Splitters and Muxers Sample

Overview

Splitters and Muxers Sample works with **Intel® Media Server Studio 2015 for Linux**.

It demonstrates how to use the **Intel® Media Server Studio – SDK** (hereinafter referred to as "**SDK**") API to create a splitter and muxer using the example FFmpeg* implementation wrapper. The splitter retrieves elementary stream from container and the muxer encapsulates frames of elementary stream into container.

Features

Splitters and Muxers Sample supports the following video formats:

Format type	
Input/output - containers	MPEG-4 Part 14 (MP4), MPEG-2 Transport Stream (M2TS)
Input/output - codecs	Video: H.264, MPEG-2 Audio: AAC, MP3

Hardware Requirements

See `<install-folder>\Media Samples Guide.pdf`.

Software Requirements

See `<install-folder>\Media Samples Guide.pdf`.

Package Contents

Splitters and Muxers Sample package consists of a shared and a static library. The first one contains the actual implementation, and the second is the dispatcher, which redirects functions calls from the application and allows to the shared library to be loaded safely: it reports to the application if the library was not found. The other function of the dispatcher is to enable custom splitters and muxers with the same API. To do this, you should change the library name to load. It is recommended for the application to use **Splitters and Muxers Sample** through the dispatcher.

How to Build the Application

See `<install-folder>\Media Samples Guide.pdf`.

Running the Software

See `<install-folder>\Media Samples Guide.pdf`.

Splitters and Muxers Sample is a shared library which can be invoked from **Full Transcoding Sample** during transcoding.

See <install-folder>/sample_full_transcode/readme-full-transcode.pdf for details.

Known Limitations

- **Splitters and Muxers Sample** does not support muxing MP3 streams at 12KHz.
- MFXSplitter_GetBitstream may return not the whole frame but one field for the interlaced streams.
- Repositioning using splitters from **Splitters and Muxers Sample** was not fully tested.
- Currently **Splitters and Muxers Sample** supports only layer 1 MPEG audio, but you can add layer 2 and 3 support.

  Firstly, modify enum mfxTrackType in mfxsmstructures.h by removing of MFX_TRACK_MPEGA and adding MFX_TRACK_MPEGA1, MFX_TRACK_MPEGA2 and MFX_TRACK_MPEGA3

  For splitters, modify GetTrackTypeByCodecID function inside ffmpeg_splitter_impl.c:

```
remove
 case AV_CODEC_ID_MP3: return MFX_TRACK_MPEGA;
and add
 case AV_CODEC_ID_MP1: return MFX_TRACK_MPEGA1;
 case AV_CODEC_ID_MP2: return MFX_TRACK_MPEGA2;
 case AV_CODEC_ID_MP3: return MFX_TRACK_MPEGA3;
```

  For muxers, modify GetCodecIDByTrackType function inside ffmpeg_mux_impl.c:

```
remove
 case MFX_TRACK_MPEGA: return AV_CODEC_ID_MP3;
and add
 case MFX_TRACK_MPEGA1: return AV_CODEC_ID_MP1;
 case MFX_TRACK_MPEGA2: return AV_CODEC_ID_MP2;
 case MFX_TRACK_MPEGA3: return AV_CODEC_ID_MP3;
```

Structure Reference

The following section describes structures which are used in splitters and muxers:

mfxDataIO

```
typedef struct {
    mfxU32 reserved1[4];
    mfxHDL pthis;
    mfxI32 (*Read) (mfxHDL pthis, mfxBitstream *bs);
    mfxI32 (*Write) (mfxHDL pthis, mfxBitstream *bs);
    mfxI64 (*Seek) (mfxHDL pthis, mfxI64 offset, mfxSeekOrigin origin);
    mfxHDL reserved2[4];
} mfxDataIO;
```

Description

This structure describes callback functions *Read*, *Write* and *Seek* that should be implemented by the application.

Members

pthis	Pointer to the file or stream with the data for i/o callbacks.
Read	Pointer to the function for reading.

Write	Pointer for the function for writing.
Seek	Pointer to the function for seeking.

Read

```
mfxI32 (*Read) (mfxHDL pthis, mfxBitstream *bs);
```

Description
This function reads data from stream object `pthis`.

