DTAPI

ENCODER CONTROL

REFERENCE

Nov 2020

Table of Contents

1. Tutorial

1.1. Introduction

To control DekTec encoder cards, three main classes are used:

1. **`DtEncPars`** for specifying the encoding parameters. It has many subclasses containing video encoding parameters, audio encoding parameters, etc.
2. **`DtEncControl`** for controlling the encoder, mainly applying parameters and starting/stopping encoding.
3. **`DtInpChannel`** for reading the encoded transport stream into an application.

This tutorial demonstrates the usage of the DTAPI encoder-control classes by example programs and code snippets. To keep the examples short, error-checking code is omitted. In production-quality code it is obviously very important to check the result of every DTAPI call.

1.2. Encoder Ports

User applications can access the encoding functionality accessed attaching a class to an encoder port, and calling methods on this class. The table below provides an overview of port numbers available on DekTec encoder cards.

Type	Port	Description	DTAPI classes
`DTA-2180`	1	SDI input	Physical port that cannot be used from DTAPI
	2	HDMI input	Physical port that cannot be used from DTAPI
	3	ASI output	Physical port that cannot be used from DTAPI
	4	Encoder	**`DtEncControl`** for controlling the encoder **`DtInpChannel`** for reading the encoded stream
`DTA-2182`	1	SDI input	Physical port that cannot be used from DTAPI
	2	SDI input	Physical port that cannot be used from DTAPI
	3	ASI output	Physical port that cannot be used from DTAPI
	4	ASI output	Physical port that cannot be used from DTAPI
	5	Encoder	**`DtEncControl`** for controlling the encoder **`DtInpChannel`** for reading the encoded stream
	6	Encoder	**`DtEncControl`** for controlling the encoder **`DtInpChannel`** for reading the encoded stream

1.3. Hello Encoder!

The example below shows a minimal code example to run a DTA-2180 encoder with SDI input, H.264 video encoding and no audio.

```
#include "DTAPI.h"

int main()
{
   // Set up encoding parameters (1)
   DtEncPars  EncPars(2180);                                    // Step 1A
   EncPars.m_VidPars.m_VidStd = DTAPI_VIDSTD_1080I50;           // Step 1B
   EncPars.m_VidPars.SetVidEncStd(DT_VIDENCSTD_H264);           // Step 1C
   EncPars.m_VidPars.SetDefaultsForProfileLevel(                // Step 1D
                        DtEncVidParsH264::PROFILE_HIGH, DtEncVidParsH264::LEVEL_AUTO);

   // Attach DtDevice object and encoder-control object to the DTA-2180 (2)
   DtDevice  Dta2180;
   DtEncControl  EncControl;
   Dta2180.AttachToType(2180);
   EncControl.AttachToPort(&Dta2180, 4);

   // Wait for the encoder to be initialized (3)
   EncControl.WaitForInitialized(20000);

   // Set encoding parameters (4)
   EncControl.SetEncPars(EncPars);

   // Start encoding (5)
   EncControl.SetOperationalState(DtEncControl::OS_RUN);

   while (!StopSignal)
      ;
}
```

The encoded output will be available on the ASI output. Simultaneously, your application can read the transport stream for further processing (e.g. forwarding to IP) from port 4. Please refer to §1.4.4 for example code.

Let's examine this code example step by step.

Step 1. Setting Up Encoding Parameters

```
   // Set up encoding parameters (1)
   DtEncPars  EncPars(2180);                                    // Step 1A
   EncPars.m_VidPars.m_VidStd = DTAPI_VIDSTD_1080I50;           // Step 1B
   EncPars.m_VidPars.SetVidEncStd(DT_VIDENCSTD_H264);           // Step 1C
   EncPars.m_VidPars.SetDefaultsForProfileLevel(                // Step 1D
                        DtEncVidParsH264::PROFILE_HIGH, DtEncVidParsH264::LEVEL_AUTO);
```

Encoding parameters are set up in object **`EncPars`** of type **`DtEncPars`**, which is the top-level DTAPI type to specify encoding parameters. The sub-steps in this example show a bare basics method to initialize the encoding parameter in a meaningful way.

This step consists of sub-steps 1A to 1D:

1A. First **EncPars** is constructed with the type number of the encoding hardware (2180 for DTA-2180) as parameter, so that **DtEncPars** "knows" what type of encoder hardware the parameters are targeted for. This is required, amongst others, so that **DtEncPars::CheckValidity**() can check whether the encoding parameters are valid on the target hardware.
The **EncPars** object is initialized with default parameters that make sense for the selected hardware, in this case the DTA-2180. By default the SDI port (port# 1) is selected as input.

1B. The "expected" input video standard is set to 1080i50 (note that the video standard must be set before setting the video *encoding* standard).
This is an important concept: DekTec video encoders do not automatically follow the video format present at the encoder input. Instead the video standard has to be specified explicitly in *m_VidStd*.
If the input format is different from the specified format, the encoding result is undefined.

1C. The video encoding standard is set to H.264. **SetVidEncStd** will set the value of the other parameters to reasonable defaults. To be able to do this, *m_VidStd* must be initialized before setting the video encoding standard.

1D. The video encoding parameters are set to defaults that are suitable for High Profile (HP), automatic level. You can also explicitly specify a level, but typically it is more convenient to use the automatic level setting.

The video encoding parameters have now been set up to reasonable values for H.264 encoding of a 1080i50 stream. No audio will be encoded. The multiplexing parameters such as PIDs and repetition rates are all set to defaults.

Step 2. Attach DtDevice and Encoder Control

```
// Attach DtDevice object and encoder-control object to the DTA-2180 (2)
DtDevice  Dta2180;
DtEncControl  EncControl;
Dta2180.AttachToType(2180);
EncControl.AttachToPort(&Dta2180, 4);
```

Similarly to using other DekTec hardware, a **DtDevice** object has to be attached the hardware, in this case the DTA-2180 HD H.264 encoder card. Then a **DtEncControl** has to be attached to port 4, which represents the audio/video encoder. Note that the SDI input is port 1 and the HDMI input is port 2.

Step 3. Wait for Encoder Initialized

```
// Wait for the encoder to be initialized (3)
EncControl.WaitForInitialized(20000);
```

Encoder hardware is a subsystem with a local processor that has a certain boot time before it is operational. The **WaitForInitialized** method must be called to wait until the encoder is booted and initialized completely. It is very important to not forget this step, otherwise other encoder-control calls may fail randomly.

Step 4. Set Encoding Parameters

```
// Set encoding parameters (4)
EncControl.SetEncPars(EncPars);
```

Apply the encoder parameters. At this stage, the encoder is still in the idle state, so the effect is just to preload the initial parameters.

Step 5. Start Encoding

```
// Start encoding (5)
EncControl.SetOperationalState(DtEncControl::OS_RUN);
```

Start encoding using the just uploaded parameters. The encoder will now start encoding and continue to do so until stopped. Please note that if your application quits, automatic close and detach operations are executed, which will also stop the encoder.

1.4. Example Code Snippets

This section exemplifies several aspects of specifying encoding parameters and controlling the encoder.

1.4.1. Custom Video-Encoding Parameters

If you want to specify custom video-encoding parameters, DekTec recommends to start with default parameters, and overwrite parameters you want to change with your own values.

```
// Set up default encoding parameters
DtEncPars  EncPars(2180);
EncPars.m_VidPars.m_VidStd = DTAPI_VIDSTD_625I50;
EncPars.m_VidPars.SetVidEncStd(DT_VIDENCSTD_H264);          // Or DT_VIDENCSTD_MP2V
EncPars.m_VidPars.SetDefaultsForProfileLevel(…);

// Change a generic video-encoding parameter:
// Set the aspect ratio to 4:3
EncPars.m_VidPars.m_AspectRatio = DT_AR_4_3;

// Change a H.264-specific video-encoding parameter:
// Increase bitrate from the default to 10Mbps for higher quality video
EncPars.m_VidPars.H264()->m_Bitrate = 10*1000*1000;

// Check integrity of video-encoding parameters
ASSERT(EncPars.CheckValidity()== DT_ENC_OK);
```

Checking the validity after building encoding parameters can be very helpful as a debugging aid. If some mistake has been made you can use the error code returned by **`CheckValidity`** to get a clue on what went wrong.

1.4.2. Selecting SDI- or HDMI Input

The encoder input can be selected with **`EncPars::m_SourcePort`**. It's best to set the source port before setting other parameters, as certain restrictions may be dependent on the input

```
DtEncPars  EncPars(2180);

EncPars.m_SourcePort = 2;                    // Port 2 is the HDMI input

EncPars.m_VidPars.m_VidStd = DTAPI_VIDSTD_1080I50;
EncPars.m_VidPars.SetVidEncStd(DT_VIDENCSTD_H264);
EncPars.m_VidPars.SetDefaultsForProfileLevel(…);

// etc.
```

1.4.3. Encoding Audio

The code below shows an example of how to add a stereo service encoded with HE-AAC to the encoding parameters.

```
// Encoding parameters with basic initialization
DtEncPars  EncPars;
EncPars.SetEncType(2180);
…

// Add audio-encoding parameters in m_AudPars[0]
EncPars.m_AudPars[0].m_Enable = true;

// Set to AAC stereo audio encoding
EncPars.m_AudPars[0].SetAudEncStd(DT_AUDENCSTD_AAC, DtEncAudPars::SVC_STEREO);

// Use audio channels 5 and 6 as input
EncPars.m_AudPars[0].m_AudChans.push_back(5);
EncPars.m_AudPars[0].m_AudChans.push_back(6);

// Set generic audio-encoding parameters
EncPars.m_AudPars[0].m_Bitrate = 192000;                    // 192kbps
EncPars.m_AudPars[0].m_Delay = 0;                           // No delay compensation
EncPars.m_AudPars[0].m_SampleRate = 48000;                  // 48kHz delay

// Set AAC-specific audio-encoding parameters:
//  - Select AAC-HE
EncPars.m_AudPars[0].Aac()->m_Profile = DtEncAudParsAac::AAC_HE;

// Check that we haven't made elementary mistakes
ASSERT(EncPars.CheckValidity()== DT_ENC_OK);
```

Similarly, you can define audio-encoding parameters for a second audio component by enabling and initializing *`m_AudPars`*[1], etc.

NOTE: The DTA-2180 can only encode audio together with video. It is not possible to encode a "radio service" without video.

1.4.4. Reading the Encoded Transport Stream

The encoded transport stream is always available on the DVB-ASI output port. You can also read the encoded transport stream into your application for further processing, e.g. for forwarding to IP, by attaching an input channel to port 4. This is illustrated in the code example below.

```
// DtDevice object for DTA-2180
DtDevice  Dta2180;

// Assumption: Dta2180 is attached to hardware and encoder is initialized

// Attach input channel for reading the transport stream
DtInpChannel  DtInpChan;
DtInpChan.AttachToPort(&Dta2180, 4);

// Start receiving
DtInpChan.SetRxControl(DTAPI_RXCTRL_RCV);

// Loop until stopped via QuitFlag (can be set from another thread)
bool  QuitFlag = false;
while (!QuitFlag)
{
   // Read FIFO load
   int  FifoLoad;
   DtInpChan.GetFifoLoad(FifoLoad);
   if (FifoLoad >= 1024)
   {
      // If sufficient load is available, read and process 1024 bytes of data
      char  TsData[1024];
      DtInpChan.Read(TsData, 1024);
      Process(TsData);
   } else {
      // Insufficient FIFO load => Sleep a while and try again
      ::Sleep(50);
   }
}
```

2. Video Encoding

2.1. Video Encoding Standards

The video encoding standards supported by DekTec encoders are defined by enumeration **DtVidEncStd**:

```
enum DtVidEncStd                    // Video encoding standard
{
   DT_VIDENCSTD_UNKNOWN,            // Unknown or not defined yet
   DT_VIDENCSTD_MP2V,               // H.262 (MPEG-2 video)
   DT_VIDENCSTD_H264,               // H.264 (AVC)
   DT_VIDENCSTD_H265                // H.265 (HEVC) (not supported yet)
};
```

H.265 (HEVC) is already defined as a video-encoding standard, but not currently supported by DekTec encoder hardware.

2.2. Profiles and Levels

Both MPEG-2 video and H.264 are video encoding standards that support a wide range of applications from mobile to high-quality UHD editing. For most applications, it is unpractical and unnecessary to support the full standard. To address this problem, *Profiles* and *Levels* have been introduced, defining subsets of the standards.

Profile	Defines a set of coding features or 'tools'.
Level	Limits the memory and processing power needed for decoders. It defines maxima for bit-stream attributes like bitrate, frame size, etc.

The combination of profile and level targets a specific application, or a class of applications. For example, Main Profile, Main Level (MP@ML) for MPEG-2 video is aimed at direct-to-home broadcasting of SD television signals. Similarly, High Profile for H.264 is a profile used for broadcasting of HDTV.

From an encoder point of view, profile and level specify a set of limits on the values that may be taken by the video encoding parameters and by certain bit-stream attributes.

2.2.1. H.264 Profiles and Levels

The following H.264 profiles are supported by DekTec hardware.

Profile	Abbr.	Target Application
Constrained Baseline Profile	**CBP**	Low-cost applications, this profile is typically used in videoconferencing and mobile applications.
Main Profile	**MP**	Broadcasting and consumer storage of SDTV.
High Profile	**HP**	Broadcasting and consumer storage of HDTV.

The table below lists the coding features supported for each of the supported H.264 profiles.

Feature	CBP	MP	HP
Bit depth	8 bits	8 bits	8 bits
Chroma formats	4:2:0	4:2:0	4:2:0
MBAFF coding	-	✓	✓
B pictures	-	✓	✓
CABAC coding	-	✓	✓
8x8 Transform	-	-	✓
Weighted prediction	-	✓	✓

The following constraints apply to the H.264 levels. Only levels supported by DekTec encoders are listed in this table.

Level	Max. decoding speed (macroblocks/s)	Max. frame size (macroblocks)	Max. video bitrate	
			HP	CBP, MP
AUTO	Level constraints are ignored			
3	40,500	1,620	12.5 Mbit/s	10 Mbit/s
3.1	108,000	3,600	17.5 Mbit/s	14 Mbit/s
3.2	216,000	5,120	25 Mbit/s	20 Mbit/s
4	245,760	8,192	25 Mbit/s	20 Mbit/s
4.1	245,760	8,192	62.5 Mbit/s	50 Mbit/s

2.2.2. H.264 Profiles and Levels – Defaults

For H.264, method **`DtVidEncPars::SetDefaultsForProfileLevel`** can be used to set default video-encoding parameters for a given video standard, H.264 profile and H.264 level.

In the tables below, **HorRes** means "horizontal rescale", **8x8** means "use 8x8 transforms" and **WP** means "weighted prediction".

