AVC Broadcast 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 many 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 AVC Broadcast Encoder Plugin for FFmpeg is a convenient way to solve this problem because it seamlessly integrates into FFmpeg.

Features:

- Use MainConcept's industry leading AVC/H.264 broadcast software encoder natively in FFmpeg.
- AVC/H.264 encoding support for up to 4:2:2 10-bit (plus 4:2:0 8-bit) and up to level 6.2 (8K).
- Pre-configured encoding profiles for professional Sony and Panasonic camcorder content creation.
- Hardware accelerated AVC/H.264 encoding powered by Intel Quick Sync Video.
- Ready-to-use presets for DASH-264 and Apple HLS-AVC.
- Use built-in multiplexers like MP4 and MXF from FFmpeg directly.

2. Software Installation

To use MainConcept AVC Broadcast 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 AVC 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 AVC Broadcast Encoder Plugin for FFmpeg if you own a valid license (optional).

3. Installation on Windows

To run the MainConcept AVC Broadcast 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 AVC Broadcast 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_1.0.0.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 AVC Broadcast *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 the MainConcept Hybrid HEVC Encoder, MainConcept AVC Broadcast Encoder and MainConcept AVC Encoder.

3.1.2 MainConcept AVC Broadcast Encoder Plugin Full Version Installation on Windows

In this short chapter, we briefly describe how to install the full version of the MainConcept AVC Broadcast 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_avc_broadcast_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 AVC Broadcast 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 AVC Broadcast 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 AVC Broadcast Encoder Plugin for FFmpeg on Debian-based 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 AVC Broadcast Encoder Plugin for FFmpeg full version (optional)

3.2.1 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_1.0.0.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 AVC Broadcast Encoder Plugin Full Version Installation on Linux

To install the MainConcept AVC Broadcast 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_avc_broadcast_encoder_installer_gcc_linux64_x64_sfx-
installer4_b4003.tar.bz2

./mainconcept_avc_encoder_plugin_linux64_full_1.0.0.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 AVC Broadcast 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.