Parameters

pthis	Pointer to the file or stream with the data to be read.
bs	Pointer to the output bitstream.

Return Value
The number of bytes successfully read.

Read

```
mfxI32 (*Write) (mfxHDL pthis, mfxBitstream *bs);
```

Description
This function writes bitstream data to the stream object.

Parameters

pthis	Pointer to the file or stream to write data.
bs	Pointer to the bitstream to be written.

Return Value
The number of bytes successfully written.

Seek

```
mfxI64 (*Seek) (mfxHDL pthis, mfxI64 offset, mfxSeekOrigin origin);
```

Description
This function sets the new byte position in the stream object.

Parameters

pthis	Pointer to the input file or stream.
offset	Number of bytes to offset from the position specified by origin.
origin	Relative byte position for the offset. See the *mfxSeekOrigin* enumerator for all available options

Return Value

The new position or any value <0 if failed. If offset is 0 and origin is `MFX_SEEK_END` it returns the file size without seeking or <0 if it is not implemented.

mfxStreamParams

```
typedef struct mfxStreamParams {
  mfxU16 reserved[22];
  mfxSystemStreamType SystemType;
  mfxU32 Flags;
  mfxU64 Duration;
  mfxU16 NumTracks;
  mfxU16 NumTracksAllocated;
  mfxTrackInfo **TrackInfo;
} mfxStreamParams;
```

Description

This structure describes stream parameters which can be used as output for splitter or input for muxer.

Members

SystemType	Container format. See the *mfxSystemStreamType* enumerator for a complete list of containers.
Flags	Stream flags, currently is not used.
Duration	The duration of the stream in units of 90 KHz .
NumTracks	Numbers of tracks in the stream.
NumTracksAllocated	Number of tracks allocated by the application. Once the application allocates `TrackInfo`, it sets `NumTracksAllocated`. Normally, the application allocates `TrackInfo` for each track and sets `NumTracksAllocated` equal to `NumTracks`. See *MFXSplitter_GetInfo* for details.
TrackInfo	Information about the elementary stream. See the *mfxTrackInfo* description for additional details.

mfxTrackInfo

```
typedef struct {
  mfxTrackType Type;
  mfxU32 SID;
  mfxU16 Enable;
  mfxU16 HeaderLength;
  mfxU8 Header[MFX_TRACK_HEADER_MAX_SIZE];
  mfxU16 reserved[16];
  union {
    mfxAudioInfoMFX AudioParam;
    mfxInfoMFX VideoParam;
 };
} mfxTrackInfo;
```

Description

This structure represents the information about the elementary stream.

Members

Type	Codec format. See the *mfxTrackType* enumerator for a complete list of codecs.
SID	Unique stream identifier.
Enable	1 if enabled, 0 otherwise.
HeaderLength	Length in bytes of the specific codec info.
Header	The codec-specific info. For example, for H.264 codec it must contain SPS/PPS NAL units.
AudioParam	Specific audio parameters. The splitter fills the next fields: `StreamInfo.NumChannel`, `StreamInfo.SampleFrequency`, `StreamInfo.Bitrate`, `StreamInfo.BitPerSample` and `CodecID`. The mandatory fields are: `StreamInfo.NumChannel`, `StreamInfo.SampleFrequency`, `StreamInfo.Bitrate`, `StreamInfo.BitPerSample`.
VideoParam	Specific video parameters. The splitter fills the next fields: `FrameInfo.Width`, `FrameInfo.Height`, `CodecProfile` and `CodecId`. The mandatory fields are: `FrameInfo.Width`, `FrameInfo.Height`, `FrameInfo.FrameRateExtD`, `FrameInfo.FrameRateExtN`.

mfxSeekOrigin

Description

This enumerator specifies the relative position from which the reposition will be performed.

List of values

MFX_SEEK_ORIGIN_BEGIN	The beginning of the file or stream.
MFX_SEEK_ORIGIN_CURRENT	The current position in the file or stream.
MFX_SEEK_ORIGIN_END	The end position in the file or stream.

mfxTrackType

Description

This enumerator specifies audio or video codec.

