DTAPI-TS Documentation

Contents

Chapter 1

Welcome

By using this software you are accepting the terms of the End User License Agreement which is available in the "End User License Agreement DTAPI-TS October 2012.pdf" file provided with this software, or from DekTec by e-mailing info@dektec.com.

1.1 Summary

The DTAPI-TS library allows you to easily integrate transport stream analysis in your own application. You can use it to extract DVB-SI or ATSC PSIP information from a stream like a list of services with their names. The library can even extract some basic information from the underlying elementary streams such as the audio/video codec used, the resolution and the number of audio channels.

1.2 Installation

To install DTAPI-TS you'll need to obtain a license from DekTec first. Once you've got a license in the form of a .dtlic file you can install DTAPI-TS.

1.2.1 Windows

On windows you proceed by running DTAPI-TS.exe. You'll be asked to select the license file you've received and also an output directory. Once you've done this you can click on "Extract DTAPI-TS". The installer will proceed to unpack DTAPITS.h and several .lib files to the directory you picked earlier. The following static link libraries are available:

Filename	#bits	Run-time library	Configuration
DTAPITSMD.lib	32	multi-threaded DLL (/MD)	release
DTAPITSMDd.lib	32	multi-threaded DLL (/MD)	debug
DTAPITSMT.lib	32	multi-threaded (/MT)	release
DTAPITSMTd.lib	32	multi-threaded (/MT)	debug
DTAPITS64MD.lib	64	multi-threaded DLL (/MD)	release
DTAPITS64MDd.lib	64	multi-threaded DLL (/MD)	debug
DTAPITS64MT.lib	64	multi-threaded (/MT)	release
DTAPITS64MTd.lib	64	multi-threaded (/MT)	debug

The correct version of the DTAPI-TS library is automatically linked to the application. This is accomplished with pragma directives in DTAPITS.h. Automatic linking can be disabled by defining _DTAPI_DISABLE_AUTO_LINK before including DTAPITS.h.

So, to use a static link library of DTAPI-TS follow these steps:

1. Copy DTAPITS.h and the right version(s) of DTAPITSxxx.lib to your project or to a standard location visible to VC++.
2. Add "#include "DTAPITS.h" to each files that uses DTAPI-TS.
3. Compile your application using compiler settinsg that match those of the lib file.

1.2.2 Linux

On linux you run the binary InstallDTAPITS from the commandline with as first argument the name of the license file. It'll check if the license is valid and if so, extract DTAPITS.h, DTAPITS.o and DTAPITS64.o to the current directory. To use DTAPI-TS in your application simply include DTAPITS.h in your source file and make sure to include DTAPITS.o (or DTAPITS64.o) in your link step.

The C++11 standard has different versions of the Application Binary Interface (ABI). You can select the CXX11 ABI version by using the _GLIBCXX_USE_CXX11_ABI compiler option. If the compiler option is not used, the compiler uses the default setting setted up at compile time of the compiler.

The GCC4.8 libraries are compatible with the CXX11 ABI version 0. The GCC5.1 libraries are compatible with the CXX11 ABI version 1.

If the compiler only supports CXX11 ABI version 0, you should always use the GCC4.8 libraries. If the compiler only supports CXX11 ABI version 1, you should always use the GCC5.1 libraries. If the compiler supports both, you can force the CXX11 ABI version with the _GLIBCXX_USE_CXX11_ABI compiler option.

1.3 Usage

DTAPI-TS usage resolves around a few main classes. First you create an instance of the DtTsLib class. Ignore the arguments, they have the correct default values and are there for checking the license. DtTsLib::CreateDtTsInfoInstance() creates an instance of the second main class, DtTsInfo. The next step is to pick the most useful DtTsInfoInput class for your use case. Pick one, create an instance of it and feed it data using the DtTsInfoInput::NewData function. The results of the analysis are in an instance of the DtTsData class which you can find as the m_Data member of DtTsInfo. See also the examples on how to get started.

Chapter 2

Revision History

The section lists the various changes made to DTAPI-TS.

Revision	Date	Change description
v1.9.0.40	2021.07.02	• Fixed error in Example 4 • Update for July2021 DTAPI changes
v1.8.0.39	2021.05.21	• Update for May2021 DTAPI changes
v1.7.0.38	2021.02.01	• Added support for J2K detection and ES info • Added suport for JPEG-XS detection • Update for Feb2021 DTAPI changes
v1.6.12.37	2020.11.16	• Update TR 101 290 2020: 100ms for PCR-error and PCR repetition error • Fix for crash on descriptor with invalid extended tag • Update for Nov2020 DTAPI changes

v1.5.11.36	2020.08.21	• Update for Aug2020 DTAPI changes
v1.5.10.35	2020.07.07	• Update for July2020 DTAPI changes
v1.5.9.34	2020.05.01	• Update for May2020 DTAPI changes
v1.5.8.33	2020.03.11	• Update for Mar2020 DTAPI changes
v1.5.7.32	2020.02.17	• Update for Feb2020 DTAPI changes
v1.5.6.31	2020.01.08	• Update for Jan2020 DTAPI changes • Fix for possible crash on illegal stream data
v1.5.5.30	2019.10.30	• Update for Oct2019 DTAPI changes
v1.5.4.29	2019.07.25	• Update for July2019 DTAPI changes
v1.5.3.28	2019.06.25	• Fix for potential HEVC parsing crash • Fix for H264 Interlaced/Non interlaced signalling not correctly parsed
v1.5.2.27	2019.05.15	• Fix for potential crash when using DT_STANDARDMODE_ATSC mode • Fix for HEVC Frame rate not always indicated • Fix for H264 videoformat did not indicate Interlaced/Non interlaced signalling

v1.5.1.26	2019.02.15	• Fix for potential crash on older PC's that did not support AVX instruction set
v1.5.0.25	2019.01.24	• Added support for TR 101 290 priority 3 errors • Added missing Linux support for Aac, Ac3, Ac4, Aes, Eac3, Tp, PtsDts, Pes, Pcr, Mpa • Added support for HEVC chroma format and bit depth fields • Fix for possible incorrect resolution for HEVC • Fix for exception during HEVC header parse (seen on Ubuntu 14.04 linker v2.24) • Fix for possible AAC header parse issue • Fix problem with memory deallocation PES packed (Windows only)

v1.4.2.22	2018.06.13	• Added support for timestamped TS packets • Added support for Teletext streams • Added support for Visual Studio 2017 (VC15) • Fix wrong m_Pid and m_TableId in P2_PCR_ACCURANCY callback indication • Fix for failing DecodeFromTable method of DtTableBat class • Fix for extracted resolution error for HEVC due to conformance_window_flag • Fix for DtapiTs::BitPtr::GetBits() Assertion • Fix problem with DEBUG version and assertions in Linux

v1.4.1.21	2017.07.18	• Added support for AC-4 audio in a DVB-stream
v1.3.3.19	2017.01.23	• Fix for rare crash in TsInfoImpl::RemovePidRef
v1.3.2.18	2016.08.24	• Fix for incomplete packet extraction in corner cases involving stuffing tables • Add AddPesPacketCallback(int Pid, DtPesCallback Callback) to .NET wrapper
v1.3.1.17	2016.06.06	• Reported H.264 frame rate was double the actual frame rate in some cases • Detect SMPTE 302M AES3 audio
v1.3.0.16	2016.02.24	• Add support for Visual Studio 2015 (VC14)
v1.2.5.15	2014.09.18	• H.264 / AVC: adjust frame size for cropping • Add DtStructuredTable subclasses and DtDesc∗ classes to .NET wrapper • Fix for possible crash while parsing E-AC3 audio • DtTsData::m_Tables was not accesible via .NET wrapper

v1.2.4.14	2014.06.25	• Descriptor parsing could fail if there were extended descriptors • Parse HEVC streams • Add MSVC 2013 libraries
v1.2.3.13	2014.03.03	• Add TR 101 290 support to .NET wrapper • Add MSVC 2012 libraries • Add code for TS file analyzer as example • Add example on how to use callbacks
v1.2.2.12	2013.08.14	• 1 and 3 segments TMCC files were not recognized correctly • Fix infinite loop if a stream contains a DVB-T2 descriptor • DtTs∗Input classes called Lock() to make sure the DtTsInfo was not changed but never unlocked the object again
v1.2.0.10	2013.04.29	• TR 101 290 support (priority 1 and 2). • Add support for more descriptors via new DtDesc∗ classes. • Compute DtTsData::m_NitTsRate also from a terrestial delivery system descriptor. • DtVideoEsInfo::m_HorzSize and DtVideoEsInfo::m_VertSize could contain garbage in some rare cases, even though DtVideoEsInfo::m_Mask indicated those fields were valid. • Don't parse PAT/CAT when they are on the wrong PID.
		• DtTablePat::DecodeFrom-Section swapped m_Pid and m_ServiceId.

v1.1.0.7	2013.02.25	• Add new classes that help to extract descriptor information from raw table data. • Changes to DtSubTableId to make it easier to filter only on certain values. • Fix for small memory leak when delete-ing a DtTsInfo object.
v1.0.4.5	2012.12.06	• Crash in case a file had sections with more than 4093 bytes.
V1.0.3.3	2012.11.13	• Protect against division by zero in case you set TsRate to 0 on a DtTsPacketInput object, invalid TsRate values are now ignored and replaced by a default. • String conversion between different charsets was broken on Linux.
V1.0.2.2	2012.09.10	• First external release

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 7

Module Documentation

7.1 Structured information from binary descriptors

Classes that help to extract information from descriptors.

Classes

- class DtapiTs::DtDescDvbAc3

  Parsed information from the DVB AC-3 descriptor.
- class DtapiTs::DtDescDvbCDelivery

  Parsed information from the DVB cable delivery system descriptor.
- class DtapiTs::DtDescDvbComponent

  Parsed information from the DVB component descriptor.
- class DtapiTs::DtDescDvbDataBroadcast

  Parsed information from the DVB data broadcast descriptor.
- class DtapiTs::DtDescDvbDataBroadcastId

  Parsed information from the DVB data broadcast id descriptor.
- class DtapiTs::DtDescDvbLinkage

  Parsed information from the DVB linkage descriptor.
- class DtapiTs::DtDescDvbLocalTimeOffset

  Parsed information from the DVB local time offset descriptor.
- class DtapiTs::DtDescDvbMultilingualComponent

  Parsed information from the DVB multilingual component descriptor.
- class DtapiTs::DtDescDvbNetworkName

  Parsed information from the DVB network name descriptor.
- class DtapiTs::DtDescDvbSDelivery

  Parsed information from the DVB satellite delivery system descriptor.
- class DtapiTs::DtDescDvbService

  Parsed information from the DVB service descriptor.
- class DtapiTs::DtDescDvbServiceList

  Parsed information from the DVB service list descriptor.
- class DtapiTs::DtDescDvbSubtitling

  Parsed information from the DVB subtitling descriptor.
- class DtapiTs::DtDescDvbTDelivery

  Parsed information from the DVB terrestrial delivery system descriptor.
- class DtapiTs::DtDescDvbTeletext

  Parsed information from the DVB teletext descriptor.

- class DtapiTs::DtDescMpegCa

  Parsed information from the conditional access descriptor.
- class DtapiTs::DtDescMpegLanguage

  Parsed information from the ISO 639 language descriptor.
- class DtapiTs::DtDescMpegPrivDataIndicator

  Parsed information from private data indicator descriptor.
- class DtapiTs::DtDescMpegRegistration

  Parsed information from the registration descriptor.
- class DtapiTs::DtDescMpegVideoStream

  Parsed information from the video stream descriptor.
- class DtapiTs::DtDescPrivLcn

  Class that can be used to parse logical channel numbers from a (list of) descriptors.
- struct DtapiTs::DtDescPrivLcn::DtLogicalChannelNumber
- struct DtapiTs::DtDescDvbLinkage::EventLinkage
- struct DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage
- struct DtapiTs::DtDescMpegLanguage::LangCode
- struct DtapiTs::DtDescDvbLocalTimeOffset::LocalTimeOffset
- struct DtapiTs::DtDescDvbLinkage::MobileHandOverInfo
- struct DtapiTs::DtDescDvbServiceList::ServiceListItem
- struct DtapiTs::DtDescDvbSubtitling::Subtitling
- struct DtapiTs::DtDescDvbTeletext::Teletext

7.1.1 Detailed Description

Classes that help to extract information from descriptors.

7.2 Structured information from binary tables

Classes that help to extract information from tables / table-sections.

Classes

- struct DtapiTs::DtTablePat::DtProgramMapping

  ServiceId to PMT Pid mapping for a single service.
- class DtapiTs::DtStructuredTable

  Base class for all structured table classes.
- class DtapiTs::DtTableBat

  Structured version of the raw data contained in a Bouquet association table.
- class DtapiTs::DtTableBatInner

  This class holds all descriptors of one sub-loop of a BAT table.
- class DtapiTs::DtTableCat

  Class that can be used to parse all descriptors in the Conditional Access Table.
- class DtapiTs::DtTableNit

  Structured version of the raw data contained in a Network Information Table.
- class DtapiTs::DtTableNitInner

  This class holds all descriptors of one sub-loop of the NIT table.
- class DtapiTs::DtTablePat

  Structured version of the raw data contained in a Program Association Table.
- class DtapiTs::DtTablePmt

  Class that can be used to parse all descriptors in the Program Map Table.
- class DtapiTs::DtTablePmtInner

  This class holds all descriptors of compoment in a PMT table.
- class DtapiTs::DtTableSdt

  Structured version of the raw data contained in a Service description table.
- class DtapiTs::DtTableSdtInner

  This class holds all descriptors of one sub-loop of the SDT table.
- class DtapiTs::DtTableTdt

  Structured version of the raw data contained in a Time and Date section.
- class DtapiTs::DtTableTot

  Structured version of the raw data contained in a Time Offset Table.

7.2.1 Detailed Description

Classes that help to extract information from tables / table-sections.

Chapter 8

Namespace Documentation

8.1 DtapiTs Namespace Reference

All DTAPITS code lives in the DtapiTs namespace.

Classes

- class DtAacEsInfo
- class DtAc3EsInfo
- class DtAc4EsInfo
- class DtAudioEsInfo

  Information about an audio elementary stream extracted from PES packets.
- class DtAudioEsInfo2
- class DtBitrate

  Some statistics about the bitrate of a PID, service or transport stream.
- class DtBitrateSettings

  Settings for the bitrate measurement sliding window.
- class DtCallback1

  Helper class to store a function pointer to a function with one argument and a pointer to the associated data.
- class DtCallback2

  Helper class to store a function pointer to a function with two arguments and a pointer to the associated data.
- class DtCallback3

  Helper class to store a function pointer to a function with 3 arguments and a pointer to the associated data.
- class DtCaSystem

  Descripion of a conditional access system.
- class DtDescDvbAc3

  Parsed information from the DVB AC-3 descriptor.
- class DtDescDvbCDelivery

  Parsed information from the DVB cable delivery system descriptor.
- class DtDescDvbComponent

  Parsed information from the DVB component descriptor.
- class DtDescDvbDataBroadcast

  Parsed information from the DVB data broadcast descriptor.
- class DtDescDvbDataBroadcastId

  Parsed information from the DVB data broadcast id descriptor.
- class DtDescDvbLinkage

  Parsed information from the DVB linkage descriptor.

- class DtDescDvbLocalTimeOffset

  Parsed information from the DVB local time offset descriptor.
- class DtDescDvbMultilingualComponent

  Parsed information from the DVB multilingual component descriptor.
- class DtDescDvbNetworkName

  Parsed information from the DVB network name descriptor.
- class DtDescDvbSDelivery

  Parsed information from the DVB satellite delivery system descriptor.
- class DtDescDvbService

  Parsed information from the DVB service descriptor.
- class DtDescDvbServiceList

  Parsed information from the DVB service list descriptor.
- class DtDescDvbSubtitling

  Parsed information from the DVB subtitling descriptor.
- class DtDescDvbTDelivery

  Parsed information from the DVB terrestrial delivery system descriptor.
- class DtDescDvbTeletext

  Parsed information from the DVB teletext descriptor.
- class DtDescMpegCa

  Parsed information from the conditional access descriptor.
- class DtDescMpegLanguage

  Parsed information from the ISO 639 language descriptor.
- class DtDescMpegPrivDataIndicator

  Parsed information from private data indicator descriptor.
- class DtDescMpegRegistration

  Parsed information from the registration descriptor.
- class DtDescMpegVideoStream

  Parsed information from the video stream descriptor.
- class DtDescPrivLcn

  Class that can be used to parse logical channel numbers from a (list of) descriptors.
- class DtDescriptor

  DtDescriptor represents a single descriptor within one of the tables.
- class DtDvbCNitInfo

  DVB-C delivery system information as extracted from the NIT.
- class DtDvbShModInfo

  DVB-SH modulation info.
- class DtDvbShNitInfo

  DVB-SH delivery system information as extracted from the NIT.
- class DtDvbShOfdmInfo

  DVB-SH OFDM modulation info.
- class DtDvbShTdmInfo

  DVB-SH TDM modulation info.
- class DtDvbSNitInfo

  DVB-S delivery system information as extracted from the NIT.
- class DtDvbT2CellInfo

  Information about a DVB-T2 cell.
- class DtDvbT2NitInfo

  DVB-T2 delivery system information as extracted from the NIT.
- class DtDvbT2SubCellInfo

  Information about a single DVB-T2 sub-cell.
- class DtDvbTNitInfo

DVB-T delivery system information as extracted from the NIT.

- class DtEac3EsInfo
- class DtEsInfoBase
- class DtHeAacEsInfo
- class DtJitterPoint
- class DtMpaEsInfo
- class DtPcr

  Class representing PCR timestamp:
- class DtPcrInfo

  Some statistics about the bitrate of a PID, service or transport stream.
- class DtPes
- class DtPidInfo

  Class that contains general information about one PID.
- class DtPtsDts

  Class representing PTS/DTS timestamp:
- class DtServiceComponentInfo

  Information about a service component.
- class DtServiceInfo

  Information about a service.
- class DtStructuredTable

  Base class for all structured table classes.
- class DtSubTableId

  Unique identifier for each sub-table.
- class DtTable

  Binary repesentation of an SI table.
- class DtTableBat

  Structured version of the raw data contained in a Bouquet association table.
- class DtTableBatInner

  This class holds all descriptors of one sub-loop of a BAT table.
- class DtTableCat

  Class that can be used to parse all descriptors in the Conditional Access Table.
- class DtTableNit

  Structured version of the raw data contained in a Network Information Table.
- class DtTableNitInner

  This class holds all descriptors of one sub-loop of the NIT table.
- class DtTablePat

  Structured version of the raw data contained in a Program Association Table.
- class DtTablePmt

  Class that can be used to parse all descriptors in the Program Map Table.
- class DtTablePmtInner

  This class holds all descriptors of compoment in a PMT table.
- class DtTableSdt

  Structured version of the raw data contained in a Service description table.
- class DtTableSdtInner

  This class holds all descriptors of one sub-loop of the SDT table.
- class DtTableSection

  Binary repesentation of one section of an SI table.
- class DtTableTdt

  Structured version of the raw data contained in a Time and Date section.
- class DtTableTot

  Structured version of the raw data contained in a Time Offset Table.

- class DtTimeDiff

  Difference between two timestamps.

- class DtTimestamp

  Abstract timestamp, do not rely on the internal representation.

- class DtTp
- class DtTr101290

  Base class for TR 101 290 support.

- class DtTr101290Error

  Information about a single TR 101 290 indicator.

- class DtTsData

  Main DtapiTs class that contains all data extracted from a transport stream.

- class DtTsInfo

  Main DtapiTs class that is used for setting parameters, adding callbacks, passing in the transport stream buffer (via a DtTsInfoInput object) and finally reading back the results.

- class DtTsInfoInput

  Abstract base class for splitting (large) input buffers into timestamped packets.

- class DtTsLib

  Main DtapiTs class used to create instances of the analysis classes.

- class DtTsPacketInput

  Input class that handles a transport stream without any timestamps.

- class DtTsTimestampedPacketInput

  Input class that handles a transport stream packets with timestamps as delivered by DTAPI when a DtInpChannel is set to DTAPI_RXMODE_TIMESTAMP32.

- class DtTsTransparentInput

  Input class that handles timestamped transparent packets as delivered by DTAPI when a DtInpChannel is set to DTAPI_RXMODE_STTRP|DTAPI_RXMODE_TIMESTAMP32.

