Contents

[Encoder Interface Usage: 2](#_Toc394403775)

[Step#1: create and destroy the encoder 2](#_Toc394403776)

[Step#2: initialize the encoder 2](#_Toc394403777)

[Step#3: invoker the encoding 3](#_Toc394403778)

[Step#4: control the encoding 4](#_Toc394403779)

[Encoder Option List: 5](#_Toc394403780)

[Encoder Parameter List: 7](#_Toc394403781)

[Encoder Return Value: CM\_RETURN 10](#_Toc394403782)

[Encoder Return Value: EVideoFrameType: 10](#_Toc394403783)

[Encoder Usage Example: 10](#_Toc394403784)

[Example 1—use base parameter 10](#_Toc394403785)

[Example 2—use extension parameter 11](#_Toc394403786)

[Decoder Usage: 13](#_Toc394403787)

[Step#1: create and destroy the decoder 13](#_Toc394403788)

[Step#2: initialize the decoder 13](#_Toc394403789)

[Step#3: invoker the decoding 13](#_Toc394403790)

[Step#4: control the decoding 14](#_Toc394403791)

[Decoder Option List: 15](#_Toc394403792)

[Decoder Parameter List: 16](#_Toc394403793)

[Decoder Usage Example: 16](#_Toc394403794)

Revisions history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Description |
| 12/23/2013 | 0.1 | Sijia Chen, Wayne Liu | Initial version |
| 12/25/2013 | 0.2 | Sijia Chen | Add encoder return value explanation |
| 12/25/2013 | 0.3 | Sijia Chen | Add explanation on range of some parameters in encoder |
| 03/04/2014 | 0.4 | Karina Li | Interface and parameters update |
| 05/23/2014 | 1.0 | Wayne Liu, Karina Li | Update for v1.0 release |
| 07/18/2014 | 1.1 | Karina Li | Add encoder usage example  Add option and parameters description |
| 10/28/2014 | 1.2 | Sijia Chen | Update API changes |

# Encoder Interface Usage:

## Step#1: create and destroy the encoder

int WelsCreateSVCEncoder(ISVCEncoder\*\* ppEncoder);

void WelsDestroySVCEncoder(ISVCEncoder\* pEncoder);

## Step#2: initialize the encoder

/\*

Initilaize encoder by using base parameters.

\*/

virtual int Initialize (const SEncParamBase\* pParam) = 0;

/\*

Initilaize encoder by using extension parameters. If the user needs to set the more details, refer to use this interface.

\*/

virtual int InitializeExt (const SEncParamExt\* pParam) = 0;

/\*

Get the default extension parameters . some times the user doesn’t care about all of the parameters. So he can get use the interface to get the default parameter, then updates these parameters he cares.

\*/

virtual int GetDefaultParams (SEncParamExt\* pParam) = 0;

virtual int Unintialize() = 0;

## Step#3: invoker the encoding

/\*

\* return: 0 - success; otherwise - failed;

\*/

virtual int EncodeFrame(const SSourcePicture\* kpSrcPic, SFrameBSInfo\* pBsInfo) = 0;

SSourcePicture:

/\*current input frame info \*/

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| int | iColorFormat | the input image color space format, currently only supports **videoFormatI420** |
| int | iStride[4] | The stride of picture buffer |
| unsigned char | pData[4] | Pointer to the source data |
| int | iPicWidth | width of picture in luminance samples |
| int | iPicHeight | height of picture in luminance samples |
| Long long | uiTimeStamp | Time stamp of the frame. |

// kpSrc = the pointer to the source luminance plane

// chrominance data:

// CbData = kpSrc + m\_iMaxPicWidth \* m\_iMaxPicHeight;

// CrData = CbData + (m\_iMaxPicWidth \* m\_iMaxPicHeight)/4;

//the application calling this interface needs to ensure the data validation between the location of [kpSrc, kpSrc+framesize-1]

SFrameBSInfo:

/\*output bitstream info\*/

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| int | iTemporalId | Temporal layer ID |
| int | iSubSeqId | The ID number of subsequence. refer to D.2.11 Sub-sequence information SEI message semantics |
| Int | iLayerNum | The number of layers |
| SLayerBSInfo | sLayerInfo[MAX\_LAYER\_NUM\_OF\_FRAME]; | Each layer bitstream information |
| EVideoFrameType | eFrameType | Output frame type |
| Long long | uiTimeStamp | Time stamp of the frame. |

SLayerBSInfo:

/\*each layer output bitstream info\*/

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| unsigned char | uiTemporalId | Temporal layer ID |
| unsigned char | uiSpatialId | Spatial layer ID |
| unsigned char | uiQualityId | Quality layer ID |
| unsigned char | uiPriorityId | This frame priority ID |
| unsigned char | uiLayerType | The current layer type |
| int | iNalCount | The number of nal in this layer |
| int | iNalLengthInByte[MAX\_NAL\_UNITS\_IN\_LAYER] | Each nal length info in current layer |
| unsigned char\* | pBsBuf | Pointer to the buffer of bitstream |

## Step#4: control the encoding

The upper layer application should ensure threading safety between calling these control interfaces and calling the encoding.