H.264 – Constrained Baseline Profile (CBP) – Defaults

Format	Levels	HorRes	Bitrate	#B Pictures	8x8	CABAC	WP
SD	3, 3.1, 3.2, 4.0, 4.1, auto	-	3.5Mbps	0	-	-	-
HD	3, 3.1, 3.2	HD is not allowed in this level **`SetDefaultsForProfileLevel`** returns **`DT_ENC_E_INV_VIDSTD`**					
HD	4.0, 4.1, auto	-	7Mbps	0	-	-	-

H.264 – Main Profile (MP) – Defaults

Format	Levels	HorRes	Bitrate	#B Pictures	8x8	CABAC	WP
SD	3, 3.1, 3.2, 4.0, 4.1, auto	-	3.5Mbps	2	-	✓	✓
HD	3, 3.1, 3.2	HD is not allowed in this level `SetDefaultsForProfileLevel` returns `DT_ENC_E_INV_VIDSTD`					
HD	4.0, 4.1, auto	-	7Mbps	0	-	✓	✓

H.264 – High Profile (HP) – Defaults

Format	Levels	HorRes	Bitrate	#B Pictures	8x8	CABAC	WP
SD	3, 3.1, 3.2, 4.0, 4.1, auto	-	3.5Mbps	2	✓	✓	✓
HD	3, 3.1, 3.2	HD is not allowed in this level `SetDefaultsForProfileLevel` returns `DT_ENC_E_INV_VIDSTD`					
HD	4.0, 4.1, auto	-	7Mbps	2	✓	✓	✓

2.2.3. MPEG-2 Video Profiles and Levels

The following MPEG-2 video profiles are supported by DekTec encoder hardware.

Profile	Abbr.	Description
Simple Profile	SP	Low-cost applications, this profile is typically used in vide-oconferencing and mobile applications.
Main Profile	MP	Broadcasting and consumer storage of SDTV.
High Profile	HP	Broadcasting and consumer storage of HDTV.

The table below lists the coding features supported for each of the supported MPEG-2 profiles.

Feature	SP	MP	HP
Aspect ratios	4:3, 16:9	4:3, 16:9	4:3, 16:9
Bit depth	8 bits	8 bits	8 bits
Chroma formats	4:2:0	4:2:0	4:2:0
B pictures	-	✓	✓
Intra DC precision	8, 9, 10	8, 9, 10	8, 9, 10, 11

The following constraints apply to the MPEG-2 video levels.

Level	Frame rates (Hz)	Max. frame width / height	Max. luminance samples/s	Max. video bitrate
auto	Level constraints are ignored			
Main (ML)	23.976, 24, 25, 29.97, 30	720 / 576	High profile: 14,475,600 All other: 10,368,000	15 Mbit/s
High (HL)	23.976, 24, 25, 29.97, 30, 50, 59.94, 60	1920 / 1152	High profile: 83,558,400 All other: 62,668,800	80 Mbit/s
High 1440	23.976, 24, 25, 29.97, 30, 50, 59.94, 60	1440 / 1152	High profile: 62,668,800 All other: 47,001,600	60 Mbit/s

2.2.4. MPEG-2 Profiles and Levels – Defaults

For MPEG-2 video, method **`DtVidEncPars::SetDefaultsForProfileLevel`** can be used to set default video-encoding parameters for a given video standard, MPEG-2 video profile and MPEG-2 video level.

In the tables below, **HorRes** means "horizontal rescale".

MPEG-2 Video – Simple Profile (SP) – Defaults

Format	Levels	HorRes	Bitrate	#B Pictures	Intra-DC Precision
SD	Main, auto	-	7Mbps	0	10 bits
SD	High 1440, High	**`SetDefaultsForProfileLevel`** returns **`DT_ENC_E_UNSUP_PRF_LVL`**			
HD	Main	HD is not allowed in this level **`SetDefaultsForProfileLevel`** returns **`DT_ENC_E_INV_VIDSTD`**			
HD	High 1440	1/2	8.5Mbps	0	10 bits
HD	High, auto	-	17Mbps	0	10 bits

MPEG-2 Video – Main Profile (MP) – Defaults

Format	Levels	HorRes	Bitrate	#B Pictures	Intra-DC Precision
SD	Main, auto	-	7Mbps	1	10 bits
SD	High 1440, High	`SetDefaultsForProfileLevel` returns `DT_ENC_E_UNSUP_PRF_LVL`			
HD	Main	HD is not allowed in this level `SetDefaultsForProfileLevel` returns `DT_ENC_E_INV_VIDSTD`			
HD	High 1440	1/2	8.5Mbps	1	10 bits
HD	High, auto	-	17Mbps	1	10 bits

MPEG-2 Video – High Profile (HP) – Defaults

Format	Levels	HorRes	Bitrate	#B Pictures	Intra-DC Precision
SD	Main, auto	-	7Mbps	1	11 bits
SD	High 1440, High	`SetDefaultsForProfileLevel` returns `DT_ENC_E_UNSUP_PRF_LVL`			
HD	Main	HD is not allowed in this level `SetDefaultsForProfileLevel` returns `DT_ENC_E_INV_VIDSTD`			
HD	High 1440	1/2	8.5Mbps	1	11 bits
HD	High	-	17Mbps	1	11 bits

3. Audio Encoding

3.1. Audio Encoding Standards

The audio encoding standards supported by DekTec encoders are defined by the enumeration type **DtAudEncStd**:

```
enum DtAudEncStd                    // Audio encoding standard
{
   DT_AUDENCSTD_UNKNOWN,            // Unknown or not defined yet
   DT_AUDENCSTD_AAC,                // AAC (AAC-LC or HE-AAC)
   DT_AUDENCSTD_AC3,                // Dolby AC-3
   DT_AUDENCSTD_DOLBY_E,            // Dolby E (pass-through only)
   DT_AUDENCSTD_EAC3,               // Dolby E-AC-3 (pass-through only)
   DT_AUDENCSTD_MP1LII,             // MPEG-1 Layer II
   DT_AUDENCSTD_PCM                 // Direct PCM embedding (SMPTE 302M)
};
```

The table below provides some additional information for each of the supported audio encoding standards:

Value	Meaning
DT_AUDENCSTD_UNKNOWN	The audio standard is not known (yet), or irrelevant.
DT_AUDENCSTD_AAC	AAC encoded audio. To select between AAC-LC, HE-AAC v1 or HE-AAC v2, use parameter **DtEncAudParsAac::***m_Profile*.
DT_AUDENCSTD_AC3	Dolby AC-3 encoded audio.
DT_AUDENCSTD_DOLBY_E	Dolby E. Currently only supported in pass-through mode.
DT_AUDENCSTD_EAC3	Dolby E-AC-3 encoded audio. Currently only supported in pass-through mode.
DT_AUDENCSTD_MP1LII	MPEG-1 Layer II audio encoding.
DT_AUDENCSTD_PCM	Direct embedding of PCM samples without encoding into an MPEG-2 transport stream according to the SMPTE 302M.

3.2. Encoded Audio Pass-Through Mode

For most audio-encoding standards it is possible to bypass the encoder and embed an encoded bit-stream directly from source into the output transport stream, by setting **DtEncAudPars::***m_SvcType* to **SVC_PASSTHROUGH**.

The table below provides additional information about pass-through mode for each of the audio encoding standards.

Value	Description of Pass-through Mode
DT_AUDENCSTD_UNKNOWN	Irrelevant, setting of service type is a “don’t care”.
DT_AUDENCSTD_AAC	Pass-through mode is supported.

	The input bitstream can be de-embedded from the SDI input and should be formatted in compliance with SMPTE 337 *Format for Non-PCM Audio and Data in an AES3 Serial Digital Audio Interface*, data type 10 or 11, and SMPTE ST 2041-3, *Format for Non-PCM Audio in AES3 — MPEG-4 AAC and HE AAC Compressed Digital Audio in ADTS and LATM / LOAS Wrappers*.
DT_AUDENCSTD_AC3	Pass-through mode is supported. The input bitstream can be de-embedded from the SDI input and should be formatted in compliance with SMPTE 337 *Format for Non-PCM Audio and Data in an AES3 Serial Digital Audio Interface*, data type 1, and SMPTE 340 *Format for Non-PCM Audio and Data in AES3 — ATSC A/52 Digital Audio Compression Standard for AC-3 and Enhanced AC-3 Data Types*.
DT_AUDENCSTD_DOLBY_E	Pass-through mode is the only supported service type. The Dolby-E input bitstream can be de-embedded from the SDI input and should be formatted in compliance with SMPTE 337 *Format for Non-PCM Audio and Data in an AES3 Serial Digital Audio Interface*, data type 28.
DT_AUDENCSTD_EAC3	Pass-through mode is the only supported service type. The input bitstream can be de-embedded from the SDI input and should be formatted in compliance with SMPTE 337 *Format for Non-PCM Audio and Data in an AES3 Serial Digital Audio Interface*, data type 16, and SMPTE 340 *Format for Non-PCM Audio and Data in AES3 — ATSC A/52 Digital Audio Compression Standard for AC-3 and Enhanced AC-3 Data Types*.
DT_AUDENCSTD_MP1LII	Pass-through mode is supported. The input bitstream can be de-embedded from the SDI input and should be formatted in compliance with SMPTE 337 *Format for Non-PCM Audio and Data in an AES3 Serial Digital Audio Interface*, data type 5, and SMPTE ST 2041-1, *Format for Non-PCM Audio in AES3 — MPEG Layer I, II, and III Audio*
DT_AUDENCSTD_PCM	Pass-through mode is not supported.

3.3. Maximum Number of Encoded Audio Services

3.3.1. DTA-2180

The table below lists the maximum number of audio services that can be encoded with the DTA-2180. In this table, "non-surround services" stands for mono, stereo or dual-mono audio services.

	Maximum Configuration #1	Maximum Configuration #2	Maximum Configuration #3
Number of surround services	2	1	0
Number of non-surround services	2	5	8

For example, if one 5.1 surround-sound service is used, 5 additional stereo pairs can be encoded.

If the number of surround-sound services is exceeded (more than 2), **DtEncPars::CheckValidity** will return error code **DT_ENC_E_EXC_NUMNONSURROUND**. If the number of surround-sound services is valid (2 or less), but the maximum configuration for the given number of surround-sound services is not met, error code **DT_ENC_E_EXC_NUMSURROUND** is returned.

If the HDMI-input is used, all audio services must use the same audio channels otherwise, **DT_ENC_E_INV_AUDCHANCONFIG** is returned.

3.4. Dolby Metadata

The default values mentioned in the descriptions below are the initial values assigned in the constructor of DekTec data structures that contain Dolby metadata members, e.g. **DtEncAudParsAc3**.

3.4.1. Dolby Digital Metadata Parameters

This section describes the metadata parameters for Dolby Digital (AC-3) audio encoding.

NOTES:

- If a parameter is suffixed with "(2)", e.g. *compr*(2), it means that parameter '2' applies to the second channel if the service type is dual mono (**SVC_DUAL_MONO**). For other service types, parameter '2' is not used.
- Parameters that are part of the extended BSI are marked: "Part of extended BSI". See description of the *xbsie* metadata parameter for more information.

acmod – Audio Coding Mode

This parameter indicates which of the main channels are in use, ranging from 3/2 to 1/0. If the MSB of *acmod* is '1', surround channels are in use and *surmixlev* follows in the bitstream. In the DekTec encoding parameters, *acmod* is called 'service type' and it is encoded in member **DtEncAudPars**::*m_SvcType*.

acmod	Mode	#Channels	Channel Ordering	DekTec Service Type
0	1+1	2	Ch1, Ch2	**SVC_DUAL_MONO**
1	1/0	1	C	**SVC_MONO**
2	2/0	2	L, R	**SVC_STEREO**
3	3/0	3	L, C, R	Not supported
4	2/1	3	L, R, S	Not supported
5	3/1	4	L, C, R, S	Not supported
6	2/2	4	L, R, SL, SR	Not supported
7	3/2	5	L, C, R, SL, SR	**SVC_SURROUND**

adconvtyp – A/D Converter Type

Part of extended BSI. This parameter allows audio previously passed through a particular A/D conversion stage to be marked as such, so that a decoder may apply the complementary D/A process.

Value	Meaning
0 (default)	Standard.
1	HDCD.

audprodie(2) – Audio Production Info Exists

This parameter indicates whether the mixing level and room type values are valid. In practice, only high-end consumer equipment implements these features. Dolby Digital Plus encoders send this information only if the settings deviate from the default. Thus, control of this parameter is for Dolby Digital only, and not provided to the user for Dolby Digital Plus.

Value	Meaning
`false` (default)	Mixing level and room type parameters are valid.
`true`	Mixing level and room type parameters are invalid (ignored).

bsmod – Bitstream Mode

This parameter describes the audio service contained within the bitstream.

Value	Meaning
0 (default)	Main audio service: complete main (CM). The bitstream is the main audio service for the program and all elements are present to form a complete audio program. This is the most common default setting. The CM service may contain from one (mono) to six (5.1) channels.
1	Main audio service: music and effects (ME). The bitstream is the main audio service for the program, minus a dialogue channel. The dialogue channel, if any, is intended to be carried by an associated dialogue service. Different dialogue services can be associated with a single ME service to support multiple languages.
2	Associated audio service: visually impaired (VI). Typically a single-channel program intended to provide a narrative description of the picture content to be decoded along with the main audio service. The VI service may be comprised of up to six channels.
3	Associated audio service: hearing impaired (HI). Typically a single-channel program intended to convey audio that has been processed for increased intelligibility and decoded along with the main audio service. The HI service may be comprised of up to six channels.
4	Associated audio service: dialogue (D).

	Typically a single-channel program intended to provide a dialogue channel for an ME service. If the ME service contains more than two channels, the D service is limited to only one channel; if the ME service is two channels, the D service can be a stereo pair. The appropriate channels of each service are mixed together. (This requires special decoders.)
5	Associated audio service: commentary (C). Typically a single-channel program intended to convey additional commentary that can be optionally decoded along with the main audio service. This service differs from a dialogue service because it contains an optional, rather than a required, dialogue channel. The C service may also be a complete mix of all program channels, comprising up to six channels.
6	Associated audio service: emergency (E). This is a single-channel service that is given priority in reproduction. When the E service appears in the bitstream, it is given priority in the decoder and the main service is muted.
7	Associated audio service: voice over (VO). This is a single-channel service intended to be decoded and mixed to the C channel (requires special decoders).

cmixlev – Center Downmix Level

When the encoded audio has three front channels (L, C, R) but the consumer has only two front speakers (left and right), *cmixlev* indicates the nominal downmix level for the C channel with respect to the L and R channels. If *cmixlev* is set to the reserved code, decoders should still reproduce audio. -4.5dB may be used in this case.

Value	Meaning	Description
0	0.707 (-3.0dB)	The C channel is attenuated 3dB and sent to L and R.
1	0.595 (-4.5dB)	The C channel is attenuated 4.5dB and sent to L and R.
2	0.500 (-6.0dB)	The C channel is attenuated 6dB and sent to L and R.
3 (default)	Reserved	

compr(2) – RF Mode Compression Words

This parameter allows a large dynamic range reduction such that a downmix will not exceed a certain peak level. The heavily compressed audio program may be desirable for certain listening situations such as movie delivery to an airline seat. The peak level limitation is useful when, for instance, a downmix will feed an RF modulator and overmodulation must be avoided.

In an encoded AC-3 bitstream, *compr* is an 8-bit field that is encoded every AC-3 frame (32ms). The first four bits of *compr* encode the gain in 6dB increments, while the following four bits indicate linear gain changes. Please refer to the Dolby documentation for more information.

NOTE: *compr* (4-bit 6dB increment, 4-bit linear) is coded similarly to *dynrng* (3-bit 6dB increment, 5-bit linear gain).

In a RDD 6 metadata stream, *compr* can be encoded in the same way as in AC-3 (see above), or alternatively RDD6 allows *compr* to be encoded as a compression profile. RDD 6 defines the following (RF[1]) compression profiles.