- class DtVideoAspectRatio
- class DtVideoEsAvcInfo
- class DtVideoEsInfo

  Information about a video elementary stream extracted from PES packets.

Typedefs

- typedef std::vector< DtCaSystem > DtCaSystemList

  List of used conditional access systems.

- typedef std::vector
  < DtJitterPoint > **DtPcrJitter**

Enumerations

- enum DtAacObjType {
  DT_AACOBJTYPE_NULL = 0,
  DT_AACOBJTYPE_AAC_MAIN = 1,
  DT_AACOBJTYPE_AAC_LC = 2,
  DT_AACOBJTYPE_AAC_SSR = 3,
  DT_AACOBJTYPE_LTP = 4,
  DT_AACOBJTYPE_SBR = 5,
  DT_AACOBJTYPE_AAC_SCALABLE = 6,
  DT_AACOBJTYPE_TWINVQ = 7,
  DT_AACOBJTYPE_CELP = 8,
  DT_AACOBJTYPE_HVXC = 9,
  DT_AACOBJTYPE_RESV1 = 10,
  DT_AACOBJTYPE_RESV2 = 11,
  DT_AACOBJTYPE_TTSI = 12,
  DT_AACOBJTYPE_MAIN_SYNTHETIC = 13,
  DT_AACOBJTYPE_WAVETABLE_SYNTHETIC = 14,
  DT_AACOBJTYPE_GENERAL_MIDI = 15,
  DT_AACOBJTYPE_ALGO_SYNTHESIS_AND_AUDIO_FXE = 16,
  DT_AACOBJTYPE_ER_AAC_LC = 17,
  DT_AACOBJTYPE_RESV3 = 18,
  DT_AACOBJTYPE_ER_AAC_LTP = 19,
  DT_AACOBJTYPE_ER_AAC_SCALABLE = 20,
  DT_AACOBJTYPE_ER_TWINVQ = 21,
  DT_AACOBJTYPE_ER_BASC = 22,
  DT_AACOBJTYPE_ER_AAC_LD = 23,
  DT_AACOBJTYPE_ER_CELP = 24,
  DT_AACOBJTYPE_ER_HVXC = 25,
  DT_AACOBJTYPE_ER_HILN = 26,
  DT_AACOBJTYPE_ER_PARAMETRIC = 27,
  DT_AACOBJTYPE_SSC = 28,
  DT_AACOBJTYPE_PS = 29,
  DT_AACOBJTYPE_MPEG_SURROUND = 30,
  DT_AACOBJTYPE_ESCAPE = 31,
  DT_AACOBJTYPE_LAYER1 = 32,
  DT_AACOBJTYPE_LAYER2 = 33,
  DT_AACOBJTYPE_LAYER3 = 34,
  DT_AACOBJTYPE_DST = 35,
  DT_AACOBJTYPE_ALS = 36,
  DT_AACOBJTYPE_SLS = 37,
  DT_AACOBJTYPE_SLS_NON_CORE = 38,
  DT_AACOBJTYPE_ER_AAC_ELD = 39,
  DT_AACOBJTYPE_SMR_SIMPLE = 40,
  DT_AACOBJTYPE_SMR_MAIN = 41 }

  AAC (exended-)object type, this indicates how the audio is encoded in an AAC stream.

- enum DtAacProfile {
  DT_AACPROFILE_UNKNOWN,
  DT_AACPROFILE_LOW_COMPLEXITY,
  DT_AACPROFILE_MAIN,
  DT_AACPROFILE_SCALABLE_SAMPLING_RATE }

  AAC profile that is used for the audio stream.

- enum DTAPITS_RESULT {

DTAPITS_OK,
DTAPITS_E_INVALID_BUF,
DTAPITS_E_TABLEID_MISMATCH,
DTAPITS_E_MISSING_DATA,
DTAPITS_E_INVALID_DESC_LEN,
DTAPITS_E_CRC_MISMATCH,
DTAPITS_E_INVALID_TAG,
DTAPITS_E_INVALID_PDS,
DTAPITS_E_INVALID_DESC,
DTAPITS_E_DESC_NOT_FOUND,
DTAPITS_E_DESC_TOO_SHORT,
DTAPITS_E_INVALID_FIELD,
DTAPITS_E_INVALID_ARG,
DTAPITS_E_NOT_SUPPORTED,
DTAPITS_E_PARSE_ERROR }

List of return values from various DTAPI-TS functions.

- enum DtAtscCcType {
  DT_CCTYPE_EIA608,
  DT_CCTYPE_EIA708 }

  Atsc closed caption type.

- enum DtAudioMode {
  DT_AUDIOMODE_UNKNOWN,
  DT_AUDIOMODE_STEREO,
  DT_AUDIOMODE_JOINT_STEREO,
  DT_AUDIOMODE_DUAL,
  DT_AUDIOMODE_MONO,
  DT_AUDIOMODE_AC3_CH1CH2,
  DT_AUDIOMODE_AC3_CENTER,
  DT_AUDIOMODE_AC3_LR,
  DT_AUDIOMODE_AC3_LCR,
  DT_AUDIOMODE_AC3_LRS,
  DT_AUDIOMODE_AC3_LCRS,
  DT_AUDIOMODE_AC3_LR_SL_SR,
  DT_AUDIOMODE_AC3_LCR_SL_SR,
  DT_AUDIOMODE_AAC_CF,
  DT_AUDIOMODE_AAC_LF_RF,
  DT_AUDIOMODE_AAC_CF_LF_RF,
  DT_AUDIOMODE_AAC_CF_LF_RF_RS,
  DT_AUDIOMODE_AAC_CF_RF_LF_LR_RR,
  DT_AUDIOMODE_AAC_CF_RF_LF_LR_RR_FLF,
  DT_AUDIOMODE_AAC_CR_RF_LF_ROF_LOF_LR_RR_FLF }

  Audio modes of the audio in elementary streams.

- enum DtDeliverySystem {
  DT_DELIVERYSYSTEM_CABLE,
  DT_DELIVERYSYSTEM_SATELLITE,
  DT_DELIVERYSYSTEM_TERRESTRIAL,
  DT_DELIVERYSYSTEM_T2,
  DT_DELIVERYSYSTEM_SH,
  DT_DELIVERYSYSTEM_INVALID = -1 }

  Types of delivey system descriptor as can be found in the DVB NIT table.

- enum **DtDescriptorType** {

DT_DESC_DVB_AC3,
DT_DESC_DVB_AC4,
DT_DESC_DVB_DVB_C_DELIVERY,
DT_DESC_DVB_COMPONENT,
DT_DESC_DVB_EAC3,
DT_DESC_DVB_DATA_BROADCAST,
DT_DESC_DVB_DATA_BROADCAST_ID,
DT_DESC_DVB_LINKAGE,
DT_DESC_DVB_LOCAL_TIME_OFFSET,
DT_DESC_DVB_MULTILINGUAL_COMPONENT,
DT_DESC_DVB_NETWORK_NAME,
DT_DESC_DVB_DVB_S_DELIVERY,
DT_DESC_DVB_SERVICE,
DT_DESC_DVB_SERVICE_LIST,
DT_DESC_DVB_SUBTITLING,
DT_DESC_DVB_DVB_T_DELIVERY,
DT_DESC_DVB_TELETEXT,
DT_DESC_MPEG_CA,
DT_DESC_MPEG_LANGUAGE,
DT_DESC_MPEG_PRIV_DATA_INDICATOR,
DT_DESC_MPEG_REGISTRATION,
DT_DESC_MPEG_VIDEO_STREAM,
DT_DESC_PRIV_LCN,
DT_DESC_UNKNOWN = -1 }

- enum DtDvbT2MisoMode {
  DT_T2MISO_SISO,
  DT_T2MISO_MISO,
  DT_T2MISO_UNK }

  SIDO/MISO mode for DVB-T2.s.

- enum DtFecOuter {
  DT_FECOUTER_NONE,
  DT_FECOUTER_RS_204_188,
  DT_FECOUTER_UNK }

  Outer forwared error correction as specified in the cable delivery descriptor.

- enum DtGuardInterval {
  DT_SH_GUARDINTERVAL_1_32,
  DT_SH_GUARDINTERVAL_1_16,
  DT_SH_GUARDINTERVAL_1_8,
  DT_SH_GUARDINTERVAL_1_4 }

  Guard interval for DVB-SH.

- enum DtMpaLayer {
  DT_MPA_LAYER_INVALID = -1,
  DT_MPA_LAYER_1,
  DT_MPA_LAYER_2,
  DT_MPA_LAYER_3 }

  MPEG audio layer.

- enum DtMpaVersion {
  DT_MPA_VERSION_INVALID = -1,
  DT_MPA_VERSION_1,
  DT_MPA_VERSION_2,
  DT_MPA_VERSION_2_5 }

  MPEG Audio version.

- enum DtPolarization {
  DT_POLAR_LIN_HOR,
  DT_POLAR_LIN_VERT,
  DT_POLAR_CIRC_LEFT,
  DT_POLAR_CIRC_RIGHT }

Polarization of the transmitted signal.

- enum DtRollOff {
  DT_SH_ROLLOFF_35,
  DT_SH_ROLLOFF_25,
  DT_SH_ROLLOFF_15,
  DT_SH_ROLLOFF_UNK }

  Roll-off factor for DVB-SH.
- enum DtScrambling {
  DT_SCRAMBLING_NONE,
  DT_SCRAMBLING_RESERVED,
  DT_SCRAMBLING_EVEN,
  DT_SCRAMBLING_ODD }

  How the last transport packet on a given pid was scrambled.
- enum DtServiceType {
  DT_SERVICETYPE_UNKNOWN,
  DT_SERVICETYPE_TELEVISION,
  DT_SERVICETYPE_RADIO }

  Very course type of service, useful for quickly identifying the content type.
- enum DtShBandwidth {
  DT_SH_BANDWITH_8MHZ,
  DT_SH_BANDWITH_7MHZ,
  DT_SH_BANDWITH_6MHZ,
  DT_SH_BANDWITH_5MHZ,
  DT_SH_BANDWITH_1_7MHZ,
  DT_SH_BANDWITH_UNK }

  Bandwidth for DVB-SH.
- enum DtShCodeRate {
  DT_SH_CODERATE_1_5,
  DT_SH_CODERATE_2_9,
  DT_SH_CODERATE_1_4,
  DT_SH_CODERATE_2_7,
  DT_SH_CODERATE_1_3,
  DT_SH_CODERATE_1_3_COMPL,
  DT_SH_CODERATE_2_5,
  DT_SH_CODERATE_2_5_COMPL,
  DT_SH_CODERATE_1_2,
  DT_SH_CODERATE_1_2_COMPL,
  DT_SH_CODERATE_2_3,
  DT_SH_CODERATE_2_3_COMPL,
  DT_SH_CODERATE_RESERVED }

  Code rate for DVB-SH.
- enum DtShModMode {
  DT_SH_MOD_QPSK,
  DT_SH_MOD_8PSK,
  DT_SH_MOD_16APSK,
  DT_SH_MOD_UNK }

  Modulation modi supported by DVB-SH.
- enum DtShModType {
  DT_SH_MOD_TDM,
  DT_SH_MOD_OFDM }

  Modulation types supported by DVB-SH.
- enum DtStandardMode {
  DT_STANDARDMODE_UNK,
  DT_STANDARDMODE_ATSC,
  DT_STANDARDMODE_DVB,
  DT_STANDARDMODE_DVB_RCS }

Used to set the mode in which the DtapiTs library operates.

- enum DtStreamType {
  DT_STREAMTYPE_INVALID = -1,
  DT_STREAMTYPE_UNKNOWN = 0x00000000,
  DT_STREAMTYPE_MPEG1_VIDEO = 0x00000001,
  DT_STREAMTYPE_MPEG2_VIDEO = 0x00000002,
  DT_STREAMTYPE_MPEG1_AUDIO = 0x00000003,
  DT_STREAMTYPE_MPEG2_AUDIO = 0x00000004,
  DT_STREAMTYPE_PRIV_SECTIONS = 0x00000005,
  DT_STREAMTYPE_PRIV_DATA = 0x00000006,
  DT_STREAMTYPE_MHEG = 0x00000007,
  DT_STREAMTYPE_DSMCC = 0x00000008,
  DT_STREAMTYPE_H_222_1 = 0x00000009,
  DT_STREAMTYPE_13818_6_TA = 0x0000000A,
  DT_STREAMTYPE_13818_6_TB = 0x0000000B,
  DT_STREAMTYPE_13818_6_TC = 0x0000000C,
  DT_STREAMTYPE_13818_6_TD = 0x0000000D,
  DT_STREAMTYPE_AUX = 0x0000000E,
  DT_STREAMTYPE_AAC = 0x0000000F,
  DT_STREAMTYPE_MPEG4_VIDEO = 0x00000010,
  DT_STREAMTYPE_HEAAC = 0x00000011,
  DT_STREAMTYPE_MPEG4_PES = 0x00000012,
  DT_STREAMTYPE_MPEG4_SECT = 0x00000013,
  DT_STREAMTYPE_13818_6_SDP = 0x00000014,
  DT_STREAMTYPE_METADATA_PES = 0x00000015,
  DT_STREAMTYPE_METADATA_SECT = 0x00000016,
  DT_STREAMTYPE_METADATA_DC = 0x00000017,
  DT_STREAMTYPE_METADATA_OC = 0x00000018,
  DT_STREAMTYPE_METADATA_SDP = 0x00000019,
  DT_STREAMTYPE_IPMP_STREAM_MP2 = 0x0000001A,
  DT_STREAMTYPE_AVC_VIDEO = 0x0000001B,
  DT_STREAMTYPE_J2K_VIDEO = 0x00000021,
  DT_STREAMTYPE_HEVC_VIDEO = 0x00000024,
  DT_STREAMTYPE_JPEGXS_VIDEO = 0x00000032,
  DT_STREAMTYPE_IPMP_STREAM = 0x0000007F,
  DT_STREAMTYPE_ATSC_DIGICYPH2 = 0x00000080,
  DT_STREAMTYPE_ATSC_AC3 = 0x00000081,
  DT_STREAMTYPE_SCTE_SUBTITLE = 0x00000082,
  DT_STREAMTYPE_SCTE_ISOCHR_DATA = 0x00000083,
  DT_STREAMTYPE_ATSC_EAC3 = 0x00000087,
  DT_STREAMTYPE_WM9_AUDIO = 0x000000E6,
  DT_STREAMTYPE_VC1_VIDEO = 0x000000EA,
  DT_STREAMTYPE_DVB_AC3 = 0x00000100,
  DT_STREAMTYPE_DVB_EAC3 = 0x00000101,
  DT_STREAMTYPE_DVB_TELETEXT = 0x00000102,
  DT_STREAMTYPE_DVB_MPE = 0x00000103,
  DT_STREAMTYPE_DVB_DATA_CAROUSEL = 0x00000104,
  DT_STREAMTYPE_DVB_INT = 0x00000105,
  DT_STREAMTYPE_SMPTE_AES3 = 0x00000106,
  DT_STREAMTYPE_DVB_AC4 = 0x00000107 }

Type of content in a given pid.

- enum DtTableType {

DT_TABLETYPE_UNKNOWN,
DT_TABLETYPE_PAT,
DT_TABLETYPE_CAT,
DT_TABLETYPE_PMT,
DT_TABLETYPE_TSDT,
DT_TABLETYPE_EMM,
DT_TABLETYPE_ECM,
DT_TABLETYPE_DVB_NITACT,
DT_TABLETYPE_DVB_NITOTH,
DT_TABLETYPE_DVB_SDTACT,
DT_TABLETYPE_DVB_SDTOTH,
DT_TABLETYPE_DVB_BAT,
DT_TABLETYPE_DVB_EITACT,
DT_TABLETYPE_DVB_EITACTS,
DT_TABLETYPE_DVB_EITOTH,
DT_TABLETYPE_DVB_EITOTHS,
DT_TABLETYPE_DVB_TDT,
DT_TABLETYPE_DVB_RST,
DT_TABLETYPE_DVB_ST,
DT_TABLETYPE_DVB_TOT,
DT_TABLETYPE_DVB_DIT,
DT_TABLETYPE_DVB_SIT,
DT_TABLETYPE_DVB_RNT,
DT_TABLETYPE_DVB_INT,
DT_TABLETYPE_DVB_RCS_RMT,
DT_TABLETYPE_DVB_RCS_SCT,
DT_TABLETYPE_DVB_RCS_FCT,
DT_TABLETYPE_DVB_RCS_TCT,
DT_TABLETYPE_DVB_RCS_SPT,
DT_TABLETYPE_DVB_RCS_CMT,
DT_TABLETYPE_DVB_RCS_TBTP,
DT_TABLETYPE_DVB_RCS_PCRPP,
DT_TABLETYPE_DVB_RCS_TMST,
DT_TABLETYPE_DVB_RCS_TIM,
DT_TABLETYPE_DVB_RCS_LL_FEC_PDT,
DT_TABLETYPE_ATSC_MGT,
DT_TABLETYPE_ATSC_TVCT,
DT_TABLETYPE_ATSC_CVCT,
DT_TABLETYPE_ATSC_RRT,
DT_TABLETYPE_ATSC_EIT,
DT_TABLETYPE_ATSC_ETT,
DT_TABLETYPE_ATSC_STT,
DT_TABLETYPE_ATSC_DCCT,
DT_TABLETYPE_ATSC_DCCSCT }

List of table types.

- enum DtTr101290Bitmask {

DT_ERR_B_P1_TS_SYNC_LOSS = 1<<DT_ERR_P1_TS_SYNC_LOSS,
DT_ERR_B_P1_SYNC_BYTE = 1<<DT_ERR_P1_SYNC_BYTE,
DT_ERR_B_P1_PAT_2 = 1<<DT_ERR_P1_PAT_2,
DT_ERR_B_P1_CONTINUITY_COUNTER = 1<<DT_ERR_P1_CONTINUITY_COUNTER,
DT_ERR_B_P1_PMT_2 = 1<<DT_ERR_P1_PMT_2,
DT_ERR_B_P1_PID = 1<<DT_ERR_P1_PID,
DT_ERR_B_P2_TRANSPORT = 1<<DT_ERR_P2_TRANSPORT,
DT_ERR_B_P2_CRC = 1<<DT_ERR_P2_CRC,
DT_ERR_B_P2_PCR_REPETITION = 1<<DT_ERR_P2_PCR_REPETITION,
DT_ERR_B_P2_PCR_DISC_IND = 1<<DT_ERR_P2_PCR_DISC_IND,
DT_ERR_B_P2_PCR_ACCURACY = 1<<DT_ERR_P2_PCR_ACCURACY,
DT_ERR_B_P2_PTS = 1<<DT_ERR_P2_PTS,
DT_ERR_B_P2_CAT = 1<<DT_ERR_P2_CAT,
DT_ERR_B_P3_NIT_ACTUAL = 1<<DT_ERR_P3_NIT_ACTUAL,
DT_ERR_B_P3_NIT_OTHER = 1<<DT_ERR_P3_NIT_OTHER,
DT_ERR_B_P3_SI_REPETITION = 1<<DT_ERR_P3_SI_REPETITION,
DT_ERR_B_P3_BUFFER = 1<<DT_ERR_P3_BUFFER,
DT_ERR_B_P3_UNREFERENCED_PID = 1<<DT_ERR_P3_UNREFERENCED_PID,
DT_ERR_B_P3_SDT_ACTUAL = 1<<DT_ERR_P3_SDT_ACTUAL,
DT_ERR_B_P3_SDT_OTHER = 1<<DT_ERR_P3_SDT_OTHER,
DT_ERR_B_P3_EIT_ACTUAL = 1<<DT_ERR_P3_EIT_ACTUAL,
DT_ERR_B_P3_EIT_OTHER = 1<<DT_ERR_P3_EIT_OTHER,
DT_ERR_B_P3_EIT_PF = 1<<DT_ERR_P3_EIT_PF,
DT_ERR_B_P3_RST = 1<<DT_ERR_P3_RST,
DT_ERR_B_P3_TDT = 1<<DT_ERR_P3_TDT,
DT_ERR_B_P3_EMPTY_BUFFER = 1<<DT_ERR_P3_EMPTY_BUFFER,
DT_ERR_B_P3_DATA_DELAY = 1<<DT_ERR_P3_DATA_DELAY,
DT_ERR_B_P1,
DT_ERR_B_P2,
DT_ERR_B_P3,
DT_ERR_B_ALL = DT_ERR_B_P1 | DT_ERR_B_P2 | DT_ERR_B_P3 }

Bitmask for the TR 101 290 errors.