/\*

\* return: 0 - success; otherwise - failed;

\*/

virtual int ForceIntraFrame(bool bIDR) = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* InDataFormat, IDRInterval, SVC Encode Param, Frame Rate, Bitrate,..

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* return: CM\_RETURN: 0 - success; otherwise - failed;

\*/

virtual int SetOption(ENCODER\_OPTION eOptionId, void\* pOption) = 0;

virtual int GetOption(ENCODER\_OPTION eOptionId, void\* pOption) = 0;

# Encoder Option List:

int CWelsH264SVCEncoder::SetOption(ENCODER\_OPTION eOptionId, void\* pOption)

|  |  |  |
| --- | --- | --- |
| Option ID | Input Format | Meaning/Constraint |
| ENCODER\_OPTION\_SVC\_ENCODE\_PARAM\_BASE | Structure of Base Param |  |
| ENCODER\_OPTION\_SVC\_ENCODE\_PARAM\_EXT | Structure of Extension Param |  |
| ENCODER\_OPTION\_IDR\_INTERVAL | int | IDR period,  0/-1 means no Intra period (only the first frame)  >0 means the desired IDR period, must be multiple of (2^temporal\_layer) |
| ENCODER\_OPTION\_FRAME\_RATE | float | Maximal input frame rate, current supported range:  MAX\_FRAME\_RATE = 30  MIN\_FRAME\_RATE = 1 |
| ENCODER\_OPTION\_BITRATE | int |  |
| ENCODER\_OPTION\_MAX\_BITRATE | int |  |
| ENCODER\_PADDING\_PADDING | int | 0:disable padding;1:padding |
| ENCODER\_OPTION\_PROFILE | SProfileInfo | Assgin the profile for each layer |
| ENCODER\_OPTION\_LEVEL | SLevelInfo | Assgin the level for each layer |
| ENCODER\_OPTION\_NUMBER\_REF | int | The number of refererence frame |
| ENCODER\_OPTION\_DELIVERY\_STATUS | SDeliveryStatus | The delivery info which is a feedback from app level |
| ENCODER\_LTR\_RECOVERY\_REQUEST | Structure of SLTRRecoverRequest |  |
| ENCODER\_LTR\_MARKING\_FEEDBACK | Structure of SLTRMarkingFeedback |  |
| ENCOCER\_LTR\_MARKING\_PERIOD | Unsigned int |  |
| ENCODER\_OPTION\_LTR | Unsigned int | 0:not enable LTR;>0 enable LTR;  LTR number is fixed to be 2 in current encoder |
| ENCODER\_OPTION\_COMPLEXITY | int (ECOMPLEXITY\_MODE) | LOW\_COMPLEXITY,  MEDIUM\_COMPLEXITY, HIGH\_COMPLEXITY |
| ENCODER\_OPTION\_ENABLE\_PREFIX\_NAL\_ADDING | Bool | false:not use Prefix NAL; true: use Prefix NAL |
| ENCODER\_OPTION\_ENABLE\_SPS\_PPS\_ID\_ADDITION | Bool | false:not adjust ID in SPS/PPS; true: adjust ID in SPS/PPS |
| ENCODER\_OPTION\_CURRENT\_PATH | string |  |
| ENCODER\_OPTION\_DUMP\_FILE | Structure of SDumpLayer | Dump layer reconstruct frame to a specified file |
| ENCODER\_OPTION\_TRACE\_LEVEL | int | Output some info accoding to the trace level |
| ENCODER\_OPTION\_TRACE\_CALLBACK | WelsTraceCallback | The trace ouput callback |
| ENCODER\_OPTION\_TRACE\_CALLBACK\_CONTEXT | Void \* | Context info of trace callback |
| ENCODER\_OPTION\_GET\_STATISTICS | Structure of SEncoderStatistics | Read only, to get the statistics of encoder (partly available) |
| ENCODER\_OPTION\_STATISTICS\_LOG\_INTERVAL | int | The interval of printing out statistics logs with info log, in the unit of milliseconds |
| ENCODER\_OPTION\_IS\_LOSSLESS\_LINK | bool | To set whether the current link is lossless, this may affect the reference selelction strategy used for screen content |

