Hybrid HEVC Encoder Plugin for FFmpeg 1.2

User Guide

Contents

1. General Overview

Nowadays FFmpeg is widely used in professional content production for file-based transcoding as well as live use cases. Although FFmpeg supports most video formats natively, there is no easy way to make use of MainConcept's industry-leading and professional codec libraries within an FFmpeg workflow. The MainConcept Hybrid HEVC Encoder Plugin for FFmpeg is a convenient way to solve this problem because it seamlessly integrates into FFmpeg.

Features:

- Use MainConcept's industry leading HEVC/H.265 software encoder natively in FFmpeg.
- Hardware accelerated HEVC/H.265 encoding powered by Intel Quick Sync Video and NVIDIA NVENC.
- Ready-to-use presets for DASH-265 and Apple HLS-HEVC up to 4K.
- HEVC/H.265 Main and Main 10, 4:2:0 and 4:2:2 support
- GPU-accelerated HEVC/H.265 encoding modes (Hybrid & Driven) on supported NVIDIA RTX, GTX and Quadro boards for significantly increased performance while providing a similar quality than the MainConcept software encoder.

2. Software Installation

To use MainConcept Hybrid HEVC Encoder Plugin for FFmpeg, you must have a compatible configuration.

Supported Operating Systems:

- Microsoft® Windows® 8, Windows 10 (64-bit)
- Linux Ubuntu 16.04 LTS (glibc2.17), CentOS 7.4 (64-bit)
- MainConcept modified version of FFmpeg 4.2.2 "Ada" (see available download below)

To run the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg the following software packages must be installed in this order:

1) A modified FFmpeg version as ready-to-use binary or as source code that you must compile yourself. The binaries can be found on the MainConcept website. The source code of the modified FFmpeg can be found here: https://github.com/MainConcept/mc-ffmpeg-omx
2) The actual MainConcept FFmpeg Plugin Demo version installer that can be downloaded from the MainConcept website: https://www.mainconcept.com/.
3) Alternatively, you can install the full version of the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg if you own a valid license (optional).

3. Installation on Windows

To run the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg on Windows the following software packages must be installed in this order:

1. Install modified FFmpeg
2. Install MainConcept FFmpeg Plugins Demo version
3. Install licensed MainConcept Hybrid HEVC Encoder Plugin for FFmpeg full version (optional)

3.1.1 Modified FFmpeg & Demo Plugin Installation on Windows

First, you need to install the required FFmpeg version for the MainConcept FFmpeg Plugins on your system. Please follow the steps below.

1. Run the "ffmpeg_static_4.2.2-omx_win64_1.2.0.exe" installer file to launch the installation wizard. In the **Welcome** dialog, click Next to proceed.

2. When the license agreement (EULA) appears on the screen, review it carefully. Click **I Agree** to accept the terms. If you do not agree, the installation process will be aborted.

3. You are asked for the destination folder, where FFmpeg should be installed. We recommend using the default location. Click **Next** to proceed.

NOTE:
You must also install the MainConcept FFmpeg Plugins Demo or the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg full version to this folder later.

4. Under Windows, you can also choose a Start Menu folder. We recommend using the default location. Click **Next** to proceed.

5. Now the installation starts. An indicator will show the installation process.

6. When the following dialog box appears, click **Finish** to complete the setup.

FFmpeg is now installed on your computer!

You must now install the MainConcept FFmpeg Plugin demo version for evaluation. It must be installed to the same location where you have installed FFmpeg before:

7. Run the "mainconcept_ffmepg_plugins_demo" installer file to launch the installation wizard. In the **Welcome** dialog, click **Next** to proceed.

8. When the license agreement (EULA) appears on the screen, review it carefully. Click **I Agree** to accept the terms. If you do not agree, the installation process will be aborted.

9. You are asked for the destination folder. However, the MainConcept FFmpeg Plugins Demo must be installed to the same folder where FFmpeg was installed before. Click **Next** to proceed.

10. Under Windows, you can also choose a Start Menu folder. We recommend using the default location. Click **Next** to proceed.

11. Now the installation starts. An indicator will show the installation process.

12. When the following dialog box appears, click **Finish** to complete the setup.

The MainConcept FFmpeg Plugins Demo is now installed on your computer! You can now start evaluating the software.

NOTE:
The MainConcept FFmpeg Plugins Demo package contains demo versions for both the MainConcept Hybrid HEVC Encoder, MainConcept AVC Broadcast Encoder and MainConcept AVC Encoder.

3.1.2 MainConcept Hybrid HEVC Encoder Plugin Full Version Installation on Windows

In this short chapter, we briefly describe how to install the full version of the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg if you already own a valid license after purchase. It must be installed to the same location where you have installed FFmpeg before.

1. If you haven't installed the modified FFmpeg yet, please follow the steps 1 – 6 from the previous chapter.

2. Run the "mainconcept_hybrid_hevc_encoder_plugin_full" installer file to launch the installation wizard. In the **Welcome** dialog, click **Next** to proceed.

3. When the license agreement (EULA) appears on the screen, review it carefully. Click **I Agree** to accept the terms. If you do not agree, the installation process will be aborted.

4. You are asked for the destination folder. However, the MainConcept MainConcept Hybrid HEVC Encoder Plugin must be installed to the same folder where FFmpeg was installed before. Click **Next** to proceed.

5. Under Windows, you can also choose a Start Menu folder. We recommend using the default location. Click **Next** to proceed.

6. When the following dialog box appears, click **Finish** to complete the setup.

The MainConcept Hybrid HEVC Encoder Plugin for FFmpeg is now installed on your computer! You now need to activate the licensed version of the software before it can be used.