- enum DtTr101290Indicator {

DT_ERR_P1_TS_SYNC_LOSS,
DT_ERR_P1_SYNC_BYTE,
DT_ERR_P1_PAT_2,
DT_ERR_P1_CONTINUITY_COUNTER,
DT_ERR_P1_PMT_2,
DT_ERR_P1_PID,
DT_ERR_P2_TRANSPORT,
DT_ERR_P2_CRC,
DT_ERR_P2_PCR_REPETITION,
DT_ERR_P2_PCR_DISC_IND,
DT_ERR_P2_PCR_ACCURACY,
DT_ERR_P2_PTS,
DT_ERR_P2_CAT,
DT_ERR_P3_NIT_ACTUAL,
DT_ERR_P3_NIT_OTHER,
DT_ERR_P3_SI_REPETITION,
DT_ERR_P3_BUFFER,
DT_ERR_P3_UNREFERENCED_PID,
DT_ERR_P3_SDT_ACTUAL,
DT_ERR_P3_SDT_OTHER,
DT_ERR_P3_EIT_ACTUAL,
DT_ERR_P3_EIT_OTHER,
DT_ERR_P3_EIT_PF,
DT_ERR_P3_RST,
DT_ERR_P3_TDT,
DT_ERR_P3_EMPTY_BUFFER,
DT_ERR_P3_DATA_DELAY,
DT_ERR_MAX }

Unique identifiers for all TR 101 290 error conditions.

- enum DtTransmissionMode {
  DT_SH_TRANSMODE_1K,
  DT_SH_TRANSMODE_2K,
  DT_SH_TRANSMODE_4K,
  DT_SH_TRANSMODE_8K }

  Transmission mode for DVB-SH.
- enum DtVideoChromaFormat {
  DT_CHROMAFORMAT_INVALID,
  DT_CHROMAFORMAT_420,
  DT_CHROMAFORMAT_422,
  DT_CHROMAFORMAT_424,
  DT_CHROMAFORMAT_444,
  DT_CHROMAFORMAT_MONO }

  Video chroma format.
- enum DtWeFlag {
  DT_WEFLAG_EAST,
  DT_WEFLAG_WEST }

  Flag indicating whether the satellite position is in the western or eastern part of the orbit.

Functions

- std::wstring DtAacObjType2String (DtAacObjType ObjType, DtAacObjType ExObjType)

  Convert DtAacObjType values to a readable string.
- std::wstring DtAacProfile2String (DtAacProfile Profile)

  Convert a DtAacProfile value to a readable string.
- std::wstring DtAtscCcType2String (DtAtscCcType Type)

  Convert a DtAtscCcType value to a readable string.

- std::wstring DtAudioMode2String (DtAudioMode ChMode)

  Convert a DtAudioMode value to a readable string.
- std::wstring DtCaSystemId2String (int CaSystemId)

  Convert a CaSystemId value to a readable string.
- std::wstring DtMpaLayer2String (DtMpaLayer Layer)

  Convert a DtMpaLayer value to a readable string.
- std::wstring DtMpaVersion2String (DtMpaVersion Version)

  Convert a DtMpaVersion value to a readable string.
- std::wstring **DtParseDvbDescriptorString** (uint8_t *Buf, int Len)
- std::wstring **DtParseDvbDescriptorStringWithLength** (uint8_t *Buf, int &Len)
- std::wstring DtVideoChromaFormat2String (DtVideoChromaFormat Format)

  Convert a DtVideoChromaFormat value to a readable string.

8.1.1 Detailed Description

All DTAPITS code lives in the DtapiTs namespace.

8.1.2 Enumeration Type Documentation

8.1.2.1 enum DtapiTs::DtAacObjType

AAC (exended-)object type, this indicates how the audio is encoded in an AAC stream.

Enumerator:

DT_AACOBJTYPE_NULL Null.

DT_AACOBJTYPE_AAC_MAIN AAC Main.

DT_AACOBJTYPE_AAC_LC AAC Low complexity.

DT_AACOBJTYPE_AAC_SSR AAC Scalable Sample Rate.

DT_AACOBJTYPE_LTP Long Term prediction.

DT_AACOBJTYPE_SBR Spectral Band Replication.

DT_AACOBJTYPE_AAC_SCALABLE AAC Scalable.

DT_AACOBJTYPE_TWINVQ TwinVQ.

DT_AACOBJTYPE_CELP Code Excited Linear Prediction.

DT_AACOBJTYPE_HVXC Harmonic Vector eXcitation Coding.

DT_AACOBJTYPE_RESV1 (Reserved)

DT_AACOBJTYPE_RESV2 (Reserved)

DT_AACOBJTYPE_TTSI Text-To-Speech Interface.

DT_AACOBJTYPE_MAIN_SYNTHETIC Main synthesis.

DT_AACOBJTYPE_WAVETABLE_SYNTHETIC Wavetable synthesis.

DT_AACOBJTYPE_GENERAL_MIDI General MIDI.

DT_AACOBJTYPE_ALGO_SYNTHESIS_AND_AUDIO_FXE Algorithmic Synthesis and Audio Effects.

DT_AACOBJTYPE_ER_AAC_LC ER AAC LC.

DT_AACOBJTYPE_RESV3 (Reserved)

DT_AACOBJTYPE_ER_AAC_LTP ER AAC LTP.

DT_AACOBJTYPE_ER_AAC_SCALABLE ER AAC Scalable.

DT_AACOBJTYPE_ER_TWINVQ ER TwinVQ.

DT_AACOBJTYPE_ER_BASC ER Bit-Sliced Arithmetic Coding.

DT_AACOBJTYPE_ER_AAC_LD ER AAC LD (Low Delay)

DT_AACOBJTYPE_ER_CELP ER CELP.

DT_AACOBJTYPE_ER_HVXC ER HVXC.

DT_AACOBJTYPE_ER_HILN ER HILN (Hermonic and Individual Lines plus Noise)

DT_AACOBJTYPE_ER_PARAMETRIC ER Parametric.

DT_AACOBJTYPE_SSC SSC (SinuSoidal Coding)

DT_AACOBJTYPE_PS Parametric Stereo.

DT_AACOBJTYPE_MPEG_SURROUND MPEG Surround.

DT_AACOBJTYPE_ESCAPE Escape value.

DT_AACOBJTYPE_LAYER1 MPEG-1/2 Layer-1.

DT_AACOBJTYPE_LAYER2 MPEG-1/2 Layer-2.

DT_AACOBJTYPE_LAYER3 MPEG-1/2 Layer-3.

DT_AACOBJTYPE_DST Direct Stream Transfer.

DT_AACOBJTYPE_ALS Audio Lossless Coding.

DT_AACOBJTYPE_SLS Scalable Lossless Coding.

DT_AACOBJTYPE_SLS_NON_CORE SLS non-core.

DT_AACOBJTYPE_ER_AAC_ELD ER AAC ELD (Enhanced Low Delay)

DT_AACOBJTYPE_SMR_SIMPLE Symbolic Music Representation Simple.

DT_AACOBJTYPE_SMR_MAIN SMR Main.

8.1.2.2 enum DtapiTs::DtAacProfile

AAC profile that is used for the audio stream.

Enumerator:

DT_AACPROFILE_UNKNOWN Invalid AAC profile.

DT_AACPROFILE_LOW_COMPLEXITY Low-Complexity profile (AAC-LC / LC-AAC)

DT_AACPROFILE_MAIN Main profile (AAC Main)

DT_AACPROFILE_SCALABLE_SAMPLING_RATE Scalable Sampling Rate profile (AAC-SSR)

8.1.2.3 enum DtapiTs::DTAPITS_RESULT

List of return values from various DTAPI-TS functions.

Enumerator:

DTAPITS_OK No errors occured.

DTAPITS_E_INVALID_BUF NULL pointer instead of valid table/section.

DTAPITS_E_TABLEID_MISMATCH Mismatch between the table id as found in the table data and the classes used for decoding. This error occurs when you try to decode a PMT with the DtTableNit class.

DTAPITS_E_MISSING_DATA The section seems to miss some data, not all fields required by the spec are available.

DTAPITS_E_INVALID_DESC_LEN One of the descriptor lengths was invalid.

DTAPITS_E_CRC_MISMATCH Invalid checksum.

DTAPITS_E_INVALID_TAG The tag in the descriptor does not match.

DTAPITS_E_INVALID_PDS An unsupported private data descriptor value was found in the given descriptor.

DTAPITS_E_INVALID_DESC Miscellaneous error while parsing descriptor.

DTAPITS_E_DESC_NOT_FOUND No descriptor was found in the given list that matches the tag/extended tag/pds value required to parse this descriptor.

DTAPITS_E_DESC_TOO_SHORT Not enough bytes for the descriptor.

DTAPITS_E_INVALID_FIELD One of the fields in a table has an invalid value.

DTAPITS_E_NOT_SUPPORTED An invalid argument was passed to the function.

DTAPITS_E_PARSE_ERROR Operation is not supported.

8.1.2.4 enum DtapiTs::DtAtscCcType

Atsc closed caption type.

Enumerator:

DT_CCTYPE_EIA608 EIA-608 closed captions.

DT_CCTYPE_EIA708 EIA-708 closed captions.

8.1.2.5 enum DtapiTs::DtAudioMode

Audio modes of the audio in elementary streams.

This information is extracted from few known stream types.

Enumerator:

DT_AUDIOMODE_UNKNOWN Unknown amount of channels.

DT_AUDIOMODE_STEREO 2 channels: stereo

DT_AUDIOMODE_JOINT_STEREO 2 channels: joint stereo

DT_AUDIOMODE_DUAL 2 channels: independently coded

DT_AUDIOMODE_MONO Mono channel.

DT_AUDIOMODE_AC3_CH1CH2 AC-3: 2 independent channels.

DT_AUDIOMODE_AC3_CENTER AC-3: center channel.

DT_AUDIOMODE_AC3_LR AC-3: left+right channels.

DT_AUDIOMODE_AC3_LCR AC-3: left+right+center channels.

DT_AUDIOMODE_AC3_LRS AC-3: left+right with surround.

DT_AUDIOMODE_AC3_LCRS AC-3: left+right+center with surround.

DT_AUDIOMODE_AC3_LR_SL_SR AC-3: left+right with surround left+right.

DT_AUDIOMODE_AC3_LCR_SL_SR AC-3: left+right+center with surround left+right.

DT_AUDIOMODE_AAC_CF AC-3: 1 channel: front-center.

DT_AUDIOMODE_AAC_LF_RF AC-3: 2 channels: front-left and front-right.

DT_AUDIOMODE_AAC_CF_LF_RF AC-3: 3 channels: front-center, front-left, front-right.

DT_AUDIOMODE_AAC_CF_LF_RF_RS AC-3: 4 channels: front-center/left/right, back-center.

DT_AUDIOMODE_AAC_CF_RF_LF_LR_RR AC-3: 5 chann: front-center/left/right, back-left/right.

DT_AUDIOMODE_AAC_CF_RF_LF_LR_RR_FLF AC-3: 6 channels.

DT_AUDIOMODE_AAC_CR_RF_LF_ROF_LOF_LR_RR_FLF AC-3: 8 channels.

8.1.2.6 enum DtapiTs::DtDeliverySystem

Types of delivey system descriptor as can be found in the DVB NIT table.

Enumerator:

DT_DELIVERYSYSTEM_CABLE Cable delivery system descriptor.
DT_DELIVERYSYSTEM_SATELLITE Satellite delivery system descriptor.
DT_DELIVERYSYSTEM_TERRESTRIAL Terrestrial delivery system descriptor.
DT_DELIVERYSYSTEM_T2 T2 delivery system descriptor.
DT_DELIVERYSYSTEM_SH SH delivery system descriptor.
DT_DELIVERYSYSTEM_INVALID No delivery system descriptor has been found.

8.1.2.7 enum DtapiTs::DtDvbT2MisoMode

SIDO/MISO mode for DVB-T2.s.

Enumerator:

DT_T2MISO_SISO SISO.
DT_T2MISO_MISO MISO.
DT_T2MISO_UNK Unknown mode.

8.1.2.8 enum DtapiTs::DtFecOuter

Outer forwared error correction as specified in the cable delivery descriptor.

Enumerator:

DT_FECOUTER_NONE No outer fec.
DT_FECOUTER_RS_204_188 Reed-solomon(188, 204) coding.
DT_FECOUTER_UNK An unknown error correction scheme has been applied.

8.1.2.9 enum DtapiTs::DtGuardInterval

Guard interval for DVB-SH.

Enumerator:

DT_SH_GUARDINTERVAL_1_32 Guard interval 1/32.
DT_SH_GUARDINTERVAL_1_16 Guard interval 1/16.
DT_SH_GUARDINTERVAL_1_8 Guard interval 1/8.
DT_SH_GUARDINTERVAL_1_4 Guard interval 1/4.

8.1.2.10 enum DtapiTs::DtMpaLayer

MPEG audio layer.

Enumerator:

DT_MPA_LAYER_1 MPEG-1/2 Layer-1.
DT_MPA_LAYER_2 MPEG-1/2 Layer-2.
DT_MPA_LAYER_3 MPEG-1/2 Layer-3.

8.1.2.11 enum DtapiTs::DtMpaVersion

MPEG Audio version.

Enumerator:

DT_MPA_VERSION_1 MPEG Audio version 1.
DT_MPA_VERSION_2 MPEG Audio version 2.
DT_MPA_VERSION_2_5 MPEG Audio version 2.5.

8.1.2.12 enum DtapiTs::DtPolarization

Polarization of the transmitted signal.

Enumerator:

DT_POLAR_LIN_HOR Linear - horizontal.
DT_POLAR_LIN_VERT Linear - vertical.
DT_POLAR_CIRC_LEFT Circular - left.
DT_POLAR_CIRC_RIGHT Circular - right.

8.1.2.13 enum DtapiTs::DtRollOff

Roll-off factor for DVB-SH.

Enumerator:

DT_SH_ROLLOFF_35 Roll-off factor 35%.
DT_SH_ROLLOFF_25 Roll-off factor 25%.
DT_SH_ROLLOFF_15 Roll-off factor 15%.
DT_SH_ROLLOFF_UNK Roll-off factor unknown.

8.1.2.14 enum DtapiTs::DtScrambling

How the last transport packet on a given pid was scrambled.

Enumerator:

DT_SCRAMBLING_NONE The packet was not scrambled.
DT_SCRAMBLING_RESERVED The packet was scrambled, but we do not know how.
DT_SCRAMBLING_EVEN The packet was scrambled with an even key.
DT_SCRAMBLING_ODD The packet was scrambled with an odd key.

8.1.2.15 enum DtapiTs::DtServiceType

Very course type of service, useful for quickly identifying the content type.

Enumerator:

DT_SERVICETYPE_UNKNOWN The service does not contain video nor audio.
DT_SERVICETYPE_TELEVISION The service contains at least one video stream.
DT_SERVICETYPE_RADIO The service contains no video but does contain at least one audio stream.

8.1.2.16 enum DtapiTs::DtShBandwidth

Bandwidth for DVB-SH.

Enumerator:

DT_SH_BANDWITH_8MHZ Bandwidth is 8Mhz.
DT_SH_BANDWITH_7MHZ Bandwidth is 7Mhz.
DT_SH_BANDWITH_6MHZ Bandwidth is 6Mhz.
DT_SH_BANDWITH_5MHZ Bandwidth is 5Mhz.
DT_SH_BANDWITH_1_7MHZ Bandwidth is 1.7Mhz.
DT_SH_BANDWITH_UNK Bandwidth is unknown.

8.1.2.17 enum DtapiTs::DtShCodeRate

Code rate for DVB-SH.

Enumerator:

DT_SH_CODERATE_1_5 1/5 standard
DT_SH_CODERATE_2_9 2/9 standard
DT_SH_CODERATE_1_4 1/4 standard
DT_SH_CODERATE_2_7 2/7 standard
DT_SH_CODERATE_1_3 1/3 standard
DT_SH_CODERATE_1_3_COMPL 1/3 complementary
DT_SH_CODERATE_2_5 2/5 standard
DT_SH_CODERATE_2_5_COMPL 2/5 complementary
DT_SH_CODERATE_1_2 1/2 standard
DT_SH_CODERATE_1_2_COMPL 1/2 complementary
DT_SH_CODERATE_2_3 2/3 standard
DT_SH_CODERATE_2_3_COMPL 2/3 complementary
DT_SH_CODERATE_RESERVED Unknown code rate.

8.1.2.18 enum DtapiTs::DtShModMode

Modulation modi supported by DVB-SH.

Enumerator:

DT_SH_MOD_QPSK WPSK modulation.
DT_SH_MOD_8PSK 8PSK modulation.
DT_SH_MOD_16APSK 16APSK modulation.
DT_SH_MOD_UNK Unknown modulation mode.

8.1.2.19 enum DtapiTs::DtShModType

Modulation types supported by DVB-SH.

Enumerator:

DT_SH_MOD_TDM TDM modulation.
DT_SH_MOD_OFDM OFDM modulation.

8.1.2.20 enum DtapiTs::DtStandardMode

Used to set the mode in which the DtapiTs library operates.

This is important because it determines how the various TableIds are interpreted and which tables are actually parsed.

Enumerator:

DT_STANDARDMODE_UNK No tables are parsed.

DT_STANDARDMODE_ATSC ATSC PSIP information is parsed.

DT_STANDARDMODE_DVB DVB-SI information is parsed.

DT_STANDARDMODE_DVB_RCS DVB-SI information is parsed.

8.1.2.21 enum DtapiTs::DtStreamType

Type of content in a given pid.

This is a property of both a service component and directly implied from that the pid that contains the data of that service component. The elements with values in the range 0..0xFF (inclusive) are defined by MPEG and are directly copied from the PMT. Values in that range that are not defined here are valid and may occur. Values outside that range are only assigned to the m_StreamType property of DtPidInfo. Those are used to determine how to decode the PES data inside that pid.

Enumerator:

DT_STREAMTYPE_INVALID Value has not been determined yet.

DT_STREAMTYPE_UNKNOWN Contents of the stream are unknown.

DT_STREAMTYPE_MPEG1_VIDEO MPEG-1 video.

DT_STREAMTYPE_MPEG2_VIDEO MPEG-2 video.

DT_STREAMTYPE_MPEG1_AUDIO MPEG-1 audio.

DT_STREAMTYPE_MPEG2_AUDIO MPEG-2 audio.

DT_STREAMTYPE_PRIV_SECTIONS Private sections.

DT_STREAMTYPE_PRIV_DATA Private data.

DT_STREAMTYPE_MHEG MHEG: interactive TV.

DT_STREAMTYPE_DSMCC Digital storage media command&control.

DT_STREAMTYPE_H_222_1 ITU-T Satellite audio-visual stream.

DT_STREAMTYPE_13818_6_TA MPEG-2 Video Clip A stream.

DT_STREAMTYPE_13818_6_TB MPEG-2 Video Clip B stream.

DT_STREAMTYPE_13818_6_TC MPEG-2 Video Clip C stream.

DT_STREAMTYPE_13818_6_TD MPEG-2 Video Clip D stream.

DT_STREAMTYPE_AUX MPEG-2 Auxiliary stream.

DT_STREAMTYPE_AAC AAC audio.

DT_STREAMTYPE_MPEG4_VIDEO MPEG-4 video.

DT_STREAMTYPE_HEAAC HE-AAC audio.

DT_STREAMTYPE_MPEG4_PES SL-packetized or FlexMux stream carried in PES packets.

DT_STREAMTYPE_MPEG4_SECT SL-packetized or FlexMux stream carried in sections.

DT_STREAMTYPE_13818_6_SDP Synchronized download protocol.

DT_STREAMTYPE_METADATA_PES Metadata carried in PES packets.

DT_STREAMTYPE_METADATA_SECT Metadata carried in metadata_sections.

DT_STREAMTYPE_METADATA_DC Metadata carried in Data carousel.

DT_STREAMTYPE_METADATA_OC Metadata carried in object carousel.