# Encoder Parameter List:

SEncParamBase:

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| int | iUsageType | Currently two type applications are supported.  0: camera video  1: screen content |
| int | iPicWidth | width of picture in luminance samples (the maximum of all layers if multiple spatial layers presents) |
| int | iPicHeight | height of picture in luminance samples((the maximum of all layers if multiple spatial layers presents) |
| int | iTargetBitrate | target bitrate |
| int | iRCMode | Rate control mode |
| float | fMaxFrameRate | maximal input frame rate |

SEncParamExt:

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| int | iUsageType | Currently two type applications are supported.  0: camera video  1: screen content |
| int | iPicWidth | width of picture in luminance samples (the maximum of all layers if multiple spatial layers presents) |
| int | iPicHeight | height of picture in luminance samples((the maximum of all layers if multiple spatial layers presents) |
| int | iTargetBitrate | target bitrate |
| int | iRCMode | Rate control mode |
| float | fMaxFrameRate | maximal input frame rate |
| int | iTemporalLayerNum | temporal layer number, max temporal layer = 4 |
| int | iSpatialLayerNum | spatial layer number,  1<= iSpatialLayerNum <= MAX\_SPATIAL\_LAYER\_NUM  MAX\_SPATIAL\_LAYER\_NUM = 4 |
| SSpatialLayerConfig | sSpatialLayers[MAX\_SPATIAL\_LAYER\_NUM]; |  |
| int | iTemporalLayerNum | temporal layer number, max temporal layer = 4 |
| int | iSpatialLayerNum | spatial layer number,  1<= iSpatialLayerNum <= MAX\_SPATIAL\_LAYER\_NUM  MAX\_SPATIAL\_LAYER\_NUM = 4 |
| unsigned int | uiIntraPeriod | period of IDR frame |
| int | iNumRefFrame | The number of the reference frame |
| unsigned int | iComplexityMode | Complexity preference of encoder (in plan) |
| bool | bEnableSpsPpsIdAddition | false:not adjust ID in SPS/PPS; true: adjust ID in SPS/PPS |
| bool | bPrefixNalAddingCtrl | false:not use Prefix NAL; true: use Prefix NAL |
| bool | bEnableSSEI | false:not use SSEI; true: use SSEI |
| int | iPaddingFlag | 0:disable padding;1:padding |
| int | iEntropyCodingModeFlag | 0:CAVLC 1:CABAC.  Currently only supports CAVLC. |
| bool | bEnableRc | False: don’t use rate control; true: use rate control |
| bool | bEnableFrameSKip | False: don’t skip frame even if VBV buffer overflow.  True: allow skipping frames to keep the bitrate within limits |
| int | iMaxBitrate | the maximum bitrate |
| int | iMaxQp | the maximum QP encoder supports |
| int | iMinQp | The minmum QP encoder supports |
| Unsigned int | uiMaxNalSize | The maximum NAL size. This value should be not 0 for dynamic slice mode, this value should be 0 with other slice modes |
| bool | bEnableLongTermReference; | 1: on, 0: off |
| int | iLTRRefNum | The number of LTR(long term reference) (in plan, arbitrary setting of this number is in plan, there is default number under different iUsageType) |
| int | iLtrMarkPeriod | The LTR marked period that is used in feedback. |
| Short | iMultipleThreadIdc | # 0: auto(dynamic imp. internal encoder); 1: multiple threads imp. disabled; > 1: count number of threads; |
| Int | iLoopFilterDisableIdc | 0: on, 1: off, 2: on except for slice boundaries |
| Int | iLoopFilterAlphaC0Offset | AlphaOffset: valid range [-6, 6], default 0 |
| Int | iLoopFilterBetaOffset | BetaOffset: valid range [-6, 6], default 0 |
| bool | bEnableDenoise; | denoise control |
| bool | bEnableBackgroundDetection | background detection control |
| bool | bEnableAdaptiveQuant | adaptive quantization control |
| bool | bEnableFrameCroppingFlag; | enable cropping source picture |
| int | bEnableSceneChangeDetect | Enable scene change detection |
| bool | bIsLosslessLink | To set whether the current link is lossless, this may affect the reference selelction strategy used for screen content |

