([64][0][0][0] / 0x0040), 48000 Hz, stereo, 157 kb/s (default)
    Metadata:
      creation_time   : 2014-10-01 16:47:05
      handler_name    : Mainconcept MP4 Sound Media Handler
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> h264 (h264_qsv))
  Stream #0:1 -> #0:1 (copy)
```

Success is indicated by h264_qsv as the encoder. The next section shows how to verify that the _qsv codecs use the GPU.

A quick note on hevc_qsv. This codec is only enabled by installing the HEVC plugin in the professional package. As a plugin, a GUID must be provided to FFmpeg. This can be found in /opt/intel/mediasdk/plugins/plugin.cfg. Here is an example command line:

```
$ ffmpeg -i ~/Videos/ToS-1920x1080.h264 -vcodec hevc_qsv -load_plugins e5400a06c74d41f5b12d430b-
baa23d0b -preset fast out.hevc
```

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.

Performance note: the current FFmpeg implementation is not well optimized yet. GPU utilization, frames-per-second, and concurrent transcode capacity are significantly less under FFmpeg than with a "native" application optimized for best Intel Media Server Studio performance.

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- SERR-
         FastB2B- DisINTx+
        Status: Cap+ 66MHz- UDF- FastB2B+ ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR-
         <PERR- INTx-
        Latency: 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. Is it 4th/5th Generation Intel Core processor or Intel Xeon processor E3-12xx v3/4 with processor graphics?
- 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 2015 R6 is CentOS 7.1.

Required kernel changes are made during installation. Check that the updated kernel is running.

```
$ uname -r
3.10.0-229.1.2.39163.MSSr4.el7.centos.x86
```

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 0.35.0
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
[h264_qsv @ 0x2340040] mfx initialization: /dev/dri/renderD128 vaInitialize successful
[h264_qsv @ 0x2340040] Initialized an internal MFX session using hardware accelerated implemen-
tation
```

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 0.35.0
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: 0.35 (libva 1.3.1)
vainfo: Driver version: 16.4.2.1.39163-ubit
vainfo: Supported profile and entrypoints
      VAProfileNone                   : VAEntrypointVideoProc
      VAProfileNone                   : <unknown entrypoint>
      VAProfileMPEG2Simple            : VAEntrypointEncSlice
      VAProfileMPEG2Simple            : VAEntrypointVLD
      VAProfileMPEG2Main              : VAEntrypointEncSlice
      VAProfileMPEG2Main              : VAEntrypointVLD
      VAProfileH264Baseline           : VAEntrypointEncSlice
      VAProfileH264Baseline           : <unknown entrypoint>
      VAProfileH264Baseline           : <unknown entrypoint>
      VAProfileH264Main               : VAEntrypointVLD
      VAProfileH264Main               : VAEntrypointEncSlice
      VAProfileH264Main               : <unknown entrypoint>
      VAProfileH264Main               : <unknown entrypoint>
      VAProfileH264High               : VAEntrypointVLD
      VAProfileH264High               : VAEntrypointEncSlice
      VAProfileH264High              : <unknown entrypoint>
      VAProfileH264High              : <unknown entrypoint>
      VAProfileVC1Simple             : VAEntrypointVLD
      VAProfileVC1Main               : VAEntrypointVLD
      VAProfileVC1Advanced           : VAEntrypointVLD
      VAProfileJPEGBaseline          : VAEntrypointVLD
      VAProfileJPEGBaseline          : VAEntrypointEncPicture
      VAProfileH264ConstrainedBaseline: VAEntrypointVLD
      VAProfileH264ConstrainedBaseline: VAEntrypointEncSlice
      VAProfileH264ConstrainedBaseline: <unknown entrypoint>
      VAProfileH264ConstrainedBaseline: <unknown entrypoint>
      VAProfileVP8Version0 _ 3        : VAEntrypointEncSlice
      VAProfileVP8Version0 _ 3        : VAEntrypointVLD
      VAProfileVP8Version0 _ 3        : <unknown entrypoint>
```

- Driver name should be iHD, with /opt/intel/mediasdk path
- Version should be 1.3.1 and driver version should match Intel Media Server SDK install
- Entry points should exist for MPEG-2 and H.264

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
```

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:

```
(cd to SDK directory)
# rpm --force -Uvh *.rpm
```

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 configure 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.