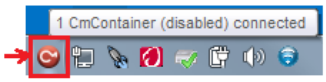
3. License Activation

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"
```

```
[thomas@centos8-1 linux-x64]$ ps ax | grep "[C]odeMeter"
 6536 ?        SLsl   0:00 /usr/sbin/CodeMeterLin -f
```

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:

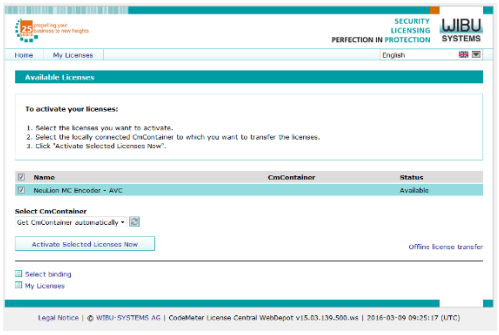
```
$ /usr/bin/CodeMeterCC –m
```

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.

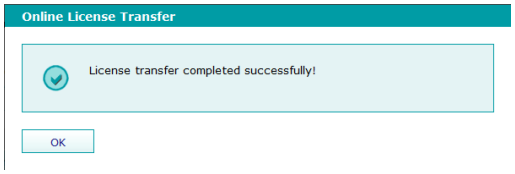
The activation link will open the CodeMeter License Central web page below. Here you can choose whether to use a USB hardware dongle (CMDongle) or a software license (CMActLicense) to activate the MainConcept AVC Broadcast Encoder Plugin for FFmpeg. In this release, please use the software activation, 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 AVC Broadcast Encoder Plugin for FFmpeg is selected. The **Select CMContainer** drop-down menu should show **Get**

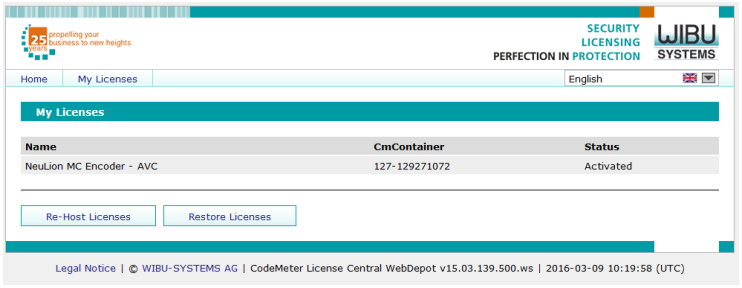
CmContainer automatically. To proceed with the activation, simply click the **Activate Selected Licenses Now** button.



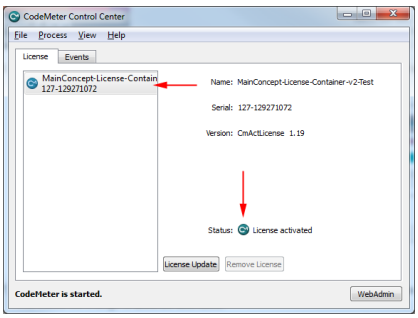
5. If successful, you will get the following notification. Press **OK** to proceed.



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



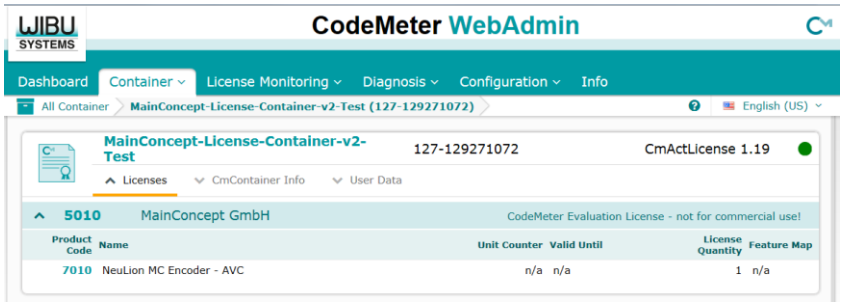
7. If you open **CodeMeter Control Center** you will also see the last activated product as confirmation. The status will change to **License activated**.



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.

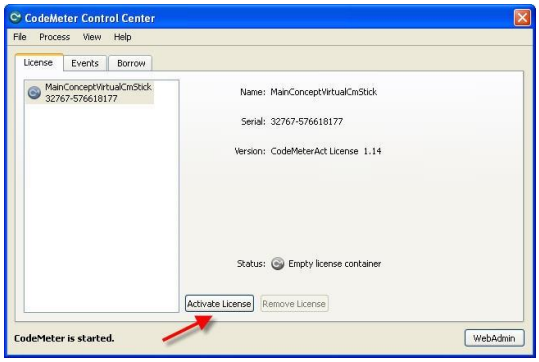
WebAdmin will open in a browser window. By choosing **Container > Licenses** you will get an overview about all installed licenses:



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.

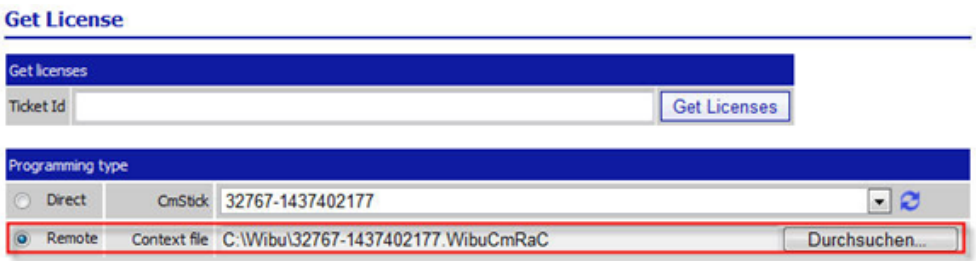
5. Transfer the stored file to a PC with internet connectivity (e.g. using an USB stick).

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.

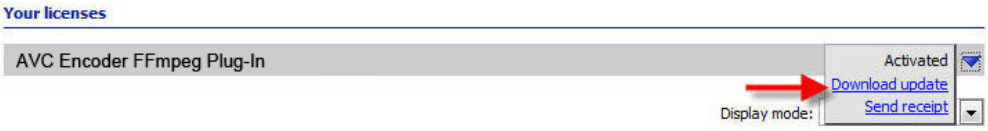
7. Tick ***Remote*** in the **Programming type** section.

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

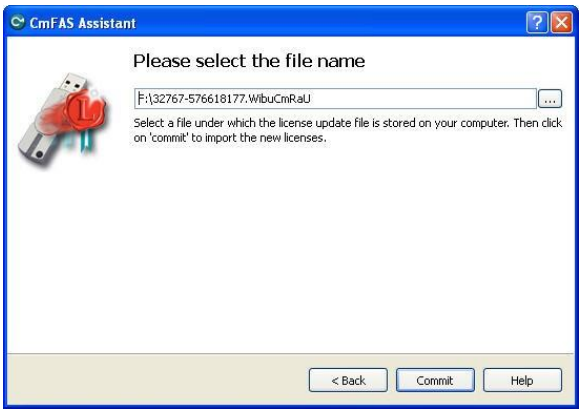


9. Confirm the activation process. After a few seconds, the status will change to ***Activated***.

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 AVC Broadcast Encoder Plugin for FFmpeg. The command line format should follow this structure:

On Windows:

*ffmpeg <ffmpeg-params> *
-c:v omx_enc_avc
*-omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video *
*-omx_param "<mc-general-params> *
*[AVC Settings] "<mc-codec-params>" *
<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_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param " force_omx_param=1:preset=H264_HIGH_10:perf_level=10:acc_type=sw:[AVC Settings]:time_scale=20000000:num_units_in_tick=1000000" "D:\1920x1080p_25p_YV12_ffmpeg.h264" -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_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "force_omx_param=0:cfg_file_path=avc_config.ini" "D:\1920x1080p_25p_YV12_ffmpeg.h264" -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_avc
*-omx_core libomxil_core.so -omx_name OMX.MainConcept.encavc.video *
*-omx_param "<mc-general-params> *
*[AVC Settings] "<mc-codec-params>" *
<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 15000k -c:v omx_enc_avc -omx_core libomxil_core.so -omx_name OMX.MainConcept.encavc.video -omx_param " force_omx_param=1:preset=H264_LONG_GOP_420_CLASS_G12:perf_level=10:acc_type=sw:[AVC Settings]:time_scale=20000000:num_units_in_tick=1000000" "D:\1920x1080p_25p_YV12_ffmpeg.h264" -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 10000k -c:v omx_enc_avc -omx_core libomxil_core.so -omx_name OMX.MainConcept.encavc.video -omx_param "force_omx_param=0:cfg_file_path=avc_config.ini" "D:\1920x1080p_25p_YV12_ffmpeg.h264" -y
```

All settings below are optional parameters:

omx_param Parameters (mc-general-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 15 for encoding. Also predefined values are allowed: • "fastest" = 1 • "faster" = 4 • "balanced" = 9 • "better_quality" = 12 • "best_quality" = 15
acc_type	Specified whether you want to use software ("sw") or Intel Quick Sync Video ("iqsv"). IQSV is a GPU encoding mode.
preset	Specifies the built-in encoder presets. Available options are "H264_BASELINE", "H264_CIF", "H264_MAIN", "H264_D1", "H264_HIGH_10", "H264_HIGH_422", "H264_DASH_L1"..."H264_DASH_L12", "H264_HLS_L1" … "H264_HLS_9", "H264_LONG_GOP_422_CLASS_G50", "H264_XAVC_4K_INTRA_CLASS_480_CBG", "H264_P2_4K_INTRA_422", "H264_INTRA_CLASS_200_RP2027", etc. 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 AVC/H.264 encoder config file containing all parameters for encoding. *-omx_param "force_omx_param=1:cfg_file_path=./avc_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 and H264_CQT mode. 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 -r 30 -vsync 0 -pix_fmt yuv420p -s 1920x1080 -i converted_file_raw.yuv -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "pass=1" ffmpeg_2pass.h264* *ffmpeg -r 30 -vsync 0 -pix_fmt yuv420p -s 1920x1080 -i converted_file_raw.yuv -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "pass=2" ffmpeg_2pass.h264*
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 -r 30 -vsync 0 -pix_fmt yuv420p -s 1920x1080 -i converted_file_raw.yuv -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "pass=1:passlogfile=mc_mpass_avc" ffmpeg_2pass.h264* *ffmpeg -r 30 -vsync 0 -pix_fmt yuv420p -s 1920x1080 -i converted_file_raw.yuv -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "pass=2:passlogfile=mc_mpass_avc" ffmpeg_2pass.h264*
[AVC Settings]	Specifies parameters from the AVC/H.264 encoder config on the command line which can be applied under "[AVC 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=H264_MAIN:acc_type=sw:[AVC Settings]:bit_rate_mode=0:bit_rate=1000000: ... : ..."* These arguments are matching the MainConcept AVC / H.264 encoder settings (see the chapter AVC/H.264 Broadcast Encoder Parameters under [AVC Settings] as a reference later in this user guide or an AVC encoder config file as an example).