SSpatialLayerConfig

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| int | iVideoWidth | width of picture in luminance samples of a layer |
| int | iVideoHeight | height of picture in luminance samples of a layer |
| float | fFrameRate | frame rate for a layer |
| int | iSpatialBitrate | target bitrate for a spatial layer |
| int | iMaxSpatialBitrate | Maximum bitrate for a spatial layer |
| EProfileIdc | uiProfileIdc | value of profile IDC (0 for auto-detection) |
| ELevelIdc | uiLevelIdc | value of level IDC (0 for auto-detection) |
| Int | iDLayerQp | Each layer QP for fixed quant case |
| SSliceConfig | sSliceCfg | slicing configuration |
|  | sSliceCfg.uiSliceMode | 0: SM\_SINGLE\_SLICE: SliceNum==1  1: SM\_FIXEDSLCNUM\_SLICE: according to SliceNum, Enabled dynamic slicing for multi-thread  2: SM\_RASTER\_SLICE: according to SlicesAssign, Need input of MB numbers each slice. In addition, if other constraint in SSliceArgument is presented, need to follow the constraints. Typically if MB num and slice size are both constrained, re-encoding may be involved.  3: SM\_ROWMB\_SLICE: according to PictureMBHeight, a row of mbs per slice  4: SM\_DYN\_SLICE: according to SliceSize, Dynamic slicing (have no idea about slice\_nums until encoding current frame) |
|  | sSliceCfg. sSliceArgument uiSliceMbNum | Used in uiSliceMode=2 |
|  | sSliceCfg. sSliceArgument uiSliceNum | Used in uiSliceMode=1 |
|  | sSliceCfg. sSliceArgument uiSliceSizeConstraint | Used in uiSliceMode=4 |

# Encoder Return Value: CM\_RETURN

Return value of parameter Initializtion: Initialize() and set option SetOption()

|  |  |  |
| --- | --- | --- |
| Value | Parameter Name | Meaning/Constraint |
| 0 | cmResultSuccess | Successful initialized |
| 1 | cmInitParaError | Found error in input parameters |
| 2 | cmMachPerfIsBad | Not supported yet |
| 3 | cmUnkonwReason | Not supported yet |
| 4 | cmMallocMemeError | Input source picture is NULL |
| 5 | cmInitExpected | Encoder is not created correctly |

# Encoder Return Value: EVideoFrameType:

Return value of encode one frame: EncodeFrame()

|  |  |  |
| --- | --- | --- |
| Value | Parameter Name | Meaning/Constraint |
| 0 | videoFrameTypeInvalid | Encoder not ready or parameters are invalidate |
| 1 | videoFrameTypeIDR | IDR frame in H.264 |
| 2 | videoFrameTypeI | I frame type |
| 3 | videoFrameTypeP | P frame type |
| 4 | videoFrameTypeSkip | Encoder decides to skip the frame, no bit stream will be ouputed |
| 5 | videoFrameTypeIPMixed | A frame where I and P slices are mixing, not supported yet |

# Encoder Usage Example:

## Example 1—use base parameter

**Step1:setup encoder**

int rv = WelsCreateSVCEncoder (&encoder\_);

ASSERT\_EQ (0, rv);

ASSERT\_TRUE (encoder\_ != NULL);

**Step2: initilize by using base parameter**

SEncParamBase param;

memset (&param, 0, sizeof (SEncParamBase));

param.iUsageType = usageType;

param.fMaxFrameRate = frameRate;

param.iPicWidth = width;

param.iPicHeight = height;

param.iTargetBitrate = 5000000;

encoder\_->Initialize (&param);

**Step3: set option, you aslo call SetOption during encoding**

**//give a example**

encoder\_->SetOption (ENCODER\_OPTION\_TRACE\_LEVEL, &g\_LevelSetting);

**Step4: encoding and store ouput bistream**

int frameSize = width \* height \* 3 / 2;

BufferedData buf;

buf.SetLength (frameSize);

ASSERT\_TRUE (buf.Length() == (size\_t)frameSize);

SFrameBSInfo info;

memset (&info, 0, sizeof (SFrameBSInfo));

SSourcePicture pic;

memset (&pic, 0, sizeof (SsourcePicture));

pic.iPicWidth = width;

pic.iPicHeight = height;

pic.iColorFormat = videoFormatI420;

pic.iStride[0] = pic.iPicWidth;

pic.iStride[1] = pic.iStride[2] = pic.iPicWidth >> 1;

pic.pData[0] = buf.data();

pic.pData[1] = pic.pData[0] + width \* height;

pic.pData[2] = pic.pData[1] + (width \* height >> 2);

for(int num = 0;num<total\_num;num++) {

//prepare input data

rv = encoder\_->EncodeFrame (&pic, &info);