3.2 Installation on Linux

To run the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg on Linux the following software packages must be installed in this order:

1. Install modified FFmpeg
2. Install MainConcept FFmpeg Plugins Demo version
3. Install licensed MainConcept Hybrid HEVC Encoder Plugin for FFmpeg full version (optional)

2.2.1 Modified FFmpeg & Demo Plugin Installation on Linux

As a first package, you must install the MainConcept modified version of FFmpeg that enables MainConcept FFmpeg Plugins:

1. Unpack the downloaded package and run the self-extracting executable and accept the EULA:

```
tar xf mc_ffmpeg_installer_gcc_linux64_x64_<version_id>.tar.bz2
./ffmpeg_static_4.2.2-omx_linux64_1.2.0.run
```

2. Install the package file according to your Linux base system:

   Debian-based Linux:

```
sudo dpkg -i ffmpeg_omx/deb/ffmpeg-static_4.2.2-0omx.0~3992_amd64.deb
```

   RPM-based Linux:

```
sudo yum localinstall ffmpeg_omx/rpm/ffmpeg-static-4.2.2-0omx.0.3992.el7.x86_64.rpm
```

3. Verify that the MainConcept modified FFmpeg is correctly installed by calling *ffmpeg* from the installation folder:

   /opt/mainconcept/ffmpeg-omx/bin/ffmpeg

```
ffmpeg version n4.2.1-456-g7af8b3b Copyright (c) 2000-2019 the FFmpeg developers built
with gcc 4.8.5 (GCC) 20150623 (Red Hat 4.8.5-39) configuration: --disable-ffplay --
disable-doc --enable-static --disable-shared --disable-debug --enable-asm --cc=gcc --
enable-x86asm --enable-omx --enable-omx_enc_avc --enable-omx_enc_hevc --extra-cflags=-
I../omxil_common/include/omx --prefix=../dist/linux-x64
```

NOTE:
You should see output containing "--enable-omx --enable-omx_enc_avc --enable-omx_enc_hevc"

FFmpeg is now installed on your computer!

You must now install the MainConcept FFmpeg Plugin demo version for evaluation:

4. Unpack the demo plugin tarball, then run the self-extracting executable and accept the EULA:

```
tar xf mc_ffmpeg_plugins_demo_installer_gcc_linux64_x64_sfx-<build_id>.tar.bz2

./mainconcept_ffmpeg_plugin_linux64_demo_<version_id>.run
```

5. Install the package files according to your Linux base system:

   Debian-based Linux:

```
cd mc_ffmpeg_plugins/deb/
sudo dpkg -i -f mcomx-core_<version_id>_amd64.deb
sudo dpkg -i -f mcomx-encavc_<version_id>_amd64.deb
sudo dpkg -i -f mcomx-enchevc_<version_id>_amd64.deb
sudo dpkg -i -f mc-encavc-demo_<version_id>_amd64.deb
sudo dpkg -i -f mc-enchevc-demo_<version_id>_amd64.deb
sudo dpkg -i -f mc-sdk-conf_<version_id>_amd64.deb

Alternatively, you can install all at once:

sudo dpkg -i *.deb
```

   RPM-based Linux:

```
cd mc_ffmpeg_plugins/rpm/
sudo yum localinstall mcomx-core_<version_id>.x86_64.rpm
sudo yum localinstall mcomx-encavc_<version_id>.x86_64.rpm
sudo yum localinstall mcomx-enchevc_<version_id>.x86_64.rpm
sudo yum localinstall mc-encavc-demo_<version_id>.x86_64.rpm
sudo yum localinstall mc-enchevc-demo_<version_id>.x86_64.rpm
sudo yum localinstall mc-sdk-conf_<version_id>.x86_64.rpm

Alternatively, you can install all at once:

sudo yum localinstall *
```

The MainConcept FFmpeg Plugins Demo is now installed on your computer! You can now start evaluating the software.

NOTE:
The MainConcept FFmpeg Plugins Demo package contains demo versions for both the MainConcept Hybrid HEVC Encoder, MainConcept AVC Broadcast Encoder and MainConcept AVC Encoder.

2.2.2 MainConcept Hybrid HEVC Encoder Plugin Full Version Installation on Linux

To install the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg full version, you must first install the evaluation version as described in the previous section. Afterwards, continue here:

1. Unpack the full version plugin tarball, then run the self-extracting executable and accept the EULA:

```
tar xf mc_ffmpeg_plugin_hybrid_hevc_encoder_installer_gcc_linux64_x64_sfx-
installer4_b4003.tar.bz2

./mainconcept_hybrid_hevc_encoder_plugin_linux64_full_1.1.0.run
```

2. Install the package files according to your Linux base system:

   Debian-based Linux:

```
sudo dpkg --force-depends -i codemeter_7.0.3918.500_amd64.deb
sudo dpkg -i mc-encavc_<version_id>_amd64.deb

We recommend the full WIBU runtime installer described above. However, if you require
a CLI only version, you can alternatively install the lite version as described below
and follow the section "1.10 CMU - CodeMeter Universal Support Tool" in WIBU CodeMeter
Administrator Manual from here:

sudo dpkg --force-depends -i codemeter-lite_7.0.3918.500_amd64.deb
```

   RPM-based Linux:

```
sudo dpkg --force-depends -i codemeter_7.0.3918.500_amd64.deb
sudo dpkg -i mc-encavc_<version_id>_amd64.deb

We recommend the full WIBU runtime installer described above. However, if you require
a CLI only version, you can alternatively install the lite version as described below
and follow the section "1.10 CMU - CodeMeter Universal Support Tool" in WIBU CodeMeter
Administrator Manual from here:

sudo dpkg --force-depends -i codemeter-lite_7.0.3918.500_amd64.deb
```

The MainConcept Hybrid HEVC Encoder Plugin for FFmpeg is now installed on your computer! You now need to activate the licensed version of the software before it can be used.