5. Force Key Frame Settings

The MainConcept AVC Broadcast 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 AVC/H.264 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 internal *idr_interval* (set it to 0), which means infinite GOP. In case you want to get additional I-frames at scene cuts, the *vcsd_mode* must be set to 1.

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 "… :idr_interval=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 "… :idr_interval=0:vcsd_mode= 1:idr_frequency=0:…"*
   ⇨ IDR-frames are at every framerate*N position plus I-frames at scene cuts. The *idr_frequency* parameter disables IDR at scene cuts, so they are simple I-frames.

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):

```
[AVC 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_avc -omx_core omxil_core.dll \
-omx_name OMX.MainConcept.encavc.video \
-omx_param “force_omx_param=1:cfg_file_path=file.ini:[AVC
Settings]:time_scale=20000000:num_units_in_tick=1000000" ffmpeg_res_filename.h264
```

ffmpeg_cmdline (Linux):

```
ffmpeg -r 27 -pix_fmt yuv420p -s 1920x1080 -i file1920x1080.yuv \
-c:v omx_enc_avc -omx_core libomxil_core.so \
-omx_name OMX.MainConcept.encavc.video \
-omx_param “force_omx_param=1:cfg_file_path=file.ini:[AVC
Settings]:time_scale=20000000:num_units_in_tick=1000000" ffmpeg_res_filename.h264
```

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.

Sony:

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_4k:acc_type=sw" output_h264_xavc_4k_3840x2160.mxf

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_4k_422:acc_type=sw" output_h264_xavc_4k_422_3840x2160.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_HD_MP4:acc_type=sw" output_h264_xavc_hd_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_HD_MXF:acc_type=sw" output_h264_xavc_hd_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_HD_INTRA_VBR:acc_type=sw" output_h264_xavc_hd_intra_vbr_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_HD_INTRA_CLASS_50_CBG:acc_type=sw" output_h264_xavc_hd_intra_50_cbg_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_HD_INTRA_CLASS_100_CBG:acc_type=sw" output_h264_xavc_hd_intra_100_cbg_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_HD_INTRA_CLASS_200_CBG:acc_type=sw" output_h264_xavc_hd_intra_200_cbg_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_4K_INTRA_CLASS_100_CBG:acc_type=sw" output_h264_xavc_4k_intra_100_cbg_3840x2160.mxf

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_4K_INTRA_CLASS_300_CBG:acc_type=sw" output_h264_xavc_4k_intra_300_cbg_3840x2160.mxf

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_4K_INTRA_CLASS_480_CBG:acc_type=sw" output_h264_xavc_4k_intra_480_cbg_3840x2160.mxf

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_4K_INTRA_CLASS_100_VBR:acc_type=sw" output_h264_xavc_4k_intra_100_vbr_3840x2160.mxf

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_4K_INTRA_CLASS_300_VBR:acc_type=sw" output_h264_xavc_4k_intra_300_vbr_3840x2160.mxf

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_XAVC_4K_INTRA_CLASS_480_VBR:acc_type=sw" output_h264_xavc_4k_intra_480_vbr_3840x2160.mxf

Panasonic:

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_INTRA_CLASS_50:acc_type=sw" output_h264intra_class_50_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_INTRA_CLASS_100:acc_type=sw" output_h264intra_class_100_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_INTRA_CLASS_200:acc_type=sw" output_h264intra_class_200_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_P2_HD_INTRA_422:acc_type=sw" output_h264intra_hd_422_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=2048:1080 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_P2_2K_INTRA_422:acc_type=sw" output_h264intra_2k_422_2048x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_P2_4K_INTRA_422:acc_type=sw" output_h264intra_4k_422_3840x2160.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_LONG_GOP_420_CLASS_G6:acc_type=sw" output_h264longgop_420_g6_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_LONG_GOP_420_CLASS_G12:acc_type=sw" output_h264longgop_420_g12_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_LONG_GOP_422_CLASS_G25:acc_type=sw" output_h264longgop_422_g25_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_LONG_GOP_422_CLASS_G50:acc_type=sw" output_h264longgop_422_g50_1920x1080.mxf

Apple HLS:

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=416:234 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HLS_L1:acc_type=sw" output_hls_l1_416x234_145kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=480:270 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HLS_L2:acc_type=sw" output_hls_l2_480x270_365kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=640:360 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HLS_L3:acc_type=iqsv" output_hls_l3_640x360_730kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=768:432 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HLS_L4:acc_type=sw" output_hls_l4_768x432_1100kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=960:540 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HLS_L5:acc_type=iqsv" output_hls_l5_960x540_2000kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HLS_L6:acc_type=sw" output_hls_l6_1280x720_3000kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HLS_L7:acc_type=iqsv" output_hls_l7_1280x720_4500kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HLS_L8:acc_type=iqsv" output_hls_l8_1920x1080_6000kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HLS_L9:acc_type=sw" output_hls_l9_1920x1080_7800kb.mp4

MPEG-DASH:

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=768:432 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L1:acc_type=sw" output_dash_l1_768x432_400kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=768:432 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L2:acc_type=sw" output_dash_l2_768x432_600kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=768:432 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L3:acc_type=iqsv" output_dash_l3_768x432_800kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L4:acc_type=sw" output_dash_l4_1280x720_1200kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L5:acc_type=iqsv" output_dash_l5_1280x720_1750kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=1280:720 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L6:acc_type=sw" output_dash_l6_1280x720_2400kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L7:acc_type=sw" output_dash_l7_1920x1080_3500kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L8:acc_type=iqsv" output_dash_l8_1920x1080_5300kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L9:acc_type=sw" output_dash_l9_1920x1080_8400kb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L10:acc_type=sw" output_dash_l10_3840x2160_15mb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L11:acc_type=sw" output_dash_l11_3840x2160_20mb.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=3840:2160 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_DASH_L12:acc_type=iqsv" output_dash_l12_3840x2160_25mb.mp4

Misc:

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=768:432 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_Baseline:acc_type=sw" output_h264baseline_768x432.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=416:234 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_CIF:acc_type=sw" output_h264baseline_416x432.mp4

ffmpeg -i C:\source_1920x1080.m2ts -vf scale=720:576 -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_D1:acc_type=iqsv" output_h264d1_720x576.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_MAIN:acc_type=sw" output_h264main_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HIGH:acc_type=iqsv" output_h264high_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_OPTIMIZED_HIGH:acc_type=sw" output_h264high_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HIGH_10:acc_type=sw" output_h264high_10_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_HIGH_422:acc_type=sw" output_h264high_422_1920x1080.mp4

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_INTRA_HIGH_10:acc_type=sw" output_h264intra_high_10_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_INTRA_CLASS_50_RP2027:acc_type=sw" output_h264intra_class_50_rp2027_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_INTRA_CLASS_100_RP2027:acc_type=sw" output_h264intra_class_100_rp2027_1920x1080.mxf

ffmpeg -i C:\source_1920x1080.m2ts -c:v omx_enc_avc -omx_core omxil_core.dll -omx_name OMX.MainConcept.encavc.video -omx_param "preset=H264_INTRA_CLASS_200_RP2027:acc_type=sw" output_h264intra_class_200_rp2027_1920x1080.mxf

8. AVC/H.264 Broadcast Encoder Parameters

The MainConcept AVC Broadcast 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 AVC/H.264 Encoder config files using TotalCode Studio. In the application, select a desired AVC/H.264 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.*

7.1 [AVC Settings]

General Parameters:

profile_id:

AVC/H.264 Profile used to encode.

- 0: Baseline
- 1: Main
- 2: Extended
- 3: High
- 4: High 10
- 5: High 4:2:2

level_id:

AVC/H.264 Level used to maintain conformance of encoded file. Depending on the level_id, there are several limits, e.g. maximal frame size in macroblocks or maximal bitrate. Please refer to Table A-1 of AVC/H.264 Encoder standard (ISO/IEC 14496-10) for details.

- 10: Level 1
- 11: Level 1.1
- 12: Level 1.2
- 13: Level 1.3
- 16: Level 1b
- 20: Level 2
- 21: Level 2.1
- 22: Level 2.2
- 30: Level 3
- 31: Level 3.1
- 32: Level 3.2
- 40: Level 4
- 41: Level 4.1
- 42: Level 4.2
- 50: Level 5
- 51: Level 5.1
- 52: Level 5.2
- 60: Level 6
- 61: Level 6.1
- 62: Level 6.2
- 100: Auto mode. The encoder detects the level according to the picture size, bit rate, frame rate, etc.

idr_frequency:

Defines how often I-frames are IDRs. Please note that you can correctly re-encode only those parts of stream which begin from IDR and end with a frame just before IDR.

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

- 0: Only first picture will be IDR,
- 1: Every I-picture will be IDR,
- 2: Every second I-picture will be IDR, etc.

idr_interval:

Maximum distance between two I-frames. The value must be a multiple of the reordering_delay field.

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

- Valid values are 0 .. 300 (default value is 33).
- Set value 0 to enable infinite GOP.
- With value 1 only I-frames are generated.

reordering_delay:

Maximum distance between two P-frames. If this value is set to 1, there will be no B-frames generated. Available range is 1 .. 8.

interlace_mode:

Specifies whether video is encoded frame-based or field-based.

- 0: Progressive
- 1: Interlaced
- 2: MABFF

def_horizontal_size:

Width of the encoded video frame.

def_vertical_size:

Height of the encoded video frame.

frame_rate:

Framerate of the encoded video.

Motion Serach Settings:

num_reference_frames:

Number of reference pictures used. Available range is between 0 .. 16.

search_range:

Motion vector search range in units of luma frame samples. This field depends on level_id. Please refer to Table A-1 of AVC/H.264 Encoder standard (ISO/IEC 14496-10).

max_l0_active:

Maximum index of reference frames in List0. Default value is 0 (Auto).

max_l1_active:

Maximum index of reference frames in List1. Default value is 0 (Auto).

Quantization Parameters:

quant_pI:

Macroblock quantization parameter value for I slices to use in constant quantization bitrate mode. Available range is 0 .. 51.

quant_pP:

Macroblock quantization parameter value for P slices to use in constant quantization bitrate mode. Available range is 0 .. 51.

quant_pB:

Macroblock quantization parameter value for B slices to use in constant quantization bitrate mode. Available range is 0 .. 51.

Bitrate Control:

bit_rate_mode:

Bitrate control mode.

- 0: CBR.
- 1: CQT
- 2: VBR

bit_rate_buffer_size:

VBV size. This field specifies the size of the Hypothetical Reference Decoder (HRD) Coded Picture Buffer (CPB) or VBV (old MPEG name). This value should be adjusted to bit_rate (CBR), respectively max_bit_rate (VBR), to avoid DTS/PTS underflows during muxing. Depending on vbv_buffer_units, this value is taken as bits (vbv_buffer_units = H264_HRD_UNIT_BIT or H264_HRD_UNIT_BIT_90K) or bytes (vbv_buffer_units = H264_HRD_UNIT_BYTE_PERCENT). Due to encoder limitations value is restricted to maximum signed integer value for bits and (maximum signed integer value)/8 for bytes correspondingly.

bit_rate:

Target bitrate (bits per second) of encoded video sequence. Due to encoder limitations value is restricted to maximum signed integer value.

max_bit_rate:

Hypothetical Stream Scheduler (HSS) delivery rate or the rate at which VBV buffer is filled, not real maximum peak rate. Maximum peak rate is unrestricted by AVC spec. Restrictions are came from HSS rate and VBV buffer size, so reducing any of them will result in peak rate reduction. This value is only needed for H264_VBR mode and should be greater than bit_rate just to give rate controller a bit more space for work. Due to encoder limitations value is restricted to maximum signed integer value.

Prediction:

inter_search_shape:

Type of subblock search for macroblocks.

Coding Mode:

entropy_coding_mode:

Entropy coding mode.

- 0: CAVLC
- 1: CABAC

use_hadamard_transform:

Use Hadamard transformation to get higher compression ratio. Available values are 0, 1 and 2 (use Hadamard transformation only on reference frames).