List of values

MFX_TRACK_MPEG2V	MPEG-2
MFX_TRACK_H264	H.264
MFX_TRACK_VC1	VC-1

MFX_TRACK_VP8	VP8
MFX_TRACK_ANY_VIDEO	Common type for video codec.
MFX_TRACK_AAC	AAC
MFX_TRACK_MPEGA	MP3
MFX_TRACK_ANY_AUDIO	Common type for audio codec.
MFX_TRACK_UNKNOWN	Unknown codec.

mfxSystemStreamType

Description

This enumerator specifies the container format.

List of values

MFX_UNDEF_STREAM	Unknown format.
MFX_MPEG2_TRANSPORT_STREAM	MPEG TS
MFX_MPEG4_SYSTEM_STREAM	MPEG-4
MFX_IVF_STREAM	IVF
MFX_ASF_STREAM	ASF
MFX_TRACK_AAC	AAC

MFXSplitter_Init

```
mfxStatus MFXSplitter_Init(mfxDataIO *data_io, mfxSplitter *spl);
```

Description

This function creates and initializes **SDK** splitter spl, identifies the input format and fills the internal info. This function must be called before any other calls. `pthis`, *`Read`* and *`Seek`* callbacks are mandatory for `data_io`.

Parameters

data_io	Pointer to the *`mfxDataIO`* object.
spl	Pointer to the output **SDK** splitter.

Return Values

MFX_ERR_NONE	The splitter was initialized successfully.
MFX_ERR_NULL_PTR	NULL input parameter or mandatory `mfxDataIO` field.
MFX_ERR_MEMORY_ALLOC	Not enough memory to allocate internal objects.
MFX_ERR_UNKNOWN	Can’t identify input format or invalid stream.

MFXSplitter_Close

```
mfxStatus MFXSplitter_Close(mfxSplitter spl);
```

Description

This function closes **SDK** splitter and frees internal objects. This function must be called after all of the splitter operations are finished.

Parameters

spl	**SDK** splitter handle.

Return Values

MFX_ERR_NONE	The function completes successfully.
MFX_ERR_NULL_PTR	Invalid splitter handle.

MFXSplitter_GetInfo

```
mfxStatus MFXSplitter_GetInfo(mfxSplitter spl, mfxStreamParams *par);
```

Description

This function retrieves and fills information about contained tracks. The `TrackInfo` from par should be allocated by the application. If `TrackInfo` structure is NULL, this function sets `NumTracks` field of the parameters to allow user allocate required number of `TrackInfo`, set `NumTracksAllocated` and pass them in the second call. This function must be called after *MFXSplitter_Init* and before any other calls. Note: the function returns `MFX_ERR_NONE` if codec or format is unsupported, but the fields `SystemType` of `mfxStreamParams` will be `MFX_UNDEF_STREAM` or the `Type` field of `TrackInfo` from `mfxStreamParams` will be `MFX_TRACK_UNKNOWN`. The function retrieves other stream info if it is possible. The application can handle this case as an error at its discretion.

Parameters

spl	`SDK` splitter handle.
par	Pointer to the output splitter parameters.

Return Values

MFX_ERR_NONE	The function identifies number of tracks or completely fills the parameters.
MFX_ERR_NULL_PTR	One of the input parameters is NULL.
MFX_ERR_MORE_DATA	Not enough TrackInfo-s were allocated.
MFX_ERR_UNKNOWN	The splitter can't retrieve stream info.

MFXSplitter_GetBitstream

```
mfxStatus MFXSplitter_GetBitstream(mfxSplitter spl, mfxU32 *track_num,
mfxBitstream *bs);
```

Description

This function returns the next frame and it's track index in the input stream. The application should call *MFXSplitter_ReleaseBitstream* after the output bitstream data is no longer needed.

Parameters

spl	**SDK** splitter handle.
track_num	The index of track in the `TrackInfo` array. Don't mix it up with SID.
bs	Pointer to the output bitstream. bs `Data, DataLength and DecodeTimeStamp` fields are mandatory. `DecodeTimeStamp` value should increase monotonically. As for `TimeStamp`, use `MFX_TIMESTAMP_UNKNOWN` if `TimeStamp` is unknown.