Value	Meaning
0	No compression.
1	Film standard compression.
2	Film light compression.
3	Music standard compression.
4	Music light compression.
5	Speech compression.
6-255	Reserved.

The DTA-2180 AC-3 encoder only supports static control of *compr*, encoded as a compression profile. A change of *compr* cannot be applied seamlessly.

copyrightb – Copyright Bit

This parameter indicates whether the encoded bitstream is copyright protected. It has no effect on AC-3 decoders, and its purpose is to provide information only.

Value	Meaning
0	Not copyright protected.
1 (default)	Copyright protected bitstream.

dialnorm(2) – Dialogue Normalization

Using the subjective level of normal spoken dialogue as a reference, the *dialnorm* value indicates how far the average dialogue level of the encoded program is below digital full scale. The valid range is 1 to 31, which is interpreted as –1 to –31dBFS. The default value is 27, corresponding to -27dBFS.

dheadphonmod – Dolby Headphone Mode

Part of extended BSI. This parameter indicates whether or not the program has been Dolby Headphone-encoded. This information is not used by the AC-3 decoder, but may be used by other portions of the audio reproduction equipment. The meaning of *dheadphonmod* is only defined if the service type is **SVC_STEREO**.

Value	Meaning
0	Not indicated.
1	Dolby headphone disabled.
2	Dolby headphone enabled.
3 (default)	Reserved, to be interpreted as “not indicated”.

[1] RF is a leftover from the “old days”. It is used to indicate that the audio compression is important when the signal is modulated with an RF modulator.

dmixmod – Preferred Stereo Downmix Mode

Part of extended BSI. This parameter indicates the type of stereo downmix preferred by the mastering engineer. This information may be used by the AC-3 decoder to automatically configure the type of stereo downmix, but may also be overridden or ignored. The meaning of *dmixmod* is only defined if the service type is **SVC_SURROUND**.

Value	Meaning
0	Not indicated.
1	Lt/Rt downmix preferred.
2	Lo/Ro downmix preferred.
3 (default)	Reserved, to be interpreted as “not indicated”.

dsurexmod – Dolby Surround EX Mode

Part of extended BSI. This parameter is used to identify the encoded audio as material encoded in Dolby Digital Surround EX. It is used only if the encoded audio has two surround channels. The behavior is similar to that of the Dolby Surround mode parameter.

Value	Meaning
0	Not indicated.
1 (default)	Not encoded in Dolby Surround EX. The decoded PCM audio should be processed by a Dolby Digital Surround EX decoder.
2	Encoded in Dolby Surround EX.

dsurmod – Dolby Surround Mode

This parameter indicates to a decoder whether the two-channel encoded bitstream contains a Dolby Surround (Lt/Rt) program that requires Dolby Pro Logic decoding.

Value Meaning	Description
0	Not encoded in Dolby Surround.
1	Encoded in Dolby Surround. The decoded PCM audio should be processed by a Dolby Pro Logic matrix decoder.
2	There is no indication.
3 (default)	Reserved, to be interpreted as “not encoded”.

dynrng(2) – Dynamic Range Gain

This parameter indicates a gain change to be applied in the AC-3 decoder in order to implement dynamic range compression. The *dynrng* values typically indicate gain reductions (cut) during loud passages and gain increases (boost) during quiet passages based on desired compression characteristics.

In an encoded AC-3 bitstream, *dynrng* is an 8-bit field that is encoded every AC-3 block (5.3ms). The first three bits of *dynrng* indicate gain in 6dB increments, while the following five bits indicate linear gain changes. Please refer to the Dolby documentation for more information.

NOTE: *dynrng* (3-bit 6dB increment, 5-bit linear) is coded similarly to *compr* (4-bit 6dB increment, 4-bit linear gain).

In a RDD 6 metadata stream, *dynrng* can be encoded in the same way as in AC-3 (see above), or alternatively *dynrng* can encoded as a compression profile. The profiles are the same as for parameter *compr*, see the description of *compr* for a table with defined profiles.

An RDD Dolby Digital Complete + Essential metadata segment contains 8 *dynrng* values. They are intended to control a AC-3 encoder that transcodes from Dolby E to AC-3. Each *dynrng* values specifies the dynamic range for 1/8 of a Dolby E frame.

The DTA-2180 AC-3 encoder only supports static control of *dynrng*, encoded as a compression profile. A change of *dynrng* cannot be applied seamlessly.

lfeon – Low Frequency Effects Channel On

Enable or disable (default) the low frequency effects (LFE) channel. This is an optional low frequency channel (<120Hz) intended to be reproduced at a level +10dB with respect to the base audio signal. The LFE channel allows high sound pressure level to be provided for low frequency sounds.

The audio service type (**`DtEncAudPars`**::*`m_SvcType`*) determines whether the LFE channel is allowed. The audio service shall have at least three channels to enable the LFE channel.

LFE Channel Allowed	Meaning
LFE channel not allowed	**`SVC_MONO`**, **`SVC_STEREO`**, **`SVC_DUAL_MONO`**
LFE channel allowed	**`SVC_SURROUND_5_1`**

lorocmixlev– Lo/Ro Center Downmix Level

Part of extended BSI. This 3-bit code indicates the level shift applied to the C channel when adding to the L and R outputs as a result of downmixing to an Lo/Ro output.

Value	Meaning
0	1.414 (+3.0dB)
1	1.189 (+1.5dB)
2	1.000 (0.0dB)
3	0.841 (-1.5dB)
4	0.707 (-3.0dB)
5	0.595 (-4.5dB)
6	0.500 (-6.0dB)
7	0

lorosurmixlev – Lo/Ro Surround Downmix Level

Part of extended BSI. This 3-bit code indicates the level shift applied to the surround channels when downmixing to an Lo/Ro output.

Value	Meaning
0, 1, 2	Reserved
3	0.841 (-1.5dB)
4	0.707 (-3.0dB)
5	0.595 (-4.5dB)
6	0.500 (-6.0dB)
7	0

ltrtcmixlev – Lt/Rt Center Mix Level

Part of extended BSI. This 3-bit code indicates the nominal down mix level of the center channel with respect to the left and right channels in an Lt/Rt downmix. The meaning of *ltrtcmixlev* is only defined if the service type is **SVC_SURROUND**.

Value	Meaning
0	1.414 (+3.0dB)
1	1.189 (+1.5dB)
2	1.000 (0.0dB)
3	0.841 (-1.5dB)
4	0.707 (-3.0dB)
5	0.595 (-4.5dB)
6	0.500 (-6.0dB)
7	0

ltrtsurmixlev – Lt/Rt Surround Mix Level

Part of extended BSI. This 3-bit code indicates the nominal down mix level of the surround channels with respect to the left and right channels in an Lt/Rt downmix. The meaning of *ltrtsurmixlev* is only defined if the service type is **SVC_SURROUND**.

Value	Meaning
0, 1, 2	Reserved
3	0.841 (-1.5dB)
4	0.707 (-3.0dB)
5	0.595 (-4.5dB)
6	0.500 (-6.0dB)
7	0

mixlevel(2) – Mixing Level

This parameter describes the peak sound pressure level (SPL) used during the final mixing session at the studio or on the dubbing stage. The peak mixing level is the acoustic level of a sine wave in a single channel whose peaks reach 100 percent in the PCM representation. The value of *mixlevel* is not typically used within the AC-3 decoder, but may be used by other parts of the audio reproduction equipment. Note that this element is present in the Dolby E frame regardless of the value of the audio production information exists flag. The valid range of *mixlevel* is 0=80dB through 31=111dB and the default value is 25=105dB.

origbs – Original Bitstream

This parameter indicates whether the encoded bitstream is the master version or a copy. It has no effect on AC-3 decoders, and its purpose is to provide information only.

Value	Meaning
0	Copied.
1 (default)	Not copied.

roomtyp(2) – Room Type

This parameter indicates the type and calibration of the mixing room used for the final audio mixing session. The value of *roomtyp* is not typically used by an AC-3 decoder, but may be used by other parts of the audio reproduction equipment.

Value	Meaning
0	Not indicated.
1	Large room, X curve monitor.
2 (default)	Small room, flat monitor.
3	Reserved, to be interpreted as “not indicated”.

surmixlev – Surround Downmix Level

If surround channels are in use, *surmixlev* indicates the nominal downmix level of the surround channels. If *surmixlev* is set to the reserved code, the decoder should still reproduce audio. –6dB may be used in this case.

Value	Meaning	Description
0	0.707 (-3.0dB)	The Ls and Rs channels are each attenuated 3dB and sent to the L and R channels, respectively.
1	0.500 (-6.0dB)	The Ls and Rs channels are each attenuated 6 dB and sent to the L and R channels, respectively.
2	0	The surround channels are discarded.
3 (default)	Reserved	

xbsie – Extended Bitstream Information Exists

This Boolean parameter indicates whether the encoded bitstream contains extended bitstream parameters. In the AC-3 bitstream, *xbsie* is split in two fields, *xbsie1* and *xbsie2*, which each enable a number of extended parameters. *xbsie1* and *xbsie2* have to be set to the same value.

AC-3 Field	Enables
xbsie1	*dmixmod*, *ltrtcmixlev*, *ltrtsurmixlev*, *lorocmixlev*, *lorosurmixlev*
xbsie2	*dsurexmode, dheadphonmod*, *adconvtyp*, *xbsi2* (reserved for future assignment), *encinfo* (reserved for use by the encoder)

The corresponding section in the AC-3 bitstream is defined as follows:

```
 :  :
xbsi1e                                  1
if (xbsi1e)
{
   dmixmod                              2
   ltrtcmixlev                          3
   ltrtsurmixlev                        3
   lorocmixlev                          3
   lorosurmixlev                        3
}
xbsi2e                                  1
if (xbsi2e)
{
   dsurexmod                            2
   dheadphonmod                         2
   adconvtyp                            1
   xbsi2                                8
   encinfo                              1
}
 :  :
```

Originally (before defining the extended bitstream syntax) this section of the AC-3 bitstream definition looked as follows:

```
 :  :
timecod1e                               1
if (timecod1e)
   timecod1                             14
timecod2e                               1
if (timecod2e)
   timecod2                             14
 :  :
```

This construction works because *xbsi1e* and *timecod1e* have opposite definitions. If this bit is '0', the bitstream contains *timecod1*, it this bit is '1' the bitstream contains the extended metadata parameters. The same applies to *xbsi2e* and *timecod2e*.

class DtEncParsBase

Base class for (most) encoding parameter classes, storing the type number of the encoder hardware. The parameter classes use the type number to implement encoder-specific parameter constraints and mappings.

The following classes are derived from **`DtEncParsBase`**: **`DtEncPars`**, **`DtEncAudPars`**, **`DtEncAudParsAc3`**, **`DtEncMuxPars`**, **`DtEncVidParsH264`**, **`DtEncVidParsMp2V`**, **`DtEncVidPars`**. The public methods **`SetEncType`** and **`GetEncType`** are available on each of these classes.

class DtEncAncPars

Class for specifying encoding and embedding parameters for data that is not audio or video.

DtEncAncPars – Public Members

Helper structure describing the ancillary data parameters.

```
class DtEncAncPars : public DtEncParsBase
{
public:
   AfdBarMode  m_AfdBarMode;       // AFD/BAR insertion mode
   CcMode  m_CcMode;               // Closed caption mode
   CcSource  m_CcSource;           // Closed caption source
   VbiFormat  m_VbiFormat;         // VBI input in MSB or LSB
   bool  m_VideoIndex;             // Enable video index processing
   bool  m_Vitc;                   // Enable VITC

   // Set encoder type (e.g. 2180)
   // First method to be called when this object is used standalone
   DTAPI_RESULT  SetEncType(int EncType);
};
```

Public Members

m_AfdBarMode

AFD/BAR insertion mode.

Value	Meaning
AFDBAR_NONE	Do not extract/insert AFD/BAR.
AFDBAR_WHENNEEDED	Insert/extract AFD/BAR as needed.
AFDBAR_ALWAYS (default)	Always extract/insert AFD/BAR.

m_CcMode

Closed caption mode.

Value	Meaning
CC_DISABLE	Do not extract/insert captions. Set *m_CcSource* to **CC_NONE**.
CC_ALL (default)	Extract/insert all captions.
CC_608B	Extract/insert EIA608B field 1 and EIA608B field 2.
CC_608B_FLD1	Extract/insert EIA608B field 1.
CC_608B_FLD2	Extract/insert EIA608B field 2.
CC_708B	Extract/insert EIA708B.

m_CcSource

Closed caption source.

Value	Meaning
CS_NONE	Closed captions not used. Set *m_CcMode* to **CC_DISABLE**.
CS_VANC	Closed caption data taken from VANC.
CS_WAVEFORM	For SD only: Decode waveform in line 21.
CS_ALL (default)	Take closed captions from VANC and/or from line 21.

m_VbiFormat

VBI input in MSB or LSB.

Value	Meaning
VBI_MSB	VBI input in MSB.
VBI_LSB (default)	VBI input in LSB.

m_VideoIndex

Enable or disable (default) video index processing.

m_Vitc

Enable or disable (default) Digital Vertical Interval Time Code (D-VITC) extraction from SDI and insertion in ??? in the transport stream.

SetEncType()

Set the type number of the encoder card for which the parameters are meant. The encoder type number can be read back with **GetEncType** (implemented in **DtEncParsBase**).
This method returns **DTAPI_E_INVALID_ARG** if the encoder type number is invalid, or if the type number is valid but it is not encoder hardware.

DtEncAncPars::CheckValidity

Check the validity of the ancillary data encoding and embedding parameters.

```
DtEncResult DtEncAncPars::CheckValidity
();
```

Function Arguments

Result

For generic result codes, see the Results table listed on the **DtEncAncPars::CheckValidity** page. The following result codes are specific for ancillary data encoding and embedding:

DT_ENC_E_INV_AFDBARMODE	The value in **DtEncAncPars**::*m_AfdBarMode* is not a valid AFD/BAR value.
DT_ENC_E_INV_CCMODE	The value in **DtEncAncPars**::*m_CcMode* is not a valid closed captioning extraction/processing mode.
DT_ENC_E_INV_CCSOURCE	The value in **DtEncAncPars**::*m_CcSource* is not a valid closed captioning source.

Remarks

DtEncAncPars::SetDefaultPars

Set defaults for the current object with ancillary data encoding and embedding parameters.

```
DtEncResult DtEncAncPars::SetDefaultPars
();
```

Function Arguments

Result

DtEncResult	Meaning
DT_ENC_OK	Defaults for the ancillary data encoding and embedding parameters have been set successfully.

Remarks

class DtEncAudPars

Class for specifying the encoding parameters for a single audio service.

DtEncAudPars – Public Members

The public members in class **DtEncAudPars** specify the generic parameters for the encoding of one audio service. An audio service is an encoded stereo pair, or a 5.1 surround service, or one of the other options, see the description of member *m_SvcType* below.