VUI Parameters:

sar_width:

Sample aspect ratio: vertical sample size in arbitrary units. Cannot be used together with picture aspect ratio parameters as soon as these parameters are interchangeable. If usage of sample aspect ratio is preferable set both picture aspect ratio parameters (pic_ar_x and pic_ar_y) to -1 and vice versa.

sar_height:

Sample aspect ratio: horizontal sample size in arbitrary units. Cannot be used together with picture aspect ratio parameters as soon as these parameters are interchangeable. If usage of sample aspect ratio is preferable set both picture aspect ratio parameters (pic_ar_x and pic_ar_y) to -1 and vice versa.

video_format:

Type of video representation.

- -1: Auto
- 0: Component
- 1: PAL
- 2: NTSC
- 3: SECAM
- 4: MAC
- 5: Unspecified

time_scale:

Number of time units that pass in one second. Use together with num_units_in_tick (frame_rate = time_scale /num_units_in_tick) to calculate exact frame rate of a stream. By default, time_scale is set to 27000000 (27 MHz clock).

num_units_in_tick:

Number of time units of a clock operating at the frequency time_scale Hz that corresponds to one clock tick. Use together with time_scale (frame_rate = time_scale /num_units_in_tick) to calculate exact frame rate of a stream.

Advanced Settings:

vbv_buffer_fullness:

Initial VBV fullness. Please refer to the HRD Buffer Model in the ISO/IEC 14496-10 specs for details. For best quality we recommend to set the VBV Buffer to the level limit.

vbv_buffer_fullness_trg:

Target VBV fullness when decoding is done.

vbv_buffer_units:

Units of VBV-fullness and buffer size.

- 0: Original units (byte percent)
- 1: All parameters are in bits
- 2: All parameters are in 90 kHz clock units

cpb_removal_delay:

CPB removal delay for the first picture (needed for segment merging). Information about CPB removal delay is carried in Picture timing SEI message. cpb_removal_delay value should be equal to the distance to the previous IDR frame given in field units. This parameter may be used for segment re-encodings or seamless joining of several H.264 streams.

bit_rate_scale:

External setting of bit_rate_scale (avoids recalculation of bitrate)

cpb_size_scale:

External setting of cpb_size_scale (avoids recalculation of bitrate)

max_frame_size:

Maximum byte size for I, P, B-reference and B frames. This value has no effect for CQT.

min_frame_size:

Minimum byte size for I, P, B-reference and B frames.

hrd_maintain:

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

- 0: Disable HRD model.
- 1: Enable HRD model.

hrd_preview:

Quality feature (turned on by default). The encoder performs preliminary analysis on defined frame window. Frame window is set by the means of encoding_buffering. Frames are delivered with expected delay in this case.

slice_arg:

Number of slices per picture. Available range is from 1 slice per picture to the height of the encoded picture in macroblock units (or macroblock pairs for H264_MBAFF). On the systems with multiple logical or physical CPU using of more than 1 slice will accelerate encoding due to independent slice encoding.

b_slice_reference:

Enables usage B pictures as a reference pictures.

b_slice_pyramid:

Enables usage of pyramidal GOP structure (...B Br B...).

cb_offset:

Chroma quality offset (-X -> increase quality, +X -> decrease quality). For High profile and higher this is Cb chroma quantization offset. For the rest of the profiles it is used as chroma quantization offset (both Cb and Cr). Should be in the range [-12, 12]. This parameter used together with cr_offset can save (or cut) a bit of quality in chroma components, thus chroma quality can be either increased (cb_offset < 0) or decreased (cb_offset > 0). Suitable values are between -2 and 2 or potentially slightly more.

cr_offset:

Chroma quality offset (-X -> increase quality, +X -> decrease quality). For High profile and higher this is Cr chroma quantization offset. For the rest of the profiles it is unused and is not written in an stream. Should be in the range [-12, 12]. This parameter used together with cb_offset can save (or cut) a bit of quality in chroma components, thus chroma quality can be either increased (cr_offset < 0) or decreased (cr_offset > 0). Suitable values are between -2 and 2 or potentially slightly more.

me_subpel_mode:

Describes subpixel motion search depth.

- 0: Full Pel
- 1: Half Pel
- 2: Quarter Pel
- 3: Quarter Pel on reference picture

me_weighted_p_mode:

Enables explicit weighted prediction usage for P-frames. Useful for compensating flashes, fades etc.

me_weighted_b_mode:

Enables explicit weighted prediction usage for B-frames.

enable_fast_intra_decisions:

Enables fast intra coding decision metrics usage to speed up encoding process. Can slightly decrease quality but will give noticeable speed improvement.

- 0: No fast intra decisions.
- 1: Fast intra decisions are enabled.
- 2: Use slow decisions on reference frames and fast - on non-reference.

enable_fast_inter_decisions:

Enables fast inter coding decision metrics usage to speed up encoding process. Can slightly decrease quality but will give noticeable speed improvement.

- 0: No fast inter decisions.
- 1: Fast inter decisions are enabled.
- 2: Use slow decisions on reference frames and fast - on non-reference.

pic_ar_x:

Picture (display) aspect ratio width. Cannot be used together with sample aspect ratio parameters as soon as these parameters are interchangeable. If usage of picture aspect ratio is preferable set both sample aspect ratio parameters (sar_width and sar_height) to -1 and vice versa.

pic_ar_y:

Picture (display) aspect ratio height. Cannot be used together with sample aspect ratio parameters as soon as these parameters are interchangeable. If usage of picture aspect ratio is preferable set both sample aspect ratio parameters (sar_width and sar_height) to -1 and vice versa.

calc_quality:

Enables quality metrics calculation.

- 0: Disabled.
- 1: PSNR metric calculation for information purposes.
- 2: SSIM metric calculation for information purposes.
- 3: PSNR and SSIM metrics calculation for information purposes.

cpu_opt:

Defines CPU acceleration to be used during encoding.

- 0: Auto
- 1: Unknown
- 2: MMX
- 3: MMX Ext.
- 4: SSE
- 5: SSE2
- 6: SSE3
- 7: SSE4
- 8: AVX
- 9: AVX2

num_threads:

Number of threads to be used during encoding.

