Streaming Media Transcoding Application (SMTA)

User Guide

July 2019

Intel Confidential

Contents

Figures

Tables

Revision History

c	Revision	Description
July 2012	1.0	Initial release.
August 2013	2.0	Adapted for SMTA version 0.2
November 2013	3.0	Updates corresponding to SMTA version 0.2
December 2013	4.0	Updates corresponding to SMTA version 0.2.3
March 2014	5.0	Updates corresponding to SMTA version 0.3.0
September 2014	6.0	Updates corresponding to SMTA version 0.4.0
November 2015	7.0	Updates corresponding to SMTA version 0.5.0
April 2016	8.0	Updates MSS version to 16.5 Update FFMPEG version to 3.0.1 Add HEVC HW decoder and HW encoder Add De-noise feature
August 2016	9.0	Porting MSS to version 16.5 PV1 Remove Live555 library dependence including audio transcoding feature Add local file input support
September 2016	10.0	Add Windows building support
July 2019	11.0	Add FEI feature support(PreENC, ENCODE) for encoder Add dynamic VPP Add FEI feature support(ENC, PAK) for encoder Add MFE feature support for encoder Add HEVC FEI feature(PreENC, ENCODE) support for encoder Add Docker file to generate SMTA transcoding container.

1. Introduction

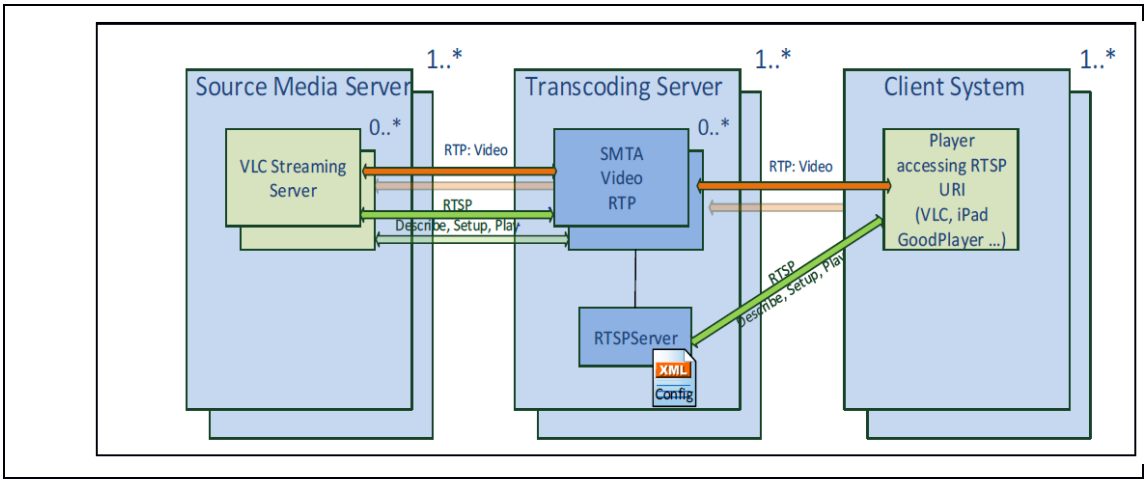
The purpose of the Streaming Media Transcoding Application (SMTA) is to demonstrate video transcoding capabilities of Intel® HD Graphics enabled processors, particularly in terms of stream density (number of streams per "system"). The SMTA takes an input AV stream from a Real Time Streaming Protocol (RTSP) source, splits out the video, decodes it, optionally applies some processing to the video frames (scaling, de-interlacing). Finally, it re-encodes the result to the specified encoding, and then transmits it to the designated host and port(s) using Real-time Transport Protocol (RTP).For encoding, SMTA support two mode: legacy mode and FEI mode. Legacy mode supports Decoder + [VPP] + Encoder pipeline. FEI mode supports Decoder + [VPP] + PREENC + Encoder, Decoder + [VPP] + PREENC + ENC + PAK, Decoder + [VPP] + ENC + PAK or Decoder + [VPP] + [PREENC/ENC/PAK/ENCODE] pipeline. [VPP]: VPP is optional.

Caution: This sample software is not fully featured, may not be fully functional, and may not be fully tested. The sample software may contain bugs or errors. Intel does not recommend use of this software in its current state for any production use. Notify Intel if you have plans to integrate any part of this software into your commercial products.

SMTA runs on Windows* 8 and Linux* (CentOS 7.2*/7.3*/7.4*), and uses the MSS Development Kit (Intel® Media Server Studio) to perform hardware-accelerated video decoding and encoding.