DT_STREAMTYPE_METADATA_SDP Metadata carried in synchronized data protocol.

DT_STREAMTYPE_IPMP_STREAM_MP2 MPEG-2 IPMP stream.

DT_STREAMTYPE_AVC_VIDEO AVC/H.264 video.

DT_STREAMTYPE_J2K_VIDEO J2k / JPEG 2000 video.

DT_STREAMTYPE_HEVC_VIDEO NOT THE OFFICIAL STREAM-TYPE (YET)

DT_STREAMTYPE_JPEGXS_VIDEO JPEGXS video.

DT_STREAMTYPE_IPMP_STREAM IPMP stream.

DT_STREAMTYPE_ATSC_DIGICYPH2 DigiCipher II/H.262 Video.

DT_STREAMTYPE_ATSC_AC3 AC-3 Audio.

DT_STREAMTYPE_SCTE_SUBTITLE SCTE Standard Subtitle.

DT_STREAMTYPE_SCTE_ISOCHR_DATA SCTE Isochronous Data.

DT_STREAMTYPE_ATSC_EAC3 E-AC-3 Audio.

DT_STREAMTYPE_WM9_AUDIO Windows Media player 9 Audio.

DT_STREAMTYPE_VC1_VIDEO VC1 Video.

DT_STREAMTYPE_DVB_AC3 AC-3 Audio.

DT_STREAMTYPE_DVB_EAC3 E-AC-3 Audio.

DT_STREAMTYPE_DVB_TELETEXT DVB Teletext.

DT_STREAMTYPE_DVB_MPE DVB MPE stream.

DT_STREAMTYPE_DVB_DATA_CAROUSEL Data carousel.

DT_STREAMTYPE_DVB_INT DVB INT.

DT_STREAMTYPE_SMPTE_AES3 SMPTE 302M AES3.

DT_STREAMTYPE_DVB_AC4 AC-4 audio.

8.1.2.22 enum DtapiTs::DtTableType

List of table types.

Used as bitmask in DtPidInfo::m_TableTypeMask

Enumerator:

DT_TABLETYPE_UNKNOWN Unkown table type. Note that while in DVB mode ATSC tables won't be recognized and the other way around.

DT_TABLETYPE_PAT Program Association Table.

DT_TABLETYPE_CAT Conditional Access Table.

DT_TABLETYPE_PMT Program Map Table.

DT_TABLETYPE_TSDT Transport Stream Description Table.

DT_TABLETYPE_EMM Entitlement Management Message.

DT_TABLETYPE_ECM Entitlement Control Message.

DT_TABLETYPE_DVB_NITACT DVB Network Information Table for current stream.

DT_TABLETYPE_DVB_NITOTH DVB Network Information Table for other stream.

DT_TABLETYPE_DVB_SDTACT DVB Service Description Table for current stream.

DT_TABLETYPE_DVB_SDTOTH DVB Service Description Table for other stream.

DT_TABLETYPE_DVB_BAT DVB Bouquet Association Table.

DT_TABLETYPE_DVB_EITACT DVB Event Information Table for current stream about present/following events.

DT_TABLETYPE_DVB_EITACTS DVB Event Information Table for current stream about present/following events.

DT_TABLETYPE_DVB_EITOTH DVB Event Information Table for other stream about scheduled events.

DT_TABLETYPE_DVB_EITOTHS DVB Event Information Table for other stream about scheduled events.

DT_TABLETYPE_DVB_TDT DVB Time Date Table.

DT_TABLETYPE_DVB_RST DVB Running Stations Table.

DT_TABLETYPE_DVB_ST DVB Stuffing Section.

DT_TABLETYPE_DVB_TOT DVB Time Offset Table.

DT_TABLETYPE_DVB_DIT DVB Discontinuity Information Table.

DT_TABLETYPE_DVB_SIT DVB Selection Information Table.

DT_TABLETYPE_DVB_RNT DVB Resolution Notification Table.

DT_TABLETYPE_DVB_INT DVB IP/MAC Notification Table.

DT_TABLETYPE_DVB_RCS_RMT DVB-RCS RCS Map Table.

DT_TABLETYPE_DVB_RCS_SCT DVB-RCS Superframe Composition Table.

DT_TABLETYPE_DVB_RCS_FCT DVB-RCS Frame Composition Table.

DT_TABLETYPE_DVB_RCS_TCT DVB-RCS Timeslot Composition Table.

DT_TABLETYPE_DVB_RCS_SPT DVB-RCS Satellite Position Table.

DT_TABLETYPE_DVB_RCS_CMT DVB-RCS Correction Message Table.

DT_TABLETYPE_DVB_RCS_TBTP DVB-RCS Terminal Burst Time Plan.

DT_TABLETYPE_DVB_RCS_PCRPP DVB-RCS PCR Packet Payload.

DT_TABLETYPE_DVB_RCS_TMST DVB-RCS Transmission Mode Support Table.

DT_TABLETYPE_DVB_RCS_TIM DVB-RCS Terminal Information Plan.

DT_TABLETYPE_DVB_RCS_LL_FEC_PDT DVB-RCS LL_FEC Parity Data Table.

DT_TABLETYPE_ATSC_MGT ATSC Master Guide Table.

DT_TABLETYPE_ATSC_TVCT ATSC Terrestrial Virtual Channel Table.

DT_TABLETYPE_ATSC_CVCT ATSC Cable Virtual Channel Table.

DT_TABLETYPE_ATSC_RRT ATSC Rating Region Table.

DT_TABLETYPE_ATSC_EIT ATSC Event Information Table.

DT_TABLETYPE_ATSC_ETT ATSC Extended Text Table.

DT_TABLETYPE_ATSC_STT ATSC System Time Table.

DT_TABLETYPE_ATSC_DCCT ATSC Directed Channel Change Table.

DT_TABLETYPE_ATSC_DCCSCT ATSC DCC Selection Code Table.

8.1.2.23 enum DtapiTs::DtTr101290Bitmask

Bitmask for the TR 101 290 errors.

Used to set callback functions for multiple errors at the same time.

See Also

enum DtTr101290Indicator

Enumerator:

DT_ERR_B_P1 Bitmask that includes all priority 1 errors.

DT_ERR_B_P2 Bitmask that includes all priority 2 errors.

DT_ERR_B_P3 Bitmask that includes all priority 3 errors.

DT_ERR_B_ALL Bitmask that includes all errors.

8.1.2.24 enum DtapiTs::DtTr101290Indicator

Unique identifiers for all TR 101 290 error conditions.

Enumerator:

DT_ERR_P1_TS_SYNC_LOSS Not synced to any transport stream. This indicator is not reset after a timeout but instead directly reset when the DTAPI-TS found sync again.

DT_ERR_P1_SYNC_BYTE First byte (sync byte) of a packet is not 0x47.

DT_ERR_P1_PAT_2 Can indicate three different errors:

1. Sections with table_id 0x00 do not occur at least every 0.5s on PID 0x0000.
2. Section with table_id other than 0x00 found on PID 0x0000.
3. Scrambling_control_field is not 00 for PID 0x0000.

DT_ERR_P1_CONTINUITY_COUNTER Lost packet, incorrect packet order or a packet occurs more than twice.

DT_ERR_P1_PMT_2 Can indicate two different errors:

1. Sections with table_id 0x02 do not occur at least every 0.5s on a PID referenced as PMT.
2. Scrambling_control_field is not 00 for all packets in a PID referenced as PMT.

Note

DT_ERR_P1_PID Not yet implemented

DT_ERR_P2_TRANSPORT Transport_error_indicator is set to 1.

DT_ERR_P2_CRC One of the following tables had an incorrect CRC:

- PAT
- CAT
- PMT
- NIT
- SDT
- BAT
- EIT
- TOT

DT_ERR_P2_PCR_REPETITION The time interval between two consecutive PCR values is more than 100ms. DtTr101290Error::m_Time will contain the actual time in ms.

DT_ERR_P2_PCR_DISC_IND The difference between two consecutive PCR values is more than 100ms without the discontinuity_indicator being set. DtTr101290Error::m_Time will contain the actual time in ms.

DT_ERR_P2_PCR_ACCURACY The PCR accuracy of the PCR values on at least one PID are not precise enough.

DT_ERR_P2_PTS PTS repetition interval is too large for a specified PID.

DT_ERR_P2_CAT Can indicate two different errors:

1. Scrambled packets found in stream but no CAT present.
2. Section with table_id other than 0x01 found on PID 0x0001.

DT_ERR_P3_NIT_ACTUAL Section with table_id 0x40 do not occur at least every 10s on PID 0x10 or occur more then 25ms.

DT_ERR_P3_NIT_OTHER A Section with table_id 0x41 do not occur at least every 10s on PID 0x0010.

DT_ERR_P3_SI_REPETITION A section repetition interval is too large or too small.

Note

DT_ERR_P3_BUFFER Not yet implemented

DT_ERR_P3_UNREFERENCED_PID The PID is not referred to by a PMT or CAT within 0.5s.

DT_ERR_P3_SDT_ACTUAL Sections with table_id 0x42 not present on PID 0x0011 for more than 2s. Sections with table_ids other than 0x42, 0x46, 0x4A or 0x72 found on PID 0x0011. Any two sections with table_id 0x42 occur on PID 0x0011 within 25ms

DT_ERR_P3_SDT_OTHER Interval between sections with the same section_number and table_id 0x46 on PID 0x0011 longer than 10s.

DT_ERR_P3_EIT_ACTUAL Section '0' with table_id 0x4E not present on PID 0x0012 for more than 2s Section '1' with table_id 0x4E not present on PID 0x0012 for more than 2s Sections with table_ids other than in the range 0x4E-0x6F or 0x72 found on PID 0x0012. Any two sections with table_id 0x4E occur on PID 0x0012 within 25ms

DT_ERR_P3_EIT_OTHER Interval between sections '0' with table_id 0x4F on PID 0x0012 longer than 10s Interval between sections '1' with table_id 0x4F on PID 0x0012 longer than 10s.

Note

DT_ERR_P3_EIT_PF Not yet implemented

DT_ERR_P3_RST Sections with table_id other than 0x71 or 0x72 found on PID 0x0013. Any two sections with table_id 0x71 occur on PID 0x0013 within 25ms

DT_ERR_P3_TDT Sections with table_id 0x70 not present on PID 0x0014 for more than 30s Sections with table_id other than 0x70, 0x72 or 0x73 found on PID 0x0014. Any two sections with table_id 0x70 occur on PID 0x0014 within 25ms

Note

DT_ERR_P3_EMPTY_BUFFER Not yet implemented

Note

DT_ERR_P3_DATA_DELAY Not yet implemented

8.1.2.25 enum DtapiTs::DtTransmissionMode

Transmission mode for DVB-SH.

Enumerator:

DT_SH_TRANSMODE_1K 1k

DT_SH_TRANSMODE_2K 2k

DT_SH_TRANSMODE_4K 4k

DT_SH_TRANSMODE_8K 8k

8.1.2.26 enum DtapiTs::DtVideoChromaFormat

Video chroma format.

Enumerator:

DT_CHROMAFORMAT_INVALID Invalid chroma format.

DT_CHROMAFORMAT_420 4:2:0

DT_CHROMAFORMAT_422 4:2:2

DT_CHROMAFORMAT_424 4:2:4

DT_CHROMAFORMAT_444 4:4:4

DT_CHROMAFORMAT_MONO Mono.

8.1.2.27 enum DtapiTs::DtWeFlag

Flag indicating whether the satellite position is in the western or eastern part of the orbit.

Enumerator:

DT_WEFLAG_EAST Satellite position is in the eastern part.

DT_WEFLAG_WEST Satellite position is in the western part.

Chapter 9

Class Documentation

9.1 DtapiTs::DtPes::DataBuffer Class Reference

Public Member Functions

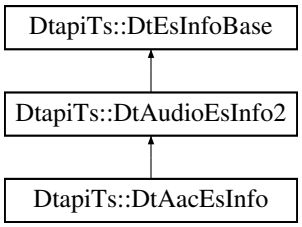
- **DataBuffer** (uint8_t *pData=NULL, int Size=0)
- **DataBuffer** (const DataBuffer &Oth)
- uint8_t * **Data** () const
- bool **IsOwnedByUs** () const
- DataBuffer & **operator=** (const DataBuffer &Oth)
- uint8_t & **operator[]** (size_t n) const
- int **Resize** (int Size)
- int **Size** () const

Protected Attributes

- uint8_t * **m_pData**
- int **m_Size**

9.2 DtapiTs::DtAacEsInfo Class Reference

Inheritance diagram for DtapiTs::DtAacEsInfo:



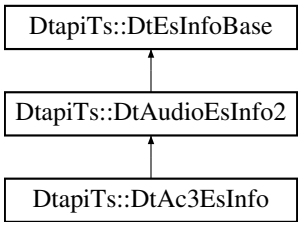
Public Member Functions

- virtual void **Clear** ()

Additional Inherited Members

9.3 DtapiTs::DtAc3EsInfo Class Reference

Inheritance diagram for DtapiTs::DtAc3EsInfo:



Public Member Functions

- virtual void **Clear** ()

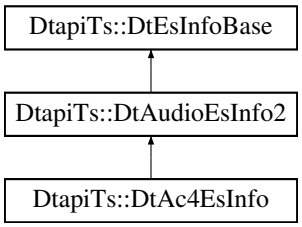
Public Attributes

- InfoField< int > **m_DialogNorm**

Additional Inherited Members

9.4 DtapiTs::DtAc4EsInfo Class Reference

Inheritance diagram for DtapiTs::DtAc4EsInfo:



Public Member Functions

- virtual void **Clear** ()

Additional Inherited Members

9.5 DtapiTs::DtAudioEsInfo Class Reference

Information about an audio elementary stream extracted from PES packets.

`#include <DTAPITS.h>`

Public Member Functions

- DtAudioEsInfo ()

  Create new DtAudioEsInfo object.

Public Attributes

- DtAacObjType m_AacExtObjType

  AAC extension object type.
- DtAacObjType m_AacObjectType

  AAC object type (used with HE-AAC, LATM/LOAS)
- DtAacProfile m_AacProfile

  AAC profile.
- int m_Ac3DialogNorm

  Dialog normalization (in dB below digital 100%)
- int m_Bitrate

  Audio bit-rate (bps)
- DtAudioMode m_ChMode

  Channel mode (Stereo, Mono, etc)
- bool m_Copyrighted

  Audio is copyrighted (=true) or not (=false)
- unsigned int m_Mask

  Indicates which fields are valid.
- DtMpaLayer m_MpaLayer

  MPEG layer (I, II, III)
- DtMpaVersion m_MpaVersion

  MPEG version (MPEG1, MPEG2, MPEG2.5)
- bool m_Original

  This is original source (=true) or a copy(=false)
- int m_Samplerate

  Audio sample rate (Hz)

Static Public Attributes

- static const unsigned int AAC_OBJECT_TYPE_FIELD = 0x00020000

  Bit set in m_Mask when m_AacObjectType and m_AacExtObjType are valid.
- static const unsigned int AAC_PROFILE_FIELD = 0x00010000

  Bit set in m_Mask when m_AacProfile is valid.
- static const unsigned int AC3_DIAL_NORM_FIELD = 0x01000000

  Bit set in m_Mask when m_Ac3DialogNorm is valid.
- static const unsigned int BITRATE_FIELD = 0x00000001

  Bit set in m_Mask when m_Bitrate is valid.
- static const unsigned int **c_AacObjectTypeField** = AAC_OBJECT_TYPE_FIELD
- static const unsigned int **c_AacProfileField** = AAC_PROFILE_FIELD
- static const unsigned int **c_Ac3DialNormField** = AC3_DIAL_NORM_FIELD
- static const unsigned int **c_BitrateField** = BITRATE_FIELD
- static const unsigned int **c_ChModeField** = CH_MODE_FIELD
- static const unsigned int **c_CopyrightField** = COPYRIGHT_FIELD
- static const unsigned int **c_MpaLayerField** = MPA_LAYER_FIELD
- static const unsigned int **c_MpaVersionField** = MPA_VERSION_FIELD
- static const unsigned int **c_OriginalField** = ORIGINAL_FIELD

- static const unsigned int **c_SamplerateField** = SAMPLERATE_FIELD
- static const unsigned int CH_MODE_FIELD = 0x00000004

  Bit set in m_Mask when m_ChMode is valid.
- static const unsigned int COPYRIGHT_FIELD = 0x00000008

  Bit set in m_Mask when m_Copyrighted is valid.
- static const unsigned int MPA_LAYER_FIELD = 0x00000200

  Bit set in m_Mask when m_MpaLayer is valid.
- static const unsigned int MPA_VERSION_FIELD = 0x00000100

  Bit set in m_Mask when m_MpaVersion is valid.
- static const unsigned int ORIGINAL_FIELD = 0x00000010

  Bit set in m_Mask when m_Original is valid.
- static const unsigned int SAMPLERATE_FIELD = 0x00000002

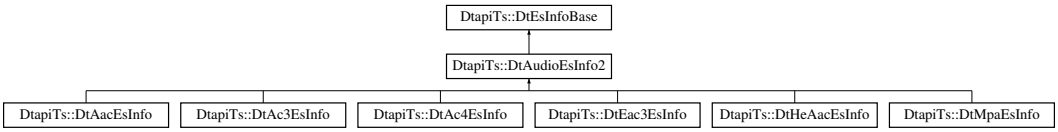
  Bit set in m_Mask when m_Samplerate is valid.

9.5.1 Detailed Description

Information about an audio elementary stream extracted from PES packets.

9.6 DtapiTs::DtAudioEsInfo2 Class Reference

Inheritance diagram for DtapiTs::DtAudioEsInfo2:



Public Types

- enum **AudioType** {
  AUDIO_TYPE_UNDEF,
  AUDIO_TYPE_AAC,
  AUDIO_TYPE_AC3,
  AUDIO_TYPE_AC4,
  AUDIO_TYPE_MPA,
  AUDIO_TYPE_EAC3,
  AUDIO_TYPE_HEAAC }

Public Member Functions

- **DtAudioEsInfo2** (AudioType Type)
- virtual void **Clear** ()
- AudioType **Type** () const

Public Attributes

- InfoField< DtAudioMode > m_AudioMode

  Audio mode (Stereo, Mono, etc)
- InfoField< int > m_BitRate

Audio bit-rate (bps)

- InfoField< bool > m_Copyrighted

  Audio is copyrighted (=true) or not (=false)

- InfoField< bool > m_Original

  This is original source (=true) or a copy(=false)

- InfoField< int > m_SampleRate

  Audio sample rate (Hz)

9.7 DtapiTs::DtBitrate Class Reference

Some statistics about the bitrate of a PID, service or transport stream.

`#include <DTAPITS.h>`

Public Member Functions

- DtBitrate ()

  Construct a new DtBitrate object.

Public Attributes

- int m_Avg

  Average bitrate.

- int m_Max

  Maximum bitrate over the measured period.

- int m_Min

  Minimum bitrate over the measured period.

9.7.1 Detailed Description

Some statistics about the bitrate of a PID, service or transport stream.

9.8 DtapiTs::DtBitrateSettings Class Reference

Settings for the bitrate measurement sliding window.

`#include <DTAPITS.h>`

Public Member Functions

- DtBitrateSettings (DtTimeDiff TimeSliceTime, int NumTimeSlices, int NumAvgValues)

  Create a new DtBitrateSettings object.

Static Public Member Functions

- static const DtBitrateSettings & MGB1 ()

  MG Profile 1 as defined in TR 101 290.

- static const DtBitrateSettings & MGB2 ()

  MG Profile 2 as defined in TR 101 290.

- static const DtBitrateSettings & MGB3 ()

  MG Profile 3 as defined in TR 101 290.
- static const DtBitrateSettings & MGB4 ()

  MG Profile 4 as defined in TR 101 290.

Public Attributes

- int m_NumAvgValues

  Number of averages to keep.
- int m_NumTimeSlices

  Number of time slices.
- DtTimeDiff m_TimeSliceTime

  Duration of each time slice.

9.8.1 Detailed Description

Settings for the bitrate measurement sliding window.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 DtapiTs::DtBitrateSettings::DtBitrateSettings (DtTimeDiff *TimeSliceTime,* int *NumTimeSlices,* int *NumAvgValues*) [inline]

Create a new DtBitrateSettings object.

