Intel® Screen Capture Decoder

Developer Reference

API Version 1.20

LEGAL DISCLAIMER

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.

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

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

Table of Contents

Overview

The Intel® Media SDK Screen Capture is a development library that exposes the media acceleration capabilities of Intel platforms for Windows Desktop front frame buffer capturing. The API library covers a wide range of Intel platforms.

The Intel® Media SDK Screen Capture package includes a hardware accelerated (HW) plug-in library exposing graphics acceleration capabilities, implemented as Intel® Media SDK Decode plug-in. The plug-in can be loaded into/used with only Intel Media SDK Hardware Library (see mediasdkusr-man.pdf document for plugin loading and more details). The package also includes a simple console sample showing how to use the Intel Media SDK Screen Capture plug-in.

This document describes Intel Media SDK Screen Capture API. To learn more about general Intel Media SDK API definition and Decode component see mediasdk-man.pdf.

Document Conventions

The Intel Screen Capture uses the Verdana typeface for normal prose. With the exception of section headings and the table of contents, all code-related items appear in the `Courier New` typeface (`mxfStatus` and `MFXInit`). All class-related items appear in all cap boldface, such as **DECODE** and **ENCODE**. Member functions appear in initial cap boldface, such as **Init** and **Reset**, and these refer to members of all three classes, **DECODE**, **ENCODE** and **VPP**. Hyperlinks appear in underlined boldface, such as **`CodecFormatFourCC`**.

Acronyms and Abbreviations

API	Application Programming Interface
Direct3D9	Microsoft* Direct3D* version 9
Direct3D11	Microsoft* Direct3D* version 11.1
DXVA2	Microsoft DirectX* Video Acceleration standard 2.0
RGB4	Thirty-two-bit RGB color format. Also known as RGB32
NV12	A color format for raw video frames
SDK	Intel® Media SDK
SDK execution	Intel® Media SDK execution
SDK functions	Intel® Media SDK functions
SDK library	Intel® Media SDK library
SDK session	Intel® Media SDK session
video memory	memory used by hardware acceleration device, also known as GPU, to hold frame and other types of video data

Architecture

Intel® Media SDK functions fall into the following categories:

DECODE	Decode compressed video streams into raw video frames
CORE	Auxiliary functions for synchronization
Misc	Global auxiliary functions

Screen capturing

Screen capture plug-in implemented through typical decoder interface with some additional features. Table 1 shows the Screen Capture features. The application can configure supported video decoding features through the video decoding initialization parameters. The application can also configure optional features through hints (additional extended buffer parameters). See *Screen capture procedure using decode plug-in / Configuration"* for more details on how to configure decoder.

Table 1: Screen capturing supported features

Screen Capturing Features	Configuration
Capture display front buffer in NV12 format	initialization parameters
Capture display front buffer in RGB4 format	initialization parameters
Display selection	hint
Dirty rectangles detection	hint

Programming Guide

This chapter describes the concepts used in programming the Intel® Media SDK and the Screen Capture.

The application must use the include file, **mfxvideo.h** (for C programming), or **mfxvideo++.h** (for C++ programming), and link the Intel Media SDK static dispatcher library, **libmfx.lib**.

Include these files:

```
#include "mfxvideo.h"    /* The SDK include file */
#include "mfxvideo++.h" /* Optional for C++ development */
#include "mfxplugin.h"   /* Plugin development */
```

Link this library:

```
libmfx.lib                /* The SDK static dispatcher library */
```

Screen capture procedure using decode plug-in

Example 1 shows the pseudo code of the video screen capturing procedure using plugin. The following describes a few key points:

- Typical decoder function **MFXVideoDECODE_DecodeHeader** is not available for screen capture plug-in since no input stream is provided to the decoder.
- The application uses the **MFXVideoDECODE_QueryIOSurf** function to obtain the number of working frame surfaces required to reorder output frames.
- The application calls the **MFXVideoDECODE_DecodeFrameAsync** function for a decoding operation, without the bitstream buffer (pointer to the bitstream structure must be set to NULL), and a free unlocked working frame surface (`work`) as input parameters. If decoding output is not available, the function returns a status code requesting additional working frame surface as follows:

  MFX_ERR_MORE_SURFACE: The function needs one more frame surface to produce any output.