SMTA comes bundled with a simple RTSP server that makes it easy to use with standard players. Clients can simply connect to the RTSP server using an RTSP Uniform Resource Identifier (URI) (for example, rtsp://smta1:8554/SplashingWaves.ts). The RTSP server starts the SMTA process according to the parameters configured for the requested stream. The RTSP server runs on the transcoder system and is configured by means of an XML file, where all required parameters for each stream are specified. The spawned SMTA process then acts as an RTSP client to the source server and sets up that side of the connection to fetch the input stream(s). The system overview is depicted in Figure 1 below:

Figure 1. System Overview



1.1. Features

The SMTA supports the following features:

- Video transcoding
  - Input formats: H.264(support up to 4K resolution), H.265(support up to 4K resolution), MPEG2(support up to 1080P resolution)
  - Output format: H.264, H.265, MPEG2. only H.264/H.265 for FEI mode. All formats support up to 4K resolution
  - Multiple stream transcoding
  - 1 to N transcoding (N is statically configured)
- Video Post Processing between decode and encode
  - Scaling
  - Control frame rate
  - De-interlacing, Interlacing
  - De-noise
  - Control composition of several input surfaces in the one output
- Input stream:
  - RTSP + separate RTP elementary streams for HEVC video
- Output stream:
  - MPEG Transport Stream
  - Optional control by RTSP (simple for clients to initiate on-demand transcode)
- Configurability
  - Encode rate control (CQP, CBR, VBR) for legacy mode and CQP only for FEI mode
  - Configurable codec profile and level
  - Configurable Group of Pictures (GOP) structure
  - Look ahead

1.2. Terminology

Table 1. Terminology

Term	Description
AAC	Advanced Audio Coding
DLL	Windows* Dynamic Link Library
GPL	GNU* General Public License
Intel® IPP	Intel® Integrated Performance Primitives
MSS	Intel® Media Server Studio
LA BRC	Look Ahead Bitrate Control algorithm
LGPL	GNU* Lesser General Public License
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SDP	Session Description Protocol
SMTA	Streaming Media Transcoding Application
UDP	User Datagram Protocol
UMC	Unified Media Classes
URI	Uniform Resource Identifier
VPP	Video Processing
HEVC	High Efficiency Video Coding
H.265	Same as HEVC
FEI	Flexible Encode Infrastructure extension
ENC	ENCode - first stage of encoding process that include motion estimation and MB mode decision
PAK	PAcK – last stag of encoding process that include bit packing
PreENC	Pre Encoding
ENCODE	Actual encoding
MFE	Multi-frame encode

1.3. Reference Documents

Table 2. Reference Documents

Document	Document No./Location
Streaming Media Transcoding Application (SMTA) Release Notes	551939

2. Network Setup

2.1. Physical Network

Intel recommends connecting the various systems, source server, and transcoder using a Gigabit Ethernet-based network, since a 1000 Mbit/s network may not run the number of parallel transcode sessions from HD sources that an Intel® HD Graphics-enabled processor system supports.

Individual client devices, especially wireless clients, can connect at lower network speeds.

2.2. IP Stack Configuration

The following recommendations apply to source servers and transcoder systems running the Linux* operating system.

Depending on the specific kernel version you are using, you may need to alter the IP stack settings for User Datagram Protocol (UDP) send buffer limits. Default values are often too low to allow the streaming of several HD videos in parallel, which results not only in UDP packets being dropped at sending, but also eventually results in defects visible at the final client.

The following command can be used to verify whether or not send errors are occurring:

```
netstat –s | grep SndbufErrors
```

If the number of `SndbufErrors` is not null, you probably need to increase the amount of memory allocated for send buffers. This can be modified using the `sysctl` command. The exact parameters can change, depending on the kernel version. Use the `sysctl –a` command to list `systctl` parameters. The following are examples of values to check:

```
net.ipv4.udp_mem and net.core.wmem_max, net.core.wmem_default
```

You can then add the required configuration to the `/etc/sysctl.conf` file. On the Ubuntu* installation used for testing, the following parameters were adjusted:

```
net.core.wmem_max = 16777216
net.core.wmem_default = 16777216
net.core.rmem_max = 16777216
net.ipv4.udp_mem = 4096 87380 16777216

net.ipv4.tcp_rmem='4096 87380 16777216'

net.ipv4.tcp_wmem='4096 65536 16777216'

net.ipv4.tcp_mem='16777216 16777216 16777216'

net.ipv4.route.flush=1
```

To take these changes into account, restart your system, or run `sysctl –p` while running under the desired transcoding load. Adjust the values until you see no more `SndbufErrors` when running the above mentioned `netstat` command.

You can check the `Documentation/networking/ip-sysctl.txt` document for your specific Linux* kernel.

3. Transcoder System Installation

SMTA currently supports the Windows* 8 (64-bit) and CentOS 7.2*/7.3*/7.4*operating systems.

3.1. System Prerequisites

3.1.1. Linux*

When running several transcode sessions on a single system, the amount of UDP traffic generated may exceed some IP stack limits. IP stack parameters must therefore be altered in order to avoid packets being discarded. See Section 2.2 , for this procedure.

3.1.2. Windows*

Make sure the Windows* firewall on the transcoder system is disabled for the local network: Open the **Control Panel**, select **System and Security**, select **Windows Firewall**, and then **Turn Windows Firewall on or off**. For the local private network to which the system is connected, select **Turn off Windows Firewall**. Check Windows updates and install them all.