3. License Activation

3.1 Online License Activation

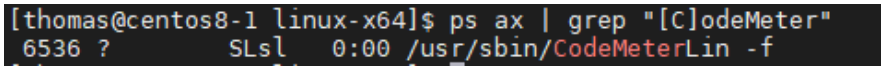
When you purchased the full licensed product, you have received a license activation link. Use this link only on the computer where you installed the software with the steps below:

1. Activation requires WIBU-Systems' Codemeter. This software is installed during the installation process of the full licensed product described above.

2. Verify that **CodeMeter runtime** is running by looking for the CodeMeter icon on your Windows taskbar:

On Linux, check if the CodeMeter daemon process is running:

```
$ ps ax | grep "[C]odeMeter"
```



You can check if Codemeter Control Center is running correctly on Linux in the menu **Applications > Accessories > CodeMeter Control Center**. You can also run this with the following command line:

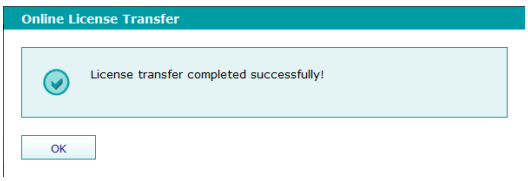
```
$ /usr/bin/CodeMeterCC -m
```

3. To activate the license, you must have an active internet connection. Open a browser on the computer where you installed the software and copy & paste the activation link.

The activation link will open the CodeMeter License Central web page below. Here you can choose whether to use a USB hardware dongle (CMDongle) or a software license (CMActLicense) to activate the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg. In this release, please use the software activation, i.e. the CmActLicense on the right side by clicking it.

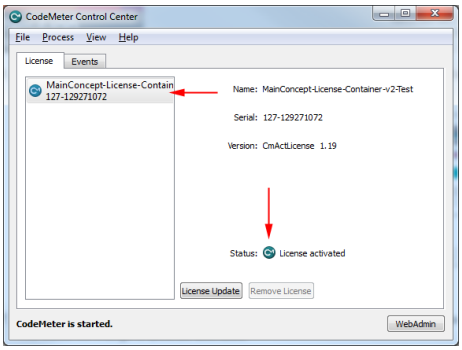
4. The following page will appear on the screen. You should ensure that the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg is selected. The **Select CMContainer** drop-down menu should show **Get CmContainer** automatically. To proceed with the activation, simply click the **Activate Selected Licenses Now** button.

5. If successful, you will get the following notification. Press **OK** to proceed.



6. After a few seconds the status will change to **Activated**. The license is now activated and can be used.

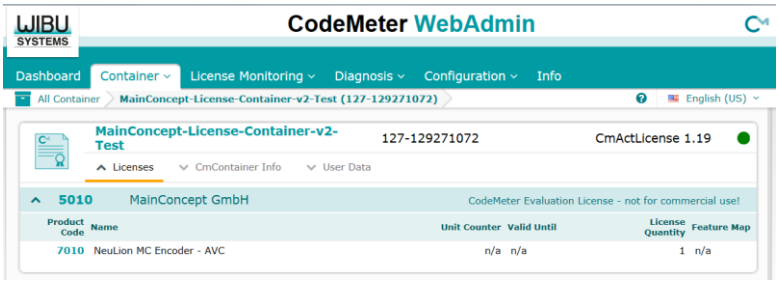
7. If you open **CodeMeter Control Center** you will also see the last activated product as confirmation. The status will change to **License activated**.



8. You will also notice that the CodeMeter icon in your task bar has changed from red to green:

9. In case you have activated more than one license you get a more detailed overview in WebAdmin.

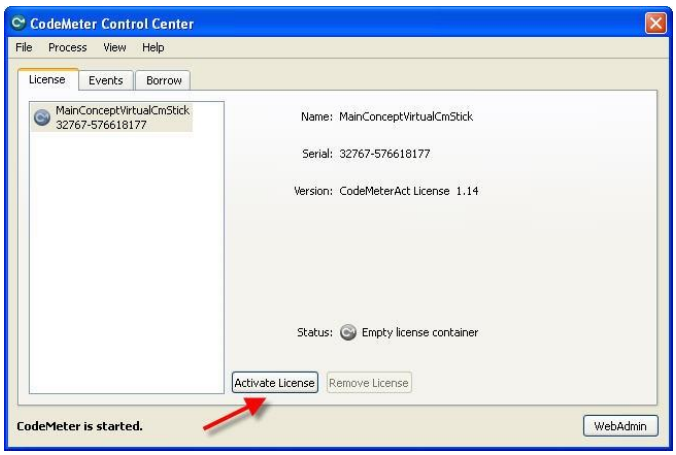
WebAdmin will open in a browser window. By choosing **Container > Licenses** you will get an overview about all installed licenses:



3.2 Offline License Activation

A license can be activated on a system without internet connectivity as follows:

1. Open the CodeMeter Control Center and press the *Activate License* button.



2. Confirm the **Welcome** screen with ***Next***.

3. Tick the option ***Create license request*** and confirm with ***Next***.

4. Select a file name. The dialog will make a suggestion. However, you can enter your own. Please note that the suffix *".WibuCmRaC"* is required.

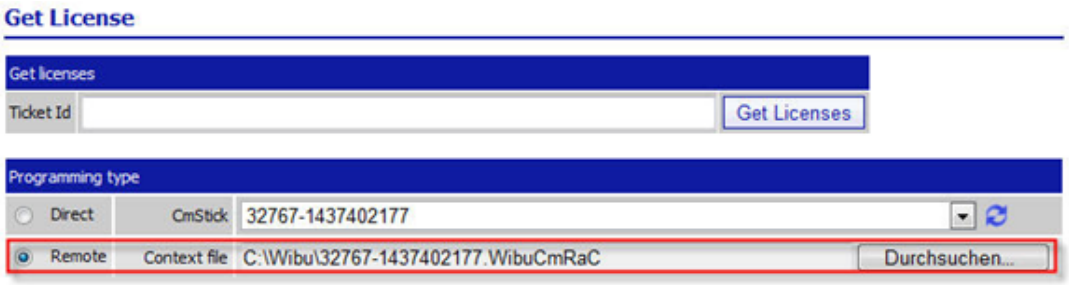
5. Transfer the stored file to a PC with internet connectivity (e.g. using an USB stick).

