



# Integrating Intel® Media SDK with FFmpeg for mux/demuxing and audio encode/decode usages

---

## Introduction

The provided samples intend to illustrate how Intel® Media SDK can be used together with the popular FFmpeg suite of components to perform container muxing and demuxing (splitting). The samples also showcase integration of rudimentary FFmpeg audio decode and encode.

The sample projects are based on the Intel Media SDK 2012 R3 samples (<http://software.intel.com/en-us/articles/vcsource-tools-media-sdk/>) with only small modifications to original code such as adding new mux/demux command line directives.

Modified areas of the code with new integration code are tagged with `/// ===== ffmpeg ... integration =====`.

FFmpeg integration functionality resides in the *FFMPEGWriter* and *FFMPEGReader* classes, which are subclasses of the generic Intel Media SDK sample file reader/writer *CSmplBitstreamWriter* and *CSmplBitstreamReader* classes, respectively.

To enable simplistic implementation of FFmpeg audio processing functionality, just enable the *DECODE\_AUDIO* or *ENCODE\_AUDIO* define directive.

Since the provided samples are based on the Intel Media SDK decode/encode samples where video stream processing is in focus, the integration of FFmpeg container handling and audio processing may seem somewhat artificial. However, the samples should sufficiently illustrate the required Intel Media SDK and FFmpeg integration points making it quite straightforward to adapt to real-life applications, which would likely entail more sophisticated approaches such as threading, etc.

The provided samples illustrate the following use cases:

1. Demux “mp4” container file containing AVC(H.264) video stream and any FFmpeg supported audio stream. Decode the AVC(H.264) video stream and audio stream.  
(using “-split” option)
2. Encode to AVC(H.264) video stream and AAC audio stream. Mux streams into “mp4” container.  
(using “-mux” option)

3. Demux "mpeg" container file containing MPEG-2 video stream and any FFmpeg supported audio stream. Decode the MPEG-2 video stream and audio stream.  
(using "-split" option)
4. Encode to MPEG-2 video stream and MPEG audio stream. Mux streams into "mpeg" container.  
(using "-mux" option)
5. Encode to AVC(H.264) or MPEG-2 video stream and Ogg Vorbis audio stream. Mux streams into "mkv" container (Matroska).  
(using "-mkv" option)

The audio encode part of the sample assumes input of raw PCM data as follows:

- Using "mp4" or "mpeg" container: 16-bit signed integer samples, 2 channels
- Using "mkv" container: 32-bit float samples, 2 channels

You can generate raw PCM audio input file by demuxing a 2 channel @ 44100Hz audio stream using the provided decode sample. The encode sample can also easily be modified to support other audio input configurations. You can use such tools as "Audacity" to convert to/from different raw formats.

The current set of samples were tested and integrated using build "2012-08-27" of FFmpeg from:

<http://ffmpeg.zeranoe.com/builds/>

Please understand that this set of samples provides a snapshot of FFmpeg integration. The FFmpeg interfaces may change at any time, thus requiring modifications to the integration code.

## Project file structure

Folder	Content	Notes
sample_decode – ffmpeg	<ul style="list-style-type: none"><li>- sample_decode.sln</li><li>- include</li><li>- src</li></ul>	Contains decode/demux sample project. FFMPEGReader class located in pipeline_decode.h/cpp
sample_encode – ffmpeg	<ul style="list-style-type: none"><li>- sample_encode.sln</li><li>- include</li><li>- src</li></ul>	Contains encode/mux sample project. FFMPEGWriter class located in pipeline_encode.h/cpp
ffmpeg	<ul style="list-style-type: none"><li>- include</li><li>- lib_win32</li><li>- lib_x64</li></ul>	Contains FFmpeg component include files and pre-built binaries. See below “Requirements” section for details
sample_common	<ul style="list-style-type: none"><li>- include</li><li>- src</li></ul>	Contains common Intel® Media SDK sample code functionality (this is a copy of the Intel Media SDK 2012 R3 sample_common folder)

## Requirements

### Intel Media SDK

The sample projects depend on Intel Media SDK API include files and dispatcher library. To be able to build the provided sample code, Intel Media SDK 2012 R2 or later must be installed. The SDK can be found here: <http://software.intel.com/en-us/articles/vcsource-tools-media-sdk/>

For more details about the Intel Media SDK samples, and Media SDK specific requirements or limitations, please refer to documentation and manuals of the SDK package.