3.2. Software Prerequisites

The software packages listed in this section must be installed on the transcoder system in order to run SMTA. These packages are not included in the SMTA software package, but can be downloaded from the hyperlinks provided below.

3.2.1. Intel® Media Server Studio

For Linux*

https://software.intel.com/en-us/intel-media-server-studio, v16.5~v16.9. It can be downloaded on VIP (contact your Intel representative to get access).

After installation of MSS, it needs to manually re-build libdispatch_shared.a according to section "Build Instructions" in /opt/intel/mediasdk/opensource/readme-dispatcher-linux.pdf and put it at /opt/intel/mediasdk/lib/lin_x64/ with root privilege.

- Additional platform-specific software requirements (X indicates a supported combination – only the hardware/OS/MSS combinations shown in Table 3 are supported).

Extension libraries – gcc-c++ and nasm: CentOS yum install gcc++ nasm

For Windows

After installation of MSS, you should add "INTELMEDIASDKROOT" to the "user variables" with the value: "C:\Program Files\Intel\Intel(R) Media Server Studio

2016\Software Development Kit", in "Control Panel" -> "System" -> "Advanced system settings" -> "Environment Variables" -> "User variables"

3.2.2. Intel® Graphics Driver

When using hardware-accelerated transcoding, the MSS relies on functionality that comes with an Intel® Graphics Driver. Intel therefore recommends you update your Intel® Graphics Driver to the latest available release.

The required driver update comes with the MSS package.

3.2.3. Microsoft Visual Studio 2015

For building Windows version, needs to install Microsoft Visual Studio 2015 Professional with full license. Choose installation options Custom->programming Languages->C++.

3.3. SMTA Installation

Unzip the complete SMTA package to the directory of your choice. It contains one subdirectory for each operating system for which pre-built binaries are provided (`lin_x64`).

3.3.1. FFmpeg* Libraries

The SMTA binaries were built against FFmpeg* 4.0 dynamic libraries. Those must be present on the transcoder system. The SMTA package includes pre-built FFmpeg* libraries suitable for SMTA and licensed under the GNU* Lesser General Public License (LGPL) v2.1.

In case you would like to rebuild them from sources, refer to Section 8 .

Note that FFmpeg* source tree contains components under various licenses, including some under the GNU* General Public License (GPL). A build that only includes the core FFmpeg* components, like that included in the SMTA package or resulting from the instructions in Section 8 , would be licensed under LGPLv2.1. Other builds, like those that can be downloaded from http://www.zeranoe.com for example, may be governed by other licenses. Make sure the licensing terms fit your intended usage.

For Windows*

The following dynamic link libraries (DLLs) must be on the path or in the current folder when SMTA executes: `avformat-57.dll, avcodec-57.dll, avutil-55.dll`.

For Linux*

The `libavformat.so.57`, `libavcodec.so.57, libswresample.so.2` and `libavutil.so.55` files must be accessible to the dynamic loader. If using the libraries included in the SMTA package, you can add "." to the `LD_LIBRARY_PATH` environment variable:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.
```

If you build the libraries from sources, they will by default be installed under `/usr/local/lib`. Therefore, this path should be present in the `/etc/ld.so.conf` file or in the `LD_LIBRARY_PATH` environment variable.

4. Source Server Installation

Intel has tested SMTA with source servers running the Linux* operating system. On the system that will provide the source streams, Intel recommends using VideoLAN* VLC media player* (VLC*) as the source stream server for H.264 and MPEG2 video format.

The default dimensioning of the Linux* IP stack may not support the amount of outgoing UDP traffic that the source server will generate. Refer to Section 2.2 to verify and adapt it.

4.1. VideoLAN* VLC Media Player* as On-demand RTSP Server

VLC* is available from http://www.videolan.org/vlc/, or from repositories of most Linux* distributions.

In order to run VLC* as an on-demand RTSP server, it must be executed with its telnet interface enabled so the streams to be served can be configured over this interface.

To run the VLC* on-demand server, execute "cvlc -L -vvv --color -I telnet --telnet-password videolan --rtsp-host 0.0.0.0 --rtsp-port 5554". This will start the VLC* RTSP server listening on port 5554, but at this stage it is not configured to serve any stream. Then, from another console, send the corresponding telnet commands to the running VLC* instance to configure the streams. This steps will configure to broadcast an AVC ES stream:

1. telnet localhost 4212
2. input password: videolan to login VLC terminal
3. new test.mp4 vod
4. setup test.mp4 input path_to_file/input.mp4
5. setup test.mp4 enabled

For more information, please visit website .

4.2. Video Files

The package does not contain any video files. You must use your own video files and adapt the XML configuration file accordingly, if applicable.

5. SMTA container configuration

In order to quickly configure the SMTA running environment, container configuration files are added. Generate transcoding image, source server image and client image by Docker file. Then trigger SMTA transcoding pipeline by running ffplay(VLC) in client.

5.1. Docker installation

Old version of Docker is named to docker or docker-engine. If the old version is installed. Please uninstall the Docker.

Install docker:

On Ubuntu:

$apt-get install docker-ce

On centos:

$yum install docker.x86_64 (or docker.io)

Start Docker:

$systemctl enable docker

$systemctl start docker

$systemctl status docker

5.2. Transcoding container

Install intel gmmlib, libva, media driver, Intel Media SDK , SMTA transcoding and dependence librarys in SMTA transcoding Dockerfile. For more information , please refer to the Dockerfile.

Generate SMTA transcoding image:

$mkdir smta_image

$cp opensouce.tar.gz SMTA.A.0.9.0-1.tar.gz Dockerfile ./smta_image

$cd smta_image

$ sudo docker build -t smta:0.2 . $(env | grep -E '_(proxy|REPO|VER)=' | sed 's/^/--build-arg /') --network=host