An audio service is made up of one or more audio 'channels'. A channel is defined as a single stream of audio samples. A stereo service is the encoding of two audio channels, while a surround service uses six audio channels. Audio channels originate (are de-embedded) from the input of the current encoder. For example, for the DTA-2180 the audio is taken from the SDI- or HDMI input. Member *m_AudChans* defines the mapping of audio streams from input to the audio encoder.

```
class DtEncAudPars : public DtEncParsBase
{
public:
  bool  m_Enable;                // Enable/disable audio service

  // Audio input configuration
  std::vector<int>  m_AudChans;  // Audio channels in the service

  // Generic (standard-independent) audio-encoding parameters
  int  m_Bitrate;                // Bitrate of encoded audio service
  int  m_Delay;                  // Audio delay in milliseconds
  int  m_SampleRate;             // Sample rate: 32000, …

  // Advanced generic audio-encoding parameters
  bool  m_AlignedPes;            // Enable aligned PES
  bool  m_VolumeAdjust;          // Enable volume adjustment
  double  m_VolumeAdjustdB;      // Volume adjustment amount in dB

  // Get and set audio encoding standard and service type
  DtAudEncStd  GetAudEncStd() const;
  DtAudEncStd  GetSvcType() const;
  DTAPI_RESULT  SetAudEncStd(DtAudEncStd, AudServiceType);

  // Audio encoding parameters for specific audio-encoding standards
  DtEncAudParsAac*  Aac() const;
  DtEncAudParsAc3*  Ac3() const;
  DtEncAudParsMp1LII*  Mp1LII() const;
  DtEncAudParsPcm*  Pcm() const;

  // Set encoder type (e.g. 2180)
  // First method to be called when this object is used standalone
  DTAPI_RESULT  SetEncType(int EncType);
};
```

Public Members

m_Enable

Enable or disable (default) encoding of this audio service. If disabled, all remaining parameters are ignored.

m_AudChans

This vector of integer specifies the input indices of the audio channels to be included in the audio service. An audio channel is defined as a single stream of audio samples.
Channel index is zero-based: The index of the first audio channel is 0, the index of the second channel is 1, etc.

Encoder Type	Channel Index Range
DTA-2180	0 through 15 for HD-SDI input. 0 through 5 for HDMI input. 0 through 31 for 3G-SDI input (not supported at the moment).

The size of vector *m_AudChans* must be set to the number of channels required for the service configuration (see *m_SvcType*). A stereo service requires two audio channels (vector size is two), while a 5.1 surround service requires six audio channels.
If the HDMI-input is selected, the first audio channel must be 0.

m_Bitrate

Integer defining the bitrate of the encoded audio service in bits per second. The table below shows the bitrates that can be used for encoded audio (32k = 32000, etc.). Dependent on the audio encoding- and service type only a subset of these bitrates may be valid. The default bitrate is 96kbps.

Bitrate Values for Encoded Audio
32k, 48k, 56k, 64k, 80k, 96k, 112k, 128k, 160k, 192k, 224k, 256k, 320k, 384k, 448k, 576k, 640k

If service type is **SVC_PASSTHROUGH**, then *m_Bitrate* has to be set to the maximum bitrate of the audio stream that is passed through.

If the audio encoding standard is **DT_AUDENCSTD_PCM**, then *m_Bitrate* has to be set to: *m_SampleRate* * *(m_BitsPerSample*+4*)* * 2.

m_Delay

Audio delay relative to the encoder's end-to-end delay, expressed in milliseconds. This delay can be used for lip-sync correction. The valid range is -100 to 400ms. The default delay is 0ms.

m_SampleRate

Integer defining the audio-channel sample rate. This parameter needs to match the source sample rate of uncompressed audio samples. The table below shows the valid sample rates. The default sample rate is 48000.

Value	Meaning
32000	32kHz, currently not supported
48000	48kHz

If service type is **SVC_PASSTHROUGH**, *m_SampleRate* has to be set to the sample rate of the source AES3 stream with the encoded audio to be passed through (typically 48kHz). This is not necessarily equal to the original source sample frequency of the audio before encoding.

m_AlignedPes

Enable (default) or disable alignment of PES packets to the start of transport packets.

m_VolumeAdjust

Enable or disable (default) adjustment of the audio volume prior to encoding. If service type is **SVC_PASSTHROUGH**, the volume cannot be adjusted and *m_VolumeAdjust* must be set to **false**.

m_VolumeAdjustdB

Volume adjustment amount applied to audio samples prior to encoding when *m_VolumeAdjust* is set to true (enabled). The valid range is from 0.0 dB to 24.0dB. The default value is 0.0 dB (no adjustment).

SetEncType()

Set the type number of the encoder card for which the parameters are meant. The encoder type number can be read back with **GetEncType** (implemented in **DtEncParsBase**).
This method returns **DTAPI_E_INVALID_ARG** if the encoder type number is invalid, or if the type number is valid but it is not encoder hardware.

DtEncAudPars::CheckValidity

Check the validity of the encoding parameters for the audio service.

```
DtEncResult DtEncVidPars::CheckValidity
(
  [in] int  SourcePort = -1        // Source port number
);
```

Function Arguments

SourcePort

Port number of the physical port connected to encoder's input. The default value of this argument is **-1**, which selects the encoder's default source port. The table below lists the source ports available on the different DekTec encoders.

Type	Source Port	Description
DTA-2180	**1 (default)**	SDI input, accepting SD-SDI and HD-SDI.
	2	HDMI input.

Result

See the Results table listed on the **DtEncPars::CheckValidity** page.

Remarks

DtEncAudPars::GetAudEncStd

Get the current audio encoding standard.

```
DtAudEncStd DtEncAudPars::GetAudEncStd();
```

Function Arguments

None.

Result

This function does not return a DTAPI_RESULT but directly returns the audio-encoding standard.

Value	Meaning
DT_AUDENCSTD_UNKNOWN	The audio standard is not known.
DT_AUDENCSTD_AAC	AAC encoded audio.
DT_AUDENCSTD_AC3	Dolby AC-3 encoded audio.
DT_AUDENCSTD_DOLBY_E	Dolby E.
DT_AUDENCSTD_EAC3	Dolby E-AC-3 encoded audio.
DT_AUDENCSTD_MP1LII	MPEG-1 Layer II audio encoding.
DT_AUDENCSTD_PCM	Direct embedding of PCM samples without encoding into an MPEG-2 transport stream according to the SMPTE 302M.

See also *3.1 Audio Encoding Standards*.

Remarks

DtEncAudPars::GetSvcType

Get the audio service type.

```
DtEncAudPars::AudServiceType DtEncAudPars::GetSvcType();
```

Function Arguments

None.

Result

Type of audio service.

Value	Meaning
SVC_DUAL_MONO	Service consisting of two mono channels.
SVC_MONO	Mono service. Not supported for AAC HEv2.
SVC_PASSTHROUGH	Pass-through mode. The audio service is already encoded, and the encoder passes through the encoded audio data.
SVC_STEREO	Stereo audio service.
SVC_SURROUND_5_1	5.1 surround sound service. Not supported for AAC HEv2 and MPEG-1 layer II.

Remarks

DtEncAudPars::SetAudEncStd

Set audio encoding standard and service type. If *AudEncStd* is new, create an internal object with encoding-standard specific parameters (**DtEncAudParsAc3**, etc.), and initialize these parameters with default values.

```
DTAPI_RESULT DtEncAudPars::SetAudEncStd
(
  [in] DtAudEncStd  AudEncStd;     // Audio encoding standard
  [in] DtAudServiceType  SvcType;  // Service type: SVC_STEREO, …
};
```

Function Arguments

AudEncStd

New audio-encoding standard.

Value	Meaning
DT_AUDENCSTD_UNKNOWN	The audio standard is not known.
DT_AUDENCSTD_AAC	AAC encoded audio.
DT_AUDENCSTD_AC3	Dolby AC-3 encoded audio.
DT_AUDENCSTD_DOLBY_E	Dolby E.
DT_AUDENCSTD_EAC3	Dolby E-AC-3 encoded audio.
DT_AUDENCSTD_MP1LII	MPEG-1 Layer II audio encoding.
DT_AUDENCSTD_PCM	Direct embedding of PCM samples without encoding into an MPEG-2 transport stream according to the SMPTE 302M.

See also *3.1 Audio Encoding Standards*.

SvcType

Type of audio service. Refer to the table below for a list of available service configurations. The default service type is **SVC_STEREO**.

Value	Meaning
SVC_DUAL_MONO	Service consisting of two mono channels.
SVC_MONO	Mono service. Not supported for AAC HEv2.
SVC_PASSTHROUGH	Pass-through mode. The audio service is already encoded, and the encoder passes through the encoded audio data.
SVC_STEREO	Stereo audio service.
SVC_SURROUND_5_1	5.1 surround sound service. Not supported for AAC HEv2 and MPEG-1 layer II.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The new audio encoding standard has been set successfully.
DTAPI_E_INVALID_ARG	The specified audio-encoding standard or the service type is invalid.
DTAPI_E_PASSTHROUGH_ONLY	Pass-through mode is not allowed for the specified audio-encoding standard.
DTAPI_E_PASSTHROUGH_INV	Pass-through mode (service type is **SVC_PASSTHROUGH**) is not allowed for the specified audio-encoding standard.

Remarks

If the audio encoding standard was already set to the standard specified in *AudEncStd*, then **SetAudEncStd** is a no-operation. In this case default parameters will not be set.

class DtEncAudParsAac

Class for specifying audio encoding parameters for AAC. The profile discriminates between AAC-LC and HE-AAC.

DtEncAudParsAac – Public Members

The public members in class **DtEncAudParsAac** specify the audio-encoding parameters for AAC. An object of this class is available through **DtEncAudPars**::*Aac()* when audio encoding standard **DT_AUDENCSTD_AAC** is selected, and the service type is not **SVC_PASSTHROUGH**.

```
class DtEncAudParsAac : public DtEncParsBase
{
public:
   AacFmt  m_ContainerFormat;     // Container format
   AacProfile  m_Profile;         // Encoder profile
   bool  m_Crc;                   // Add CRC to data packets
   AacVersion  m_Version;         // MPEG-2 or MPEG-4 AAC
   bool  m_LowLoad;               // Use low-load encoding algorithm
};
```

Public Members

m_ContainerFormat

Defines the container format to be used for encapsulating the encoded AAC audio data.

Value	Meaning
CF_ADTS (default)	Audio Data Transport Stream (ADTS) container format.
CF_LATM	Low Overhead Audio Transport Multiplex (LATM) container format. Incompatible with m_EncVersion=**AAC_MP2**

m_Profile

Defines the AAC profile.

Value	Meaning
AAC_LC (default)	Low Complexity (LC) profile
AAC_HE	High Efficiency profile (HE-AAC or HE-AAC v1). Uses spectral band replication (SBR) to enhance the compression efficiency in the frequency domain.
AAC_HEv2	High Efficiency version 2 profile (HE-AAC v2). Uses spectral band replication (SBR) and parametric stereo (PS) to enhance the compression efficiency of stereo signals. Only supported for service type **SVC_STEREO**.

For HE-AAC, an end-to-end delay (**DtVidPars**::*m_EndToEndDelay*) of 150ms is not supported.

m_Crc

Enable or disable (default) a 16-bit CRC appended to the AAC data packets.

m_EncVersion

Defines the AAC version.

Value	Meaning
AAC_MP4 (default)	MPEG-4 AAC.
AAC_MP2	MPEG-2 AAC.

m_Lowload

Enable or disable (default) a low-load algorithm for encoding AAC. If disabled, "full mode" is used.

DtEncAudParsAac::CheckValidity

Check the validity of the AAC audio encoding parameters.

```
DtEncResult DtEncAudParsAac::CheckValidity
();
```

Validity Checks

For AAC, the following minimum and maximum service bitrates apply for the different service types:

Service Type	AAC-LC	HE-AAC v1	HE-AAC v2
SVC_MONO	32-192kbps	32-96kbps	Not supported
SVC_STEREO	32-384kbps	32-192kbps	32-96kbps
SVC_DUAL_MONO	32-384kbps	32-192kbps	Not supported
SVC_SURROUND_5_1	96-640kbps	96-640kbps	Not supported

Result

See the Results table listed on the **DtEncPars::CheckValidity** page.

Remarks

class DtEncAudParsAc3

Class for specifying audio encoding parameters for AC-3.

DtEncAudParsAc3 – Public Members

The public members in class **DtEncAudParsAc3** specify the audio-encoding parameters for AC3. An object of this class is available through **DtEncAudPars**::*Ac3()* when audio encoding standard **DT_AUDENCSTD_AC3** is selected, and the service type is not **SVC_PASSTHROUGH**.

```
class DtEncAudParsAc3 : public DtEncParsBase
{
public:
  bool  m_DynRangeCtrl1;          // Enable normal compression
  bool  m_DynRangeCtrl2;          // Enable secondary compression
  bool  m_LfeChannel;             // Enable LFE channel
  bool  m_LfeFilter;              // Enable LFE lowpass filter
  bool  m_SurroundDelay;          // Enable surround channel delay

  // Dolby metadata
  int   m_DialNorm;               // Dialog normalisation
  bool  m_DcFilter;               // Enable DC filter
  int   m_CompChar;               // Global compression profile
  int   m_DComp;                  // Line mode profile
  int   m_D2Comp;                 // Line mode profile second channel
  int   m_CComp;                  // RF mode profile
  int   m_C2Comp;                 // RF mode profile second channel
  bool  m_Deemphasis;             // Enable digital deemphasis
  bool  m_BwFilter;               // Enable bandwidth filter
  bool  m_Phase90;                // 90-degree surround phase shift
  bool  m_Xbsi2Ex;                // Enable extended bitstream ind
  int   m_HeadphoneMode;          // Dolby headphone mode
  int   m_AdConvType;             // A/D converter type
  int   m_MixingLevel;            // Mixing level
  bool  m_Copyright;              // Copyright flag
  bool  m_OriginalBs;             // Original bitstream flag
  int   m_BitstreamMode;          // Bitstream mode
  int   m_RoomType;               // Room type
  int   m_SurroundMode;           // Dolby surround mode
  bool  m_Xbsi1Ex;                // Enable extended bitstream ind.
  bool  m_AdvDrc;                 // Enable advanced DRC
  int   m_CenterMixLevel;         // Center mix level
  int   m_SurroundMixLevel;       // Surround mix level
  int   m_DownMixMode;            // Preferred stereo downmix mode
  int   m_LtRtCenterMixLevel;     // Lt/Rt center mix level
  int   m_LtRtSurroundMixLevel;   // Lt/Rt surround mix level
  int   m_LoRoCenterMixLevel;     // Lo/Ro center mix level
  int   m_LoRoSurroundMixLevel;   // Lo/Ro surround mix level
  int   m_SurroundExMode;         // Dolby surround EX mode
  bool  m_SurroundAttn;           // 3dB surround attenuation flag
  bool  m_AudioProdInfo;          // Audio production info
};
```

Public Members

m_DynRangeCtrl1
Enable (default) or disable normal dynamic-range reduction.

m_DynRangeCtrl2
Enable or disable (default) large dynamic-range reduction.

m_LfeFilter
Enable or disable (default) the LE low-pass filter (120Hz).

m_SurroundDelay
Enable or disable (default) additional delay of the surround channel.

Public Members - Dolby metadata

m_DialNorm

m_DcFilter
Enable (default) or disable DC filter.

m_CompChar
Global compression profile.

Value	Meaning
0	No compression.
1 (default)	Film standard compression.
2	Film light compression.
3	Music standard compression.
4	Music light compression.
5	Speech compression.

m_DComp
Line mode profile.

Value	Meaning
0	No compression.
1	Film standard compression.
2	Film light compression.
3	Music standard compression.
4	Music light compression.
5	Speech compression.
7 (default)	Unspecified

m_D2Comp

Line mode profile for second channel.

