

# Splitters and Muxers Sample

[Overview](#)

[Features](#)

[Software Requirements](#)

[Package Contents](#)

[How to Build the Application](#)

[Running the software](#)

[Structure Reference](#)

[Enumerator Reference](#)

[Splitters API](#)

[Muxers API](#)

[Known Limitations](#)

[Legal Information](#)

## Overview

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

It demonstrates how to use **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 container formats:

Input/output	MPEG-4 Part 14 (MP4), MPEG-2 Transport Stream (M2TS)
--------------	--

and codecs:

Input/output	Video: H.264, MPEG-2 Audio: AAC, MP3
--------------	---

## 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 (.so), 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 .so 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.

The dispatcher contains the following:

### <install-folder>/sample\_spl\_mux/dispatcher/

CMakeLists.txt

CMake file for the "Splitters and muxers" dispatcher that controls a build process.

### <install-folder>sample\_spl\_mux/dispatcher/include/

mux\_exposed\_functions\_has\_impl.h

Header file with modified "Splitters and muxers" sample muxers functions.

mux\_exposed\_functions\_list.h

Header file with muxers functions that were used without modifications.

spl\_exposed\_functions\_has\_impl.h

Header file with modified "Splitters and muxers" sample splitters functions.

spl\_exposed\_functions\_list.h

Header file with splitters functions that were used without modifications.

### <install-folder>/sample\_spl\_mux/dispatcher/src/

spl\_mux\_dispatcher.c

Source file with the modified "Splitters and muxers" functions implementations.

The shared library module contains the following:

### <install-folder>/sample\_spl\_mux/

readme-splitters-muxers.pdf

This file.

### <install-folder>/sample\_spl\_mux/api/

mfxsmstructures.h

Header file with the structures definitions.

mfxsplmux.h	Header file with the API functions definitions.
mfxsplmux++.h	Header file with the splitters and muxers classes of c++ wrapper.
<install-folder>/sample_spl_mux/include/	
spl_mux_defs.h	Header file with definitions
adts_muxer.h	Header file for the AAC ADTS header writer functions definitions.
ffmpeg_mux_impl.h	Header file for the muxers functions definitions.
ffmpeg_reader_writer.h	Header file for splitters and muxers callbacks definitions .
ffmpeg_splitter_impl.h	Header file for the splitters functions definitions.
<install-folder>/sample_spl_mux/src/	
adts_muxer.c	Source file for the AAC ADTS header writer functions implementations.
ffmpeg_mux_impl.c	Source file for the muxers functions implementations.
ffmpeg_reader_writer.c	Source file for the splitters and muxers callbacks implementation.
ffmpeg_splitter_impl.c	Source file for the splitters functions implementation.

## How to Build the Application

See <install-folder>/Media Samples Guide.pdf for general build instructions, including how to resolve FFmpeg\* dependency.

## Using custom Splitters and Muxers

See <install-folder>/Media Samples Guide.pdf for general build instructions, including how to resolve FFmpeg\* dependency.

## Running the Software

**Splitters and Muxers Sample** is a Shared Library (.so) which can be invoked from **Full Transcoding Sample** during transcoding.

See <install-folder>/sample\_full\_transcode/readme-full-transcode.pdf for details.

## Structure Reference

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

### mfxDataIO

#### Definition

```
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

<code>pthis</code>	Pointer to the file or stream with the data for i/o callbacks.
<a href="#">Read</a>	Pointer to the function for reading.
<a href="#">Write</a>	Pointer for the function for writing.
<a href="#">Seek</a>	Pointer to the function for seeking.

Callback functions should be complied with the following interface:

### Read

#### Syntax

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

#### Parameters

<code>pthis</code>	Pointer to the file or stream with the data to be read.
<code>bs</code>	Pointer to the output bitstream.

#### Description

This function reads data from stream object `pthis`.

### Return Value

The number of bytes successfully read.

## Write

### Syntax

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

### Parameters

<code>pthis</code>	Pointer to the file or stream to write data.
<code>bs</code>	Pointer to the bitstream to be written.

### Description

This function writes bitstream data to the stream object.

### Return Value

The number of bytes successfully written.

## Seek

### Syntax

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

### Parameters

<code>pthis</code>	Pointer to the input file or stream.
<code>offset</code>	Number of bytes to offset from the position specified by <code>origin</code> .
<code>origin</code>	Relative byte position for the offset. See the <a href="#">mfxSeekOrigin</a> enumerator for all available options.

### Description

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

### Return Value

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

## mfxStreamParams

### Definition

```
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 <a href="#">mfxSystemStreamType</a> 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 <code>TrackInfo</code> , it sets <code>NumTracksAllocated</code> . Normally, the application allocates <code>TrackInfo</code> for each track and sets <code>NumTracksAllocated</code> equal to <code>NumTracks</code> . See <a href="#">MFXSplitter_GetInfo</a> for details.
TrackInfo	Information about the elementary stream. See the <a href="#">mfxTrackInfo</a> description for additional details.