- Upon successful decoding, the **MFXVideoDECODE_DecodeFrameAsync** function returns **MFX_ERR_NONE**. However, the decoded frame data (identified by the `disp` pointer) is not yet available because the **MFXVideoDECODE_DecodeFrameAsync** function is asynchronous. The application must use the **MFXVideoCORE_SyncOperation** function to synchronize the decoding operation before retrieving the decoded frame data.

```
mfxSession session;
MFXInit(MFX_IMPL_HARDWARE, 1.17, &session);
MFXVideoUSER_Load(session, MFX_PLUGINID_CAPTURE_HW, version);
mfxVideoParam *in;// input parameters structure
mfxVideoParam *out;// output parameters structure
/* allocate structures and fill input parameters structure, zero unused fields */
MFXVideoDECODE_Query(session, in, out);
/* check supported parameters */
MFXVideoDECODE_QueryIOSurf(session, &in, &request);
allocate_pool_of_frame_surfaces(request.NumFrameSuggested);
MFXVideoDECODE_Init(session, &in);
sts=MFX_ERR_MORE_DATA;
for (;;) {
       find_unlocked_surface_from_the_pool(&work);
       sts=MFXVideoDECODE_DecodeFrameAsync(session,NULL,work,&disp,&syncp);
       if (sts==MFX_ERR_MORE_SURFACE) continue;
       … // other error handling
       if (sts==MFX_ERR_NONE) {
              MFXVideoCORE_SyncOperation(session, syncp, INFINITE);
              do_something_with_decoded_frame(disp);
       }
}
MFXVideoDECODE_Close(session);
free_pool_of_frame_surfaces();
MFXClose(session)
```

Example 1: Video decoding pseudo code

Configuration

The Screen Capture plug-in configures the screen capturing pipeline operation based on parameters specified in the **mfxVideoParam** structure.

Example 2 shows how to configure the SDK screen capturing.

```
    /* configure the mfxVideoParam structure */
    mfxVideoParam conf;

    memset(&conf,0,sizeof(conf));
    conf.IOPattern=MFX_IOPATTERN_OUT_VIDEO_MEMORY;

    conf.mfx.CodecId=MFX_CODEC_CAPTURE;
    conf.mfx.FrameInfo.FourCC=MFX_FOURCC_NV12;
    conf.mfx.FrameInfo.ChromaFormat=MFX_CHROMAFORMAT_YUV420;
    conf.mfx.FrameInfo.Width=conf.mfx.FrameInfo.CropW=1920;
    conf.mfx.FrameInfo.Height=1088;
    conf.mfx.FrameInfo.CropH=1080;

    /* video decoding initialization */
    MFXVideoDECODE_Init(session, &conf);
```

Example 2: Screen capture configuration pseudo code

Transcoding Procedures

The application can use other Intel Media SDK components and Screen Capture plugin together for transcoding operations. For example, video processing functions to resize with resolution less than initial content and then encode to H.264 or H.265, or directly encode to JPEG in RGB32. For more details on building transcoding pipelines please see mediasdk-man.pdf.

For more details and pseudo code examples, please refer to **Transcoding Procedures** section in the mediasdk-man.pdf.

Plug-in does not support capturing with a specified frame rate, so that frame is captured immediately after plug-in's **MFXVideoDECODE_DecodeFrameAsync** function call. This means that one frame could be captured several times and some other frame could be skipped.
It is application responsibility to call plug-in with constant frequency to get a constant FPS.