Value	Meaning
0	No compression.
1	Film standard compression.
2	Film light compression.
3	Music standard compression.
4	Music light compression.
5	Speech compression.
7 (default)	Unspecified

m_CComp

RF mode profile.

Value	Meaning
0	No compression.
1	Film standard compression.
2	Film light compression.
3	Music standard compression.
4	Music light compression.
5	Speech compression.
7 (default)	Unspecified

m_C2Comp

RF mode profile for second channel.

Value	Meaning
0	No compression.
1	Film standard compression.
2	Film light compression.
3	Music standard compression.
4	Music light compression.
5	Speech compression.
7 (default)	Unspecified

m_Deemphasis

Enable or disable (default) digital deemphasis.

m_BwFilter

Enable or disable (default) bandwidth filter.

m_Phase90
Enable (default) or disable 90-degree phase shift for surround.

m_Xbsi2Ex

m_HeadphoneMode

m_AdConvType

m_MixingLevel

m_Copyright

m_OriginalBs

m_RoomType

m_SurroundMode

m_Xbsi1Ex

m_AdvDrc

m_CenterMixLevel

m_SurroundMixLevel

m_DownMixMode

m_LtRtCenterMixLevel

m_LtRtSurroundMixLevel

m_LoRoCenterMixLevel

m_LoRoSurroundMixLevel

m_SurroundExMode

m_SurroundAttn
3dB surround attenuation flag.

Value	Meaning
false (default)	3dB surround attenuation disabled.
true	3dB surround attenuation enabled.

m_AudioProdInfo

DtEncAudParsAc3::CheckValidity

Check the validity of the AC-3 audio encoding parameters.

```
DtEncResult DtEncAudParsAc3::CheckValidity
();
```

Validity Checks

For AC-3, the following minimum, maximum and typical service bitrates apply for the different service types.

Service Type	Minimum	Maximum	Typical
SVC_MONO	56kbps	640kbps	96kbps
SVC_STEREO	96kbps	640kbps	192kbps
SVC_DUAL_MONO	96kbps	640kbps	192kbps
SVC_SURROUND_5_1	224kbps	640kbps	448kbps

Result

For generic result codes, see the Results table listed on the **DtEncPars::CheckValidity** page. The following result codes are specific for AC-3 audio encoding:

DT_ENC_E_INV_DOLBYMETADATA	One or more Dolby metadata settings are invalid.

Remarks

class DtEncAudParsMp1LII

Class for specifying audio encoding parameters for MPEG 1 layer II.

DtEncAudParsMp1LII – Public Members

The public members in class **DtEncAudParsMp1LII** specify the audio-encoding parameters for MPEG-1 layer II. An object of this class is available through **DtEncAudPars**::*Mp1LII()* when audio encoding standard **DT_AUDENCSTD_MP1LII** is selected, and the service type is not **SVC_PASSTHROUGH**.

```
class DtEncAudParsMp1LII: public DtEncParsBase
{
public:
  bool  m_Crc;                    // Add CRC to data packets
};
```

m_Crc
Enable or disable (default) a 16-bit CRC appended to the MPEG-1 layer II data packets.

DtEncAudParsMp1LII::CheckValidity

Check the validity of the MPEG-1 layer II audio encoding parameters.

```
DtEncResult DtEncAudParsMp1LII::CheckValidity
();
```

Validity Checks

For MPEG-1 layer II audio encoding, the following minimum and maximum service bitrates apply for the different service types:

Service Type	Bitrate Range
SVC_MONO	32-192kbps
SVC_STEREO	64, 96-384kbps
SVC_DUAL_MONO	64, 96-384kbps
SVC_SURROUND_5_1	Not supported

Result

See the Results table listed on the **DtEncPars::CheckValidity** page.

Remarks

class DtEncAudParsPcm

Class for specifying audio encoding parameters for mapping PCM samples in AES3 frames directly (without encoding) into an MPEG-2 transport stream according to SMPTE 302M-2002. The encoder will generate one PES packet with audio samples per video frame.

DtEncAudParsPcm – Public Members

The public members in class **DtEncAudParsPcm** specify the parameters for mapping PCM samples to a transport stream without encoding.

```
class DtEncAudParsPcm : public DtEncParsBase
{
public:
  int  m_BitPerSample;          // #bits per sample: 16, 20 or 24
};
```

m_BitPerSample
Number of bits per audio sample. Valid values are 16 (default), 20 and 24.

Remarks

The audio service type in **DtEncAudPars**::*m_SvcType* must be set to **SVC_STEREO**.

DtEncAudParsPcm::CheckValidity

Check the validity of the parameters for embedding PCM samples in a transport stream without encoding.

```
DtEncResult DtEncAudParsPcm::CheckValidity
();
```

Validity Checks

The audio service type **DtEncAudPars**::*m_SvcType* must be set to **SVC_STEREO**.

The bitrate in **DtEncAudPars**::*m_Bitrate* must be set to a specific value (see table below), dependent on the specified number of bits per sample (**DtEncAudParsPcm**::*m_BitPerSample*).

Number of Bits / Sample	Required Bitrate Setting
16	1920kbps
20	2304kbps
24	2688kbps

Result

For generic result codes, see the Results table listed on the **DtEncPars::CheckValidity** page. The following result codes are specific for PCM sample embedding:

DT_ENC_E_INV_BITRATE_PCM	The elementary-stream bitrate set for a PCM audio service is not compatible with the number of bits per sample for that service.

Remarks

class DtEncControl

DtEncControl

Top-level class for controlling DekTec video encoder hardware.

```
class DtEncControl;
```

Video encoder hardware may require significant time to boot and to initialize. When **DtEncControl** is attached, the underlying hardware may still be booting or initializing. While this is the case, no operations can be performed on the encoder (**DTAPI_E_INITIALIZING** will be returned). Should the encoder hardware encounter a fatal error, all further operations will return an error code (**DTAPI_E_IN_ERROR_STATE**). To rectify this, detach from the channel and attach again. This process will reboot the encoder hardware.

Two methods are provided to check the initialization status:

DtEncControl::GetOperationalState to poll the current initialization state of the encoder;

DtEncControl::WaitForInitialized to wait, optionally with a timeout, until the initialization state is **OS_NORMAL**.

When a **DtEncControl** object is detached, the encoder will be brought back to the reset state. This may take some time, so that the user application has to wait for the normal state again when a **DtEncControl** object is attached.

DtEncControl::AttachToPort

Attach the encoder-control object to an encoder port. Only one encoder-control object can be attached to an encoder port at a time (exclusive access).

```
DTAPI_RESULT DtEncControl::AttachToPort
(
  [in] DtDevice*  pDtDvc,          // Device object
  [in] int  Port,                  // Physical port number (1…#ports)
  [in] bool  ProbeOnly=false       // Just check whether channel is in use
);
```

Function Arguments

pDtDvc

Pointer to the device object that represents a DekTec encoder device. The device object must have been attached to the device hardware.

Port

Physical port number (must be an encoder port) to which the encoder-control object is attached.

DekTec has the following devices with encoder ports available.

Type	Port	Encoder type
DTA-2180	**4**	Magnum D7Pro encoder

ProbeOnly

Probe whether the encoder is in use, but do not actually attach.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	Channel object has been attached successfully to the port.
DTAPI_E_ATTACHED	Channel object is already attached.
DTAPI_E_DEVICE	Pointer *pDtDvc* is not valid or the device object is not attached to a hardware device.
DTAPI_E_DEV_DRIVER	Unclassified failure in device driver.
DTAPI_E_IN_USE	Another channel object is already attached to this port.
DTAPI_E_NO_ENCODER	*Port* is not an encoder.
DTAPI_E_NO_SUCH_PORT	Invalid port number for this device.

Remarks

DtEncControl::Detach

Detach encoder-control object from the encoder hardware and free resources.

```
DTAPI_RESULT DtEncControl::Detach
();
```

Function Arguments

Result

DTAPI_RESULT	Meaning
DTAPI_OK	Encoder-control object has been detached successfully.
DTAPI_E_NOT_ATTACHED	Encoder-control object is not attached to encoder hardware, so it cannot be detached.

Remarks

DtEncControl::GetEncPars

Get the current encoding parameters. This method will always return the parameters that have been set with **DtEncControl::SetEncPars**.

```
DTAPI_RESULT DtEncControl::GetEncPars
(
 [out] DtEncPars&  EncPars         // Retrieved encoding parameters
);
```

Function Arguments

EncPars

Output argument that receives the current set of encoding parameters.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The encoding parameters have been retrieved successfully.
DTAPI_E_FAN_FAIL	The fan is failing. No operations can be performed.
DTAPI_E_IN_ERROR_STATE	The encoder is in a fatal error state and no operations can be performed.
DTAPI_E_INITIALIZING	The encoder is initializing and no operations can be performed.
DTAPI_E_NO_POWER	The power has not been connected properly to the encoder hardware. No operations can be performed.
DTAPI_E_NOT_ATTACHED	Encoder-control object is not attached to encoder hardware.

Remarks

DtEncControl::GetOperationalState

Get the current operational state of the encoder hardware.

```
DTAPI_RESULT DtEncControl::GetOperationalState
(
 [out] OpState&  State              // Operational state
);
```

Function Arguments

State

Receives the current operational state of the encoder hardware.

Value	Meaning
OS_BOOTING	The encoder hardware is booting. This may take considerable time, up to 11s for the DTA-2180 and DTA-2182.
OS_INIT	The encoder hardware is initializing. This may take considerable time, but not as long as booting. For the DTA-2180 or DTA-2182, the maximum initialization time is TBDs.
OS_IDLE	The encoder hardware is initialized, but still idle (not encoding). In this state parameters can be specified and encoding can be started with **SetOperationalState**.
OS_RUN	The encoder hardware is encoding. Parameters can be changed and encoding can be stopped.
OS_ERROR	The encoder hardware is in a fatal error state. The only way to recover from this situation is attempting a reboot by detaching and re-attaching (which will cause a reboot), or by an explicit reboot with **DtEncControl**::*Reboot*.
OS_FAN_FAIL	The fan is failing and caused the encoder stop. The only way to recover from this situation is attempting a reboot by detaching and re-attaching (which will cause a reboot), or by an explicit reboot with **DtEncControl**::*Reboot*.
OS_NO_POWER	The external power connection of the encoder hardware (6-pin PCIe power connector) is not connected to a power supply. The only way to recover from this situation is attempting a reboot by detaching and re-attaching (which will cause a reboot), or by an explicit reboot with **DtEncControl**::*Reboot*.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The operational state has been retrieved successfully.
DTAPI_E_NOT_ATTACHED	Encoder-control object is not attached to encoder hardware.

Remarks

DtEncControl::IsSeamless

Check whether the transition from the current to a new set of encoding parameters can be performed seamlessly, or not. A transition is seamless if no artefacts are visible. For many encoding-parameter transitions the encoder has to be stopped and restarted, causing a non-seamless transition.

```
DTAPI_RESULT DtEncControl::IsSeamless
(
  [in] const DtEncPars&  NewPars   // New encoding parameters
 [out] bool&  Seamless             // Transition is seamless yes/no
);
```

Function Arguments

NewPars

The new encoding parameters.

Seamless

Indicates whether the transition can be performed seamlessly.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The check for a seamless transition has been performed successfully.
DTAPI_E_FAN_FAIL	The fan is failing. No operations can be performed.
DTAPI_E_IN_ERROR_STATE	The encoder is in a fatal error state and no operations can be performed.
DTAPI_E_NO_POWER	The power has not been connected properly to the encoder hardware. No operations can be performed.
DTAPI_E_INITIALIZING	The encoder is initializing and no operations can be performed.
DTAPI_E_INVALID_PARS	The new encoding parameters are invalid.
DTAPI_E_NOT_ATTACHED	Encoder-control object is not attached to encoder hardware.

Remarks

DtEncControl::Reboot

Reboot the encoder hardware.

WARNING. Rebooting encoder hardware takes considerable time (about 11s for the DTA-2180 and DTA-2182) and is not necessary except as "emergency measure" if the encoder is stuck (e.g. due to an issue with the power or fan).

```
DTAPI_RESULT DtEncControl::Reboot
();
```

Function Arguments

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The encoder is rebooting.
DTAPI_E_NOT_ATTACHED	Encoder-control object is not attached to encoder hardware.

Remarks

DtEncControl::SetEncPars

Set the encoding parameters. Only the full set of parameters can be specified, incremental changes are not supported.

If the encoder is not running (state is not **OS_RUN**), the encoding parameters will become active when the encoder state is changed to **OS_RUN**.

If the encoder is running (state is **OS_RUN**), the encoding parameters will be applied on the fly, and **SetEncPars** will return when the encoder is running with the new parameter set. If possible, DTAPI will change the parameters seamlessly. Otherwise the encoder will be stopped and restarted with the new parameters.

```
DTAPI_RESULT DtEncControl::SetEncPars
(
  [in] const DtEncPars&  EncPars   // New encoding parameters
);
```

Function Arguments

EncPars

New set of encoding parameters.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The new parameters have been set successfully.
DTAPI_E_FAN_FAIL	The fan is failing. No operations can be performed.
DTAPI_E_IN_ERROR_STATE	The encoder is in a fatal error state and no operations can be performed.
DTAPI_E_INITIALIZING	The encoder is initializing and no operations can be performed.
DTAPI_E_INVALID_PARS	The new encoding parameters are invalid.
DTAPI_E_LICENSE	The new encoding parameters need a license that is not available.
DTAPI_E_NO_POWER	The power has not been connected properly to the encoder hardware. No operations can be performed.
DTAPI_E_NOT_ATTACHED	Encoder-control object is not attached to encoder hardware.

Remarks

DtEncControl::SetOperationalState

Set the operational state in order to start or stop encoding.

```
DTAPI_RESULT DtEncControl::SetOperationalState
(
  [in] const OpState&  State       // New operational state
);
```

Function Arguments

State

New operational state.

Value	Meaning
OS_IDLE	Go to the idle state. If the current operational state is **OS_RUN**, encoding is stopped.
OS_RUN	Go to the run state. If the current operational state is **OS_IDLE**, encoding is started.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The new operational state has been applied successfully.
DTAPI_E_FAN_FAIL	The fan is failing. No operations can be performed.
DTAPI_E_IN_ERROR_STATE	The encoder is in a fatal error state and no operations can be performed.
DTAPI_E_INITIALIZING	The encoder is initializing and the operational state cannot yet be set.
DTAPI_E_INVALID_ARG	The argument *State* has an invalid value.
DTAPI_E_NO_POWER	The power has not been connected properly to the encoder hardware. No operations can be performed.
DTAPI_E_NOT_ATTACHED	Encoder-control object is not attached to encoder hardware.

Remarks

It's an error to call **SetOperationalState** while the encoder is booting or initializing (operational state is **OS_BOOTING** or **OS_INIT**).

DtEncControl::WaitForInitialized

Wait until the encoder hardware is initialized, this is operational state is not **OS_BOOTING** and not **OS_INIT**.

```
DTAPI_RESULT DtEncControl::WaitForInitialized
(
  [in] int  TimeOut                // Timeout in ms
);
```

Function Arguments

TimeOut

Timeout in milliseconds. The valid range is 1 to 20.000ms (20 seconds). An infinite time out is not supported.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The encoder hardware is initialized.
DTAPI_E_FAN_FAIL	The fan is failing. No operations can be performed.
DTAPI_E_IN_ERROR_STATE	The encoder is in a fatal error state and no operations can be performed.
DTAPI_E_NO_POWER	The external power of the encoder is not connected.
DTAPI_E_NOT_ATTACHED	Encoder-control object is not attached to encoder hardware.
DTAPI_E_TIMEOUT	The timeout period expired while waiting for the encoder hardware to boot and initialize.