## mfxTrackInfo

## Definition

```
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 <a href="#">mfxTrackType</a> 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.Bitrates, StreamInfo.BitPerSample and CodecID. The mandatory fields are: StreamInfo.NumChannel, StreamInfo.SampleFrequency, StreamInfo.Bitrates and 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 and FrameInfo.FrameRateExtN.

## Enumerator Reference

The following section contains splitters and muxers enumerators.

### mfxSeekOrigin

## Description

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

## Name/Description

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.

### Name/Description

Video codecs:

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.

Audio codecs:

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.

### Name/Description

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

## Splitters API

This part describes splitters API.



## MFxSplitter\_Init

### Syntax

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

### Parameters

data_io	Pointer to the <a href="#">mfxDataIO</a> object.
spl	Pointer to the output SDK splitter.

### 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`.

### Return Status

MFx_ERR_NONE	The splitter was initialized successfully.
MFx_ERR_NULL_PTR	NULL input parameter or mandatory <code>mfxDataIO</code> 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

### Syntax

```
mfxStatus MFxSplitter_Close(mfxSplitter spl);
```

### Parameters

spl	SDK splitter handle.
-----	----------------------

### Description

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

### Return Status

MFx_ERR_NONE	The function completes successfully.
MFx_ERR_NULL_PTR	Invalid splitter handle.

## MFxSplitter\_GetInfo

### Syntax

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

## Parameters

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

## 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.

## Return Status

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

## MFXSplitter\_GetBitstream

### Syntax

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

## Parameters

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

## 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.

## Return Status

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

## MFXSplitter\_ReleaseBitstream

### Syntax

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

### Parameters

<code>spl</code>	SDK splitter handle.
<code>bs</code>	Pointer to the input bitstream.

### Description

This function releases resources after [MFXSplitter\\_GetBitstream](#) call.

## Return Status

<code>MFX_ERR_NONE</code>	The function completes successfully.
---------------------------	--------------------------------------

## MFXSplitter\_Seek

### Syntax

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

### Parameters

<code>spl</code>	SDK splitter handle.
<code>timestamp</code>	Time stamp to reposition in units of 90 KHz.

### Description

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

## Return Status

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.

## Muxers API

This part describes muxers API.

### MFxMuxer\_Init

#### Syntax

```
mfxStatus MFxMuxer_Init(mfxStreamParams* par, mfxDatIO *data_io,
mfxMuxer *mux);
```

#### Parameters

par	Pointer to the input muxer parameters.
data_io	Pointer to the <a href="#">mfxDatIO</a> object.
mux	Pointer to the output SDK muxer.

#### 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`.

## Return Status

MFx_ERR_NONE	The muxer was initialized successfully.
MFx_ERR_NULL_PTR	NULL input parameter or mandatory <code>mfxDatIO</code> field.
MFx_ERR_MEMORY_ALLOC	Not enough memory to allocate internal objects.
MFx_ERR_UNKNOWN	The muxer can't be initialized.

### MFxMuxer\_Close

#### Syntax

```
mfxStatus MFxMuxer_Close(mfxMuxer mux);
```

#### Parameters

mux	SDK muxer handle.
-----	-------------------

#### Description

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

### Return Status

<code>MXF_ERR_NONE</code>	The function completes successfully.
<code>MXF_ERR_NULL_PTR</code>	Invalid muxer handle.
<code>MXF_ERR_UNKNOWN</code>	Not all of the internal objects were released successfully.

## MXFMuxer\_PutBitstream

### Syntax

```
mxStatus MXFMuxer_PutBitstream(mxMuxer mux, mxU32 track_num,
mxBitstream *bs, mxU64 duration);
```

### Parameters

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

### Description

This function puts the next frame to the output stream.

### Return Status

<code>MXF_ERR_NONE</code>	The function completes successfully.
<code>MXF_ERR_NULL_PTR</code>	One of the input parameters is <code>NULL</code> .
<code>MXF_ERR_UNKNOWN</code>	The muxer can't put the frame.

**Note:** See <msdk\_install-folder>/media\_server\_studio\_sdk\_release\_notes.pdf for `mxStatus`, `mxBitstream`, `mxAudioInfoMXF` and `mxInfoMXF` description.

## Known Limitations

- **Splitters and Muxers sample** does not support muxing MP3 streams at 12KHz
- `MXFSplitter_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;
```

## Legal Information

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 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 and the Intel logo 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