6. During the purchasing process you received a **License Ticket ID**. Click or copy and paste the provided link into your browser.

   The browser will access the **MainConcept License Center** where the product will automatically show up.

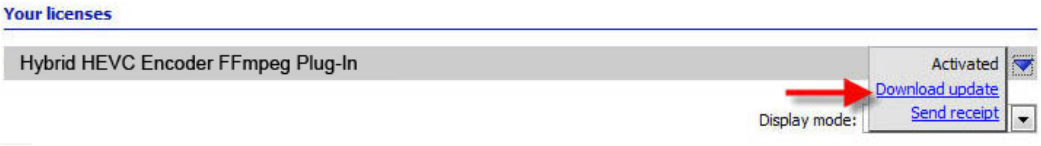
7. Tick ***Remote*** in the **Programming type** section.

8. Pick the context file (license request) you have created before on your target system. Press **Activate now**.

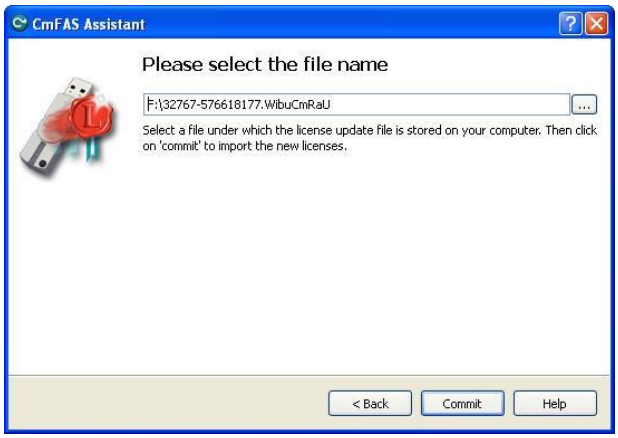


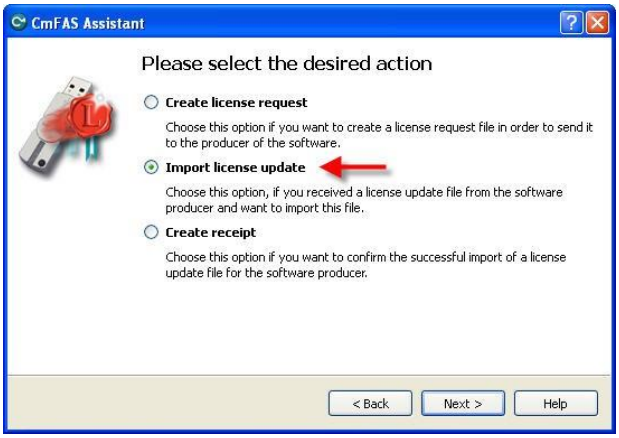
9. Confirm the activation process. After a few seconds, the status will change to ***Activated***.

10. The download of the update file ("*.WibuCmRaU") should automatically start. In case of browser problems such as pop-up blocker or extended security settings, please go to the activated item and press ***Download update*** in the drop-down menu.



11. Transfer the activation file ("*.WibuCmRaU") to your target system where the Plugin for FFmpeg is installed. Open the **Control Center** again, go to **Activate License** and pick **Import license update** this time, selecting the file you have saved before. Confirming with ***Commit*** will activate the license.





The license is now activated.

4. Plugin Usage

In the following we briefly describe how to use the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg. The command line format should follow this structure:

On Windows:

```
ffmpeg <ffmpeg-params> \
        -c:v omx_enc_hevc
        -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video \
        -omx_param "<mc-general-params> \
                [HEVC Settings] "<mc-codec-params>" \
                [HEVC Layer 0000] "<mc-codec-params-layer0>" \
        <ffmpeg-output-parameters>
```

Here is a command-line example:

```
ffmpeg -r 25.000000 -pix_fmt yuv420p -s 1920x1080 -i "D:\1920x1080p_25p_YV12.yuv" -b:v 3500k -
c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -
omx_param "force_omx_param=1:preset=main:
perf_level=10:acc_type=nvenc:acc_mode=hybrid:[HEVC
Settings]:time_scale=20000000:num_units_in_tick=1000000:[HEVC Layer
0000]:bit_rate_mode=3:hrd_conformance=2" "D:\1920x1080p_25p_YV12_ffmpeg.hevc" -y
```

Here is a command-line example using a config file:

```
ffmpeg -r 25.000000 -pix_fmt yuv420p -s 1920x1080 -i "D:\1920x1080p_25p_YV12.yuv" -b:v 3500k -
c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -
omx_param "force_omx_param=0:cfg_file_path=hevc_config.ini"
"D:\1920x1080p_25p_YV12_ffmpeg.hevc" -y
```

On Linux:

NOTE:
The modified ffmpeg executable is installed in /opt/mainconcept/ffmpeg-omx/bin/. For the following examples make sure this folder is in your search path!

```
ffmpeg <ffmpeg-params> \
        -c:v omx_enc_hevc
        -omx_core libomxil_core.so -omx_name OMX.MainConcept.enchevc.video \
        -omx_param "<mc-general-params> \
                [HEVC Settings] "<mc-codec-params>" \
                [HEVC Layer 0000] "<mc-codec-params-layer0>" \
        <ffmpeg-output-parameters>
```