- 0: Auto mode.
- 1: Single threaded encoding.
- 2 ..256: More threads will use more CPUs/cores.

live_mode:

Reduces the number of reencodings to one; disables consequent reencoding of frames.

buffering:

Defines the encoder input queue length in frames. Specifies the size of a buffer for input pictures. Available range is 0 .. 900. Value of 0 sets minimal possible buffer for the current settings. Beware of large values, since source frames may need substantial amount of memory to store. This feature is useful for live capture as it may decrease frame dropping. Usage of this feature is required for hrd_preview.

min_quant:

Minimum quantization parameter to use. Value range is 0 to 51.

max_quant:

Maximum quantization parameter to use. Value range is 0 to 51.

max_slice_size:

Maximum slice size in bits. This specifies the maximum number of bits for a slice. Default value is 0 (no limit imposed).

encoding_buffering:

Number of frames to buffer encoding queue. Available range is 0 .. 900 (shall be less than or equal to buffering and greater). This feature is useful for graceful maintaining of target HRD buffer fullness (vbv_buffer_fullness_trg). Usage of this feature is required for hrd_preview.

low_delay:

Forces minimum encoding delay. Settings that affect encoding latency will be disabled. Default value is 0. If low_delay is set to 1 the encoder overwrites other settings to encode IPPP stream without frames buffering. low_delay can be used in two modes: synchronous (detach_thread = 0) and asynchronous (detach_thread = 1). In synchronous mode the encoder freezes calling thread in put frame and resumes the thread after a frame been encoded. In asynchronous mode the encoder works in a separate thread. All frames to be encoded are stored in a frames pool. The size of it defined by the setting buffering.

air_mode:

Defines adaptive intra refresh mode. Using this feature may be useful for improving error robustness of encoded video.

- 0: Off
- 1: Slow. 1 (SD) or 2 (HD) rows of intra macroblocks in every P frame.
- 2: Medium. 2 (SD) or 4 (HD) rows of intra macroblocks in every P frame.
- 3: Fast. 3 (SD) or 6 (HD) rows of intra macroblocks in every P frame.
- 4: Split. 1 row of intra macroblocks per number of rows.

detach_thread:

Runs encoder core in a new thread.

constrained_intra_pred:

Constrained intra prediction for improving error resilience.

In-loop Filter:

use_deblocking_filter:

Indicates whether to use deblocking for smoothing video frames.

- 0: Do not use deblocking filter.
- 1: Use deblocking filter.

deblocking_alphaC0_offset:

Specifies the offset used in accessing alpha deblocking filter table for filtering operations controlled by the macroblocks within a slice. The value shall be in the range of -6 to +6, inclusive.

deblocking_beta_offset:

Specifies the offset used in accessing beta deblocking filter table for filtering operations controlled by the macroblocks within a slice. The value shall be in the range of -6 to +6, inclusive.

adaptive_deblocking:

Enables the usage of deblocking filter with alternative offsets related to the quantizer or with standard ones. With alternative offsets, encoder applies less deblocking for lower quantizers in order to keep more details in the encoded video).

Type Parameters:

video_type:

AVC/H.264 encoder preset.

video_pulldown_flag:

Specifies the NTSC pulldown generated in the video stream. It should only be used if the input (original) video frame rate is 23.976 or 24 frames per second. When enabled it encodes 23.976fps as 29.97 (59.94) fps or 24fps as 30 (60) fps, using the pic_struct flag in the Picture timing SEI message. The frame_rate parameter should be specified as target framerate (i.e. 29.97, 30, 59.94 or 60). Video pulldown can only be only to a frame, i.e. interlace_mode can be equal only to H264 Progressive or H264 MBAFF.

- 0: No pulldown.
- 1: 2:3 pulldown (frame 1 is 2 fields, frame 2 is 3 fields, ...). 23.97 and 24 played as 29.97 and 30, respectively.
- 2: 3:2 pulldown (frame 1 is 3 fields, frame 2 is 2 fields, ...). 23.97 and 24 played as 29.97 and 30, respectively.
- 4: 2:3 pulldown (frame doubling, frame tripling, ...). 23.97/24 played as 59.94 and 60, respectively.
- 5: 3:2 pulldown (frame tripling, frame doubling, ...). 23.97/24 played as 59.94 and 60, respectively.
- 6: Auto. Adaptive mode.
- 7: 2:2 pulldown. Every frame is displayed twice.

- 8: 3:3 pulldown. Every frame is displayed three times.

File / Stream Parameters:

stream_type:

Defines the sort of NALU types being written into stream.

- 0: Stream Type I. NALUs + filler data
- 1: Stream Type I SEI. NALUs + filler data + SEI messages.
- 2: Stream Type II. All NALU type.
- 3: Stream Type II, except SEI messages.

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

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

chroma_format:

Chrominance sampling.

- 1: 4:0:0
- 2: 4:2:0
- 3: 4:2:2

vui_presentation:

Configures video usability information (VUI) parameters header appearing in SPS. Parameter can be either set to 0 and encoder configures VUI parameters automatically depending on the other settings.

- 0: Auto. Configures the video usability information appearing in the SPS header automatically (depending on the other encoder settings).
- 1: Customize. Enable combination of the following flags for configure the video usability information appearing in the SPS.
- 2: Enable aspect_ratio_info_present_flag.
- 4: Enable overscan_info_present_flag.
- 8: Enable video_signal_type_present_flag.
- 10: Enable colour_description_present_flag. The "video_signal_type_present_flag" should be enabled as well.
- 20: Enable chroma_loc_info_present_flag (not supported).

- 40: Enable timing_info_present_flag.
- 80: Enable nal_hrd_parameters_present_flag.
- 100: Enable vcl_hrd_parameters_present_flag.
- 200: Enable pic_struct_present_flag.
- 400: Enable bitstream_restriction_flag.

write_au_delimiters:

Write access unit delimiters. Access unit delimiters may be used to indicate the type of slices present in a primary coded picture and to simplify the detection of the boundary between access units (e.g. frames).

write_seq_end_code:

Write sequence end code. End of sequence code may be used to specify that the next subsequent access unit in the bitstream in decoding order shall be an IDR access unit, starting with new sequence parameter set.

- 0: Do not write sequence end code.
- 1: Write sequence end code at the end of a stream.
- 2: Write sequence end code before IDR picture and at the end of a stream.

write_timestamps:

Write picture timing information into Picture Timing SEI of the encoded stream.

- 0: Do not write timecode information.
- 1: Write timecodes using format with full_timestamp_flag = 0 and optional timecode fields (seconds_flag, minutes_flag, hour_flag).
- 2: Write timecodes using format with full_timestamp_flag = 1 and persistent timecode fields.

timestamp_offset:

Frame based offset (in number of frames) for timestamps (i.e DTS/PTS). Can be used for example for segment reencoding to guarantee continuous timings.

drop_frame_timecode:

Use NTSC drop frame timecode notation for 29.97 and 59.94 target frame rates.

write_single_sei_per_nalu:

Encapsulates each SEI message into its own NAL unit.

write_seq_par_set:

Controls writing of sequence parameter set (SPS) into a stream.