Parameters

TimeSliceTime,:	The duration of each time slice.
NumTime-Slices,:	The number of timeslices in the window.
NumAvgValues,:	Each time a new slices is added (and the oldest one removed) from the window, the average value is computed and stored. This parameter indicates the number of averages to store. It can be used to have a longer running average besides the normal average.

9.8.3 Member Data Documentation

9.8.3.1 int DtapiTs::DtBitrateSettings::m_NumAvgValues

Number of averages to keep.

If duration=0.1s and there ae 10 slices, each average will be over 1s. If NumAvgValues is 50, you can see a running average over the last 5s.

9.9 DtapiTs::DtCallback1< TArg1 > Class Template Reference

Helper class to store a function pointer to a function with one argument and a pointer to the associated data.

`#include <DTAPITS.h>`

Public Types

- typedef void(**DtCallbackFunc**)(void *pOpaque, TArg1)

Public Member Functions

- **DtCallback1** (DtCallbackFunc *Func, void *pOpaque=NULL)
- void **operator()** (TArg1 Arg1)

Static Public Member Functions

- template<class Tcls , void(Tcls::*)(TArg1) Tfunc>
  static void CallbackWrapper (void *pOpaque, TArg1 Arg1)

  Static helper function that calls a class member function.

9.9.1 Detailed Description

template<typename TArg1>class DtapiTs::DtCallback1< TArg1 >

Helper class to store a function pointer to a function with one argument and a pointer to the associated data.

9.10 DtapiTs::DtCallback2< TArg1, TArg2 > Class Template Reference

Helper class to store a function pointer to a function with two arguments and a pointer to the associated data.

`#include <DTAPITS.h>`

Public Types

- typedef void(**DtCallbackFunc**)(void *pOpaque, TArg1, TArg2)

Public Member Functions

- **DtCallback2** (DtCallbackFunc *Func, void *pOpaque=NULL)
- void **operator()** (TArg1 Arg1, TArg2 Arg2)

Static Public Member Functions

- template<class Tcls , void(Tcls::*)(TArg1, TArg2) Tfunc>
  static void CallbackWrapper (void *pOpaque, TArg1 Arg1, TArg2 Arg2)

  Static helper function that calls a class member function.

9.10.1 Detailed Description

template<typename TArg1, typename TArg2>class DtapiTs::DtCallback2< TArg1, TArg2 >

Helper class to store a function pointer to a function with two arguments and a pointer to the associated data.

9.11 DtapiTs::DtCallback3< TArg1, TArg2, TArg3 > Class Template Reference

Helper class to store a function pointer to a function with 3 arguments and a pointer to the associated data.

`#include <DTAPITS.h>`

Public Types

- typedef void(**DtCallbackFunc**)(void *pOpaque, TArg1, TArg2, TArg3)

Public Member Functions

- **DtCallback3** (DtCallbackFunc *Func, void *pOpaque=NULL)
- void **operator()** (TArg1 Arg1, TArg2 Arg2, TArg3 Arg3)

Static Public Member Functions

- template<class Tcls , void(Tcls::*)(TArg1, TArg2, TArg3) Tfunc>
  static void CallbackWrapper (void *pOpaque, TArg1 Arg1, TArg2 Arg2, TArg3 Arg3)

  Static helper function that calls a class member function.

9.11.1 Detailed Description

template<typename TArg1, typename TArg2, typename TArg3>class DtapiTs::DtCallback3< TArg1, TArg2, TArg3 >

Helper class to store a function pointer to a function with 3 arguments and a pointer to the associated data.

9.12 DtapiTs::DtCaSystem Class Reference

Descripion of a conditional access system.

`#include <DTAPITS.h>`

Public Member Functions

- DtCaSystem (int CaSystemId, int Pid)

  Create a new DtCaSystem object.

Public Attributes

- int m_CaSystemId

  CA_System_Id as defined by DVB.
- int m_Pid

  PID that contains the EMM or ECM stream.

9.12.1 Detailed Description

Descripion of a conditional access system.

This can either describe a EMM or a ECM stream depending on the context where this class is used.

9.12.2 Member Data Documentation

9.12.2.1 int DtapiTs::DtCaSystem::m_CaSystemId

CA_System_Id as defined by DVB.

See Also

DtCaSystemId2String

9.13 DtapiTs::DtDescDvbAc3 Class Reference

Parsed information from the DVB AC-3 descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_Asvc

  Used to indicate the main services that this service can be associated with.
- int m_Bsid

  The AC-3 coding version.
- int m_ComponentType

  The type of audio carried in the AC-3 ES, this field should have the same value as the component-tag field in a component descriptor.
- std::vector< uint8_t > m_InfoByte

  Additional info bytes reserved for future use.
- int m_MainId

  Identifier for the main audio service.

9.13.1 Detailed Description

Parsed information from the DVB AC-3 descriptor.

9.13.2 Member Data Documentation

9.13.2.1 int DtapiTs::DtDescDvbAc3::m_Asvc

Used to indicate the main services that this service can be associated with.

-1 if this field is not present.

9.13.2.2 int DtapiTs::DtDescDvbAc3::m_Bsid

The AC-3 coding version.

-1 if this field is not present.

9.13.2.3 int DtapiTs::DtDescDvbAc3::m_ComponentType

The type of audio carried in the AC-3 ES, this field should have the same value as the component-tag field in a component descriptor.

-1 if this field is not present.

9.13.2.4 int DtapiTs::DtDescDvbAc3::m_MainId

Identifier for the main audio service.

-1 if this field is not present.

9.14 DtapiTs::DtDescDvbCDelivery Class Reference

Parsed information from the DVB cable delivery system descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_FecInner

  The inner FEC scheme used.
- int m_FecOuter

  The outer FEC scheme used.
- __int64 m_Frequency

  Frequency of the transmitted signal in Hz.
- int m_Modulation

  Modulation scheme.
- int m_SymbolRate

  Symbol rate in symbols/s.

9.14.1 Detailed Description

Parsed information from the DVB cable delivery system descriptor.

9.15 DtapiTs::DtDescDvbComponent Class Reference

Parsed information from the DVB component descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_ComponentTag

  Identifier for the component this descriptor describes.
- int m_ComponentType

  The type of the video/audio component.
- std::wstring m_Description

  A textual description of the stream.
- std::string m_LangCode

  ISO 639 language code, language of the component.
- int m_StreamContent

  The type (video, audio, ...) of stream.

9.15.1 Detailed Description

Parsed information from the DVB component descriptor.

9.15.2 Member Data Documentation

9.15.2.1 int DtapiTs::DtDescDvbComponent::m_ComponentTag

Identifier for the component this descriptor describes.

9.16 DtapiTs::DtDescDvbDataBroadcast Class Reference

Parsed information from the DVB data broadcast descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_ComponentTag

  Identifier for the component this descriptor describes.
- int m_DataBroadcastId

  Identifier for the data broadcast specification that is used to broadcast the data in this network.
- std::wstring m_Description

Textual description of the component.

- std::string m_LangCode

  ISO 639 language code of m_Description.

- std::vector< uint8_t > m_SelectorBytes

  A sequence of selector bytes.

9.16.1 Detailed Description

Parsed information from the DVB data broadcast descriptor.

9.16.2 Member Data Documentation

9.16.2.1 int DtapiTs::DtDescDvbDataBroadcast::m_ComponentTag

Identifier for the component this descriptor describes.

Optional, may be set to 0.

9.16.2.2 int DtapiTs::DtDescDvbDataBroadcast::m_DataBroadcastId

Identifier for the data broadcast specification that is used to broadcast the data in this network.

Values are allocated in TS 101 162.

9.16.2.3 std::vector<uint8_t> DtapiTs::DtDescDvbDataBroadcast::m_SelectorBytes

A sequence of selector bytes.

The syntax and semantics of this field are defined by the broadcast specification identified by m_DataBroadcastId.

9.17 DtapiTs::DtDescDvbDataBroadcastId Class Reference

Parsed information from the DVB data broadcast id descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.

- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_DataBroadcastId

  Identifier for the data broadcast specification that is used to broadcast the data in this network.

- std::vector< uint8_t > m_SelectorBytes

  A sequence of selector bytes.

9.17.1 Detailed Description

Parsed information from the DVB data broadcast id descriptor.

9.17.2 Member Data Documentation

9.17.2.1 int DtapiTs::DtDescDvbDataBroadcastId::m_DataBroadcastId

Identifier for the data broadcast specification that is used to broadcast the data in this network.

Values are allocated in TS 101 162.

9.17.2.2 std::vector<uint8_t> DtapiTs::DtDescDvbDataBroadcastId::m_SelectorBytes

A sequence of selector bytes.

The syntax and semantics of this field are defined by the broadcast specification identified by m_DataBroadcastId.

9.18 DtapiTs::DtDescDvbLinkage Class Reference

Parsed information from the DVB linkage descriptor.

`#include <DTAPITS.h>`

Classes

- struct EventLinkage
- struct ExtendedEventLinkage
- struct MobileHandOverInfo

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- union {
  EventLinkage m_Event
  Event linkage information.
  MobileHandOverInfo m_MobileHandOver
  Mobile hand-over information.
  };
- std::vector< ExtendedEventLinkage > m_ExtendedEvents

  All extended event linkage information.
- int m_LinkageType

  The type of linkage to e.g. information.
- int m_OrigNetworkId

  Indicates the network that contains the information service.

- std::vector< uint8_t > m_PrivData

  Private data bytes.
- int m_ServiceId

  The service indentifier that is linked too.
- int m_TransportStreamId

  Indicates the transport stream that contains the information service.

9.18.1 Detailed Description

Parsed information from the DVB linkage descriptor.

9.18.2 Member Data Documentation

9.18.2.1 EventLinkage DtapiTs::DtDescDvbLinkage::m_Event

Event linkage information.

Only valid when m_LinkageType==0x0D

9.18.2.2 std::vector<ExtendedEventLinkage> DtapiTs::DtDescDvbLinkage::m_ExtendedEvents

All extended event linkage information.

Only valid when m_LinkageType==0x0E.

9.18.2.3 MobileHandOverInfo DtapiTs::DtDescDvbLinkage::m_MobileHandOver

Mobile hand-over information.

Only valid when m_LinkageType==0x08

9.18.2.4 int DtapiTs::DtDescDvbLinkage::m_OrigNetworkId

Indicates the network that contains the information service.

9.18.2.5 int DtapiTs::DtDescDvbLinkage::m_TransportStreamId

Indicates the transport stream that contains the information service.

9.19 DtapiTs::DtDescDvbLocalTimeOffset Class Reference

Parsed information from the DVB local time offset descriptor.

`#include <DTAPITS.h>`

Classes

- struct LocalTimeOffset

Public Member Functions

- void Clear ()
  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- std::vector< LocalTimeOffset > m_TimeOffsets
  All local time offsets.

9.19.1 Detailed Description

Parsed information from the DVB local time offset descriptor.

9.20 DtapiTs::DtDescDvbMultilingualComponent Class Reference

Parsed information from the DVB multilingual component descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()
  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_ComponentTag
  Identifier for the component this descriptor describes.
- std::vector< std::pair
  < std::string, std::wstring > > m_Descriptions
  Mapping from ISO 639 language code to textual description in that language.

9.20.1 Detailed Description

Parsed information from the DVB multilingual component descriptor.

9.20.2 Member Data Documentation

9.20.2.1 int DtapiTs::DtDescDvbMultilingualComponent::m_ComponentTag

Identifier for the component this descriptor describes.

9.20.2.2 std::vector<std::pair<std::string, std::wstring> > DtapiTs::DtDescDvbMultilingualComponent::m_Descriptions

Mapping from ISO 639 language code to textual description in that language.

9.21 DtapiTs::DtDescDvbNetworkName Class Reference

Parsed information from the DVB network name descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- std::wstring m_NetworkName

  The name of the network that is described by the NIT that contains this descriptor.

9.21.1 Detailed Description

Parsed information from the DVB network name descriptor.

9.21.2 Member Data Documentation

9.21.2.1 std::wstring DtapiTs::DtDescDvbNetworkName::m_NetworkName

The name of the network that is described by the NIT that contains this descriptor.

9.22 DtapiTs::DtDescDvbSDelivery Class Reference

Parsed information from the DVB satellite delivery system descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_FecInner

  The inner FEC scheme used.
- __int64 m_Frequency

  Frequency of the transmitted signal in Hz.
- bool m_IsDvbS2

  Indicates whether this is a DVB-S2 or DVB-S signal.
- int m_ModType

The modulation scheme used:

- int m_OrbitalPosition

  Orbital position in 0.1 degrees (192 means 19.2°)

- int m_Polarization

  The polarization of the transmitted signal.

- int m_RollOff

  Encoded roll-off factor used in DVB-S2.

- int m_SymbolRate

  Symbol rate in symbols/s.

- int m_WestEastFlag

  Indication of whether the satellite position is in the eatern or western part of the orbit.

9.22.1 Detailed Description

Parsed information from the DVB satellite delivery system descriptor.

9.22.2 Member Data Documentation

9.22.2.1 bool DtapiTs::DtDescDvbSDelivery::m_IsDvbS2

Indicates whether this is a DVB-S2 or DVB-S signal.

9.22.2.2 int DtapiTs::DtDescDvbSDelivery::m_ModType

The modulation scheme used:

- 0: Auto
- 1: QPSK
- 2: 8PSK
- 3: 16-QAM

9.22.2.3 int DtapiTs::DtDescDvbSDelivery::m_RollOff

Encoded roll-off factor used in DVB-S2.

Only meaningfull when m_IsDvbS2 is true.

9.22.2.4 int DtapiTs::DtDescDvbSDelivery::m_WestEastFlag

Indication of whether the satellite position is in the eatern or western part of the orbit.

0 means west, 1 means east.

9.23 DtapiTs::DtDescDvbService Class Reference

Parsed information from the DVB service descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- std::wstring m_ProviderName

  Name of the service provider.
- std::wstring m_ServiceName

  Name of the service.
- int m_ServiceType

  Service type field as defined by table 87 in ETSI EN 300 468.

9.23.1 Detailed Description

Parsed information from the DVB service descriptor.

9.23.2 Member Data Documentation

9.23.2.1 int DtapiTs::DtDescDvbService::m_ServiceType

Service type field as defined by table 87 in ETSI EN 300 468.

9.24 DtapiTs::DtDescDvbServiceList Class Reference

Parsed information from the DVB service list descriptor.

`#include <DTAPITS.h>`

Classes

- struct ServiceListItem

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- std::vector< ServiceListItem > m_Services

  List of services described in this descriptor.

9.24.1 Detailed Description

Parsed information from the DVB service list descriptor.

9.25 DtapiTs::DtDescDvbSubtitling Class Reference

Parsed information from the DVB subtitling descriptor.

`#include <DTAPITS.h>`

Classes

- struct Subtitling

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- std::vector< Subtitling > m_SubtitleDescs

  Description of the various subtitle streams.

9.25.1 Detailed Description

Parsed information from the DVB subtitling descriptor.

9.26 DtapiTs::DtDescDvbTDelivery Class Reference

Parsed information from the DVB terrestrial delivery system descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_Bandwidth

  Bandwidth of the transmission.
- __int64 m_CentreFrequency

  Frequency of the transmitted signal in Hz.

- int m_CodeRateHpStream

  Inner FEC scheme used for high-priority stream.
- int m_CodeRateLpStream

  Inner FEC scheme used for low priority stream.
- int m_Constellation

  Constellation pattern.
- int m_GuardInterval

  Guard interval.
- int m_HierarchyInformation

  This specifies whether the transmission is hierarchical.
- int m_MpeFecIndicator

  If 0 at least one elementary stream used MPE-FEC.
- bool m_OtherFreqFlag

  True if one or more other frequencies are used.
- bool m_Priority

  True if this is a high priority stream.
- int m_TimeSlicingIndicator

  If 1, time slicing is not used.
- int m_TransmissionMode

  Transmission mode.

9.26.1 Detailed Description

Parsed information from the DVB terrestrial delivery system descriptor.

9.26.2 Member Data Documentation

9.26.2.1 int DtapiTs::DtDescDvbTDelivery::m_HierarchyInformation

This specifies whether the transmission is hierarchical.

9.27 DtapiTs::DtDescDvbTeletext Class Reference

Parsed information from the DVB teletext descriptor.

`#include <DTAPITS.h>`

Classes

- struct Teletext

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- std::vector< Teletext > m_TeletextLoop

  All items from the loop in this descriptor.

9.27.1 Detailed Description

Parsed information from the DVB teletext descriptor.

9.28 DtapiTs::DtDescMpegCa Class Reference

Parsed information from the conditional access descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_CaPid

  The PID that contains either ECM or EMM information.
- int m_CaSystemId

  The type fo CA system in the associated ECM and/or EMM streams.
- std::vector< uint8_t > m_PrivData

  Contents of the private bytes in the descriptor.

9.28.1 Detailed Description

Parsed information from the conditional access descriptor.

9.28.2 Member Data Documentation

9.28.2.1 int DtapiTs::DtDescMpegCa::m_CaPid

The PID that contains either ECM or EMM information.

9.28.2.2 int DtapiTs::DtDescMpegCa::m_CaSystemId

The type fo CA system in the associated ECM and/or EMM streams.

See Also

DtCaSystemId2String.

9.29 DtapiTs::DtDescMpegLanguage Class Reference

Parsed information from the ISO 639 language descriptor.

`#include <DTAPITS.h>`

Classes

- struct LangCode

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- std::vector< LangCode > m_Codes

  List of all audio codes defined in this descriptor.

9.29.1 Detailed Description

Parsed information from the ISO 639 language descriptor.

9.29.2 Member Data Documentation

9.29.2.1 std::vector<LangCode> DtapiTs::DtDescMpegLanguage::m_Codes

List of all audio codes defined in this descriptor.

9.30 DtapiTs::DtDescMpegPrivDataIndicator Class Reference

Parsed information from private data indicator descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- unsigned int m_PrivDataInd

  Value of the private data indicator field.

9.30.1 Detailed Description

Parsed information from private data indicator descriptor.

9.31 DtapiTs::DtDescMpegRegistration Class Reference

Parsed information from the registration descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- std::vector< uint8_t > m_AdditionalInfo

  Extra bytes in the descriptor.
- unsigned int m_FormatIdentifier

  Unique format identifier.

9.31.1 Detailed Description

Parsed information from the registration descriptor.

9.32 DtapiTs::DtDescMpegVideoStream Class Reference

Parsed information from the video stream descriptor.

`#include <DTAPITS.h>`

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- int m_ChromaFormat

  2-bit field coded like chroma_format in H.262.
- bool m_ConstrainedParameter

  If true the video stream contains only constrained ISO/IEC 1172-2 video data.
- int m_FrameRateCode

  A 4-bit code that indicates the frame rate.

- bool m_FrameRateExtension

  True when at least one of frame_rate_extension_n or frame_rate_extension_d fields in the stream are non-zero.
- bool m_Mpeg1Only

  If true this the stream that this descriptor describes holds only ISO/IEC 11172-2 data.
- bool m_MultipleFrameRates

  Flag that indicates multiple frame rates may be present in the video stream.
- int m_ProfileLevelIndication

  See profile_and_level_indication field in ITU-T Rec H.262.
- bool m_StillPicture

  If true the stream contains only still pictures.

9.32.1 Detailed Description

Parsed information from the video stream descriptor.

The tag of this descriptor is 2.

9.32.2 Member Data Documentation

9.32.2.1 int DtapiTs::DtDescMpegVideoStream::m_ChromaFormat

2-bit field coded like chroma_format in H.262.

Note

This field is only valid when m_Mpeg1Only is false.

9.32.2.2 bool DtapiTs::DtDescMpegVideoStream::m_ConstrainedParameter

If true the video stream contains only constrained ISO/IEC 1172-2 video data.

9.32.2.3 int DtapiTs::DtDescMpegVideoStream::m_FrameRateCode

A 4-bit code that indicates the frame rate.

See section 6.3.3 of ITU-T Rec H.262.

9.32.2.4 bool DtapiTs::DtDescMpegVideoStream::m_FrameRateExtension

True when at least one of frame_rate_extension_n or frame_rate_extension_d fields in the stream are non-zero.

Note

This field is only valid when m_Mpeg1Only is false.

9.32.2.5 bool DtapiTs::DtDescMpegVideoStream::m_Mpeg1Only

If true this the stream that this descriptor describes holds only ISO/IEC 11172-2 data.