Here is a command-line example:

```
ffmpeg -r 25.000000 -pix_fmt yuv420p -s 1920x1080 -i "D:\1920x1080p_25p_YV12.yuv" -b:v 3500k -c:v omx_enc_hevc -omx_core libomxil_core.so -omx_name OMX.MainConcept.enchevc.video -omx_param "force_omx_param=1:preset=main:perf_level=10:acc_type=nvenc:acc_mode=hybrid:[HEVC Settings]:time_scale=20000000:num_units_in_tick=1000000:[HEVC Layer 0000]:bit_rate_mode=3:hrd_conformance=2" "D:\1920x1080p_25p_YV12_ffmpeg.hevc" -y
```

Here is a command-line example using a config file:

```
ffmpeg -r 25.000000 -pix_fmt yuv420p -s 1920x1080 -i "D:\1920x1080p_25p_YV12.yuv" -b:v 3500k -c:v omx_enc_hevc -omx_core libomxil_core.so -omx_name OMX.MainConcept.enchevc.video -omx_param "force_omx_param=0:cfg_file_path=hevc_config.ini" "D:\1920x1080p_25p_YV12_ffmpeg.hevc" -y
```

All settings below are optional parameters:

omx_param Parameters (mc-general-params / mc-codec-params):

Function	Description
force_omx_param	Specifies whether to prioritize the FFmpeg generic global options or the MainConcept codec specific settings: • **0**: The force_omx_param flag is not set, i.e. FFmpeg 's generic global options will overwrite the omx_param settings used by the MainCocept codecs. • **1**: The FFmpeg generic global options are ignored, i.e. the MainConcept codec specific settings are exclusively used.
perf_level	Specifies the predefined performance level 1 to 31 for encoding. Default is 15. Also predefined values are allowed: • "fastest" = 1 • "faster" = 8 • "balanced" = 15 • "better_quality" = 20 • "best_quality" = 30
acc_type	Specified whether you want to use software ("sw"), Intel Quick Sync Video ("iqsv") or NVIDIA NVENC ("nvenc") encoding. IQSV and NVENC are GPU encoding modes.

acc_mode	Special encoding modes for NVIDA RTX boards: • "full": All encoding is done on the GPU. • "driven": Software pre-analysis and rate control, hardware encoding (all frames are encoded by the GPU) • "hybrid": Software pre-analysis and rate control, hybrid encoding (some frames are encoded by the GPU, others are software encoded).
preset	Specifies the builtin encoder presets. Available options are "main", "main10", "main_422_10", "4k", "4k10", "hls1"..."hls12", "dash1"..."dash12". For more details, please refer to the sample command-lines in the chapter Command Line Examples of this user guide.
cfg_file_path	Specifies path to HEVC/H.265 encoder config file containing all parameters for encoding. *-omx_param "force_omx_param=1:cfg_file_path=./hevc_config.ini"*
pass	2-pass encoding is an option to achieve better visual quality but with strict target bitrate and HRD compliance. It cannot be used in live encoding. In the first encoding pass, called analyze, intermediate statistics are saved for further usage in the second pass where the actual encoding will take place. 2-pass encoding can take up to twice longer than a single-pass encoding. • **1**: Analysis pass. • **2**: Encoding pass that uses statistics saved in a temporary file during the analysis pass. For 2-pass encoding, you need to run FFmpeg twice, the first pass covers the analyzing step and the second one the actual encoding step. Here are command-line examples: *ffmpeg -s 720x576 -pix_fmt yuv420p -i /home/Projects/ffmpeg_omx/test/streams/input.yuv -c:v omx_enc_hevc -omx_core /home/scum/Projects/omx_sdk_battoolbar/bin/libomxil_core.so -omx_name OMX.MainConcept.enchevc.video -b:v 500000 -profile main -omx_param "pass=1" mc_ffmpeg_2pass.h265* *ffmpeg -s 720x576 -pix_fmt yuv420p -i /home/Projects/ffmpeg_omx/test/streams/input.yuv -c:v omx_enc_hevc -omx_core /home/scum/Projects/omx_sdk_battoolbar/bin/libomxil_core.so -omx_name OMX.MainConcept.enchevc.video -b:v 500000 -profile main -omx_param "pass=2" mc_ffmpeg_2pass.h265*
passlogfile	Specifies the filename and path where to store the log file that includes the data and statistics from the first analyzing pass. This parameter is optional. If not specified the file is created with default naming in the current working directory. Here are command-line examples:

	ffmpeg -s 720x576 -pix_fmt yuv420p -i /home/Projects/ffmpeg_omx/test/streams/input.yuv -c:v omx_enc_hevc -omx_core /home/scum/Projects/omx_sdk_battoolbar/bin/libomxil_core.so -omx_name OMX.MainConcept.enchevc.video -b:v 500000 -profile main -omx_param "pass=1:passlogfile=mc_mpass_hevc" mc_ffmpeg_2pass.h265 *ffmpeg -s 720x576 -pix_fmt yuv420p -i /home/Projects/ffmpeg_omx/test/streams/input.yuv -c:v omx_enc_hevc -omx_core /home/scum/Projects/omx_sdk_battoolbar/bin/libomxil_core.so -omx_name OMX.MainConcept.enchevc.video -b:v 500000 -profile main -omx_param "pass=2:passlogfile=mc_mpass_hevc" mc_ffmpeg_2pass.h265*
[HEVC Settings]	Specifies parameters from the HEVC/H.265 encoder config on the command line which can be applied under "[HEVC Settings]". Please use : as a separator and = as a value set for the key. They must have the same structure and order as they would appear in an *.ini file, e.g. *-omx_param "force_omx_param=1:preset=4k:acc_type=nvenc:acc_mode=hybrid:[HEVC Settings]:max_num_reencodings=0:input_filtering=2: ... :[HEVC Layer 0000]:bit_rate_mode=3:hrd_conformance=2: ... : ..."* These arguments are matching the MainConcept HEVC / H.265 encoder settings (see the chapter HEVC/H.265 Encoder Parameters under [HEVC Settings] as a reference later in this user guide or an HEVC encoder config file as an example).
[HEVC Layer 0000]	Specifies parameters from the HEVC/H.265 encoder config on the command line which can be applied under for layer 0 ("[HEVC Layer 0000]"). Please use : as a separator and = as a value set for the key. They must have the same structure and order as they would appear in an *.ini file, e.g. *-omx_param "force_omx_param=1:preset=4k:acc_type=nvenc:acc_mode=hybrid:[HEVC Settings]:max_num_reencodings=0:input_filtering=2: ... :[HEVC Layer 0000]:bit_rate_mode=3:hrd_conformance=2: ... : ..."* These arguments are matching the MainConcept HEVC / H.265 encoder settings (see the chapter HEVC/H.265 Encoder Parameters under [HEVC Layer 0000] as a reference later in this user guide or an HEVC encoder config file as an example).