- 0: SPS once per IDR.
- 1: SPS once per I frame.

write_pic_par_set:
Controls writing of picture parameter set (PPS) into a stream.

- 0: PPS once per IDR.
- 1: PPS once per I-frame.
- 2: PPS once per picture.

log2_max_poc:
Specifies custom log2_max_pic_order_cnt_lsb_minus4 value. Available range is [4, 16], default value is 8.
log2_max_frame_num:
Specifies custom log2_max_frame_num_minus4 value. Available range is [4, 16], default value is 8.

pic_order_cnt_type:
Specifies custom pic_order_cnt_type value. Available values are 0 (default) and 2. Picture order count type 2 can only be used for streams with identical decoding and presentation orders, i.e. containing no B frames.

pic_order_present_flag:
Controls bottom_field_pic_order_in_frame_present_flag value in PPS (e.g. for SBTVD-T).

fixed_frame_rate:
Controls fixed_frame_rate_flag in VUI.

frame_based_timing:
Write frame rate in VUI (instead of field rate by default).

- 0: VUI's time_scale / num_units_in_tick equals to the field rate.
- 1: VUI's time_scale / num_units_in_tick equals to the frame rate.

Scene Detection:
vcsd_mode:
Visual content scene detection (VCSD). Feature is useful for getting better quality on scene changes.
If you set the FFmpeg's generic *force_key_frames* parameter, please also read chapter 5 Force Key Frame Settings carefully.

- 0: Disable scene detection.
- 1: Sets IDR frame at any scene change.

Advanced GOP Settings:
min_idr_interval:
Minimum number of frames between I-frames. This parameter forces usual distance between I-frames if SCD detects I-frame after less than min_idr_interval frames from previous one. The SCD forced I-frame can be discarded when distance to a next chapter point is less than min_idr_interval. Available values from 1 to idr_interval if idr_interval > 0, otherwise min_idr_interval can be from 1 (which means no limitation for

minimal GOP length) to maximal signed 32-bit integer (which means in practice that no I-frames will be inserted at scene changes).

adaptive_b_frames:

Enables adaptive B-frames placement. This parameter is used together with vcsd_mode and allows to use adaptive B-frames placement depending on sequence complexity.

- 0: Fixed placement.
- 1: Adaptive placement.

idr_frequency:

Defines how often I-frames are IDRs.

- 0: Only first picture will be IDR,
- 1: Every I-picture will be IDR,
- 2: Every second I-picture will be IDR, etc.

field_order:

Sets field order.

- 0: Top field first.
- 1: Bottom field first

fixed_i_position:

Sets I-frame at multiple of idr_interval positions.

isoleted_gops:

Allows to limit referencing for frames come after non-IDR I-frame to frames before non-IDR I-picture. It may be useful for avoiding certain stream seeking problems when idr_frequency is not equal to 1 (i.e. some I-frames are non-IDR I-frames).

- 0: Do not limit referencing.
- 1: Only allow the first group of B-frames to use frames before non-IDR I-frame as a reference picture.

Advanced ME Settings:

fast_multi_ref_me:

Enables fast decisions for multi-ref motion estimation.

- 0: Use greedy search mode (slower but can achieve better quality).
- 1: Use heuristic search mode (significantly faster but can loss some quality).
- 2: Use mode 0 on reference frames and mode 1 - on non-reference.

fast_sub_block_me:

Enables fast decisions for sub-block motion estimation.

- 0: Use greedy search mode (slower but can achieve better quality).
- 1: Use heuristic search mode (significantly faster but can loss some quality).
- 2: Use mode 0 on reference frames and mode 1 - on non-reference.

allow_out_of_pic_mvs:

Controls using pixels beyond the picture boundaries for motion compensation.

- 0: Do not use pixels outside the picture boundaries.
- 1: Enables out of picture motion vectors.

constrained_ref_list:

Use constrained reference picture list.

- 0: Do not apply any constraints,
- 1: Use only nearest I/P-frames for B-frame motion compensation, do not use reference B-frames for P-frame motion compensation.

Advanced Intra Settings:

enable_intra_big:

Allows to use 16x16 intra prediction modes in intra slices. Parameter can be set to 0, 1 or combination of flags. 0 indicates 16x16 intra prediction modes won't be used for I-slices. 1 means all 16x16 intra prediction modes will be used. Combination of following flags can be used to disable specific 16x16 intra prediction modes.

- 1: Enable 16x16 intra prediction modes usage.
- 2: Disable vertical 16x16 intra prediction mode.
- 4: Disable horizontal 16x16 intra prediction mode.
- 0: Disable plane 16x16 intra prediction mode.

enable_intra_8x8:

Allows to use 8x8 intra prediction modes in intra slices. Parameter can be set to 0, 1 or combination of flags. 0 indicates 8x8 intra prediction modes won't be used for I-slices. 1 means all 8x8 intra prediction modes will be used. Combination of following flags can be used to disable specific 8x8 intra prediction modes:

- 1: Enable 8x8 intra prediction modes usage.
- 2: Disable vertical 8x8 intra prediction mode.
- 4: Disable horizontal 8x8 intra prediction mode.
- 10: Disable diagonal down left 8x8 intra prediction mode.
- 20: Disable diagonal down right 8x8 intra prediction mode.
- 40: Disable vertical right 8x8 intra prediction mode.
- 80: Disable horizontal down 8x8 intra prediction mode.
- 100: Disable vertical left 8x8 intra prediction mode.