Screen capture with DirtyRect detection using decode plug-in

Example 3 shows the pseudo code of the video screen capturing with DirtyRect feature usage procedure using plugin. The following describes a few key points:

- Capturing procedure with DirtyRect detection is almost like typical screen capture pipeline described by **Screen capture procedure using decode plug-in** section above;
- DirtyRect feature should be turned on during **MFXVideoDECODE_Init** function call by setting corresponded parameter (`EnableDirtyRect`) in the provided extended buffer **mfxExtScreenCaptureParam**;
- **mfxExtDirtyRect** extended buffer should be attached to the provided surfaces into the **MFXVideoDECODE_DecodeFrameAsync** function. Detected dirty rectangles will be reported into this extended buffer after **MFXVideoCORE_SyncOperation** function call.

```
mfxSession session;
MFXInit(MFX_IMPL_HARDWARE, 1.17, &session);
MFXVideoUSER_Load(session, MFX_PLUGINID_CAPTURE_HW, version);
mfxVideoParam par; //input parameters structure
mfxExtScreenCaptureParam extScPar; //input extended parameters structure
  extScPar.Header.BufferId = MFX_EXTBUFF_SCREEN_CAPTURE_PARAM;
  extScPar.Header.BufferSz = sizeof(extPar);
  extScPar.EnableDirtyRect = 1;

//Atach extended parameter structure to the input parameters:
mfxExtBuffer* buffers = (mfxExtBuffer*) &extScPar;
par.ExtParam = &buffers;
par.NumExtParam = 1;

MFXVideoDECODE_QueryIOSurf(session, &in, &request);

//allocate surface pool with mfxExtDirtyRect buffer attached to each surface
allocate_pool_of_frame_surfaces_and_attach_dirtyrect_buf(request.NumFrameSuggested);

MFXVideoDECODE_Init(session, &in);
sts=MFX_ERR_MORE_DATA;
for (;;) {
       find_unlocked_surface_from_the_pool(&work);
       sts=MFXVideoDECODE_DecodeFrameAsync(session,NULL,work,&disp,&syncp);
       if (sts==MFX_ERR_MORE_SURFACE) continue;
       … // other error handling
       if (sts==MFX_ERR_NONE) {
              MFXVideoCORE_SyncOperation(session, syncp, INFINITE);
              do_something_with_decoded_frame_and_dirty_rect_info(disp);
       }
}
MFXVideoDECODE_Close(session);
free_pool_of_frame_surfaces();
MFXClose(session)
```

Example 3: Video decoding with DirtyRect detection pseudo code

Screen capture with display selection

Screen capture plug-in supports display selection for systems with enabled virtual display. The following describes a few key points:

- Capturing procedure with display selection is exactly like typical screen capture pipeline described by **Screen capture procedure using decode plug-in** section above, the only difference is initialization;
- Display selection is done during **MFXVideoDECODE_Init** function call by setting corresponded parameter (`DisplayIndex`) in the provided extended buffer **mfxExtScreenCaptureParam**;
- `DisplayIndex` represents the id field in the `DISPLAYCONFIG_PATH_TARGET_INFO` structure reported by `QueryDisplayConfig` function;

- Display selection feature is available on systems with virtual displays only (without physical display connected) and RGB4 output fourcc format. For more information about supported configurations please refer to the release notes.

Known limitations

There are several known issues and limitations in usage models of screen capture:

Constant frame rate capturing

Plug-in does not support capturing with a specified frame rate, so that frame is captured immediately after plug-in's **MFXVideoDECODE_DecodeFrameAsync** function call. This means that one frame could be captured several times and some other frame could be skipped.
It is application responsibility to call plug-in with constant frequency to get a constant FPS.

DirectX 11 and RGB4 output fourcc configuration particularity

Please note that in case of DirectX 11 implementation, video memory type, and RGB4 surface format usage, the application frame allocator needs to allocate the surfaces using DXGI_FORMAT_AYUV format because OS runtime will block RGB surface allocation with BIND_DECODER flag and decoder output view. In any other configuration cases, e.g. DirectX 9 implementation, system or opaque memory type, or NV12 output format, special frame allocation is not needed.

Capturing through regular DXGI DesktopDuplication or DirectX 9 front buffer capturing method

Plug-in supports screen capturing by slow regular DXGI DesktopDuplication or DirectX 9 front buffer capturing methods. This mode is used when **MFXVideoDECODE_Init** function call returns warning **MFX_WRN_PARTIAL_ACCELERATION** or if the current decoder library session was initialized as a software-based session (**MFX_IMPL_SOFTWARE**).