NOTE:
The plugin does not support the SABET feature in FFmpeg right now, therefore, only LAYER0 is used.

5. Force Key Frame Settings

The MainConcept HEVC Encoder Plugin offers an option to force the current frame to become an IDR frame with FFmpeg. This option is essential for adaptive bitrate streaming formats like Apple HLS and MPEG-DASH for creating segments that only occur at key frames. This will provide smooth proper playback of the segments.

You can use FFmpeg's generic *force_key_frames* parameter to specify when a new IDR frame should be set. However, you also need to define some MainConcept HEVC/H.265 Video Encoder parameters via *-omx_param*. So if *-force_key_frames 'expr:gte(t,n_forced*N)'* is present (*N* is the number of seconds) then you should disable *fixed_intra_position* (set it to 0), which means irregular IRAP (Intra Random Access Pictures) placement.

There are currently two typical use cases to set *force_key_frames* in FFmpeg:

1) *…-force_key_frames 'expr:gte(t,n_forced*N) … -omx_param "… :fixed_intra_position=0:vcsd_mode= 0:…"*
   ⇨ IDR-frames are only at every framerate*N position.

2) *…-force_key_frames 'expr:gte(t,n_forced*N) … -omx_param "… :fixed_intra_position=0:…"*
   ⇨ IDR-frames are at every framerate*N position plus I-frames (CRA) at scene cuts.

6. Parameter Priority Order

There is a specific priority order when passing the parameters to FFmpeg on the command-line. You should be aware of it, because this can have a crucial impact on the expected encoder output:

omx_param: This has the highest priority and overrides everything what was previously set with a cfg_file or via FFmpeg options.

cfg_file: It only overrides FFmpeg options.

FFmpeg options: They have the lowest priority.

Example:

file.ini (setting framerate to 24 fps):

```
[HEVC Settings]
time_scale=27000000
num_units_in_tick=1125000
```

ffmpeg_cmdline (Windows):

```
ffmpeg -r 27 -pix_fmt yuv420p -s 1920x1080 -i file1920x1080.yuv \
-c:v omx_enc_hevc -omx_coreomxil_core.dll \
-omx_name OMX.MainConcept.enchevc.video \
-omx_param “force_omx_param=1:cfg_file_path file.ini:[HEVC
Settings]:time_scale=20000000:num_units_in_tick=1000000" ffmpeg_res_filename.hevc
```

ffmpeg_cmdline (Linux):

```
ffmpeg -r 27 -pix_fmt yuv420p -s 1920x1080 -i file1920x1080.yuv \
-c:v omx_enc_hevc -omx_core libomxil_core.so \
-omx_name OMX.MainConcept.enchevc.video \
-omx_param “force_omx_param=1:cfg_file_path file.ini:[HEVC
Settings]:time_scale=20000000:num_units_in_tick=1000000" ffmpeg_res_filename.hevc
```

The resulting file will have 20 fps (as defined in omx_param) instead of 24 fps which was pre-defined in the cfg_file and defined in the FFmpeg settings as 27 fps.

7. Command Line Examples

NOTE:
The command-line examples below are Windows specific. For running the examples on Linux you need to slightly modify them. Instead of “-omx_core omxil_core.dll” you need to specify “-omx_core libomxil_core.so” on Linux.

Apple HLS:

NOTE:
The command-line examples for Apple HLS do not work out-of-the-box. Depending on your use case you need to specify the correct parameters for -force_key_frames 'expr:gte(t,n_forced*N)' *where* N *is the number of seconds.*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=416:234 -force_key_frames *'expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param “preset=hls1:acc_type=sw:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*” output_hls_l1_416x234.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=480:270 -force_key_frames *'expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param “preset=hls2:acc_type=sw:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*” output_hls_l2_480x270.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=640:360 -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls3:acc_type=sw:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l3_640x360.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=768:432 -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls4:acc_type=sw:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l4_768x432.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=960:540 -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls5:acc_type=iqsv:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l5_960x540.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls6:acc_type=sw:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l6_1280x720.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls7:acc_type=nvenc:[HEVC Settings]:acc_mode=full:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l7_1280x720.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:960 -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls8:acc_type=sw:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l8_1280x960.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls9:acc_type=nvenc:acc_mode=hybrid:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l9_1920x1080.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=2560:1440 -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls10:acc_type=nvenc:acc_mode=driven:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l10_2560x1440.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls11:acc_type=sw:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l11_3840x2160.mp4*

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -force_key_frames '*expr*:gte(t,n_forced*N) *-c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=hls12:acc_type=iqsv:[HEVC Settings]:fixed_intra_position=0*:vcsd_mode= 0*" output_hls_l12_3840x2160.mp4*

MPEG-DASH:

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=768:432 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash1:acc_type=sw" output_dash_l1_768x432.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=768:432 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param -preset=dash2:acc_type=sw" output_dash_l2_768x432.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash3:acc_type=sw" output_dash_l3_1280x720.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash4:acc_type=sw" output_dash_l4_1280x720.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash5:acc_type=iqsv" output_dash_l5_1280x720.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash6:acc_type=nvenc:acc_mode=full" output_dash_l6_1280x720.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash7:acc_type=sw" output_dash_l7_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash8:acc_type=sw" output_dash_l8_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash9:acc_type=sw" output_dash_l9_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash10:acc_type=nvenc:acc_mode=hybrid" output_dash_l10_3840x2160.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash11:acc_type=nvenc:acc_mode=driven" output_dash_l11_3840x2160.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=dash12:acc_type=iqsv" output_dash_l12_3840x2160.mp4

Misc:

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=main:acc_type=sw" output_main_1280x720.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=main10:acc_type:iqsv" output_main10_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=4k:acc_type=nvenc:acc_mode=hybrid" output_4k_3840x2160.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_hevc -omx_core omxil_core.dll -omx_name OMX.MainConcept.enchevc.video -omx_param "preset=4k10:acc_type=nvenc:acc_mode=full" output_4k10_1920x1080.mp4

8. HEVC/H.265 Encoder Parameters

The MainConcept Hybrid HEVC Encoder Plugin for FFmpeg comes with a couple of sample config (*.ini) files that cover the following encoder settings and parameters. You can modify them manually in a normal text editor and pass them to FFmpeg via -omx_param “cfg_file_path=<...>.

NOTE:
*It is also possible to create additional HEVC/H.265 Encoder config files using TotalCode Studio. In the application, select a desired HEVC/H.265 preset and then export the settings within the config file. Simply go to **File** > **Export** > **Target Settings** and specify a name for the *.ini file. The “<...>.video.ini file can be used in the FFmpeg plugin now.*

8.1 [HEVC Settings]

General Parameters:

chroma_format:

Chroma format of the encoded picture.

- 1: 4:2:0
- 2: 4:2:2

bit_depth_chroma:

Bit depth of the encoded chrominance samples. Please note that bit_depth_chroma and bit_depth_luma need to be set to the same value.

- 8: 8-bit
- 10: 10-bit

bit_depth_luma:

Bit depth of the encoded luminance samples. Please note that bit_depth_luma and bit_depth_chroma need to be set to the same value.

- 8: 8-bit
- 10: 10-bit

max_dec_pic_buffering:

Maximum number of pictures in decoded picture buffer. This is restricted by level ID. Value range is [1, 16].

min_src_pic_buffering:

Maximum number of pictures in source picture buffer. Buffering of source pictures is used for preliminary analysis. Increasing of input buffer size will increase memory requirements and latency of pictures encoding, but positively affects visual quality. To achieve the lowest latency of frames in encoder this value must be set to 0. Value range is [0, 500]. Default value is 115.

max_src_pic_buffering:

Minimum encoding frames per second. Specifies the minimal encoding frame rate for automatic settings control. Values range is [0, 120], 0 means control is switched off.

time_scale:

Timing info use together with num_units_in_tick (frame rate in fps = time_scale/num_units_in_tick)

time_scale is the number of time units that pass in one second. By default, time_scale is set to 27000000 (27 MHz clock). These two values are used for calculating the exact frame rate of an encoded stream, where *fps = time_scale/num_units_in_tick*.

persistent_parsets:

Makes SPS and PPS persistent for the whole encoder performance range. In this case, the headers contain all features signaled as “turned on” and do not depend on the encoder settings. Nevertheless, not all the features are used during encoding.
Value is in range [0, 1].

temporal_sub_layers:

Number of temporal sub-layers. Specifies number of temporal sub-layers. Value range is [1, 3].

Picture Preprocessing Options:

input_filtering:

Input pictures filtering. Specifies filtering mode of the incoming frames.

- 0: Disable input filtering.
- 1: Apply median filter by 3x3 kernel.
- 2: Apply mean filter by 3x3 kernel.

Picture Pre-analysis Options:

aq_mode:

Adaptive quantization mode.

- 0: Disable adaptive quantization.
- 1: Adaptive quantization according to an external QP map.
- 2: Temporal adaptive quantization
- 3: Adaptive quantization without additional input, RDO-driven.

GOP Options:

min_intra_period:

Minimum GOP length. Minimum distance between two consequent I pictures. This parameter is intended for use with enabled Scene change detection feature only. Value range is [1, hevc_v_settings::max_intra_period]. Default value is 1.

max_intra_period:

Maximum GOP length. Maximum distance between two consequent I pictures. This parameter is not responsible for IRAP (Intra Random Access Pictures) placement by itself. There is a dedicated parameter for IRAP control - hevc_v_settings::irap_period. Value range is [1, 360]. Default value is 200.

irap_period:

IRAP period in GOP units. Number of GOPs between two consequent IRAP pictures.

- 0: Only the first picture is IRAP.
- 1: Every GOP begins with IRAP.
- n: Every nth GOP begins with IRAP.

fixed_intra_position:

Fixed position of I-frames helps to achieve regular IRAP placement when Scene Change Detection is used. The encoder inserts I-frames at positions that is the multiple of hevc_v_settings::max_intra_period. Only these pictures are taken into account for IRAP placement, I-frames inserted by SCD are just ignored. Finally, the encoder places IRAPs at the positions of hevc_v_settings::max_intra_period x hevc_v_settings::irap_period.

If you set the FFmpeg's generic *force_key_frames* parameter, please also read chapter 5 Force Key Frame Settings carefully.