Remarks

class DtEncMuxPars

Class for specifying transport-stream multiplexing and system parameters for encoders.

DtEncMuxPars::EsPars

Helper structure describing the multiplexing parameters for an elementary stream.

```
struct EsPars
{
public:
   int  m_Pid;                    // PID; -1=disabled
   int  m_StreamId;               // Stream ID of the elementary stream

   // Set encoder type (e.g. 2180)
   // First method to be called when this object is used standalone
   DTAPI_RESULT  SetEncType(int EncType);
};
```

Public Members

m_Pid

PID on which the elementary stream is multiplexed. The valid range for *m_Pid* is 16 to 8190 (0x1FFE). MPEG-2, DVB and ATSC reserve certain low PID values, refer to the respective standards. A value of -1 indicates that the elementary stream is not multiplexed in the output transport stream.

m_StreamId

Stream ID assigned to the elementary stream. The valid range for *m_StreamId* for video is 224 to 239 and for audio is 192 to 223, except for AC-3 audio, which is limited to 189 (private stream).

SetEncType()

Set the type number of the encoder card for which the parameters are meant. The encoder type number can be read back with **GetEncType** (implemented in **DtEncParsBase**).
This method returns **DTAPI_E_INVALID_ARG** if the encoder type number is invalid, or if the type number is valid but it is not encoder hardware.

DtEncMuxPars – Public Members

The constructor of **DtEncMuxPars** fills the object with default values which are listed below in the member descriptions.

```
class DtEncMuxPars : public DtEncParsBase
{
public:
   // Overall (elementary-stream independent) parameters
   int  m_Bitrate;                // Total multiplex output rate
   int  m_TsId;                   // Transport stream ID

   // PIDs and stream IDs
   int  m_PcrPid;                 // PCR PID
   int  m_PmtPid;                 // PMT PID
   EsPars  m_VidEsPars;           // Encoded video PID and stream ID

   // Scheduling intervals
   int  m_PatInterval;            // PAT interval in ms
   int  m_PmtInterval;            // PMT interval in ms
   int  m_PcrInterval;            // PCR interval in ms

   // Set encoder type (e.g. 2180)
   // First method to be called when this object is used standalone
   DTAPI_RESULT  SetEncType(int EncType);
};
```

Public Members

m_Bitrate

Transport stream output bitrate. The valid range for the Magnum D7Pro (DTA-2180, DTA-2182) is 2.5Mbps to 128Mbps. The default value depends on the selected format: 10Mbps for SD and 20Mbps for HD.

m_TsId

Transport stream ID inserted into the output transport stream. The valid range is 0 through 0xFFFF and the default value is 0.

m_PcrPid

PID of the stream into which the PCRs are inserted. If the PCR PID is the same as the video PID (*m_VidPid*), then the PCR values are inserted into the video elementary stream. Otherwise a separate PCR elementary stream is generated.
The default value for PCR PID is 0x100, which is the same default as for the video PID, so by default the video PID contains PCRs.

m_PmtPid

PID of the PMT (Program Map Table). The default value for PMT PID is 0x50.

m_VidEsPars

PID and stream ID of the video elementary stream stored in an **EsPars** object. The default value for the video PID is 0x100. The valid range for the video stream ID is 0 to 255, and the default value is 224.

m_PatInterval
Interval (ms) with which the PAT (Program Association Table) is scheduled in the output transport stream. The valid range is 1 to 100ms, and the default value is 50ms.

m_PmtInterval
Interval (ms) with which the PMT (Program Map Table) is scheduled in the output transport stream. The valid range is 1 to 100ms and the default value is 50ms.

m_PcrInterval
Interval (ms) with which PCR values are scheduled into the output transport stream. The valid range is 1 to 40ms and the default value is 35ms.

SetEncType()
Set the type number of the encoder card for which the parameters are meant. The encoder type number can be read back with **GetEncType** (implemented in **DtEncParsBase**).
This method returns **DTAPI_E_INVALID_ARG** if the encoder type number is invalid, or if the type number is valid but it is not encoder hardware.

DtEncMuxPars::CheckValidity

Check the validity of the multiplexing parameters.

```
DtEncResult DtEncMuxPars::CheckValidity
();
```

Function Arguments

Result

See the Results table listed on the **`DtEncPars::CheckValidity`** page.

Remarks

class DtEncPars

Top-level class for specifying encoding parameters.

DtEncPars – Public Members

The public members in class **DtEncVidPars** each specify one aspect of the encoding parameters. The constructor of **DtEncVidPars** automatically invokes the constructor of the sub-classes, which in turn fills all sub-members with their default value.

```
class DtEncPars : public DtEncParsBase
{
public:
   int  m_SourcePort;              // Source port number
   DtEncMuxPars  m_AncPars;        // Ancillary parameters
   DtEncMuxPars  m_MuxPars;        // Multiplexing parameters
   DtEncVidPars  m_VidPars;        // Video encoding parameters

   // Audio encoding parameters per audio service
   std::vector<DtEncAudPars>  m_AudPars;
};
```

Public Members

m_SourcePort

Port number of the physical port connected to encoder's input. The table below lists the source ports available on the different DekTec encoders.

Type	Source Port	Description
DTA-2180	1 (default)	SDI input, accepting SD-SDI and HD-SDI.
	2	HDMI input.

m_AncPars

Encoding and embedding parameters for data that is not audio or video: AFD/BAR insertion, closed captioning, video index, VITC insertion.

m_MuxPars

Parameters specifying the structure of the transport stream generated by the encoder, and specifying system-level parameters such as the end-to-end delay.

m_VidPars

Video encoding parameters, specifying both the video preprocessing and the actual video encoding parameters. The video PID and stream ID are located in **m_MuxPars**.

m_AudPars

Audio encoding parameters per service. This vector is sized to the maximum number of audio services that can be encoded when the **DtEncPars** object is constructed with an *EncType* argument, or when **DtEncPars**::**SetEncType**() is called.

Each **`DtEncAudPars`** object has an enable flag, so the actual number of encoded audio services depends on the number of 'enabled' objects. The audio PIDs and stream IDs are located in **`m_MuxPars`**.

DtEncPars::CheckValidity

Check the validity of the encoding parameters. If the parameters are invalid, return a result value indicating the reason of invalidity. If the encoding parameters have multiple errors, the first detected error is returned. This needs not be the "most important" error.

```
DtEncResult DtEncPars::CheckValidity
(
  [in] bool  SkipRateChecks = false   // Skip checks related to bitrates
);
```

Function Arguments

SkipRateChecks

If this argument is **true**, checks related to bitrates are skipped. This enables checking, for example, the validity of the encoding parameters when the transport-stream bitrate has not been set yet. The default value of this argument is **false**, which means that bitrates are checked.

Result

DtEncResult	Meaning
DT_ENC_OK	The encoding parameters are valid.
DT_ENC_E_AUDBITRATETOOLOW	The encoded-audio bitrate is too low for the audio service type.
DT_ENC_E_AUDBITRATETOOHIGH	The encoded-audio bitrate is too high for the audio service type.
DT_ENC_E_EXC_NUMAAC	The number of AAC audio services is too high for the current encoder hardware.
DT_ENC_E_EXC_NUMDOLBYE	The number of Dolby-E audio services is too high for the current encoder hardware.
DT_ENC_E_EXC_NUMNONSURROUND	The number of non-surround audio services (mono, dual mono, stereo) is too high for the current encoder hardware.
DT_ENC_E_EXC_NUMSURROUND	The number of surround services is too high for the current encoder hardware. For the DTA-2180, the maximum number of surround services is 2.
DT_ENC_E_INV_AACPAR	The value of one of the AAC enumeration parameters is invalid.
DT_ENC_E_INV_AC3MODE	The value of one of the AC-3 enumeration parameters is invalid.
DT_ENC_E_INV_AFDBARMODE	The value in **DtEncAncPars**::*m_AfdBarMode* is not a valid AFD/BAR value.
DT_ENC_E_INV_ASPECTRATIO	The value in **DtEncVidPars**::*m_AspectRatio* is not a valid aspect ratio.
DT_ENC_E_INV_AUDBITRATE	The audio service bitrate is invalid.

DT_ENC_E_INV_AUDCHANCONFIG	The audio channel configuration (mapping of audio channels to audio encoder) is invalid, or not possible for the current encoder hardware.
DT_ENC_E_INV_AUDCHANIDX	The audio channel index is invalid.
DT_ENC_E_INV_AUDDELAY	The audio delay is invalid.
DT_ENC_E_INV_AUDENCSTD	The audio encoding standard is invalid.
DT_ENC_E_INV_AUDPID	One or more audio PIDs are invalid.
DT_ENC_E_INV_AUDSAMPLERATE	The audio sample rate is invalid, or not all audio channels are using the same sample rate.
DT_ENC_E_INV_BITPERSAMPLE	The number of bits per audio sample is invalid.
DT_ENC_E_INV_BITRATE_PCM	The elementary-stream bitrate set for a PCM audio service is not compatible with the number of bits per sample for that service.
DT_ENC_E_INV_BITRATE_TS	The transport-stream bitrate is out of range or incompatible with the current profile and level.
DT_ENC_E_INV_BITRATE_VID	The video bitrate is out of range or incompatible with the current profile and level.
DT_ENC_E_INV_CCMODE	The value in **DtEncAncPars**::*m_CcMode* is not a valid closed captioning extraction/processing mode.
DT_ENC_E_INV_CCSOURCE	The value in **DtEncAncPars**::*m_CcSource* is not a valid closed captioning source.
DT_ENC_E_INV_CODINGMODE	The coding mode is invalid, or the combination of end-to-end delay, coding mode and number of B pictures is invalid or incompatible with the current profile and level.
DT_ENC_E_INV_DOLBYMETADATA	One or more Dolby metadata settings are invalid.
DT_ENC_E_INV_DUPLICATEPIDS	One or more duplicate PID's are used.
DT_ENC_E_INV_END2ENDDELAY	The end-to-end delay is invalid.
DT_ENC_E_INV_ENTROPYENC	The entropy encoding (e.g. in *m_Cabac*) is invalid.
DT_ENC_E_INV_FRAMERATE	The frame rate in *m_VidStd* is invalid or incompatible with the current profile and level, and/or the frame rate in **DtEncAudParsPcm**::*m_FrameRate* is invalid.
DT_ENC_E_INV_FRAMESIZE	The frame size in *m_VidStd* is invalid or incompatible with the current profile and level.
DT_ENC_E_INV_GOPSIZE	The GOP size.
DT_ENC_E_INV_HEAACE2EDELAY	The end-to-end delay is incompatible in combination with HE-AAC (v1/v2) audio encoding. On the DTA-2180, the end-to-end delay may not be 150ms for HE-AAC.
DT_ENC_E_INV_IDRFREQ	Invalid IDR frequency.

DT_ENC_E_INV_ILIMAGE	The value in **DtEncVidPars**::*m_InpLossImage* is not a valid input loss image.
DT_ENC_E_INV_INTRADCPREC	The value in *m_IntraDcPrecision* is out of range.
DT_ENC_E_INV_INTRAVLCFMT	The value in *m_IntraVlcFmt* is not a valid intra-VLC format.
DT_ENC_E_INV_LEVEL	Invalid H.264 or MPEG-2 video level.
DT_ENC_E_INV_NUMBPICTURES	The number of B pictures between I or P pictures is outside its valid range or incompatible with the current profile and level.
DT_ENC_E_INV_NUMCHANNELS	The number of specified channels does not match the number of audio source channels required for the audio service.
DT_ENC_E_INV_PATITV	The PAT table interval is outside its valid range.
DT_ENC_E_INV_PIXDEPTH	The value in **DtEncVidPars**::*m_PixelDepth* is not valid or incompatible with the current profile and level.
DT_ENC_E_INV_PMTITV	The PMT table interval is outside its valid range.
DT_ENC_E_INV_PMTPID	The PMT PID is outside its valid range.
DT_ENC_E_INV_PCRITV	The PCR table interval is outside its valid range.
DT_ENC_E_INV_PCRPID	The PCR PID is outside its valid range.
DT_ENC_E_INV_PROFILE	Invalid H.264 or MPEG-2 video profile.
DT_ENC_E_INV_QSCALETYPE	The value in *m_QScaleType* is not valid.
DT_ENC_E_INV_QUANTTABLE	The value in *m_QuantizationTable* is not valid.
DT_ENC_E_INV_RESCALEHOR	The value in **DtEncVidPars**::*m_HorResolutionRescaled* is not a valid horizontally rescaled resolution for the specified video encoding standard.
DT_ENC_E_INV_SOURCEPORT	The value in *m_SourcePort* is invalid.
DT_ENC_E_INV_STREAMID	The stream ID is outside its valid range.
DT_ENC_E_INV_TELECINE	Invalid inverse telecine operation: the frame rate corresponding to the specified video standard is not 59.94Hz or 60Hz
DT_ENC_E_INV_TRANSBLOCKSIZE	The value in *m_8x8Transform* is not valid or incompatible with the current profile and level.
DT_ENC_E_INV_TSID	The transport-stream ID is outside its valid range.
DT_ENC_E_INV_TYPE	The type number set with **SetEncType()** is not valid encoder hardware.
DT_ENC_E_INV_UVSAMPLING	The value in **DtEncVidPars**::*m_UvSampling* is not valid or incompatible with the current profile and level.
DT_ENC_E_INV_VBVDELAY	The VBV delay is invalid or incompatible with the current profile and level.

DT_ENC_E_INV_VIDENCSTD	The video encoding standard in *m_VidPars* has not been set or is invalid for the current encoder hardware.
DT_ENC_E_INV_VIDPID	The video PID is outside its valid range.
DT_ENC_E_INV_VIDSTD	The value in **DtEncVidPars**::*m_VidStd* is not a valid video standard, not supported by the encoder or incompatible with the current profile and level.
DT_ENC_E_INV_VOLUMEADJUST	The volume adjustment is outside its valid range, or volume adjustment is set while in pass-through mode.
DT_ENC_E_TYPE_NOT_SET	No type number has been set with **SetEncType()**.

Remarks

DtEncPars::FromXml

Initialize the encoding parameters in this **DtEncPars** object from an XML string.

```
DTAPI_RESULT DtEncPars::FromXml
(
  [in] const wstring&  XmlString  // Parameters encoded in XML
);
int DtEncPars::GetEncType();
```

Function Arguments

XmlString
XML string containing the encoding parameters in a serialized form.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The encoding parameters have been initialized successfully.
DTAPI_E_XML_ELEM	A required element in the XML string is missing at the expected location.
DTAPI_E_XML_SYNTAX	The XML string is not well-formed XML.

Remarks

DtEncPars::*FromXml* accepts XML strings that are incomplete, e.g. generated by an older DTAPI version in which certain encoding parameters were missing.

- For all parameters except video-encoding parameters, **DtEncPars**::*FromXml* leaves **DtEncPars** fields that are not present in the XML string untouched. We therefor recommend to initialize the **DtEncPars** objects with default values before invoking **DtEncPars**::*FromXml*.
- For video-encoding parameters this works differently: **DtEncPars**::*FromXml* starts by reading the video standard, the video-encoding standard, the profile and the level. These values are passed to **SetDefaultsForProfileLevel**, which sets the video-encoding parameters to sensible default values for the given profile and level. This way, if a certain video-encoding parameter is not contained in the XML string, the corresponding will get a reasonable value anyway.