9.32.2.6 bool DtapiTs::DtDescMpegVideoStream::m_MultipleFrameRates

Flag that indicates multiple frame rates may be present in the video stream.

9.32.2.7 int DtapiTs::DtDescMpegVideoStream::m_ProfileLevelIndication

See profile_and_level_indication field in ITU-T Rec H.262.

Note

This field is only valid when m_Mpeg1Only is false.

9.33 DtapiTs::DtDescPrivLcn Class Reference

Class that can be used to parse logical channel numbers from a (list of) descriptors.

`#include <DTAPITS.h>`

Classes

- struct DtLogicalChannelNumber

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT **Parse** (const DtDescriptor &Descriptor)
- DTAPITS_RESULT **Parse** (const std::vector< DtDescriptor > &Descriptors)

Public Attributes

- std::vector
  < DtLogicalChannelNumber > m_Lcns

  Logical channel numbers.

9.33.1 Detailed Description

Class that can be used to parse logical channel numbers from a (list of) descriptors.

The LCN descriptor has tag 0x83. There are several different documents from multiple organisations that describe the content in different ways. This class is therefor dependend on a correct private data specifier descriptor that precedes the LCN descriptor. If no private data specifier descriptor is present, parsing the descriptor will fail. The following private data specifiers are supported:

- 0x00000028: EACEM
- 0x00000029: NorDig
- 0x00000037: FreeView (NZ)
- 0x0000233A: Independent Television Commission (D-Book)
- 0x00003200-0x0000320F: Australian Terrestrial Television Networks

9.34 DtapiTs::DtDescriptor Class Reference

DtDescriptor represents a single descriptor within one of the tables.

`#include <DTAPITS.h>`

Public Member Functions

- **DtDescriptor** (const DtDescriptor &)
- DtDescriptor & **operator=** (DtDescriptor)

Public Attributes

- uint8_t * m_Buf

  The actual contents of this descriptor.
- int m_DescriptorType

  Type of descriptor if it could be determined, otherwise -1.
- uint8_t m_ExtendedTag

  The extension descriptor (tag 0x7F) will be split by DTAPI-TS it's individual parts.
- int m_Len

  Length of the m_Buf array.
- __int64 m_Pds

  The private data descriptor specifier from the previous data descriptor in the same table section and loop as this descriptor.
- uint8_t m_Tag

  Descriptor tag.

Friends

- void swap (DtDescriptor &First, DtDescriptor &Second)

  Swap the contents of two descriptor objects.

9.34.1 Detailed Description

DtDescriptor represents a single descriptor within one of the tables.

9.34.2 Member Data Documentation

9.34.2.1 int DtapiTs::DtDescriptor::m_DescriptorType

Type of descriptor if it could be determined, otherwise -1.

See Also

enum DtDescriptorType

9.34.2.2 uint8_t DtapiTs::DtDescriptor::m_ExtendedTag

The extension descriptor (tag 0x7F) will be split by DTAPI-TS it's individual parts.

Each part will have m_Tag set to 0x7F and this variable set to the extended tag descriptor. For other descriptors this will be set to 0.

9.34.2.3 __int64 DtapiTs::DtDescriptor::m_Pds

The private data descriptor specifier from the previous data descriptor in the same table section and loop as this descriptor.

Otherwise -1

9.35 DtapiTs::DtDvbCNitInfo Class Reference

DVB-C delivery system information as extracted from the NIT.

`#include <DTAPITS.h>`

Public Attributes

- int Constellation

  Modulation scheme, one of DTAPI_MOD_(16|32|64 |128|256|TYPE_UNK).
- int FecInner

  Code rate, one of DTAPI_MOD_∗.
- DtFecOuter FecOuter

  Outer FEC.
- __int64 Frequency

  Frequency in Hz.
- int SymbolRate

  Symbol rate in symbols/s.

9.35.1 Detailed Description

DVB-C delivery system information as extracted from the NIT.

This information only applies when the transport stream was distributed via cable.

9.35.2 Member Data Documentation

9.35.2.1 int DtapiTs::DtDvbCNitInfo::Constellation

Modulation scheme, one of DTAPI_MOD_(16|32|64 |128|256|TYPE_UNK).

9.36 DtapiTs::DtDvbShModInfo Class Reference

DVB-SH modulation info.

`#include <DTAPITS.h>`

Public Attributes

- union {
  DtDvbShOfdmInfo m_Ofdm
  Ofdm modulation info.
  DtDvbShTdmInfo m_Tdm
  Tdm modulation info.
  };

- int m_CommonMultiplier

  The length increment in Interleaving Units between two consecutive taps of the physical interleaver belonging to the late tap part.
- bool m_CompleteInterleaver

  When false, only m_CommonMultiplier is set, otherwise the other interleaver information is valid too.
- bool m_InterleaverPresence

Whether or not interleaver information is present.

- DtShModType m_ModType

  Modulation type.

- int m_NofLateTaps

  The number of taps of the physical time interleaver that belong to the late tap part.

- int m_NofSlices

  The number of slices over which the physical time interleaver spans.

- int m_NonLateIncrement

  The length increment between two consecutive taps belonging to the same non-late slice of the physical interleaver.

- int m_SliceDistance

  The number of SH frames between two consecutive slices of the physical time interleaver.

9.36.1 Detailed Description

DVB-SH modulation info.

9.36.2 Member Data Documentation

9.36.2.1 int DtapiTs::DtDvbShModInfo::m_CommonMultiplier

The length increment in Interleaving Units between two consecutive taps of the physical interleaver belonging to the late tap part.

Only valid when m_InterleaverPresense==true.

9.36.2.2 bool DtapiTs::DtDvbShModInfo::m_CompleteInterleaver

When false, only m_CommonMultiplier is set, otherwise the other interleaver information is valid too.

Only meaningfull when m_InterleaverPresence is `true`.

9.36.2.3 int DtapiTs::DtDvbShModInfo::m_NofLateTaps

The number of taps of the physical time interleaver that belong to the late tap part.

Only valid when m_InterleaverPresense==true && m_CompleteInterleaver==true.

9.36.2.4 int DtapiTs::DtDvbShModInfo::m_NofSlices

The number of slices over which the physical time interleaver spans.

Only valid when m_InterleaverPresense==true && m_CompleteInterleaver==true.

9.36.2.5 int DtapiTs::DtDvbShModInfo::m_NonLateIncrement

The length increment between two consecutive taps belonging to the same non-late slice of the physical interleaver.

Actual length increment is computed by multiplying this field with m_CommonMultiplier. Only valid when m_InterleaverPresense==true && m_CompleteInterleaver==true

9.36.2.6 int DtapiTs::DtDvbShModInfo::m_SliceDistance

The number of SH frames between two consecutive slices of the physical time interleaver.

Only valid when m_InterleaverPresense==true && m_CompleteInterleaver==true

9.37 DtapiTs::DtDvbShNitInfo Class Reference

DVB-SH delivery system information as extracted from the NIT.

`#include <DTAPITS.h>`

Public Attributes

- int m_DiversityMode

  4-bit field that describes the diversity modes, possible values:
- std::vector< DtDvbShModInfo > m_ModInfo

  One (in a SFN) or more (in a non-SFN) sets of modulation info.

9.37.1 Detailed Description

DVB-SH delivery system information as extracted from the NIT.

This information only applies when the transport stream was distributed as DVB-SH signal.

9.37.2 Member Data Documentation

9.37.2.1 int DtapiTs::DtDvbShNitInfo::m_DiversityMode

4-bit field that describes the diversity modes, possible values:

```
Value (binary)    paTS   FEC diversity    FEC at phy    FEC at link
0000              no     no               no            no
1000              yes    no               no            no
1101              yes    yes              no            yes
1110              yes    yes              yes           no
1111              yes    yes              yes           yes
Other values; reserved for future use
```

9.37.2.2 std::vector<DtDvbShModInfo> DtapiTs::DtDvbShNitInfo::m_ModInfo

One (in a SFN) or more (in a non-SFN) sets of modulation info.

9.38 DtapiTs::DtDvbShOfdmInfo Class Reference

DVB-SH OFDM modulation info.

`#include <DTAPITS.h>`

Public Attributes

- DtShBandwidth m_Bandwith

  OFDM bandwidth.
- DtShCodeRate m_CodeRate

  Code rate.
- bool m_CommonFrequency

  True if the modulation is used over a common frequency, false otherwise.

- int m_Constellation

  Stream constellation and hierarchy, see table 125 of EN 300 468 v1.13.1 for more information.
- DtGuardInterval m_GuardInterval

  Guard interval.
- int m_Priority

  Indication of the streams hierarchical priority, interpretation depends on m_Constellation.
- DtTransmissionMode m_TransmissionMode

  Transmission mode.

9.38.1 Detailed Description

DVB-SH OFDM modulation info.

9.38.2 Member Data Documentation

9.38.2.1 bool DtapiTs::DtDvbShOfdmInfo::m_CommonFrequency

True if the modulation is used over a common frequency, false otherwise.

9.38.2.2 int DtapiTs::DtDvbShOfdmInfo::m_Constellation

Stream constellation and hierarchy, see table 125 of EN 300 468 v1.13.1 for more information.

9.38.2.3 int DtapiTs::DtDvbShOfdmInfo::m_Priority

Indication of the streams hierarchical priority, interpretation depends on m_Constellation.

9.39 DtapiTs::DtDvbShTdmInfo Class Reference

DVB-SH TDM modulation info.

`#include <DTAPITS.h>`

Public Attributes

- DtShCodeRate m_CodeRate

  Code rate.
- DtShModMode m_ModMode

  Modulation mode used.
- DtPolarization m_Polarization

  Polarization of the transmitted signal.
- DtRollOff m_RollOff

  Roll-off factor.
- int m_SymbolRate

  The TDM symbol rate.

9.39.1 Detailed Description

DVB-SH TDM modulation info.

9.39.2 Member Data Documentation

9.39.2.1 int DtapiTs::DtDvbShTdmInfo::m_SymbolRate

The TDM symbol rate.

See table 122 of EN 300 468 v1.13.1 for detailed information.

9.40 DtapiTs::DtDvbSNitInfo Class Reference

DVB-S delivery system information as extracted from the NIT.

`#include <DTAPITS.h>`

Public Attributes

- int FecInner
  Code rate, one of DTAPI_MOD_∗.
- __int64 Frequency
  Frequency in Hz.
- int InputStreamIdentifier
  DVB-S2 ISI (input stream identifier).
- bool IsDvbS2
  True for DVB-S2, false for DVB-S.
- int ModType
  Modulation type.
- double OrbitalPosition
  Satellite position in orbit.
- DtPolarization Polarization
  Polarization of the transmitted signal.
- int RollOff
  DVB-S(2) rolloff factor. DTAPI_MOD_ROLLOFF_∗.
- bool S2FieldsPresent
  Indicates whether the fields specific for DVB-S2 contain valid values.
- int ScramblingSequenceIndex
  Index of the DVB-S2 physical layer scrambling sequence.
- int SymbolRate
  Symbol rate in symbols/s.
- DtWeFlag WestEastFlag
  Flag indicating the direction of the satellite.

9.40.1 Detailed Description

DVB-S delivery system information as extracted from the NIT.

This information only applies when the transport stream was distributed via satellite.

9.40.2 Member Data Documentation

9.40.2.1 int DtapiTs::DtDvbSNitInfo::InputStreamIdentifier

DVB-S2 ISI (input stream identifier).

Only valid when `S2FieldsPresent` is true.

9.40.2.2 int DtapiTs::DtDvbSNitInfo::ModType

Modulation type.

Possible values: DTAPI_MOD_ (DVBS_QPSK|DVBS2_QPSK|DVBS2_8PSK|TYPE_UNK)

9.40.2.3 bool DtapiTs::DtDvbSNitInfo::S2FieldsPresent

Indicates whether the fields specific for DVB-S2 contain valid values.

These fields are ScramblingSequenceIndex and InputStreamIdentifier

9.40.2.4 int DtapiTs::DtDvbSNitInfo::ScramblingSequenceIndex

Index of the DVB-S2 physical layer scrambling sequence.

Only valid when `S2FieldsPresent` is true.

9.41 DtapiTs::DtDvbT2CellInfo Class Reference

Information about a DVB-T2 cell.

`#include <DTAPITS.h>`

Public Attributes

- int m_CellId

  Unique identifier for this cell in a network.
- std::vector< __int64 > m_CentreFrequencies

  Centre frequencies used by this cell.
- std::vector< DtDvbT2SubCellInfo > m_SubCellInfo

  Sub-cell information.

9.41.1 Detailed Description

Information about a DVB-T2 cell.

9.42 DtapiTs::DtDvbT2NitInfo Class Reference

DVB-T2 delivery system information as extracted from the NIT.

`#include <DTAPITS.h>`

Public Attributes

- int m_Bandwith

  Bandwidth for the modulated signal, one of DTAPI_DVBT2_(1_7MHZ|5MHZ|6MHZ|7MHZ|8MHZ|10MHZ|UNK)
- std::vector< DtDvbT2CellInfo > m_CellInfo

  List of cells used by this DVB-T2 network.
- int m_GuardInterval

  Guard interval, can be one of DTAPI_DVBT2_GI_(1_128|1_32|1_16|19_256|1_8|19|128 |1_4|UNK)

- DtDvbT2MisoMode m_MisoMode

  SIDO/MISO mode.
- bool m_OtherFrequencyUsed

  Indicates whether or not other frequencies (non-FTS case) or other groups of frequencies (TFS case) are in use.
- int m_PlpId

  Identifier for this data PLP, unique in the DVB-T2 system.
- int m_T2SystemId

  Identifier for the current system, uqniue in the DVB-T2 network.
- bool m_TfsArangement

  Indicates whether a TFS arrangement is in place.
- int m_TransmissionMode

  Transmission mode, can be one of DTAPI_DVBT2_FFT_(1K|2K|4K|8K|16K|32K|UNK)

9.42.1 Detailed Description

DVB-T2 delivery system information as extracted from the NIT.

This information only applies when the transport stream was distributed as DVB-T2 terrestrial signal.

9.42.2 Member Data Documentation

9.42.2.1 bool DtapiTs::DtDvbT2NitInfo::m_OtherFrequencyUsed

Indicates whether or not other frequencies (non-FTS case) or other groups of frequencies (TFS case) are in use.

9.42.2.2 int DtapiTs::DtDvbT2NitInfo::m_PlpId

Identifier for this data PLP, unique in the DVB-T2 system.

9.42.2.3 int DtapiTs::DtDvbT2NitInfo::m_T2SystemId

Identifier for the current system, uqniue in the DVB-T2 network.

9.43 DtapiTs::DtDvbT2SubCellInfo Class Reference

Information about a single DVB-T2 sub-cell.

`#include <DTAPITS.h>`

Public Attributes

- int m_SubCellId

  Identifier for a sub-cell, unique for the single DVB-T2 cell this sub-cell belongs to.
- __int64 m_TransposerFrequency

  Transposer frequency in Hz.

9.43.1 Detailed Description

Information about a single DVB-T2 sub-cell.

9.43.2 Member Data Documentation

9.43.2.1 int DtapiTs::DtDvbT2SubCellInfo::m_SubCellId

Identifier for a sub-cell, unique for the single DVB-T2 cell this sub-cell belongs to.

9.44 DtapiTs::DtDvbTNitInfo Class Reference

DVB-T delivery system information as extracted from the NIT.

`#include <DTAPITS.h>`

Public Attributes

- int Bandwith

  Bandwidth of the signal.
- __int64 CentreFrequency

  Centre frequency in Hz.
- int CodeRateHpStream

  Code rate for the high priority part of an hierarchical transmission.
- int CodeRateLpStream

  DTAPI_MOD_(1_2|2_3|3_4|5_6|7_8|CR_UNK)
- int Consellation

  Constellation pattern, one of DTAPI_MOD_DVBT_(QPSK|QAM16|QAM64|CO_UNK)
- int GuardInterval

  Guard interval, can be one of DTAPI_MOD_DVBT_G_1_(4|8|16|32)
- int HierarchyInformation

  Whether or not the transmission is hierarchical.
- bool IsHighPriority

  Indicates this is a high-priority stream.
- bool MpeFecUsed

  True when MPE-FEC is used on the stream.
- bool OtherFrequencyUsed

  Set to true if the transport stream indicates that one or more other frequencies are used too.
- bool TimeSlicingUsed

  Indicates whether or not time slicing is used.
- int TransmissionMode

  Transmission mode.

9.44.1 Detailed Description

DVB-T delivery system information as extracted from the NIT.

This information only applies when the transport stream was distributed as terrestrial signal.

9.44.2 Member Data Documentation

9.44.2.1 int DtapiTs::DtDvbTNitInfo::Bandwith

Bandwidth of the signal.

One of DTAPI_MOD_DVBT_(8MHZ|7MHZ|6MHZ|5MHZ|BW_UNK).

9.44.2.2 int DtapiTs::DtDvbTNitInfo::CodeRateHpStream

Code rate for the high priority part of an hierarchical transmission.

Only valid when `HierarchyInformation` is set to `DTAPI_MOD_DVBT_INDEPTH`. Can be one of DTAPI_-MOD_(1_2|2_3|3_4|5_6|7_8|CR_UNK)

9.44.2.3 int DtapiTs::DtDvbTNitInfo::HierarchyInformation

Whether or not the transmission is hierarchical.

One of DTAPI_MOD_DVBT_(INDEPTH|NATIVE)

9.44.2.4 bool DtapiTs::DtDvbTNitInfo::OtherFrequencyUsed

Set to true if the transport stream indicates that one or more other frequencies are used too.

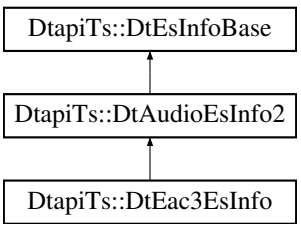
9.44.2.5 int DtapiTs::DtDvbTNitInfo::TransmissionMode

Transmission mode.

Can be one of DTAPI_MOD_DVBT_(2K|4K|8K|MD_UNK)

9.45 DtapiTs::DtEac3EsInfo Class Reference

Inheritance diagram for DtapiTs::DtEac3EsInfo:



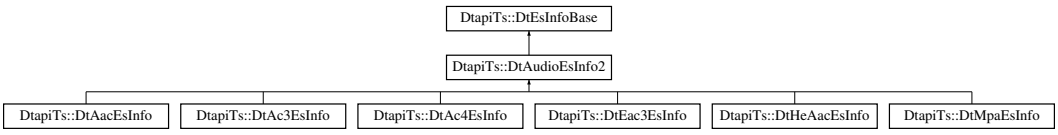
Public Member Functions

- virtual void **Clear** ()

Additional Inherited Members

9.46 DtapiTs::DtEsInfoBase Class Reference

Inheritance diagram for DtapiTs::DtEsInfoBase:



Classes

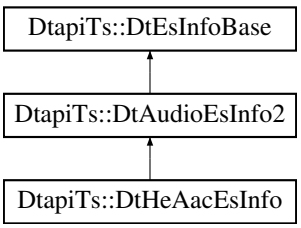
- class InfoField

Public Member Functions

- virtual void **Clear** ()=0

9.47 DtapiTs::DtHeAacEsInfo Class Reference

Inheritance diagram for DtapiTs::DtHeAacEsInfo:



Public Member Functions

- virtual void **Clear** ()

Additional Inherited Members

9.48 DtapiTs::DtJitterPoint Class Reference

Public Attributes

- double **m_AcErr**
- double **m_OjErr**
- DtTimestamp **m_Timestamp**

9.49 DtapiTs::DtDescPrivLcn::DtLogicalChannelNumber Struct Reference

Public Attributes

- bool m_IsVisible

  Should this channel be visible to the end-user?
- int m_Lcn

  Logical channel number.
- int m_ServiceId

  Service identifier.

9.49.1 Member Data Documentation

9.49.1.1 bool DtapiTs::DtDescPrivLcn::DtLogicalChannelNumber::m_IsVisible

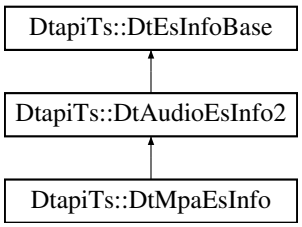
Should this channel be visible to the end-user?

Note

This field is not always present. If it's not present it'll default to true.