```
$    docker run --it  smta:0.1      \
            -v /var/tmp:/var/tmp        \
            -v /dev:/dev          \
```

```
-v /home/media:/home/media

--privileged=true          \

--network=host     \

/bin/bash
```

In transcoding server, you can use a simple RTSP server to stream RTP package of smta output.

Note:

a. The version of Installed packages are specified in Dockerfile. Please make appropriate corrections when you use it according to you needed.
b. SMTA.A.0.9.3-2-pv5.tar.gz is used in Dockerfile. If you need the difference SMTA source code, please correct the Dockerfile.

5.3. Client container

Client can be in or out of the container.

If you need to have a client container. Please use the following commands.

```
$mkdir client_image

$cp Dockerfile ./client_image

$cd client_image

$docker build –t client:01 .

$docker run –it      client:0.1\

            --device=/dev/dri:/dev/dri             \

            -v /var/tmp:/var/tmp          \

            -v /dev:/dev          \

             -v /home/media:/home/media

            --privileged=true          \

            --network=host     \

            /bin/bash
```

$ffplay rtsp://ip:port/output-stream-name #if you have a RTSP server in transcoding server.

Or

$ffplay rtp://ip:port #you can directly receive RTP package from transcoding server.

Client dockfile:

```
FROM centos:7.4.1708

WORKDIR /home

RUN  yum install -y epel-release rpm && \

    yum repolist && \

    rpm --import http://li.nux.ro/download/nux/RPM-GPG-KEY-nux.ro
&& \

    rpm -Uvh http://li.nux.ro/download/nux/dextop/el7/x86_64/nux-
dextop-release-0-1.el7.nux.noarch.rpm&& \

    yum repolist && \

    yum install -y ffmpeg
```

5.4. Source server container

#Docker run –name=source_container --privileged=true --network=host centos:7.4.1708 /bin/bash

cvlc -L -vvv --color -I telnet --telnet-password videolan --rtsp-host 0.0.0.0 --rtsp-port 5554

6. Transcoder Command Line

The chart focus on how to run SMTA manually. SMTA supports two different means of specifying transcoding parameters. All parameters can be specified directly on the command line, in which case there can only be a single 1-to-1 session configured. The following syntax must be used in this case:

```
StreamingMediaTranscoder –params
-i::<codec>::rtsp <rtsp_uri>
-o::h264|mpeg2::rtp[::ts] <output_ip> <output_port>
-w <width> -h <height> -b <bitrate> -f <frame_rate>
[additional options]
```

It also supports specifying parameters in a parameter file by using the syntax below:

```
StreamingMediaTranscoder –par <par_file>
```

The provided parameter file must contain the parameters for one or more sessions. For each session, use the same syntax as on the command line where the –params approach is used. In this case, the sessions can be 1-to-1 or can be combined to set up a 1-to-N configuration, where one input stream is transcoded to multiple output streams with different bitrate, resolution, frame rate, or other target parameters.

When using the RTSP server, the `<cmd>` node of each stream corresponds to an invocation of the Streaming Media Transcoder and must use this syntax.

Table 3 below describes the supported command line parameters.

Table 3. Supported Transcoder Command Line Parameters

Parameter	Description
`-name <session_name>`	Session name for logging and console title.

`-i::<codec>::rtsp <rtsp_uri>`
OR
`-i::<codec>`
OR
`–i::source`
	Selects codec for the video input bitstream and specifies the RTSP URI of the source stream.
	When `-i::<codec>` is specified, selects codec for the local video and specifies local video file full path.
	Source stream can exist in separate video elementary streams.
	Supported codecs: h264, h265, mpeg2
	Refer to Section 6.1 for the meaning and usage of the `–i::source` option.

Parameter	Description

`-o::<codec>::rtp[::ts] <output_ip> <output_port>`
OR
`-o::null`
OR
`-o::sink`
	Specifies the output video codec, the format and the destination IP address and port for the RTP packets carrying the transcoded stream.
	Supported codecs: h264, h265, mpeg2
	When using `-o::h265::rtp`, an HEVC elementary video stream is sent to the specified destination.
	When using `-o::h264|mpeg2::rtp::ts`, a MPEG2 Transport Stream is used
	When `-o::null` is specified, the transcoder only performs the decode operation and discards the resulting frames. No output is produced.
	Refer to Section 6.1 for the meaning and usage of the `-o::sink` option.

`-f <frame_rate>`	Output video frame rate (frames per second).
`-u speed|quality|balanced|<value>`	Target usage. Can be specified as "speed," "balanced," or "quality," or as an integer value passed directly to MSS.
`-rc <rateControl>`	Specifies the encoder rate control method to be used: CBR, VBR, CQP, AVBR. Default: CBR For FEI mode, only support CQP
`-la`	Use look ahead bitrate control algorithm (LA BRC) for H.264 encoder. Supported on the 4th Generation Intel® Core™ processors. Note: It should be used with –lad option.
`-lad <depth>`	Depth parameter for the LA BRC, the number of frames to be analyzed before encoding. In the range [10,100].
`-b <bitrate>`	Output video bit rate (Kbits per second). Please set the value more than 1500.
`-qp <QP>`	Defines the QP value to be used when constant QP (CQP) rate control method is selected.
`-cprofile <profile>`	Codec profile to use (default: let encoder decide)

Parameter	Description
`-clevel <level>`	Codec level to be used by the encoder. By default, the encoder automatically selects a suitable level. Allowed values for AVC codec: 1, 1b, 11, 12, 13, 2, 21, 22, 3, 31, 32, 4, 41, 42, 5, 51, 52. Allowed values for MPEG2 codec: Low, Main, High, High1440. Allowed values for HEVC codec: 1, 2, 21, 3, 31, 4, 41, 42, 5, 51, 52, 6, 61, 62.
`-gopPicSize <size>`	Number of pictures within the GOP.
`-gopRefDist <dist>`	Distance between I- or P- key frames.
`-gopOptFlag CLOSED|STRICT`	Allows specifying additional GOP flags. Repeat `gopOptFlag` option for each desired flag.
`-idrInterval <interval>`	IDR frame interval in terms of I-Frames.
`-w <destinationPictureWidth>`	When different from the source width, VPP resizing is in effect.
`-h <destinationPictureHeight>`	When different from the source height, VPP resizing is in effect.
`-deinterlace`	Configures VPP to perform de-interlacing of the input video stream. For h264 interlace encode, ignore this parameter.
`-denoise <level>`	Enable denoise algorithm, Level is optional. Value range [0, 100]
`[-tff|-bff]`	Output stream is interlaced, top field first for tff, bottom field first for bff. If not specified output stream is the same as input.
`-loop [loop number]`	By default, when reaching the end of the input stream, the transcoder will close the output stream and terminate the transcoding session. Use this option to have the transcoder resume reading the input from the start while keeping the output stream active.

Parameter	Description
-log <log_flag>[:<log_flag>[:...]]	Turns on logging of specific functional areas according to specified log flags. Allowed flags can be found in the log_flags.h source file under MediaTranscoder/Common/include. Additionally, the special log flag "ALL" may be used to turn full logging at once. This is only recommended for detailed troubleshooting.
-mfe_frames <N>	Maximum number of frames to be combined in multi-frame encode pipeline 0 - Default for platform will be use Support MFE by h264 legacy and FEI(Encode) encoder
-mfe_mode <N>	Multi-frame encode operation mode - should be the same for all sessions 0, MFE operates as DEFAULT mode, decided by SDK if MFE enabled 1, MFE is disabled 2, MFE operates as AUTO mode 3, MFE operates as MANUAL mode Support MFE by h264 legacy and FEI(Encode) encoder
-mfe_timeout <N>	Multi-frame encode timeout in milliseconds - set per sessions control Support MFE by h264 legacy and FEI(Encode) encoder
-inputbyFPS	Input local data to decoder with FPS speed.
-ppyr	Enables P-pyramid. (disable - by default)
-gpb <on, off>	Make HEVC encoder use regular P-frames.(on - by default)