- 0: Irregular IRAP placement
- 1: IRAP placement at every (hevc_v_settings::max_intra_period x hevc_v_settings::irap_period)

vscd_mode:

Scene Change Detection (SCD) is intended to improve encoded content quality by inserting I picture at the beginning of new scenes. Enabling SCD has side effects, it affects normal I-picture/IRAP placement. Different length of scenes causes a different GOPs length. In this case GOP length is restricted by the range from hevc_v_settings::min_intra_period to hevc_v_settings::max_intra_period. In addition, more frequent I-pictures reduce the distance between two consequent IRAP pictures. To avoid influence of SCD on IRAP placement use hevc_v_settings::fixed_intra_position feature.

If you set the FFmpeg's generic *force_key_frames* parameter, please also read chapter 5 Force Key Frame Settings carefully.

- 0: Disable SCD
- 1: Enable SCD (default)

max_num_ref_pics_p:

Maximum number of reference pictures used on P-pictures from list0. Value range is [1,16].

max_num_b_pics:

Maximum number of B-pictures between consecutive I and P pictures. If this value is set to 0, there will be no B-pictures generated. Value range is [0, 7].

adaptive_num_b_pics:

Adaptive number of B-frames. Reduces distance between two consequent P-frames according to temporal complexity.

- 0: Disable
- 1: Enable

Performance Options:

num_threads:

Number of threads to be used during encoding. 0 means auto mode (default), 1 means single threaded encoding, more threads will use more CPUs/cores. Value range is [0, 256].

num_parallel_pics:

The number of pictures encoded simultaneously. 0 means auto mode (default), value in range from 1 to 8 manually set parallel encoding pictures number.

min_encoding_fps:

Specifies the minimal encoding frame rate for automatic settings control. Values range is [0, 120], 0 means control is switched off (default).

max_num_reencodings:

The maximum number of re-encoding attempts per picture to achieve rate control goals (e.g. bit rate match or HRD conformance). Value in range from 0 to 10, 0 means that re-encodings are disabled.

hw_acceleration:

Hardware acceleration. Choose hardware encoder type. Use hevc_v_settings::hw_acc_idx to specify device if several devices match hw_acceleration. Use hevc_v_settings::hw_acc_mode to specify what is done on hardware.

- 0: Pure software encoding mode.
- 1: Intel Quick Sync Video hardware acceleration.
- 2: NVIDIA NVENC hardware acceleration.

hw_acc_indices:

Device index for hardware acceleration. hw_acc_indices is used together with hevc_v_settings::hw_acceleration and helps to pick a particular device from the list of compatible hardware.

- 0: n/a
- 1: Ignored. Any available device is used.
- 2: Index of NVIDIA CUDA devices.

hw_acc_mode:

Hardware acceleration mode for supported NVIDIA boards. hw_acc_mode is used together with hevc_v_settings::hw_acceleration to choose what is delegated to hardware.

- 0: Pure hardware encoding.
- 1: Software analysis, hardware encoding.
- 2: Software analysis, hybrid (HW/SW) encoding.

Video Signal Description:

input_colorimetry.signal_range:

Specify signal range of input stream. Required if you need to change signal range for encoding:

- 0: Auto
- 1: Full (0 – 255)
- 2: Short (limited) (16 – 235)

output_colorimetry.signal_range:

Specify signal range of output stream if you need to change it. You need to set this option to Short (2) as soon as your input is Short (Limited): Default is Full (without auto detection of input):

- 0: Auto
- 1: Full (0 – 255) (default)
- 2: Short (limited) (16 – 235)

8.2 [HEVC Layer 0000]

Layer Parameters:

width:

Encoded pictures width. Values range is [64, 8192].

height:

Encoded pictures height. Values range is [64, 4352].

level:

Level ID. This parameter specifies the level to maintain conformance of the encoding layer according to the ITU-T H.265 recommendation.

- 30: Level 1
- 60: Level 2
- 63: Level 2.1
- 90: Level 3
- 93: Level 3.1
- 120: Level 4
- 123: Level 4.1
- 150: Level 5
- 153: Level 5.1
- 156: Level 5.2
- 180: Level 6

- 183: Level 6.1
- 186: Level 6.2
- 1000: Auto mode. The encoder detects the level according to the picture size, bit rate, frame rate, etc.

high_tier:

This parameter specifies the tier to maintain conformance of layer.

- 0: Main tier
- 1: High tier

wavefront_processing:

Enables/disables Wavefront Parallel Processing (WPP).

- 0: Disable WPP
- 1: Enable WPP

num_slices:

Number of slices per picture. The encoder does not support dependent slice segments, so each slice is represented by a single independent slice segment. The slice borders are always aligned to an integer number of CTU rows. The number of CTU rows in a slice is calculated by division of the total number of rows in a picture by num_slices. Usage of slices in encoding can improve CPU utilization. Values range is [1, height_of_picture_in_ctus].

Rate Control Settings:

bit_rate_mode:

Rate control mode.

Value	HDR on	HDR off
0	Constant bitrate (CBR)	Average bitrate
1	N/A	Constant quantization (CQT)
2	Variable bitrate (VBR)	Average bitrate
3	Average quantization (AQP)	Average quantization (AQP)

hrd_conformance:

HRD conformance. Maintain NAL-based hypothetical reference decoder model.

- 0: Do not provide HRD conformance.
- 1: Provide strict HRD conformance, reencode on underflows until hevc_v_layer::max_qp is reached.
- 2: Try to provide HRD conformance, reencode on underflows until hevc_v_layer::max_qp or hevc_v_settings::max_num_reencodings is reached.

bit_rate:

Target bitrate for rate control in bits per second.

hss_rate:

Maximum bitrate of layer for HEVC_VBR and HEVC_CBR modes in bits per second. For constant bitrate this value should match bit_rate.

cpb_size:

Coded picture buffer (CPB) size in bits of layer.

9. Customer Care

For feedback and assistance with using the MainConcept Hybrid HEVC Encoder Plugin for FFmpeg, please contact our Customer Care team at apps.support@mainconcept.com.

10. Copyright Notice

Edition: November 2020