- 200: Disable horizontal up 8x8 intra prediction mode.

enable_intra_4x4:

Allows to use 4x4 intra prediction modes in intra slices. Parameter can be set to 0, 1 or combination of flags. 0 indicates 4x4 intra prediction modes won't be used for I-slices. 1 means all 4x4 intra prediction modes will be used. Combination of following flags can be used to disable specific 4x4 intra prediction modes:

- 1: Enable 4x4 intra prediction modes usage.
- 2: Disable vertical 4x4 intra prediction mode.
- 4: Disable horizontal 4x4 intra prediction mode.
- 10: Disable diagonal down left 4x4 intra prediction mode.
- 20: Disable diagonal down right 4x4 intra prediction mode.
- 40: Disable vertical right 4x4 intra prediction mode.
- 80: Disable horizontal down 4x4 intra prediction mode.
- 100: Disable vertical left 4x4 intra prediction mode.
- 200: Disable horizontal up 4x4 intra prediction mode.

enable_intra_pcm:

Allows to use PCM intra prediction mode in intra slices.

enable_inter_big:

Allows to use 16x16 intra prediction modes in inter slices. Parameter can be set to 0, 1 or combination of flags. 0 indicates 16x16 intra prediction modes won't be used for inter slices. 1 means all 16x16 intra prediction modes will be used. Combination of following flags can be used to disable specific 16x16 intra prediction modes:

- 1: Enable 16x16 intra prediction modes usage.
- 2: Disable vertical 16x16 intra prediction mode.
- 4: Disable horizontal 16x16 intra prediction mode.
- 10: Disable plane 16x16 intra prediction mode.

enable_inter_8x8:

Allows to use 8x8 inter prediction modes in inter slices. Parameter can be set to 0, 1 or combination of flags. 0 indicates 8x8 inter prediction modes won't be used for I-slices. 1 means all 8x8 inter prediction modes will be used. Combination of following flags can be used to disable specific 8x8 inter prediction modes:

- 1: Enable 8x8 inter prediction modes usage.
- 2: Disable vertical 8x8 inter prediction mode.
- 4: Disable horizontal 8x8 inter prediction mode.
- 10: Disable diagonal down left 8x8 inter prediction mode.
- 20: Disable diagonal down right 8x8 inter prediction mode.
- 40: Disable vertical right 8x8 inter prediction mode.
- 80: Disable horizontal down 8x8 inter prediction mode.

- 100: Disable vertical left 8x8 inter prediction mode.
- 200: Disable horizontal up 8x8 inter prediction mode.

enable_inter_4x4:

Allows to use 4x4 inter prediction modes in intra slices. Parameter can be set to 0, 1 or combination of flags. 0 indicates 4x4 inter prediction modes won't be used for I-slices. 1 means all 4x4 inter prediction modes will be used. Combination of following flags can be used to disable specific 4x4 inter prediction modes:

- 1: Enable 4x4 inter prediction modes usage.
- 2: Disable vertical 4x4 inter prediction mode.
- 4: Disable horizontal 4x4 inter prediction mode.
- 10: Disable diagonal down left 4x4 inter prediction mode.
- 20: Disable diagonal down right 4x4 inter prediction mode.
- 40: Disable vertical right 4x4 inter prediction mode.
- 80: Disable horizontal down 4x4 inter prediction mode.
- 100: Disable vertical left 4x4 inter prediction mode.
- 200: Disable horizontal up 4x4 inter prediction mode.

enable_inter_pcm:

Allows to use PCM inter prediction mode in inter slices.

Advanced Rate Distortion Optimization (RDO) Settings:

fast_rd_optimization:

Fast Rate Distortion (RD) methods.

- 0: Use greedy RD-optimization method without any simplifications. Very slow, but can achieve highest possible quality. It is recommended to set up this flag to 0 only when all fast decision and algorithms are disabled and all parameters which can increase quality are enabled.
- 1: Use simplified RD-optimization method. It is much faster but can lose some quality (usual very small amount). It is recommended to use this value in usual encoding scenarios.
- 2: Use mode 0 on reference frames and mode 1 - on non-reference ones.

quant_mode:

Specifies quantization optimization mode.

- 0: Use reference plain quantization method without any optimization. It is for debug/testing purposes only.
- 1: Use Mode1 quant optimization. In most cases produce better results than Mode0, but never worse.
- 2: Use Mode2 quant optimization. In most cases produce better results than Mode1 especially for Chroma components. It is recommended to increase cb_offset and cr_offset by 1 or even 2 so luminance component becomes even better while chroma will be the same as for Mode1.

grain_mode:

Granular noise optimization mode. Currently not used.

grain_opt_strength:

Scalable film grain optimization. This parameter is used to limit the use of intra prediction on videos with intense noise/film grain. The actual value defines penalty distortion for intra prediction modes in inter frames, which is introduced due to the fact that intra-coded macroblocks usually look a lot blockier than inter-coded ones. It is recommended to use higher strengths for very grainy or noisy material and lower strengths for less grainy material. It also makes sense to set the strength to 50 every time, although it may drop PSNR by up to 0.5 dB, but visual quality will be superior to strength 0. Computational complexity of this feature is usually negligible, so it is not considered in further descriptions of speed/quality presets. Available range is from 0 (off) to 100.

Psycho-Visual Enhancement:

adaptive_quant_strength:

Specifies the strengths for every mode of adaptive quantization. Positive values denote coarser quantization for macroblocks with brightness / contrast / complexity higher than the average value for the current picture and vice versa. Available range for every element is -100..100, default value is 0.

- 0: Adaptive quantization mode is driven by macroblocks' brightness.
- 1: Adaptive quantization mode is driven by macroblocks' contrast.
- 2: Adaptive quantization mode is driven by macroblocks' complexity.

denoise_strength_y:

Denoise strength for luma [0..100].

denoise_strength_c:

Denoise strength for chroma [0..100].

black_norm_level:

Specifies pre-encoding black normalization level. Makes any luma sample value less than or equal to black_norm_level will be set to 16. Available range is 0..255.

Scaling Lists:

seq_scaling_matrix_present_flag:

Specifies that scaling_list_present_flags are present. If scaling_matrix_present_flag is set to 0, it specifies that scaling_list_present_flags are not present and flat matrices will be used for all scaling lists (ITU-T Rec. H.264, 7.4.2.1.1).

- 1: Apply corresponding user defined scaling list will be used for i-th scaling list.
- 0: The corresponding default matrices will be used for i-th scaling list.

seq_scaling_list_present_flag:
Specifies that the syntax structure for scaling list i is present in the sequence parameter set (SPS).

Value	Block Size	MB Prediction Type	Component
0	4x4	Intra	Y
1	4x4	Intra	Cb
2	4x4	Intra	Cr
3	4x4	Inter	Y
4	4x4	Inter	Cb
5	4x4	Inter	Cr
6	8x8	Intra	Y
7	8x8	Inter	Y

9. Customer Care

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

10. Copyright Notice

Edition: November 2020