In case if regular DXGI DesktopDuplication or DirectX 9 front buffer capturing and HW accelerated encoding is required, Decoder should be initialized in a separated SW session and then joined to the Encoder HW session by **MFXJoinSession** SDK function. For more details, please refer to **Multiple Sessions** section in the mediasdk-man.pdf.

Protected content capturing

Plug-in does not support protected content capturing. Performance degradations might be observed if there is running application with OPM (Output Protection Manager) session even without actual protected content playing is open.

Capturing through regular DXGI DesktopDuplication or DirectX 9 front buffer capturing methods mode is activated during runtime (after initialization) if plug-in fails to perform HW accelerated screen capturing because of OPM session presence in the system.

Performance decrease during entering and exiting fullscreen mode

A performance decrease for a several frames might be observed during starting and stopping fullscreen mode of any application.

Resolution change up

Resolution change up (with bigger resolution values than on initialization) is not supported during runtime. Decoder re-initialization (**MFXVideoDECODE_Close** and then **MFXVideoDECODE_Init** with new parameters) is required.

Function Reference

The Screen Capture does not define any new functions in addition to standard Intel Media SDK function set, please check mediasdk-man.pdf and mediasdkusr-man.pdf for the SDK functions description.

Structure Reference

For a complete list of standard Intel Media SDK structure set, please check mediasdk-man.pdf and mediasdkusr-man.pdf for the SDK structures' description.

mfxExtScreenCaptureParam

Definition

```
typedef struct
{
    mfxExtBuffer    Header;

    mfxU32          DisplayIndex;
    mfxU16          EnableDirtyRect;
    mfxU16          EnableCursorCapture;
    mfxU16          reserved[24];
} mfxExtScreenCaptureParam;
```

Description

The `mfxExtScreenCaptureParam` additional options for screen capturing.

Members

`Header.BufferId`	Must be **`MFX_EXTBUFF_SCREEN_CAPTURE_PARAM`**
`Header.BufferSz`	Must be `sizeof(mfxExtScreenCaptureParam)`
`DisplayIndex`	Display index for screen capturing; this value is reserved and must be zero.
`EnableDirtyRect`	Enable DirtyRect detection feature.
`EnableCursorCapture`	Enable cursor capturing; this value is reserved and must be zero.

Change History

This structure is available since SDK API 1.17.

Enumerator Reference

For a complete list of standard Intel Media SDK enumerator set, please check mediasdk-man.pdf and mediasdkusr-man.pdf for the SDK enumerators' description.

CodecFormatFourCC

Description

The `CodecFormatFourCC` enumerator itemizes codecs in the FourCC format.

Name/Description

`MFX_CODEC_AVC`	AVC, H.264, or MPEG-4, part 10 codec
`MFX_CODEC_MPEG2`	MPEG-2 codec
`MFX_CODEC_VC1`	VC-1 codec
`MFX_CODEC_HEVC`	HEVC codec
`MFX_CODEC_CAPTURE`	Screen capture pseudo-codec id

Change History

This enumerator is available since SDK API 1.0.

SDK API 1.8 added `MFX_CODEC_HEVC` definition.

SDK API 1.15 added `MFX_CODEC_CAPTURE` definition.

ExtendedBufferID

Description

The `ExtendedBufferID` enumerator itemizes and defines identifiers (`BufferId`) for extended buffers or video processing algorithm identifiers.

Name/Description

`MFX_EXTBUFF_SCREEN_CAPTURE_PARAM`	This extended buffer defines additional control parameters for screen capturing. See the **`mfxExtScreenCaptureParam`** structure for details. The application can attach this buffer to the **`mfxVideoParam`** structure for screen capture initialization.

This enumerator is available since SDK API 1.0.

SDK API 1.17 adds `MFX_EXTBUFF_SCREEN_CAPTURE_PARAM`.