9.50 DtapiTs::DtMpaEsInfo Class Reference

Inheritance diagram for DtapiTs::DtMpaEsInfo:



Public Member Functions

- virtual void **Clear** ()

Public Attributes

- InfoField< DtMpaLayer > **m_Layer**
- InfoField< DtMpaVersion > **m_Version**

Additional Inherited Members

9.51 DtapiTs::DtPcr Class Reference

Class representing PCR timestamp:

`#include <DTAPITS.h>`

Public Member Functions

- **DtPcr** (__int64 PcrFlat=PCR_INVALID)
- __int64 **GetFlat** () const
- DtPcr & **GetFromTp** (const class DtTp &Tp)
- bool IsValid () const

  True, if a valid PCR is present.
- DtPcr **operator+** (__int64 Offset) const
- DtPcr **operator+=** (__int64 Offset)
- DtPcr **operator-** (__int64 Offset) const
- __int64 **operator-** (const DtPcr &) const
- DtPcr **operator-=** (__int64 Offset)
- bool **PutIntoTp** (class DtTp &Tp) const
- double **ToSec** () const

Static Public Attributes

- static const __int64 **PCR_CLOCK_FREQ** = 27000000LL
- static const __int64 **PCR_HALFRANGE** = 0x100000000LL * 300LL
- static const __int64 **PCR_INVALID** = LLONG_MIN
- static const __int64 **PCR_RANGE** = 0x200000000LL * 300LL

Protected Attributes

- __int64 **m_PcrFlat**

9.51.1 Detailed Description

Class representing PCR timestamp:

- Overloaded operators for PCR arithmetic.

9.52 DtapiTs::DtPcrInfo Class Reference

Some statistics about the bitrate of a PID, service or transport stream.

`#include <DTAPITS.h>`

Public Member Functions

- DtPcrInfo ()

  Construct a new DtPcrInfo object.

Public Attributes

- double m_AcMax

  Maximum AC jitter error during last window (ns)
- double m_AcStdDev

  Standard deviation of AC jitter.
- double m_AvgRate

  Average number of PCRs per second.
- double m_Df

  Network delay factor (difference between min OJ and max OJ errors.
- double m_OjMax

  Maximum OJ jitter error during last window (ns)
- double m_OjStdDev

  Standard deviation of OJ jitter.
- int m_TsRate

  Transport stream rate according to PCRs or -1.

9.52.1 Detailed Description

Some statistics about the bitrate of a PID, service or transport stream.

9.52.2 Member Data Documentation

9.52.2.1 double DtapiTs::DtPcrInfo::m_Df

Network delay factor (difference between min OJ and max OJ errors.

9.53 DtapiTs::DtPes Class Reference

Classes

- class DataBuffer

Public Types

- enum **PesStreamId** {
  STREAMID_PMT = 0xBC,
  STREAMID_PRIV1 = 0xBD,
  STREAMID_PADDING = 0xBE,
  STREAMID_PRIV2 = 0xBF,
  STREAMID_AUDIO_MIN = 0xC0,
  STREAMID_AUDIO_MAX = 0xDF,
  STREAMID_VIDEO_MIN = 0xE0,
  STREAMID_VIDEO_MAX = 0xEF,
  STREAMID_ECM = 0xF0,
  STREAMID_EMM = 0xF1 }

Public Member Functions

- **DtPes** (uint8_t *pData=NULL, int Size=0, int NumValid=0)
- bool **Add** (const DtTp &Tp, bool &MoreData)
- bool **Add** (const DtTp &Tp)
- void **Clear** ()
- uint8_t * **Data** (int &Size) const
- bool **HasExtendedHeader** (int StreamId) const
- uint8_t * **Header** (int &Size) const
- bool **IsComplete** () const
- DtPtsDts **PtsDts** () const
- bool **PtsDts** (const DtPtsDts &)
- int **StreamId** () const

Static Public Attributes

- static const int **BUF_DEFAULT_SIZE** = 32*1024
- static const int **BUF_GROW_SIZE** = 8*1024
- static const int **BUF_MAX_SIZE** = 8*1024*1024

Protected Types

- enum **PesStatus** {
  PES_EMPTY,
  PES_BUILD,
  PES_COMPLETE }

Protected Member Functions

- bool **Add** (const uint8_t *pBuf, int Size)
- int **PesPacketLenght** () const

Protected Attributes

- DataBuffer **m_Buf**
- int **m_NumValid**
- enum DtapiTs::DtPes::PesStatus **m_Status**

9.54 DtapiTs::DtPidInfo Class Reference

Class that contains general information about one PID.

`#include <DTAPITS.h>`

Public Member Functions

- DtPidInfo ()

  Construct a new DtPidInfo object.
- DtPidInfo (const DtPidInfo &Other)

  Create a copy of an existing DtPidInfo object.
- virtual ~DtPidInfo ()

  Destroy the DtPidInfo object.
- std::wstring GetDescription () const

  Get a textual description of the contents of this pid.
- bool HasTableType (DtTableType Type) const

  Test whether a given table has been seen on this pid.
- DtPidInfo & operator= (const DtPidInfo &Other)

  Assign the contents of another DtPidInfo object to this object.

Public Attributes

- std::list< int > m_AtscTypes

  ATSC table types (as defined in MGT)
- DtAudioEsInfo * m_AudioEs

  Audio elementary stream information, can be NULL.
- DtBitrate m_Bitrate

  Bitrate statistics.
- int m_CcErrors

  Number of continuity count errors.
- DtPcrInfo m_Pcrs

  Statistics about PCRs in this PID.
- int m_Pid

  Pid number, 0..8191 inclusive.
- DtScrambling m_Scrambled

  How the last packet with this PID was scrambled.
- bool m_SeenBefore

  True after we've seen the first packet from this PID.

- int m_StreamId

  Stream ID extracted from PES header.
- DtStreamType m_StreamType

  Stream type based on PMT.
- DtVideoEsInfo * m_VideoEs

  Video elementary stream information, can be NULL.

Protected Attributes

- __int64 m_TableTypeMask

  Mask of PSI(P)/SI tables that are valid on this PID.

9.54.1 Detailed Description

Class that contains general information about one PID.

A PID without any packets can still get a DtPidInfo object to hold stream information, this will happen for for example if the PID is referenced in a PMT.

Examples:

example3.cpp.

9.54.2 Member Function Documentation

9.54.2.1 std::wstring DtapiTs::DtPidInfo::GetDescription ( ) const

Get a textual description of the contents of this pid.

In case of a known elementary stream type or known tables the name of those will be returned. If the contents of this pid have not been recognized (yet) this function will return an empty string.

Returns

: String representing the pid contents or an empty string.

Examples:

example3.cpp.

9.54.2.2 bool DtapiTs::DtPidInfo::HasTableType (DtTableType *Type*) const

Test whether a given table has been seen on this pid.

Parameters

Type,:	The table type to test for.

Returns

: True if the table has occured on this pid, false otherwise.

9.54.3 Member Data Documentation

9.54.3.1 bool DtapiTs::DtPidInfo::m_SeenBefore

True after we've seen the first packet from this PID.

Can be false in case we have extracted information about this PID from the contents of other PIDs but no packets from this PID have arrived yet.

9.54.3.2 __int64 DtapiTs::DtPidInfo::m_TableTypeMask [protected]

Mask of PSI(P)/SI tables that are valid on this PID.

Don't use this directly, use HasTableType instead.

9.55 DtapiTs::DtTablePat::DtProgramMapping Struct Reference

ServiceId to PMT Pid mapping for a single service.

`#include <DTAPITS.h>`

Public Attributes

- int m_Pid

  Pid that contains the PMT for this service.
- int m_ServiceId

  A unique identifier for a service in this TS.

9.55.1 Detailed Description

ServiceId to PMT Pid mapping for a single service.

9.56 DtapiTs::DtPtsDts Class Reference

Class representing PTS/DTS timestamp:

`#include <DTAPITS.h>`

Public Member Functions

- DtPtsDts (__int64 PtsFlat=PTSDTS_INVALID, __int64 DtsFlat=PTSDTS_INVALID)

  PTS as "flat" 64-bit number (range 33 bits)
- double **DtsToSec** () const
- __int64 **GetDtsFlat** () const
- DtPtsDts & **GetFromPes** (const class DtPes &Pes)
- __int64 **GetPtsFlat** () const
- bool IsDtsValid () const

  True, if a valid DTS is present.
- bool IsPtsValid () const

  True, if a valid PTS is present.
- DtPtsDts **operator+** (__int64)
- DtPtsDts **operator+=** (__int64 Offset)
- DtPtsDts **operator-** (__int64 Offset)
- DtPtsDts **operator-=** (__int64 Offset)

- double **PtsToSec** () const
- bool **PutIntoPes** (class DtPes &Pes) const

Static Public Attributes

- static const __int64 **PTSDTS_CLOCK_FREQ** = 27000000LL/300LL
- static const __int64 **PTSDTS_HALFRANGE** = 0x100000000LL
- static const __int64 **PTSDTS_INVALID** = LLONG_MIN
- static const __int64 **PTSDTS_RANGE** = 0x200000000LL

Protected Attributes

- __int64 **m_DtsFlat**
- __int64 m_PtsFlat

  DTS as "flat" 64-bit number (range 33 bits)

9.56.1 Detailed Description

Class representing PTS/DTS timestamp:

- Overloaded operators for PTS/DTS arithmetic.

9.57 DtapiTs::DtServiceComponentInfo Class Reference

Information about a service component.

`#include <DTAPITS.h>`

Public Member Functions

- DtServiceComponentInfo (int Pid)

  Create a new DtServiceComponentInfo object.

Public Attributes

- int m_BroadcastId

  Broadcast ID from descriptor or -1.
- DtCaSystemList m_CaSystems

  List of conditional access systems specific for this component.
- int m_ComponentType

  Component type as defined in EN 300 458 Table 26.
- std::wstring m_Description

  A textual descripton of this service component.
- bool m_HasAc3Desc

  True when an Ac3 descriptor was found.
- bool m_HasAc4Desc

  True when an Ac4 descriptor was found.
- bool m_HasAES3Desc

  True when an AES3 registration descr. was found.
- bool m_HasEAc3Desc

True when an E-Ac3 descriptor was found.

- bool m_HasPrivateDataDesc

  True if a private data indicator descriptor was found for this component.

- bool m_HasTeletextDesc

  True when an a teletext descriptor was found.

- int m_Pid

  Pid that carries the elementary stream.

- int m_StreamContent

  Stream content as defined in EN 300 458 Table 26.

- DtStreamType m_StreamType

  Streamtype as given by PMT.

9.57.1 Detailed Description

Information about a service component.

9.57.2 Member Data Documentation

9.57.2.1 DtCaSystemList DtapiTs::DtServiceComponentInfo::m_CaSystems

List of conditional access systems specific for this component.

See Also

DtServiceInfo::m_CaSystems
DtTsInfo::m_CaSystems

9.57.2.2 std::wstring DtapiTs::DtServiceComponentInfo::m_Description

A textual descripton of this service component.

This text is part of the transport stream. When not available this string will be empty.

9.57.2.3 bool DtapiTs::DtServiceComponentInfo::m_HasPrivateDataDesc

True if a private data indicator descriptor was found for this component.

9.58 DtapiTs::DtServiceInfo Class Reference

Information about a service.

`#include <DTAPITS.h>`

Public Types

- typedef std::vector
  < DtServiceComponentInfo > DtComponentList

  List of service components.

Public Member Functions

- DtServiceInfo ()

  Create a new DtServiceInfo object.
- const std::wstring & GetName () const

  Helper function to get the name of the service.
- bool InService (int Pid) const

  Helper function to determine is a specific PID is part of this service (i.e.

Public Attributes

- int m_AvgBitrate

  Average bitrate.
- DtCaSystemList m_CaSystems

  List of conditional access systems that are valid for all components of this service.
- DtComponentList m_Components

  List of components that make up this service.
- std::wstring m_ExtendedName

  Extended name of the service.
- int m_OrigServiceType

  Service type from SDT/VCT, -1 if unknown.
- int m_PcrPid

  Pid that contains the PCR for this service or -1.
- int m_PmtPid

  Pid that contains the PMT for this service.
- int m_ProgramNumber

  Program/service number.
- std::wstring m_ProviderName

  Name of the provider (only for DVB)
- DtServiceType m_ServiceType

  Simplified service type based on m_StreamType in the various PIDs.
- std::wstring m_ShortName

  Short name of the service.

9.58.1 Detailed Description

Information about a service.

Examples:

example1.cpp, and example3.cpp.

9.58.2 Member Function Documentation

9.58.2.1 const std::wstring& DtapiTs::DtServiceInfo::GetName ( ) const

Helper function to get the name of the service.

It'll return the extended name if it's non-empty, otherwise it'll return the short name.

Returns

: The name of this service, can be empty.

Examples:

example1.cpp, and example3.cpp.

9.58.2.2 bool DtapiTs::DtServiceInfo::InService (int *Pid*) const

Helper function to determine is a specific PID is part of this service (i.e.

the contains a component on the specified PID)

9.58.3 Member Data Documentation

9.58.3.1 DtCaSystemList DtapiTs::DtServiceInfo::m_CaSystems

List of conditional access systems that are valid for all components of this service.

See Also

DtServiceInfo::m_CaSystems
DtTsInfo::m_CaSystems

9.58.3.2 int DtapiTs::DtServiceInfo::m_OrigServiceType

Service type from SDT/VCT, -1 if unknown.

The interpretation depends on the stream type.

9.58.3.3 int DtapiTs::DtServiceInfo::m_ProgramNumber

Program/service number.

Unique identification for each service.

Examples:

example1.cpp.

9.58.3.4 DtServiceType DtapiTs::DtServiceInfo::m_ServiceType

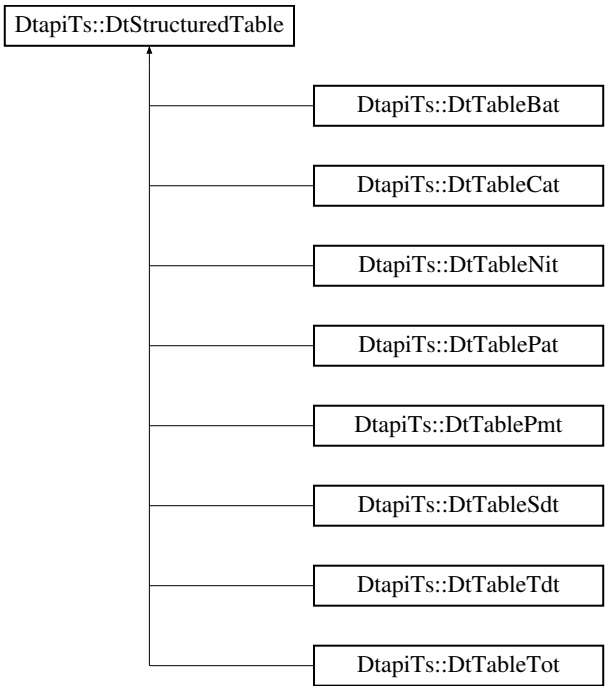
Simplified service type based on m_StreamType in the various PIDs.

9.59 DtapiTs::DtStructuredTable Class Reference

Base class for all structured table classes.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtStructuredTable:



Public Member Functions

- virtual void Clear ()=0

  Clear all data contained in this class.
- virtual DTAPITS_RESULT DecodeFromSection (const DtTableSection *)=0

  Parse the given TableSection and append all information to the current object.
- virtual DTAPITS_RESULT DecodeFromTable (const DtTable *)

  First clear all information and than parse all sections in the given table.

9.59.1 Detailed Description

Base class for all structured table classes.

9.59.2 Member Function Documentation

9.59.2.1 virtual DTAPITS_RESULT DtapiTs::DtStructuredTable::DecodeFromTable (const DtTable *) [virtual]

First clear all information and than parse all sections in the given table.

The extract information will be made available in this object. If an error occurs while parsing one of the table sections, the parsing will continue with the next section. Finally one of the errors will be returned.

9.60 DtapiTs::DtSubTableId Class Reference

Unique identifier for each sub-table.

`#include <DTAPITS.h>`

Public Member Functions

- DtSubTableId ()

  Create a new DtSubTableId object, initializes all fields to -1.
- DtSubTableId (int Pid, int TableId)

  Create a new DtSubTableId object for a subtable on the given Pid and with TableId.
- bool Matches (const DtSubTableId &Filter) const

  Does the current object match the filter? Every field in the filter that is set to -1 will be ignored, every other field has to be an exact match with the current object.
- bool operator< (const DtSubTableId &) const

  Compare two DtSubTableId objects.
- bool operator== (const DtSubTableId &) const

  Compare two DtSubTableId objects to see if they are exactly the same.

Public Attributes

- int m_Pid

  Pid this sub-table was found on.
- int m_TableId

  Table identifier, first byte of every section.
- int m_TableIdExt [3]

  Extended table id.

9.60.1 Detailed Description

Unique identifier for each sub-table.

Every field has either a meaningfull value or is set to -1. This also holds for the various callback functions. If you don't care about the value of a specific field you can set it to -1.

9.60.2 Member Function Documentation

9.60.2.1 bool DtapiTs::DtSubTableId::Matches (const DtSubTableId & *Filter*) const

Does the current object match the filter? Every field in the filter that is set to -1 will be ignored, every other field has to be an exact match with the current object.

9.60.2.2 bool DtapiTs::DtSubTableId::operator< (const DtSubTableId &) const

Compare two DtSubTableId objects.

Will sort numerically by the following fields (in this order): m_Pid, m_TableId, m_TableIdExt.

9.61 DtapiTs::DtTable Class Reference

Binary repesentation of an SI table.

`#include <DTAPITS.h>`

Public Member Functions

- DtTable ()

  Create a new (empty) DtTable object.
- DtTable (const DtTable &Other)

  Create a copy of an existing DtTable object.
- virtual ~DtTable ()

  Remove this object.
- DtTable & operator= (const DtTable &Other)

  Make this object a copy of an other DtTable object.

Public Attributes

- int m_LastSectionNum

  The number of the last valid section.
- DtTimestamp m_LastSeen

  Last time this table was complete.
- DtTableSection * m_Sections [DT_NUM_TABLE_SECTIONS]

  Array with pointers to the various table sections.
- int m_Version

  The version of this table.

9.61.1 Detailed Description

Binary repesentation of an SI table.

9.61.2 Constructor & Destructor Documentation

9.61.2.1 DtapiTs::DtTable::DtTable (const DtTable & *Other*)

Create a copy of an existing DtTable object.

Parameters

Other,:	the object to copy.

9.61.3 Member Function Documentation

9.61.3.1 DtTable& DtapiTs::DtTable::operator= (const DtTable & *Other*)

Make this object a copy of an other DtTable object.

Parameters

Other,:	the object to copy.

9.61.4 Member Data Documentation

9.61.4.1 DtTableSection* DtapiTs::DtTable::m_Sections[DT_NUM_TABLE_SECTIONS]

Array with pointers to the various table sections.

Each pointer can be NULl to indicate the given section is not available.

9.61.4.2 int DtapiTs::DtTable::m_Version

The version of this table.

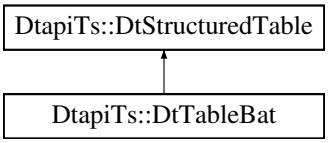
This should increment on every update.

9.62 DtapiTs::DtTableBat Class Reference

Structured version of the raw data contained in a Bouquet association table.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTableBat:



Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT DecodeFromSection (const DtTableSection *)

  Parse the given TableSection and append all information to the current object.
- DtTableBatInner * FindTs (int TsId, int OrigNetworkId)

  Search through m_TsLoop and return the first DtTableBatInner with a matching transport id and original network id.

Public Attributes

- std::vector< DtDescriptor > m_BouquetDescs

  All descriptors for bouquet.
- int m_BouquetId

  A unique identifier for this bouquet.
- std::vector< DtTableBatInner > m_TsLoop

  Structured information about each TS loop.

9.62.1 Detailed Description

Structured version of the raw data contained in a Bouquet association table.

9.62.2 Member Function Documentation

9.62.2.1 DtTableBatInner* DtapiTs::DtTableBat::FindTs (int *TsId,* int *OrigNetworkId*)

Search through m_TsLoop and return the first DtTableBatInner with a matching transport id and original network id.

If no object matches it'll return NULL.