The following options is for the FEI encoder:	
-preenc <ds_strength>	Use extended FEI interface PREENC (RC is forced to constant QP). If ds_strength parameter is missed or less than 2, PREENC is used on the full resolution, otherwise PREENC is used on downscaled (by VPP resize in ds_strength times) surfaces. For FEI mode, this parameter must be set.

Parameter	Description
-encode	Use extended FEI interface FEI ENCODE (RC is forced to constant QP). For FEI mode, this parameter must be set.
-search_path <path_method>	Defines shape of search path. (0 -full, 1-diamond for AVC FEI) (1 - diamond, 2 - full, 0 - default (full) for HEVC FEI)
-sub_mb_part_mask <mask_hex>	Specifies which partitions should be excluded from search (default is 0x00 - enable all).
-sub_pel_mode <mode_hex>	Specifies sub pixel precision for motion estimation 0x00-0x01-0x03 integer-half-quarter (Default is 0x03).
-intra_part_mask <mask_hex>	Specifies what blocks and sub-blocks partitions are enabled for intra prediction (default is 0).
-adaptive_search	Enables adaptive search for motion estimation.
-search_window <window_size>	Specifies one of the predefined search path and windows size. In range [1, 8] (5 is default). If non-zero value specified: -ref_window_w / _h, -len_sp are ignored.
-ref_window_w <width>	Width of search region (should be multiple of 4), maximum allowed search window w*h is 2048 for one direction and 1024 for bidirectional searching.
-ref_window_h <height>	Height of search region (should be multiple of 4), maximum allowed is 32.
-len_sp <length>	Defines number of search units in search path. In range [1, 63] (default is 57).
-repartition_check	Enables additional sub pixel and bidirectional refinements (ENC, ENCODE)
-multi_pred_l0	MVs from neighbor MBs will be used as predictors for L0 prediction list (ENC, ENCODE)
-multi_pred_l1	MVs from neighbor MBs will be used as predictors for L1 prediction list (ENC, ENCODE)
-num_active_P <numRefs>	Number of maximum allowed references for P frames (up to 4(default)); for PAK only limit is 16

Parameter	Description
-num_active_BL0 <numRefs>	Number of maximum allowed backward references for B frames (up to 4(default)); for PAK only limit is 16
-num_active_BL1 <numRefs>	Number of maximum allowed forward references for B frames (up to 2(default) for interlaced, 1(default) for progressive); for PAK only limit is 16
-dblk_idc <value>	Value of DisableDeblockingIdc (default is 0), in range [0,2]
-dblk_alpha <value>	Value of SliceAlphaC0OffsetDiv2 (default is 0), in range [-6,6]
-dblk_beta <value>	Value of SliceBetaOffsetDiv2 (default is 0), in range [-6,6]
-chroma_qpi_offset <first_offset>	First offset used for chroma qp in range [-12, 12] (used in PPS)
-s_chroma_qpi_offset <second_offset>	Second offset used for chroma qp in range [-12, 12] (used in PPS)
-transform_8x8_mode_flag	Enables 8x8 transform, by default only 4x4 is used (used in PPS)
-adjust_distortion	If enabled adds a cost adjustment to distortion, default is RAW distortion (ENC, ENCODE)
-n_mvpredictors_P_l0 <num>	Number of MV predictors for l0 list of P frames, up to 4 is supported (default is 1) (AVC ENC, ENCODE)
-n_mvpredictors_B_l0 <num>	Number of MV predictors for l0 list of B frames, up to 4 is supported (default is 1) (AVC ENC, ENCODE)
-n_mvpredictors_B_l1 <num>	Number of MV predictors for l1 list, up to 4 is supported (default is 0) (AVC ENC, ENCODE)
-colocated_mb_distortion	Provides the distortion between the current and the co-located MB. It has performance impact (ENC, ENCODE) do not use it, unless it is necessary
-mbsize	With this options size control fields will be used from MB control structure (only ENC+PAK)
-mvout <file>	Use this to output MV predictors(AVC FEI)

Parameter	Description
-mvin <file>	Use this input to set MV predictor for AVC FEI. PREENC and ENC (ENCODE) expect different structures.
-mbcode <file>	File to output per MB information (structure mfxExtFeiPakMBCtrl) for each frame
-repack_preenc_mv	Use this in pair with -mvin to import preenc MVout directly
-num_predictorsL0 <num>	Number of maximum L0 predictors (default - assign depending on the frame type (HEVC FEI ENCODE)
-num_predictorsL1 <num>	Number of maximum L1 predictors (default - assign depending on the frame type (HEVC FEI ENCODE)
-mvPBlocksize <size>	External MV predictor block size (0 - no MVP, 1 - MVP per 16x16, 2 - MVP per 32x32, 7 - use with -mvpin (HEVC FEI ENCODE)
-forceCtuSplit	Force splitting CTU into CU at least once (HEVC FEI ENCODE)
-num_framepartitions <num>	Number of partitions in frame that encoder processes concurrently (1, 2, 4, 8 or 16) (HEVC FEI ENCODE)
-qrep	Quality predictor MV repacking before encode

6.1. 1-to-N Transcoding Configuration

In order to connect sessions together, the input session must specify "-o::sink" as output and the output sessions must specify "-i::source" as input. This tells SMTA that the decoded surfaces produced by the input session should be made available to input sessions and that input sessions use those decoded surfaces as input. The input and output sessions must exist within the same process and must therefore be configured using a parameter file that lists all involved sessions, using the -par command line switch.

There can only be one "sink" defined; hence a single parameter file can only support a single 1-to-N setup (a single input stream transcoded to multiple output streams).

6.2. Examples

6.2.1. 1-to-1 Streaming Transcoding

Below are two examples of 1-to-1 streaming transcoding:

```
StreamingMediaTranscoder -params -i::h264::rtsp
rtsp://192.168.1.100:5554/test.mp4 -o::h264::rtp 192.168.1.200
29500 -h 1080 -w 1920 -b 4000 -f 29.97
```

The above command line takes H.264 video from the specified RTSP URI (can correspond to elementary video streams or a MPEG2 Transport Stream), transcodes it to 1920 x 1080p H.264 output with a bit rate of 4000 and a frame rate of 29.97, and sends it to port 29500 on 192.168.1.100.

6.2.2. 1-to-N Streaming Transcoding

1-to-N transcode provides the functionality of transcoding one stream into N streams with different bitrate, resolution, frame rate, etc.

Below is an example of 1-to-N streaming transcoding:

```
StreamingMediaTranscoder –par 1toN.par
```

Contents of 1toN.par:

```
-name DancingGuy -stats –loop -i::h264::rtsp
rtsp://192.168.1.100:5554/DancingGuy.ts -o::sink -join
-i::source -o::h264::rtp::ts 192.168.1.12 29500 -h 360 -w 640 -b
2000 -f 29.97 -join
-i::source -o::h264::rtp::ts 192.168.1.12 29502 -h 1080 -w 1920
-b 5000 -f 29.97 –join
-i::source -o::mpeg2::rtp::ts 192.168.1.12 29504 -h 1080 -w 1920
-b 5000 -f 29.97 -idrInterval 1 -gopPicSize 30 -gopRefDist 4
  –join
```

The above par file takes H264 TS video as an input, and then transcodes it into three MPEG-TS streams: one output stream is 640 x 360 H.264 with a bit rate set as 2000 bps and frame rate set as 29.97 fps, and sends it to port 29500 on IP address 192.168.1.12; the second stream is 1920 x 1080 H.264 output with a bit rate set as 5000 bps and frame rate set as 29.97 fps, and then sends it to port 29502 on IP address 192.168.1.12. The third one is MPEG2 output with a GOP size of 30, GOP Reference distance set as 4 and one sequence header info before every 1th I-Frame. Transcoding loops on the input stream and statistics collection is enabled during the transcoding.

```
-name DancingGuyES -log ALL -i::h264::rtsp
rtsp://192.168.1.100:5554/DancingGuy.ts -o::sink -join
-i::source -o::h264::rtp 192.168.1.12 29500 -h 360 -w 640 -b
2000 -f 29.97 -join
-i::source -o::h264::rtp 192.168.1.12 29502 -h 1080 -w 1920 -b
5000 -f 29.97 –join
-i::source -o::mpeg2::rtp 192.168.1.12 29504 -h 1080 -w 1920 -b
5000 -f 29.97 -idrInterval 1 -gopPicSize 30 -gopRefDist 4
  –join
```

The above par file takes H264 TS video as an input and transcodes it into three Elementary Streams with the log file enabled.

7. Building SMTA

SMTA comes with source code for the transcoder itself and for the RTSP server.

7.1. FFmpeg*

The SMTA package does not include FFmpeg* 4.0 source. You can obtain the original FFmpeg* 4.0 source codes from http://www.ffmpeg.org/releases/ffmpeg-4.0.tar.gz.

7.1.1. For Windows*

The SMTA depends FFmpeg dynamic link libraries (DLLs). Make sure these files are in folder "dist": avcodec.lib, avcodec-57.def, avcodec-57.dll, avformat.lib, avformat-57.def, avformat-57.dll, avutil.lib, avutil-55.def, avutil-55.dll.

7.1.2. For Linux*

Put ffmpeg-4.0.tar.gz in "imports/ffmpeg/" directory before the SMTA build, the SMTA build process will auto build FFmpeg. When the SMTA build process will complete, the FFmpeg libraries are in the "`dist`" directory one level above the SMTA sources directory, just as in the layout of the complete SMTA package. This directory contains the versioned `.so` files and symbolic links to satisfy the linker: `libavformat.so` → `libavformat.so.57`, `libavcodec.so` → `libavcodec.so.57`, `libavutil.so` → `libavutil.so.55`.

7.2. Building SMTA on Linux*

The SMTA depends on some third libraries, such as FFmpeg, and will build them together by invoking make from the root of the SMTA source tree:

```
make -j8 smta
```

note: the default value of compilation

LIBMFX_ENABLE=1

LIBDISPATCH_ENABLE=0

AVC_FEI_ENABLE=1

HEVC_ENABLE=1

HEVC_FEI_ENABLE=0

MFX_INCLUDE_PATH=/opt/intel/mediasdk/include/mfx

If your MSDK LIBRARY is libdispatch_shared.a, please compile smta as the following command.