### FFmpeg

The provided sample projects do not include FFmpeg include and binary files. To be able to build the projects the required FFmpeg files for Windows\* must therefore be downloaded from <http://ffmpeg.zeranoe.com/builds/> (other Windows builds of FFmpeg also exists, but these have not been verified with this integration code)

Download the following packages:

1. “<build\_id>-<arch>-dev.7z” archive file (from “<arch>-bit Builds (Dev)” section on the webpage)
2. “<build\_id>-<arch>-shared.7z” archive file (from “<arch>-bit Builds (Shared)” section on the webpage)

From the above packages:

- Copy content of “include” folder in (1) to “ffmpeg/include” folder

- Copy \*.lib content of "lib" folder in (1) to "ffmpeg/lib\_<arch>" folder
- Copy \*.dll content of "bin" folder in (2) to "ffmpeg/lib\_<arch>" folder

Note: "<arch>" is either win32 or x64.

### **msinttypes**

The folder "ffmpeg/include/msinttypes-r26" contains parameter type bridge required to be able to build FFmpeg project in Microsoft Visual Studio.

Download the required include files from <http://code.google.com/p/msinttypes/>

Note: The solution/projects were created using Microsoft Visual Studio\* 2010, but there is nothing preventing the environment to be back-ported to older versions of Visual Studio, if needed.

## **How to build**

1. Open the solution (".sln") file in either "sample\_decode – ffmpeg" or "sample\_encode – ffmpeg" folder
2. Select desired build configuration: Debug/Release, Win32/x64
3. Build the solution

## How to execute workloads

Below are some example command line workloads.

1. Demux mp4 container and decode the AVC(H.264) video stream. If the container includes audio stream, it will be decoded into "audio.dat" (assuming sample has been built with "DECODE\_AUDIO")

```
sample_decode.exe h264 -i file.mp4 -hw -d3d -split -o video.yuv
```

2. Demux mpeg container and decode the MPEG-2 video stream. If the container includes audio stream, it will be decoded into "audio.dat" (assuming the sample was built with "DECODE\_AUDIO")

```
sample_decode.exe mpeg2 -i file.mpg -hw -d3d -split -o video.yuv
```

3. Encode raw YUV video data into AVC(H.264) video stream. Mux into mp4 container. If the sample was built with "ENCODE\_AUDIO" the audio will be encoded using AAC encoder (raw audio PCM data read from file "audio.dat").

```
sample_encode.exe h264 -i video.yuv -w 640 -h 480 -o out.mp4 -hw -d3d -mux -b 1000 -f 30
```

4. Encode raw YUV video data into MPEG-2 video stream. Mux into mpeg container. If the sample was built with "ENCODE\_AUDIO" the audio will be encoded using mpeg encoder (raw audio PCM data read from file "audio.dat").

```
sample_encode.exe mpeg2 -i video.yuv -w 640 -h 480 -o out.mpg -hw -d3d -mux -b 1000 -f 30
```

5. Encode raw YUV video data into the AVC(H.264) video stream. Mux into Matroska (mkv) container. If the sample was built with "ENCODE\_AUDIO" the audio will be encoded using Ogg Vorbis encoder (raw audio PCM data read from file "audio.dat").

```
sample_encode.exe h264 -i video.yuv -w 640 -h 480 -o out.mkv -hw -d3d -mkv -b 1000 -f 30
```

## Known issues

- MPEG2 muxing results in “buffer underflow” warning. However, the warning does not seem to impact the content or validity of the resulting mpeg container.

## References

- Intel Media SDK  
<http://software.intel.com/en-us/articles/vcsource-tools-media-sdk/>
- Intel Media SDK forum  
<http://software.intel.com/en-us/forums/intel-media-sdk/>
- FFmpeg  
<http://ffmpeg.org/>
- FFmpeg Windows builds  
<http://ffmpeg.zeranoe.com/builds/>
- FFmpeg tutorial  
<http://www.codediesel.com/tutorial/ffmpeg-a-beginners-guide-part-1/>

## Notices

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>  
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel and the Intel logo are trademarks of Intel Corporation in the US and/or other countries.

Copyright © 2012 Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.