ASSERT\_TRUE (rv == cmResultSuccess);

if (info.eFrameType != videoFrameTypeSkip && cbk != NULL) {

//output bitstream

}

}

**Step5: teardown encoder**

if (encoder\_) {

encoder\_->Uninitialize();

WelsDestroySVCEncoder (encoder\_);

}

## Example 2—use extension parameter

**The same operation on Step#1,#3,#4,#5 with Example 1—use base parameter**

**Step 2:-- initialize by using extension parameter**

SEncParamExt param;

encoder->GetDefaultParams (&param);

param.iUsageType = usageType;

param.fMaxFrameRate = frameRate;

param.iPicWidth = width;

param.iPicHeight = height;

param.iTargetBitrate = 5000000;

param.bEnableDenoise = denoise;

param.iSpatialLayerNum = layers;

if (sliceMode != SM\_SINGLE\_SLICE && sliceMode != SM\_DYN\_SLICE) //SM\_DYN\_SLICE don't support multi-thread now

param.iMultipleThreadIdc = 2;

for (int i = 0; i < param.iSpatialLayerNum; i++) {

param.sSpatialLayers[i].iVideoWidth = width >> (param.iSpatialLayerNum - 1 - i);

param.sSpatialLayers[i].iVideoHeight = height >> (param.iSpatialLayerNum - 1 - i);

param.sSpatialLayers[i].fFrameRate = frameRate;

param.sSpatialLayers[i].iSpatialBitrate = param.iTargetBitrate;

param.sSpatialLayers[i].sSliceCfg.uiSliceMode = sliceMode;

if (sliceMode == SM\_DYN\_SLICE) {

param.sSpatialLayers[i].sSliceCfg.sSliceArgument.uiSliceSizeConstraint = 600;

param.uiMaxNalSize = 1500;

}

}

param.iTargetBitrate \*= param.iSpatialLayerNum;

encoder\_->InitializeExt (&param);

# Decoder Usage:

## Step#1: create and destroy the decoder

int WelsCreateDecoder(ISVCDecoder\*\* ppDecoder);

void WelsDestroyDecoder(ISVCDecoder\* pDecoder);

## Step#2: initialize the decoder

virtual long Initialize(const SDecodingParam\* pParam) = 0;

virtual long Unintialize() = 0;

## Step#3: invoker the decoding

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Description:

\* Decompress one frame or slice, and output I420 decoded data.

\*

\* Input parameters:

\* Parameter TYPE Description

\* kpSrc const unsigned char\* the h264 stream to be decoded

\* kiSrcLen const int the length of h264 steam

\* ppDst unsigned char\*\* buffer pointer of decoded data (YUV)

\* pDstInfo SBufferInfo\* information provided to API including width, height, etc

\* pStride int\* output stride

\* iWidth int& output width

\* iHeight int& output height

\*

\* return: if decode frame success return 0, otherwise corresponding error returned.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//the following API is for slice level decoding

virtual DECODING\_STATE DecodeFrame2(

const unsigned char\* kpSrc,

const int kiSrcLen,

unsigned char\*\* ppDst,

SBufferInfo\* pDstInfo);

//the following API is for frame level decoding

Virtual DECODING\_STATE DecodeFrame(

const unsigned char\* kpSrc,

const int kiSrcLen,

unsigned char\*\* ppDst,

int\* pStride,

int& iWidth,

int& iHeight );

//Note: DecodeFrameEx() is not used.

//Note: for slice level DecodeFrame2() (4 parameters input), whatever the function return value is, the outputted data of I420 format will only be available when pDstInfo->iBufferStatus == 1,. (e.g., in multi-slice cases, only when the whole picture is completely reconstructed, this variable would be set as 1.)