Return Values

MFX_ERR_NONE	The function completes successfully.
MFX_ERR_NULL_PTR	One of the input parameters is NULL.
MFX_ERR_NOT_ENOUGH_BUFFER	Means that the application holds the packets and does not call *`MFXSplitter_ReleaseBitstream`* for a long time.
MFX_ERR_MORE_DATA	The splitter need more data or reached the end of the file, bs Data should be NULL. If one of the elementary streams has finished, the splitter returns last bs Data for this particularly stream with `MFX_BITSTREAM_EOS` in bs DataFlag and returns `MFX_ERR_NONE`
MFX_ERR_UNKNOWN	The splitter can't get next frame.

MFXSplitter_ReleaseBitstream

```
mfxStatus MFXSplitter_ReleaseBitstream(mfxSplitter spl, mfxBitstream
*bs);
```

Description

This function releases resources after *`MFXSplitter_GetBitstream`* call.

Parameters

spl	**SDK** splitter handle.
bs	Pointer to the input bitstream.

Return Values

MFX_ERR_NONE	The function completes successfully.

MFXSplitter_Seek

```
mfxStatus MFXSplitter_Seek(mfxSplitter spl, mfxU64 timestamp);
```

Description

This function seeks to the key frame at position specified as `timestamp`.

Parameters

spl	**SDK** splitter handle.
timestamp	Time stamp to reposition in units of 90 KHz.

Return Values

MFX_ERR_NONE	The function completes successfully.
MFX_ERR_NULL_PTR	Invalid splitter handle.
MFX_ERR_UNKNOWN	The splitter can't seek at specified position.

MFXMuxer_Init

```
mfxStatus MFXMuxer_Init(mfxStreamParams* par, mfxDataIO *data_io,
mfxMuxer *mux);
```

Description

This function creates and initializes **SDK** muxer `mux`, sets the output format and fills internal info. This function must be called firstly. `pthis`, *`Seek`* and *`Write`* callbacks are mandatory for `data_io`.

Parameters

par	Pointer to the input muxer parameters.
data_io	Pointer to the `mfxDataIO` object.
mux	Pointer to the output **SDK** muxer.

Return Values

MFX_ERR_NONE	The muxer was initialized successfully.
MFX_ERR_NULL_PTR	NULL input parameter or mandatory *`mfxDataIO`* field.
MFX_ERR_MEMORY_ALLOC	Not enough memory to allocate internal objects.
MFX_ERR_UNKNOWN	The muxer can't be initialized.

MFXMuxer_Close

```
mfxStatus MFXMuxer_Close(mfxMuxer mux);
```

Description

This function closes **SDK** muxer and frees internal objects. This function must be called after all of the muxer operations are finished.

Parameters

mux	**SDK** muxer handle.

Return Values

MFX_ERR_NONE	The function completes successfully.
MFX_ERR_NULL_PTR	Invalid muxer handle.
MFX_ERR_UNKNOWN	Not all of the internal objects were released successfully.

MFXMuxer_PutBitstream

```
mfxStatus MFXMuxer_PutBitstream(mfxMuxer mux, mfxU32 track_num,
mfxBitstream *bs, mfxU64 duration);
```

Description

This function puts the next frame to the output stream.

Parameters

mux	**SDK** muxer handle.
track_num	Stream index for the input frame.
bs	Pointer to the input bitstream. `The Data, DataLength, FrameType` (`MFX_FRAMETYPE_I` or not) and `DecodeTimeStamp` fields are mandatory. Use `MFX_TIMESTAMP_UNKNOWN` if `TimeStamp` is unknown.
duration	Frame duration in units of 90 KHz or 0 if it is unknown.

Return Values

MFX_ERR_NONE	The function completes successfully
MFX_ERR_NULL_PTR	One of the input parameters is NULL.
MFX_ERR_UNKNOWN	The muxer can’t put the frame.

Note: See `<msdk_install-folder>/ media_server_studio_sdk_release_notes.pdf` for `mfxStatus, mfxBitstream, mfxAudioInfoMFX and mfxInfoMFX` description.

Legal Information

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 by visiting *Intel's Web Site*.

MPEG is an international standard for video compression/decompression promoted by ISO. Implementations of MPEG CODECs, or MPEG enabled platforms may require licenses from various entities, including Intel Corporation.

Intel, the Intel logo, Intel Core are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804