```
make -j8 smta LIBDISPATCH_ENABLE=1
MFX_INCLUDE_PATH=/opt/intel/mediasdk/include
```

7.3. Building SMTA on Windows*

SMTA comes with solution and project files for Microsoft Visual Studio* 2015. Simply open the solution file and build the solution. Select **Release** as configuration and **X64** as the target platform.

:The environment variable must be set before you launch Visual Studio*

1. Installing CMake Toolset
2. Add "MFX_HOME" to the "user variables" environment variable:

   ```
   C:\Program Files\Intel\Intel(R) Media Server Studio 2016\Software Development Kit
   ```
3. Open CMake and set "where is the source code" and "where to build the binaries".
4. Click Configure and choose "Visual studio 14 2015 Win64" to generate VS project.
5. Open imediapro-linear_transcoder.sln in "where to build the binaries".
6. Choose Release as building type then build solution.

8. Building FFmpeg* Libraries

The following procedure describes how to build the `libavformat*`, `libavcodec*`, and `libavutil*` libraries. Those libraries are part of the FFmpeg* project. When built as described below, they only contain components licensed under the LGPL v2.1.

The SMTA package does not contain the complete FFmpeg 4.0 source, please place the ffmpeg-4.0.tar.gz under `sources/imports/ffmpeg`. This matches the source tarball at .

10.1 Linux*

1. Install YASM* in case it is not yet present on your system:

   For Ubuntu*:

   ```
   $ sudo apt-get install yasm
   ```

   For SLES*:

   ```
   $ sudo zypper install yasm
   ```

   For CentOS*:

   ```
   $ sudo yum install yasm
   ```

1. Then execute the following commands:

   ```
   $ cd ffmpeg-4.0
   $ ./configure --enable-shared --disable-static       \
         --enable-runtime-cpudetect --disable-debug   \
         [--disable-avfilter] [--disable-swresample] [--disable-
         swscale]
   $ make -j4
   $ sudo make install
   $ sudo /sbin/ldconfig
   ```

The optional `--disable-avfilter --disable-swresample --disable-swscale` arguments are meant to disable building libraries that SMTA does not use. You can omit those if you prefer to build the complete suite of FFmpeg* libraries.

The above procedure will build and install FFmpeg* dynamically loadable libraries. If you prefer to rebuild SMTA from sources and statically link against FFmpeg* libraries, replace the configure command with the following:

```
$ ./configure --enable-runtime-cpudetect \
      [--disable-avfilter --disable-swresample --disable-
      swscale]
```

10.2 For Windows*

FFmpeg is built in Mingw64 system.

1. Install Mingw64 and MSYS.
2. Set environment variables in “Control Panel” -> “System” -> “Advanced system setting” -> “environment variables” -> “User Variables”. Adding “MINGW64” (C:\mingw64\bin) path to “PATH” environment variable.
3. Download yasm-1.3.0-win64.exe and move it to C:\mingw64\bin as named yasm.exe
4. Add text “C:\mingw64\msys\1.0\msys.bat, add “call "C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin \amd64\vcvars64.bat” to head of msys.bat which can be found in folder “C:\mingw64\msys\1.0”.
5. Run msys.bat and change directory to “imports/ffmpeg”.
6. Running “make smta “, this will compile and install files to “dist” directory in root folder of source.

9. Known Issues and Limitations

The following known issues and limitations apply to this release of SMTA:

1. HEVC encoder only support main profile.
2. When decoding or transcoding HEVC video, video frame rate is not more than 30.
3. When decoding or transcoding HEVC video, only support ES format.
4. When transcoding H264 (4K) and HEVC (1080p) to HEVC (1080p), transcoded stream cannot be played smoothly by VLC.
5. When doing 1 to N streaming for H.264 elementary clips, jitters will happen and the video cannot be played to end.
6. H264 and MPEG2 file transcoding to H264 with 'speed, balanced, quality' works well but the size of output is abnormal.
7. The bitrate value and fps of output stream is abnormal for some video.
8. On Windows, when encoding HEVC video, the output stream play with shake.
9. On Windows, the parameter '-u' works abnormal.
10. The FEI encoder supports only H.264/H.265
11. On Windows, the FEI encoder is not supported.
12. On Centos, the pipeline is noVPP and includes FEI ENCPAK or PREENC-ENCODE, SMTA process blocks in multiple processes.

10. New in this Release

- Add FEI feature support(PreENC, ENCODE) for encoder
- Add dynamic VPP
- Add FEI feature support(ENC, PAK) for encoder
- Add MFE feature support for encoder
- Add HEVC FEI feature(PreENC, ENCODE) support for encoder
- Add Dockerfile to generate SMTA transcoding container

11. Third-party Components

SMTA utilizes certain third-party software components. Such third-party software is copyrighted by their respective owners as indicated below.

11.1. FFmpeg*

SMTA uses FFmpeg's* libavformat for muxing and demuxing of MPEG2 Transport Streams.

FFmpeg* (Copyright (c) 2000-2008 Fabrice Bellard, et al.) is free software licensed under the GNU Lesser General Public License (LGPL). Some parts are released under other licenses, including GPL but SMTA does not require any of the components under GPL.

More information about FFmpeg* can be found at http://ffmpeg.org.