/\* (in plan, coming soon)

\* This function parse input bitstream only, and rewrite possible SVC syntax to AVC syntax

\* return: 0 - success; otherwise -failed;

\*/

virtual DECODING\_STATE EXTAPI DecodeParser (const unsigned char\* pSrc,

const int iSrcLen,

SParserBsInfo\* pDstInfo) = 0;

## Step#4: control the decoding

virtual int SetOption(DECODER\_OPTION eOptionId, void\* pOption) = 0;

virtual int GetOption(DECODER\_OPTION eOptionId, void\* pOption) = 0;

# Decoder Option List:

|  |  |  |
| --- | --- | --- |
| Option ID | Input Format | Meaning/Constraint |
| DECODER\_OPTION\_DATAFORMAT | int | color format, now supports 23 only (I420) |
| DECODER\_OPTION\_END\_OF\_STREAM | bool | end of stream flag |
| DECODER\_OPTION\_VCL\_NAL | bool | feedback whether or not have VCL NAL in current AU for application layer |
| DECODER\_OPTION\_TEMPORAL\_ID | int | if have VCL NAL in current AU, then feedback the temporal ID |
| DECODER\_OPTION\_FRAME\_NUM | int | Indicate frame\_num |
| DECODER\_OPTION\_IDR\_PIC\_ID | int | Indicate current IDR\_ID |
| DECODER\_OPTION\_LTR\_MARKING\_FLAG | bool | read only, indicating if LTR\_marking SE flag is used in current AU |
| DECODER\_OPTION\_LTR\_MARKED\_FRAME\_NUM | int | Read only, indicating the frame\_num of current AU marked as LTR |
| DECODER\_OPTION\_ERROR\_CON\_IDC | int | Indicate error concealment method  0: disable  1: frame\_copy  2: slice\_copy (default) |
| DECODER\_OPTION\_TRACE\_LEVEL | int | Output some info accoding to the trace level |
| DECODER\_OPTION\_TRACE\_CALLBACK | WelsTraceCallback | The trace ouput callback |
| DECODER\_OPTION\_TRACE\_CALLBACK\_CONTEXT | Void \* | Context info of trace callback |
| DECODER\_OPTION\_GET\_STATISTICS | Structure of SDecoderStatistics | Read only, to get the decoder statistics (in plan) |

# Decoder Parameter List:

(Note: some parameters in mentioned structure are not explained because they are to be removed)

SDecodingParam: (Note: some of the members may not be used for now.)

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| char\* | pFileNameRestructed | File name of restructed frame used for PSNR calculation based debug; |
| EVideoFormatType | eOutputColorFormat | color space format to be outputed |
| unsigned int | uiCpuLoad | CPU load |
| unsigned char | uiTargetDqLayer | target dq layer number |
| ERROR\_CON\_IDC | eEcActiveIdc | Whether active error concealment feature in decoder |
| SVideoProperty | sVideoProperty | Video stream property |

SVideoProperty

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| unsigned int | size | size of structure |
| VIDEO\_BITSTREAM\_TYPE | eVideoBsType | Video stream type (AVC/SVC) |

SBufferInfo:

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| int | iBufferStatus | 0: data not ready, 1: data ready |
| union UsrData |  | Output buffer info, see following tables |

union UsrData:

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| SSysMEMBuffer | sSysMEMBuffer | output with memory, see following tables |

SSysMEMBuffer

|  |  |  |
| --- | --- | --- |
| Format | Parameter Name | Meaning/Constraint |
| int | iWidth | width of decoded pic |
| int | iHeight | height of decoded pic |
| int | iFormat | type is “EVideoFormatType”, see codec\_def.h |
| int | iStride[2] | stride |

# Decoder Usage Example:

A dummy process for using the decoder could be: (for static library)

1. ISVCDecoder \*pSvcDecoder; //declare a decoder

unsigned char \*pBuf = …; //input: encoded bitstream start position; should include start code prefix

int iSize = …; //input: encoded bitsteam length; should include the size of start code prefix

unsigned char \*pData[3] = …; //output: [0~2] for Y,U,V buffer

SBufferInfo sDstBufInfo; //in-out: declare the output buffer info

memset(&sDstBufInfo, 0, sizeof(SBufferInfo));

2. CreateDecoder(pSvcDecoder); //create a decoder

3. SDecodingParam sDecParam = {0}; //declare required param

sDecParam.sVideoProperty.eVideoBsType = VIDEO\_BITSTREAM\_AVC;

4. Initialize(&sDecParam); //initialize the param and decoder context, allocate memory

5. iRet = DecodeFrame2(pBuf, iSize, pData, &sDstBufInfo); //do actual decoding for slice level; this can be done in a loop until data ends

If (iRet != 0) //Decode failed

{RequestIDR or something like that.}

if (sDstBufInfo.iBufferStatus==1) //pData can be used for render.

{ output pData[0], pData[1], pData[2]; }

6. Uninitialize(); //Uninitialize the decoder and memory free.

7. DestroyDecoder(); //Destroy the decoder