This means that with respect to defaults, **DtEncPars**::*FromXml* operates asymmetrically between video-encoding parameters and other parameters. Video encoding parameters are automatically initialized to defaults, while the user has to set defaults for the other parameters before calling this routine.

DtEncPars::IsSeamless

Check whether the transition from one set of encoding parameters to another can be performed seamlessly, or not. A transition is seamless if no artefacts are visible. For many encoding-parameter transitions the encoder has to be stopped and restarted, causing a non-seamless transition.

```
static DTAPI_RESULT DtEncPars::IsSeamless
(
  [in] const DtEncPars&  OldPars,  // Old encoding parameters
  [in] const DtEncPars&  NewPars,  // New encoding parameters
 [out] bool&  Seamless             // Transition is seamless yes/no
);
```

Function Arguments

OldPars, NewPars

The old and the new encoding parameters for the transition.

Seamless

Indicates whether the transition is seamless.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The check for a seamless transition has been performed successfully.
DTAPI_E_INVALID_PARS	The old or new encoding parameters are invalid.

Remarks

DtEncPars::MinTsRate

Compute the minimum transport-stream rate – valid for the current encoder type number – corresponding to the encoding parameters in this **DtEncPars** object.

This method can be used to find a suitable value to initialize **m_MuxPars**.*m_Bitrate*.

```
int DtEncPars::MinTsRate
();
```

Function Arguments

Result

The minimum transport-stream rate required for this set of encoding parameters. If the encoding parameters are invalid, -1 is returned.

Remarks

MinTsRate() checks the validity of the encoding parameters, but as this method is meant to compute a valid value for the transport-stream bitrate, parameter **m_MuxPars**.*m_Bitrate* is not used in the validity check.

DtEncPars::NumAudPars

Method that returns the number of **DtEncAudPars** objects available. This is the maximum number of audio services that can be encoded. Each **DtEncAudPars** object has an enable flag, so the actual number of encoded audio services depends on the number of 'enabled' objects.

```
int DtEncPars::NumAudPars
();
```

Function Arguments

Result

The number of **DtEncAudPars** objects (audio parameter structures) available in vector **DtEncPars**::*m_AudPars* (the size of this vector).

Remarks

DtEncPars::ReqNumLicPoints

Method that computes the number of license points required for the specified audio-encoding standard (e.g. for Dolby AC-3), given the encoding parameters in this **DtEncPars** object. This routine accumulates the number of license points required for each enabled audio service that uses the specified audio-encoding standard.

```
DTAPI_RESULT DtEncPars::ReqNumLicPoints
(
  [in] DtAudEncStd  AudEncStd;     // Audio encoding standard
 [out] int&  NumPoints;            // Number of license points required
);
```

Function Arguments

AudEncStd

Audio-encoding standard for which to determine the number of required license points. See *3.1 Audio Encoding Standards* for a list of supported audio encoding standards.

NumPoints

The computed number of license points required for the specified audio-encoding standard.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The number of required license points has been computed successfully.
DTAPI_E_NOT_INITIALIZED	The encoding parameters in this **DtEncPars** objects have not been initialized properly, they are invalid.

Remarks

DtEncPars::SetEncType

Set the type number of the encoder card for which the encoding parameters are meant. The encoder type number can be read back with **GetEncType**.

```
DTAPI_RESULT DtEncPars::SetEncType
(
  [in] int  EncType                // Type of encoder card (e.g. 2180)
);
int DtEncPars::GetEncType();
```

Function Arguments

EncType
Type number of the encoder card, e.g. 2180.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The encoder type number has been set successfully.
DTAPI_E_INVALID_ARG	The encoder type number is invalid or the type number is valid but the device is not an encoder.

Remarks

DtEncPars::SetVidEncDefaultPars

Set default video encoding parameters for a given video-encoding standard (e.g. H.264) and video standard (e.g. 1080i50). Before calling this routine the encoder type must have been set with **SetEncType**.

```
DTAPI_RESULT DtEncPars::SetVidEncDefaultPars
(
  [in] DtVidEncStd  VidEncStd      // Encoding standard: DT_VIDENC_STD_xxx
  [in] int  VidStd;                // Video standard: DTAPI_VIDSTD_xxx
);
```

Function Arguments

VidEncStd

Video encoding standard, either **DT_VIDENCSTD_H264** or **DT_VIDENCSTD_MP2V**.

VidStd

Video standard coded as a **DTAPI_VIDSTD_xxx** constant.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The video-encoding parameters have been set to their defaults successfully.
DTAPI_E_INVALID_ARG	The video encoding standard or the video standard is invalid.
DTAPI_E_NOT_INITIALIZED	The encoder type number has not been set yet.

Remarks

DtEncPars::ToXml

Serialize the encoding parameters in this **DtEncPars** object into an XML string.

```
DTAPI_RESULT DtEncPars::ToXml
(
 [out] wstring&  XmlString          // Parameters encoded in XML
);
int DtEncPars::GetEncType();
```

Function Arguments

XmlString

XML string receiving the serialized encoding parameters.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The encoding parameters have been initialized successfully.

Remarks

class DtEncVidPars

Class for specifying video encoding parameters.

DtEncVidPars – Public Members

The public members in class **DtEncVidPars** specify the video-input settings, which are parameters that are not directly related to encoding, but rather to the preprocessing of the video. For specifying the video-encoding parameters, the class has a private member 'video-encoding standard', which is accessible through a get and set accessor. The video-encoding parameters are stored in subordinate classes **DtEncVidParsH264** for H.264 and **DtEncVidParsMp2V** for MPEG-2 video. An object for one of these classes is created when **SetVidEncStd** is called.

```
class DtEncVidPars : public DtEncParsBase
{
public:
   // Video input settings
   DtAspectRatio  m_AspectRatio;  // Aspect ratio: 4x3, 16x9 or 14x9
   bool  m_Dithering;             // 10- to 8-bit input dithering on/off
   int  m_HorResolutionRescaled; // Rescale to this horizontal resolution
   InpLossImage  m_InpLossImage;  // Image used when input sync is lossed
   bool  m_InvTelecineDetect;     // Enable inverse telecine operation
   int  m_PixelDepth;             // Number of bits per pixel
   UvSampling  m_UvSampling;      // Chroma sampling: 4:2:0 or 4:2:2
   int  m_VidStd;                 // Video standard: DTAPI_VIDSTD_xxx

   // System parameter, but strongly connected to video encoding parameters
   int  m_EndToEndDelay;          // End-to-end delay in ms

   // Get and set video encoding standard
   DtVidEncStd  GetVidEncStd() const;
   DTAPI_RESULT  SetVidEncStd(DtVidEncStd);

   // Video encoding parameters for H.264 or MPEG-2 video
   DtEncVidParsH264*  H264() const;
   DtEncVidParsMp2V*  Mp2V() const;

   // Constructor
   DtEncVidPars(int EncType = -1);

   // Set encoder type (e.g. 2180)
   // First method to be called when this object is used standalone
   DTAPI_RESULT  SetEncType(int EncType);
};
```

The behavior of the constructor depends on the value of *EncType*. If *EncType* is -1, the default value, then minimal initialization is applied and most members remain uninitialized. If a valid *EncType* is passed to the constructor, **DtEncVidPars** fills the object with default values that are documented in the member descriptions below.

Public Members

m_AspectRatio

Aspect ratio signaled in the encoded video stream.

Value	Meaning
DT_AR_4_3	4x3, not supported for HD
DT_AR_16_9 (default)	16x9
DT_AR_14_9	14x9, not supported for HD

m_Dithering

If **true**, use dithering to reduce the 10-bit video input data to 8 bits. If **false**, truncate 10-bit input words to 8 bits. The default value is truncate (**false**).

m_HorResolutionRescaled

Horizontally rescale the input video to this resolution before encoding. The values allowed are dependent on the width of the input video, as listed in the table below.

Input video width	Value	Meaning
any	**0** (default)	Disable rescaling
1920 pixels	**1440**	Scale by 3/4
	1280	Scale by 2/3
	960	Scale by 1/2
1280 pixels	**960**	Scale by 3/4
	640	Scale by 1/2
720 pixels	**704**	Drop 16 pixels
	640	Scale by 8/9 (square pixels)
	544	Scale by 3/4
	528	Drop 16, scale by 3/4
	480	Scale by 2/3
	352	Drop 16, scale by 1/2

m_InpLossImage

Enumeration that specifies the image to be used for encoding when the input signal to the encoder is lost. The default value is **IL_COLORBARS**.

Value	Meaning
IL_BLACKFRAME	Encode black frames upon loss of input signal
IL_COLORBARS (default)	Encode color bars upon loss of input signal

m_InvTelecineDetect

Detect telecine (film) patterns in the input video and perform the inverse operation.

m_InvTelecineDetect can only be set to true if the frame rate is 59.94Hz or 60Hz. The default value for *m_InvTelecineDetect* is **false**.
To convert 24-Hz film material to 60-Hz video, one easy-to-implement method is to convert two 24-Hz frames to five 60-Hz fields by duplicating one field of the second frame. This technique is called 2:3 pulldown, as the first frame is copied to two fields, while the second frame is converted to three fields.
If setting *m_InvTelecineDetect* is **true**, the 2-3 pattern is automatically detected in the 60Hz video, and if found the video signal is converted back to 24-Hz by deleting duplicated fields.

m_PixelDepth

Pixel depth used for encoding. For the moment only 8-bit is supported and this is the default. In future versions of DTAPI a pixel depth of 10 bits may be supported.

m_UvSampling

Chroma subsampling pattern used. In the current version of DTAPI 4:2:0 is the only supported value.

Value	Meaning
UV_420 (default)	4:2:0
UV_422	4:2:2. At the moment this value is not supported.

m_VidStd

Video standard (not the video-*encoding* standard!) coded as a **DTAPI_VIDSTD_xxx** constant. The encoder hardware expects this video standard at its input. Currently progressive segmented frame formats (e.g. **DTAPI_VIDSTD_1080PSF30**) are not supported.
The default video standard is **DTAPI_VIDSTD_1080I50** for 1080i50.

m_EndToEndDelay

End-to-end delay in ms. The valid values for the Magnum D7Pro (DTA-2180, DTA-2182) are 150ms, 200ms, 350ms and 650ms. The default value is 650ms.

There are specific constraints for each value of end-to-end delay. Please refer to *m_GopSize*, *m_GopNumBPictures*, *m_CodingMode* and **DtEncAudParsAac::***m_Profile* for the details of these constraints.

H264()

Pointer to the H.264 video-encoding parameters. Will return **NULL** if the video-encoding standard is not **DT_VIDENCSTD_H264**.

Mp2V()

Pointer to the MPEG-2 video encoding parameters. Will return **NULL** if the video-encoding standard is not **DT_VIDENCSTD_MP2V**.

SetEncType()

Set the type number of the encoder card for which the parameters are meant. The encoder type number can be read back with **GetEncType** (implemented in **DtEncParsBase**).
This method returns **DTAPI_E_INVALID_ARG** if the encoder type number is invalid, or if the type number is valid but it is not encoder hardware.

DtEncVidPars::CheckValidity

Check the validity of the video encoding parameters.

```
DtEncResult DtEncVidPars::CheckValidity
();
```

Function Arguments

Result

See the Results table listed on the **`DtEncPars::CheckValidity`** page.

Remarks

DtEncVidPars::Es2TpRate

Static function to convert a video elementary-stream bitrate (without transport-packet overhead) to a video transport-packet bitrate (the bitrate of video stream packaged in transport packets).

```
static DTAPI_RESULT DtEncVidPars::Es2TpRate
(
  [in] int  PcrInterval;          // PCR interval in ms
  [in] int  VidStd;               // Video standard: DTAPI_VIDSTD_xxx
  [in] int  EsRate;               // Video elementary-stream bitrate
 [out] int&  TpRate;              // Video transport-packet bitrate
};
```

Function Arguments

PcrInterval

Interval (ms) with which PCR values are scheduled into the video stream. For the valid range, refer to **DtEncMuxPars**.*m_PcrInterval*.

Value	Meaning
0	Another elementary stream is used to carry PCR.

VidStd

Video standard (not the video-*encoding* standard!) coded as a **DTAPI_VIDSTD_xxx** constant.

EsRate

Bitrate of the encoded video elementary stream in bits per second.

TpRate

Output argument that receives the bitrate of the video transport packets in bits per second.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The video transport-packet bitrate was computed successfully.
DTAPI_E_INVALID_ARG	The value in *PcrInterval* or in *EsRate* is not valid.
DTAPI_E_INVALID_VIDSTD	The value in *VidStd* is not a valid video standard.

Remarks

DtEncVidPars::H264

Get a pointer to the embedded H.264 video encoding parameters.

```
DtEncVidParsH264* DtEncVidPars::H264
();
```

Function Arguments

Result

*DtEncVidParsH264**

Pointer to a **DtEncVidParsH264** object containing the H.264 video encoding parameters. If the video encoding standard stored in **DtEncVidPars** is not H.264 (**DT_VIDENCSTD_H264**), NULL is returned.

DtEncVidPars::Mp2V

Get a pointer to the embedded MPEG-2 video encoding parameters.

```
DtEncVidParsMp2V* DtEncVidPars::Mp2V
();
```

Function Arguments

Result

*DtEncVidParsMp2V**

Pointer to a **DtEncVidParsMp2V** object containing the MPEG-2 video encoding parameters. If the video encoding standard stored in **DtEncVidPars** is not MPEG-2 video (**DT_VIDENCSTD_MP2V**), NULL is returned.

DtEncVidPars::SetDefaultsForProfileLevel

Set default video-encoding parameters for a given profile and level.

```
DtEncResult DtEncVidPars::SetDefaultsForProfileLevel
(
  [in] H264Profile  Profile        // H.264 profile: PROFILE_MAIN, …
  [in] H264Level  Level            // H.264 profile: LEVEL_1_0, …
);
DtEncResult DtEncVidPars::SetDefaultsForProfileLevel
(
  [in] Mp2VProfile  Profile        // MPEG-2 video profile: PROFILE_MAIN, …
  [in] Mp2VLevel  Level            // MPEG-2 video profile: LEVEL_1_0, …
);
```

Function Arguments

Profile

Defines the H.264 or MPEG-2 video profile. Please refer to *§1 Profiles and Levels* for a description of profiles, and the values supported for DekTec encoders.

Level

Defines the H.264 or MPEG-2 video profile. Please refer to *§1 Profiles and Levels* for a description of profiles, and the values supported for DekTec encoders.

Result

DtEncResult	Meaning
DT_ENC_OK	The new video encoding parameters have been set successfully.
DT_ENC_E_INV_PROFILE	Invalid H.264 or MPEG-2 video profile, or the profile is not supported by the current encoder hardware.
DT_ENC_E_INV_LEVEL	Invalid H.264 or MPEG-2 video level, or the level is not supported by the specified profile and/or current encoder hardware.
DT_ENC_E_INV_TYPE	The type number set with **SetEncType()** is not valid encoder hardware.
DT_ENC_E_INV_VIDSTD	The value in **DtEncVidPars**::*m_VidStd* is not a valid video standard, not supported by the encoder or incompatible with the current profile and level.
DT_ENC_E_TYPE_NOT_SET	No type number has been set with **SetEncType()**.

Remarks

Parameters **DtEncVidPars**.*m_VidStd*, **DtEncVidPars**.*m_HorResolutionRescaled* and **DtEncVidPars**.*m_AspectRatio*, which have dependencies to the profile and level constraints, are not affected by this method and need to be set separately.

DtEncVidPars::SetVidEncStd, GetVidEncStd

Set the video encoding standard. If *VidEncStd* changes (different from the value stored in the **DtEncVidPars** object), **SetVidEncStd** deletes the old parameters object, if any, and creates a new internal object with encoding-standard specific parameters (either **DtEncVidParsH264** or **DtEncVidParsMp2V**), and initializes these parameters with default values.

Before calling **SetVidEncStd**, set the encoder type and the video standard **DtEncVidPars**::*m_VidStd* to a valid value. This is required so that meaningful defaults can be computed.

The video-encoding standard can be read back with **GetVidEncStd**.

```
DTAPI_RESULT DtEncVidPars::SetVidEncStd
(
  [in] DtVidEncStd  VidEncStd;     // Video encoding standard
};
DtVidEncStd DtEncVidPars::GetVidEncStd();
```

Function Arguments

VidEncStd

Video-encoding standard.

Value	Meaning
DT_VIDENCSTD_H264	H.264 (AVC)
DT_VIDENCSTD_H265	H.265; Not supported by DekTec encoder hardware yet
DT_VIDENCSTD_MP2V	MPEG-2 video

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The new video encoding standard has been set successfully.
DTAPI_E_ENC_TYPE_NOTSET	The encoder type has not been set.
DTAPI_E_INVALID_ARG	The specified video-encoding standard is invalid.
DTAPI_E_INVALID_ENC_TYPE	Encoder type has not been set to a valid DekTec type number, or it is not encoder hardware.

Remarks

If the video encoding standard was already set to the standard specified in *VidEncStd*, then **SetVidEncStd** is a no-operation. In this case default parameters will not be set.

DtEncVidPars::Tp2EsRate

Static function to convert a video transport-packet bitrate (the bitrate of video stream packaged in transport packets) to a video elementary-stream bitrate (without transport-packet overhead).

```
static DTAPI_RESULT DtEncVidPars::Tp2EsRate
(
  [in] int  PcrInterval;          // PCR interval in ms
  [in] int  VidStd;               // Video standard: DTAPI_VIDSTD_xxx
  [in] int  TpRate;               // Video transport-packet bitrate
 [out] int&  EsRate;              // Video elementary-stream bitrate
};
```

Function Arguments

PcrInterval

Interval (ms) with which PCR values are scheduled into the video stream. For the valid range, refer to **DtEncMuxPars**.*m_PcrInterval*.

Value	Meaning
0	Another elementary stream is used to carry PCR.

VidStd

Video standard (not the video-*encoding* standard!) coded as a **DTAPI_VIDSTD_xxx** constant.

TpRate

Bitrate of the video transport packets in bits per second.

EsRate

Output argument that receives the bitrate of the encoded video elementary stream in bits per second.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The video transport-packet bitrate was computed successfully.
DTAPI_E_INVALID_ARG	The value in *PcrInterval* or in *TpRate* is not valid.
DTAPI_E_INVALID_VIDSTD	The value in *VidStd* is not a valid video standard.

Remarks

DtEncVidPars::TpRate

Compute the video transport-packet bitrate (the bitrate of video stream packaged in transport packets) for the video-encoding parameters in the **DtEncVidPars** object. The PCR interval must be specified as an argument.

```
int DtEncVidPars::TpRate
(
  [in] int  PcrInterval;          // PCR interval in ms
};
```

Function Arguments

PcrInterval

Interval (ms) for which PCR values are scheduled into the video transport packets. For the valid range, refer to **DtEncMuxPars**.*m_PcrInterval*.

Value	Meaning
0	Another elementary stream is used to carry PCR.

Result

The computed video transport-packet bitrate. If the encoding parameters or *PcrInterval* are invalid, -1 is returned.

Remarks

class DtEncVidParsH264

Class for specifying H.264 video encoding parameters.

DtEncVidParsH264 – Public Members

The public members in class **`DtEncVidParsH264`** specify the video-encoding parameters for H.264 (AVC).

```
class DtEncVidParsH264 : public DtEncParsBase
{
public:
   // Generic
   H264Profile  m_Profile;          // H.264 profile: PROFILE_MAIN, …
   H264Level  m_Level;              // H.264 profile: LEVEL_1_0, …
   int  m_Bitrate;                  // Bitrate of the encoded video stream
   int  m_VbvDelayMax;              // Maximum VBV delay in milliseconds

   // GOP parameters
   bool  m_ClosedGop;               // Use closed GOPs
   int  m_GopSize;                  // GOP size in #frames (-1=auto)
   int  m_GopNumBPictures;          // Number of B pictures (-1=auto)

   // Specialized H.264 encoding parameters
   bool  m_8x8Transform;            // Enable 8x8 transforms
   bool  m_Cabac;                   // Enable CABAC, use CAVLC otherwise

   // H.264 video encoding parameters, specific to the Magnum D7Pro
   bool  m_AdaptiveQuantization;  // Enable adaptive quantization
   bool  m_ChromaScalingList;     // Enable chroma scaling list
   CodingMode  m_CodingMode;      // Coding mode: CM_FRAME, …
   int  m_IdrFrequency;           // Frequency of IDRs
   bool  m_IntraScoreAvg;         // Use averaged intra scoring
   int  m_QuantizationTable;      // Quantization table to use
   bool  m_WeightedPrediction;    // Enable weighted prediction
};
```

m_Profile

Defines the H.264 profile. Please refer to *§1 Profiles and Levels* for a description of profiles.

Value	Meaning
`PROFILE_CONSTRAINED_BASE`	H.264 Constrained Baseline Profile (CBP)
`PROFILE_MAIN`	H.264 Main Profile (MP)
`PROFILE_HIGH` (default)	H.264 High Profile (HP)
`PROFILE_AVCI50`	AVC-Intra 50 profile: Currently not supported by DekTec encoder hardware.
`PROFILE_AVCI100`	AVC-Intra 100 profile: Currently not supported by DekTec encoder hardware.

m_Level

Defines the H.264 level. Please refer to *§1 Profiles and Levels* for a description of levels.

Value	Meaning
LEVEL_AUTO (default)	Level constraints are ignored
LEVEL_1	Level 1
LEVEL_1.1	Level 1.1
LEVEL_1.2	Level 1.2
LEVEL_1.3	Level 1.3
LEVEL_2	Level 2
LEVEL_2.1	Level 2.1
LEVEL_2.2	Level 2.2
LEVEL_3	Level 3
LEVEL_3.1	Level 3.1
LEVEL_3.2	Level 3.2
LEVEL_4	Level 4
LEVEL_4.1	Level 4.1
LEVEL_4.2	Level 4.2
LEVEL_5	Level 5
LEVEL_5.1	Level 5.1

m_Bitrate

Bitrate of the encoded video in bits per second. The bitrate must be between 256kbps and 80Mbps. The default bitrate is 80Mbps.

m_VbvDelayMax

Maximum VBV delay in milliseconds. The value must be between 30 and 1000ms, with -1 (default) meaning "automatic" maximum VBV delay.

m_ClosedGop

If **true**, close every GOP. If **false**, open GOPs are used. The default is open GOPs (**false**).

m_GopSize

The size of each GOP in number of frames. The valid range is 1 through 300, or -1 (default) indicating "automatic" GOP size. The table below describes the meaning of "automatic GOP size" as a function of the system delay.

System delay	Meaning of value -1 (automatic GOP size)
150ms/250ms/350ms	GOP size is infinite. This means that there will be I-fields only at scene changes and all other fields will be P-fields using Continuous Decoder Refresh (CDR).
650ms	GOP size is 300 frames

m_GopNumBPictures

Number of B pictures between I/P pictures, which specifies the GOP structure of the encoded video (see the table below). A value of -1 (default) indicates that the encoder "automatically" chooses the GOP structure. The table below describes the meaning of "automatic GOP size" as a function of the system delay.

m_GopNumBPictures	GOP structure
-1 (default)	Automatic
0	IP
1	IPB
2	IPBB
3	IPBBB

The following constraints apply as a function of the system delay.

System delay	Maximum number of B pictures
150ms	No B pictures can be generated
200ms/350ms	2
650ms	3 if the frame rate in **DtEncVidPars**::*m_VidStd* is 50, 59.94 or 60Hz. Otherwise 2 B pictures is the maximum.

m_8x8Transform

Enable 8x8 transforms. If 8x8 transforms are not enabled, the less efficient scheme 4x4 is used. The default is **true**.

m_Cabac

Enable CABAC (Context-Adaptive Binary Arithmetic Coding), an advanced form of entropy encoding. If CABAC is not enabled, the simpler scheme CAVLC (Context-Adaptive Variable-Length Coding) is used. The default is CABAC enabled (**true**).

m_AdaptiveQuantization

Enable adaptive quantization. The default value is **true**.

m_ChromaScalingList

Enable chrominance scaling list to get a better video quality for Sarnoff-like content. The default is to disable chrominance scaling (**false**).

m_CodingMode

The coding mode to use.

Value	Meaning
CM_AUTO (default)	Automatically select an appropriate coding mode
CM_FRAME	Frame coding for progressive video
CM_FIELD	Field coding for interlaced video
CM_MBAFF	MBAFF coding for interlaced video

The following constraints apply as a function of the system delay.

System delay	Supported
150ms	Only mode **CM_AUTO** is supported
200ms	**CM_MBAFF** is not supported
350ms	**CM_MBAFF** is not supported
650ms	**CM_FIELD** mode is supported for interlaced content if B pictures are disabled. Otherwise **CM_AUTO** will select **CM_MBAFF** for interlaced or **CM_FRAME** for progressive.

m_IdrFrequency

Frequency of IDRs relative to I frames: 0=No IDR frames, 1=every I-frame, 2=every second I-frame, etc. The valid range is 0 through 255. The default is no IDR frames (value 0).

m_IntraScoreAvg

Use averaged intra score to compute the QP (Quantization Parameter) increase. The default value is **false**.

m_QuantizationTable

Quantization table to use for encoding. A default table (value 0) and five custom tables are defined (value 1 through 5).

m_WeightedPrediction

Enable weighted prediction. The default value is **true**.

class DtEncVidParsMp2V

Class for specifying MPEG-2 video encoding parameters.

DtEncVidParsMp2V – Public Members

The public members in class **`DtEncVidParsMp2V`** specify the video-encoding parameters for MPEG-2 video.

```
class DtEncVidParsMp2V : public DtEncParsBase
{
public:
   // Generic
   Mp2VProfile  m_Profile;         // MPEG-2 video profile: PROFILE_MAIN, …
   Mp2VLevel  m_Level;             // MPEG-2 video profile: LEVEL_1_0, …
   int  m_Bitrate;                 // Bitrate of the encoded video stream
   int  m_VbvDelayMax;             // Maximum VBV delay in milliseconds

   // GOP parameters
   bool  m_ClosedGop;              // Use closed GOPs
   int  m_GopSize;                 // GOP size in #frames (-1=auto)
   int  m_GopNumBPictures;         // Number of B pictures (-1=auto)

   // Specialized MPEG-2 video encoding parameters
   bool  m_AlternateScan;          // Alternate scan for VLC coefficients
   int  m_IntraDcPrecision;        // #bits used for intra-DC values
   IntraVlcFormat  m_IntraVlcFmt; // Format to use for intra-vlc
   bool  m_LowDelayFlag;           // Set the low delay flag
   QScaleType  m_QScaleType;       // Linear or non-linear quantization

   // MPEG-2 video encoding parameters, specific to the Magnum D7Pro
   bool  m_AdaptiveQuantization; // Enable adaptive quantization
   int  m_IdrFrequency;            // Frequency of IDRs
   int  m_QuantizationTable;       // Quantization table to use
};
```

`m_Profile`

Defines the MPEG-2 video profile. Please refer to *§1 Profiles and Levels* for a description of profiles.

Value	Meaning
`PROFILE_SIMPLE`	MPEG-2 video simple profile
`PROFILE_MAIN`	MPEG-2 video Main Profile (MP)
`PROFILE_HIGH` (default)	MPEG-2 video High Profile (HP)
`PROFILE_422P`	MPEG-2 video 422P profile. Not supported by current DekTec encoder hardware.

m_Level

Defines the MPEG-2 video level. Please refer to *§1 Profiles and Levels* for a description of levels.

Value	Meaning
LEVEL_AUTO (default)	Level constraints are ignored
LEVEL_HIGH	High Level (HL)
LEVEL_HIGH1440	High 1440 Level
LEVEL_MAIN	Main Level (ML)

m_Bitrate

Bitrate of the encoded MPEG-2 video in bits per second. The bitrate must be between 512kbps and 80Mbps. The default bitrate is 80Mbps.

m_VbvDelayMax

Maximum VBV delay in milliseconds. The value must be between 30 and 728ms, with -1 (default) meaning "automatic" maximum VBV delay.

m_ClosedGop

If **true**, close every GOP. If **false**, open GOPs are used. The default is open GOPs (**false**).

m_GopSize

The size of each GOP in number of frames. The valid range is 1 through 300, or -1 (default) indicating "automatic" GOP size. The table below describes the meaning of "automatic GOP size" as a function of the system delay.

System delay	Meaning of value -1 (automatic GOP size)
150ms/250ms/350ms	GOP size is infinite. This means that there will be I-fields only at scene changes and all other fields will be P-fields using Continuous Decoder Refresh (CDR).
650ms	GOP size is 132 frames

m_GopNumBPictures

Number of B pictures between I/P pictures, which specifies the GOP structure of the encoded video (see the table below). A value of -1 (default) indicates that the encoder "automatically" chooses the GOP structure. The table below describes the meaning of "automatic GOP size" as a function of the system delay.

m_GopNumBPictures	GOP structure
-1	Automatic
0	IP
1	IPB
2	IPBB
3	IPBBB

The following constraints apply as a function of the system delay.

System delay	Maximum number of B pictures
150ms	No B pictures can be generated
200ms/350ms	1
650ms	2 if the frame rate in **DtEncVidPars**::*m_VidStd* is 50, 59.94 or 60Hz. Otherwise 1 B picture is the maximum.

m_AlternateScan
Use alternate scan pattern for VLC coefficients. The default is **true**.

m_IntraDcPrecision
Number of bits used for intra-DC values: 8...11, or -1 for a 'dynamic' number of bits to use. The default intra-DC precision is 8 bits.

m_IntraVlcFmt
The intra VLC format to use.

Value	Meaning
IV_ALTERNATE (default)	Alternate intra-VLC format
IV_MPEG1	MPEG-1 intra-VLC format

m_LowDelayFlag
Set the low delay flag (no B pictures). The default value is **false**.

m_QScaleType
Type of quantization scale.

Value	Meaning
QS_LINEAR	Linear quantization scale
QS_NONLINEAR (default)	Non-linear quantization scale

m_AdaptiveQuantization
Enable adaptive quantization. The default value is **true**.

m_IdrFrequency
Frequency of IDRs relative to I frames: 0=No IDR frames, 1=every I-frame, 2=every second I-frame, etc. The valid range is 0 through 255. The default is no IDR frames (value 0).

m_QuantizationTable
Quantization table to use for encoding. A default table (value 0) and five custom tables are defined (value 1 through 5).