9.63 DtapiTs::DtTableBatInner Class Reference

This class holds all descriptors of one sub-loop of a BAT table.

`#include <DTAPITS.h>`

Public Attributes

- int m_OrigNetworkId

  Original network identifier.
- int m_TransportStreamId

  Transport stream identifier.
- std::vector< DtDescriptor > m_TsDescriptors

  All descriptors for this transport stream loop.

9.63.1 Detailed Description

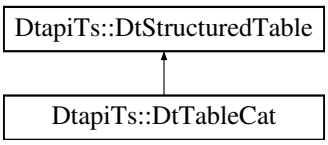
This class holds all descriptors of one sub-loop of a BAT table.

9.64 DtapiTs::DtTableCat Class Reference

Class that can be used to parse all descriptors in the Conditional Access Table.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTableCat:



Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT DecodeFromSection (const DtTableSection *)

  Parse the given TableSection and append all information to the current object.

Public Attributes

- std::vector< DtDescriptor > m_CaDescriptors

  Conditional access descriptors.

9.64.1 Detailed Description

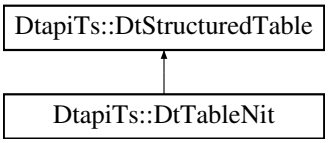
Class that can be used to parse all descriptors in the Conditional Access Table.

9.65 DtapiTs::DtTableNit Class Reference

Structured version of the raw data contained in a Network Information Table.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTableNit:



Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT DecodeFromSection (const DtTableSection *)

  Parse the given TableSection and append all information to the current object.
- DtTableNitInner * FindTsLoop (int TsId, int OrigNetworkId=-1)

  Search through m_TransportStreamLoop and return the first DtTableNitInner with a matching transport stream id and NetworkId.

Public Attributes

- std::vector< DtDescriptor > m_NetworkDescriptors

  All network descriptors found in the first loop of the NIT.
- int m_NetworkId

  Unique identifier for this network.
- std::vector< DtTableNitInner > m_TransportStreamLoop

  Structured information about the inner loop.

9.65.1 Detailed Description

Structured version of the raw data contained in a Network Information Table.

9.65.2 Member Function Documentation

9.65.2.1 DtTableNitInner* DtapiTs::DtTableNit::FindTsLoop (int *TsId*, int *OrigNetworkId* = -1)

Search through m_TransportStreamLoop and return the first DtTableNitInner with a matching transport stream id and NetworkId.

NetworkId can be set to -1 to match every DtTableNitInner that matches the given TsId. If no object matches it'll return NULL.

9.65.3 Member Data Documentation

9.65.3.1 std::vector<DtDescriptor> DtapiTs::DtTableNit::m_NetworkDescriptors

All network descriptors found in the first loop of the NIT.

9.66 DtapiTs::DtTableNitInner Class Reference

This class holds all descriptors of one sub-loop of the NIT table.

`#include <DTAPITS.h>`

Public Member Functions

- **DtTableNitInner** (int TsId, int OrigNetworkId)

Public Attributes

- int m_OriginalNetworkId

  Unique identifier for the original network.
- std::vector< DtDescriptor > m_TransportDescriptors

  All descriptors for this transport stream.
- int m_TransportStreamId

  Unique identifier for one TS within a network.

9.66.1 Detailed Description

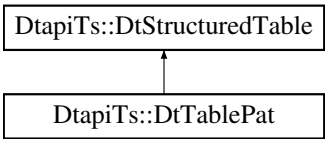
This class holds all descriptors of one sub-loop of the NIT table.

9.67 DtapiTs::DtTablePat Class Reference

Structured version of the raw data contained in a Program Association Table.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTablePat:



Classes

- struct DtProgramMapping

  ServiceId to PMT Pid mapping for a single service.

Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT DecodeFromSection (const DtTableSection *)

  Parse the given TableSection and append all information to the current object.

Public Attributes

- std::vector< DtProgramMapping > m_ProgramMap

  List of all <program number, PID> mappings.
- int m_TsId

  Transport stream identifier.

9.67.1 Detailed Description

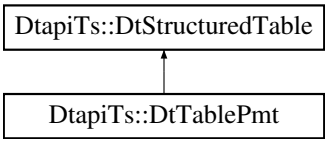
Structured version of the raw data contained in a Program Association Table.

9.68 DtapiTs::DtTablePmt Class Reference

Class that can be used to parse all descriptors in the Program Map Table.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTablePmt:



Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT DecodeFromSection (const DtTableSection *)

  Parse the given TableSection and append all information to the current object.

Public Attributes

- std::vector< DtTablePmtInner > m_Components

  Structured information about all components.
- int **m_PcrPid**
- std::vector< DtDescriptor > m_PmtDescriptors

  Descriptors for the complete stream.

9.68.1 Detailed Description

Class that can be used to parse all descriptors in the Program Map Table.

9.69 DtapiTs::DtTablePmtInner Class Reference

This class holds all descriptors of compoment in a PMT table.

`#include <DTAPITS.h>`

Public Attributes

- std::vector< DtDescriptor > m_CompDescriptors

  All descriptors for this service component.
- int m_Pid

  Pid used to broadcast this elementary stream.
- int m_StreamType

  Elementary stream type.

9.69.1 Detailed Description

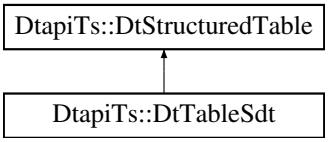
This class holds all descriptors of compoment in a PMT table.

9.70 DtapiTs::DtTableSdt Class Reference

Structured version of the raw data contained in a Service description table.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTableSdt:



Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT DecodeFromSection (const DtTableSection *)

  Parse the given TableSection and append all information to the current object.
- DtTableSdtInner * FindService (int SvcId)

  Search through m_Services and return the first DtTableSdtInner with a matching service id.

Public Attributes

- std::vector< DtTableSdtInner > m_Services

  Structured information about each service.
- int m_TransportStreamId

  Unique identifier for the transport stream described by this SDT.

9.70.1 Detailed Description

Structured version of the raw data contained in a Service description table.

9.70.2 Member Function Documentation

9.70.2.1 DtTableSdtInner* DtapiTs::DtTableSdt::FindService (int *SvcId*)

Search through m_Services and return the first DtTableSdtInner with a matching service id.

If no object matches it'll return NULL.

9.70.3 Member Data Documentation

9.70.3.1 int DtapiTs::DtTableSdt::m_TransportStreamId

Unique identifier for the transport stream described by this SDT.

9.71 DtapiTs::DtTableSdtInner Class Reference

This class holds all descriptors of one sub-loop of the SDT table.

`#include <DTAPITS.h>`

Public Attributes

- bool m_EitPresentFollowing

  If true this indicates that EIT_present_following information for this service is present in the current transport stream.
- bool m_EitSchedule

  If true EIT scheduling information is present in the current transport stream.
- bool m_FreeCaMode

  When false all components are unscrambled, when true one or more components may be scrambled.
- int m_RunningStatus

  Indication of the status of this service.
- std::vector< DtDescriptor > m_ServiceDescriptors

  All descriptors for this transport stream.
- int m_ServiceId

  Service identifier for this program.

9.71.1 Detailed Description

This class holds all descriptors of one sub-loop of the SDT table.

9.71.2 Member Data Documentation

9.71.2.1 bool DtapiTs::DtTableSdtInner::m_EitPresentFollowing

If true this indicates that EIT_present_following information for this service is present in the current transport stream.

9.71.2.2 bool DtapiTs::DtTableSdtInner::m_EitSchedule

If true EIT scheduling information is present in the current transport stream.

9.71.2.3 bool DtapiTs::DtTableSdtInner::m_FreeCaMode

When false all components are unscrambled, when true one or more components may be scrambled.

9.72 DtapiTs::DtTableSection Class Reference

Binary repesentation of one section of an SI table.

`#include <DTAPITS.h>`

Public Member Functions

- DtTableSection ()

  Create a new (empty) DtTableSection object.
- DtTableSection (const DtTableSection &Other)

  Copy an exising DtTableSection object.
- virtual ~DtTableSection ()

  Remove this object.
- DtTableSection & operator= (const DtTableSection &Other)

  Make this object a copy of an other DtTableSection object.

Public Attributes

- uint8_t * m_Buffer

  Data in this section.
- int m_BufSize

  Size of buffer.
- DtTimestamp m_TimeCompleted

  Timestamp the last TS packet that contained part of this section was received.

9.72.1 Detailed Description

Binary repesentation of one section of an SI table.

9.72.2 Constructor & Destructor Documentation

9.72.2.1 DtapiTs::DtTableSection::DtTableSection (const DtTableSection & *Other*)

Copy an exising DtTableSection object.

Parameters

Other,:	the object to copy.

9.72.3 Member Function Documentation

9.72.3.1 DtTableSection& DtapiTs::DtTableSection::operator= (const DtTableSection & *Other*)

Make this object a copy of an other DtTableSection object.

Parameters

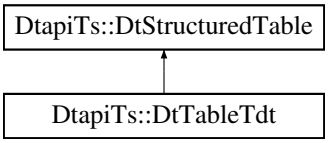
Other,:	the object to copy.

9.73 DtapiTs::DtTableTdt Class Reference

Structured version of the raw data contained in a Time and Date section.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTableTdt:



Public Member Functions

- void Clear ()

  Clear all data contained in this class.
- DTAPITS_RESULT DecodeFromSection (const DtTableSection *)

  Parse the given TableSection and append all information to the current object.

Public Attributes

- __int64 m_UtcTime

  Current time and date in UTC and MJD.

9.73.1 Detailed Description

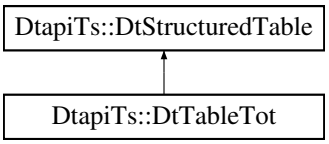
Structured version of the raw data contained in a Time and Date section.

9.74 DtapiTs::DtTableTot Class Reference

Structured version of the raw data contained in a Time Offset Table.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTableTot:



Public Member Functions

- void Clear ()

  Clear all data contained in this class.

- DTAPITS_RESULT DecodeFromSection (const DtTableSection *)

  Parse the given TableSection and append all information to the current object.

Public Attributes

- std::vector< DtDescriptor > m_Descriptors

  All descriptors for time offset table.
- __int64 m_UtcTime

  Current time and date in UTC and MJD.

9.74.1 Detailed Description

Structured version of the raw data contained in a Time Offset Table.

9.75 DtapiTs::DtTimeDiff Class Reference

Difference between two timestamps.

`#include <DTAPITS.h>`

Public Member Functions

- DtTimeDiff (__int64 Diff=0)

  Create a new DtTimeDiff object.
- bool **operator!=** (const DtTimeDiff &Other) const
- DtTimeDiff operator* (int Mul) const

  Multiply this time difference with an integer.
- DtTimeDiff operator* (unsigned int Mul) const

  Multiply this time difference with an unsigned integer.
- DtTimeDiff operator* (__int64 Mul) const

  Multiply this time difference with a 64bit integer.
- DtTimeDiff operator* (double Mul) const

  Multiply this time difference with a double.
- __int64 operator/ (const DtTimeDiff &Other) const

  Divide two DtTimeDiff objects to get the ratio between them.
- DtTimeDiff operator/ (int Div) const

  Get a new time difference object by dividing this one in `Div` equal timeframes.
- DtTimeDiff **operator**/ (__int64 Div) const
- bool operator< (const DtTimeDiff &Other) const

  Compare two DtTimeDiff objects.
- bool **operator==** (const DtTimeDiff &Other) const
- bool **operator**> (const DtTimeDiff &Other) const
- double **ToMSeconds** () const
- double ToSeconds () const

  Convert this DtTimeDiff object to the amount of seconds it represents.

Static Public Member Functions

- static const DtTimeDiff **MSECOND** ()
- static const DtTimeDiff SECOND ()

  Constant that can be used to convert to/from normal time units.

Public Attributes

- __int64 m_Diff

  Internal difference value.

9.75.1 Detailed Description

Difference between two timestamps.

Can be added/subtracted to a DtTimestamp or can be converted to seconds.

9.75.2 Member Function Documentation

9.75.2.1 bool DtapiTs::DtTimeDiff::operator< (const DtTimeDiff & *Other*) const [inline]

Compare two DtTimeDiff objects.

Returns true if `this` time difference is smaller than the object `Other`.

9.76 DtapiTs::DtTimestamp Class Reference

Abstract timestamp, do not rely on the internal representation.

`#include <DTAPITS.h>`

Public Member Functions

- DtTimestamp (__int64 Time=LLONG_MIN)

  Create a new DtTimestamp object that is invalid.
- DtTimestamp (const DtTimestamp &Other)

  Create a copy of an existing DtTimestamp object.
- bool operator!= (const DtTimestamp &Other) const

  Compare two DtTimestamp objects for inequality.
- DtTimestamp operator+ (const DtTimeDiff &Diff) const

  Add a timespan to this object and return the result.
- DtTimestamp & operator+= (const DtTimeDiff &Diff)

  Add a timespan to the current object and return it.
- DtTimeDiff operator- (const DtTimestamp &Time) const

  Compute the difference between two timestamp objects.
- DtTimestamp operator- (const DtTimeDiff &Diff) const

  Subtract a timespan from this object and return the result.
- bool operator< (const DtTimestamp &Other) const

  Compare two DtTimestamp objects to see which one is earlier.
- bool operator== (const DtTimestamp &Other) const

  Compare two DtTimestamp objects for equality.
- bool **operator**> (const DtTimestamp &Other) const

Static Public Member Functions

- static const DtTimestamp INVALID ()

  Helper function that returns an invalid timestamp.

Public Attributes

- __int64 m_Time

  Internal representation of a timestamp.

9.76.1 Detailed Description

Abstract timestamp, do not rely on the internal representation.

Two DtTimestamp objectscan be subtracted to get a DtTimeDiff object which can be converted to seconds.

9.77 DtapiTs::DtTp Class Reference

Public Types

- enum **TpAdaptationField** {
  AF_RESERVED = 0,
  AF_PAYLOAD_ONLY = 1,
  AF_ADAP_ONLY = 2,
  AF_ADAP_PAYLOAD = 3 }
- enum **TpScrambleControl** {
  SC_NOT_SCRAMBLED = 0,
  SC_RESERVED = 1,
  SC_EVEN_CW = 2,
  SC_ODD_CW = 3 }

Public Member Functions

- **DtTp** (uint8_t *pTpBuf=NULL, int TpSize=0)
- **DtTp** (const DtTp &)
- uint8_t * **AdaptationField** (int &Size) const
- TpAdaptationField **AdaptationFieldControl** () const
- void **AdaptationFieldControl** (TpAdaptationField)
- int **ContinuityCounter** () const
- void **ContinuityCounter** (int)
- void **DeepCopy** (const DtTp &)
- bool **DiscontinuityIndicator** () const
- bool **HasAdaptationField** () const
- bool **HasPayload** () const
- bool **HoldsPcr** () const
- bool **IsScrambled** () const
- bool **IsSyncValid** () const
- bool **IsValid** () const
- **operator const uint8_t** * ()
- DtTp & **operator=** (const DtTp &)
- uint8_t * **Payload** (int &Size) const
- bool **PayloadUnitStartIndicator** () const
- void **PayloadUnitStartIndicator** (bool)
- DtPcr **Pcr** () const
- void **Pcr** (DtPcr &)
- int **Pid** () const
- void **Pid** (int)
- void **ShallowCopy** (const DtTp &)

- bool **TransportErrorIndicator** () const
- void **TransportErrorIndicator** (bool)
- TpScrambleControl **TransportScrambleControl** () const
- void **TransportScrambleControl** (TpScrambleControl)

Static Public Attributes

- static const int **TP_PID_MAX** = 8191
- static const int **TP_PID_MIN** = 0
- static const int **TP_SIZE_MIN** = 188
- static const int **TP_SYNC_BYTE** = 0x47

9.78 DtapiTs::DtTr101290 Class Reference

Base class for TR 101 290 support.

`#include <DTAPITS.h>`

Public Member Functions

- void **ResetAll** (bool ResetCount=false)

Public Attributes

- DtTr101290Error **m_Indicators** [DT_ERR_MAX]

9.78.1 Detailed Description

Base class for TR 101 290 support.

Contains an array with all indicators that can be used to check the error counts and whether an indicator is currently on or off.

9.79 DtapiTs::DtTr101290Error Class Reference

Information about a single TR 101 290 indicator.

`#include <DTAPITS.h>`

Public Member Functions

- void **Reset** (bool ResetCount=false)

Public Attributes

- int m_ErrCount

  Number of times this error has occured.
- bool m_IsSet

  Is this indicator currently set? m_IsSet will be automatically reset by DTAPI-TS after a certain timeout expires.
- bool m_Latched

Has this indicator been set since the last reset? This var is not reset by DTAPI-TS.

- std::wstring m_Msg

  Human-readable error message.

- DtSubTableId m_TableId

  Table for which this error occured last.

- double m_Time

  Duration for this error.

- DtTimestamp m_Timestamp

  Timestamp this error occured last.

9.79.1 Detailed Description

Information about a single TR 101 290 indicator.

9.79.2 Member Data Documentation

9.79.2.1 int DtapiTs::DtTr101290Error::m_ErrCount

Number of times this error has occured.

Will increase even when m_IsSet is already true.

9.79.2.2 bool DtapiTs::DtTr101290Error::m_IsSet

Is this indicator currently set? m_IsSet will be automatically reset by DTAPI-TS after a certain timeout expires.

9.79.2.3 bool DtapiTs::DtTr101290Error::m_Latched

Has this indicator been set since the last reset? This var is not reset by DTAPI-TS.

9.79.2.4 double DtapiTs::DtTr101290Error::m_Time

Duration for this error.

Exact meaning depends on the indicator.

9.80 DtapiTs::DtTsData Class Reference

Main DtapiTs class that contains all data extracted from a transport stream.

`#include <DTAPITS.h>`

Public Types

- typedef std::map< int, DtServiceInfo > DtSvcInfoMap

  Type used to map program numbers to their service information.

- typedef std::map< DtSubTableId, DtTable * > DtTableMap

  Type used to map a unique table identifier to the contents of that table.

Public Member Functions

- DtTsData ()

  Create a new empty DtTsData object.
- DtTsData (const DtTsData &)

  Copy an existing DtTsData object.
- ~DtTsData ()

  Clean up a DtTsData object.
- __int64 GetNitFrequency ()

  Get the frequency as defined in any of the above delivery descriptors, otherwise -1.
- DtTsData & operator= (const DtTsData &)

  Overwrite the current object with values from another DtTsData object.

Public Attributes

- union {
  DtDvbCNitInfo m_Cable
  Info from DVB cable delivery system descriptor.
  DtDvbSNitInfo m_Satellite
  Info from DVB satellite delivery system descriptor.
  DtDvbTNitInfo m_Terrestrial
  Info from DVB terrestial delivery system descriptor.
  };
- DtCaSystemList m_CaSystems

  List of conditional access systems that are valid for all components of all services in this stream See also DtServiceInfo::m_CaSystems and DtServiceComponentInfo::m_CaSystems.
- DtDeliverySystem m_DeliverySystem

  Indicates which of the following nit info objects contains valid data.
- int m_ErrIndErrors

  Number of packets dropped because the error indicator bit was set.
- bool m_InSync

  Whether we're currently in sync.
- std::wstring m_NetworkName

  Name of the network this TS was send on.
- int m_NitTsRate

  Transport stream rate as indicated by the NIT if available, otherwise -1.
- int m_PacketSize

  Size of each packet.
- DtPidInfo * m_PidInfo [DT_NUM_PIDS]

  Information about all PIDs in this stream.
- DtSvcInfoMap m_ServiceInfo

  Service information.
- DtDvbShNitInfo * m_SH

  Info from DVB SH delivery system descriptor.
- DtStandardMode m_StandardMode

  Standard mode used to parse the tables.
- int m_SyncByteErrors

  Number of packets dropped because sync byte was incorrect.
- int m_SyncLossCounter

  Number of times we lost synchronization.
- DtDvbT2NitInfo * m_T2

Info from DVB T2 delivery system descriptor.

- DtTableMap m_Tables

  All completed tables.

- bool m_TmccDataValid

  True if the ISDB-T information parsed from the TMCC data is valid.

- int m_TransportStreamId

  Transport stream identifier.

- int m_TsRate

  The rate of the incoming data normalized to 188-byte packets (in bits/s)

9.80.1 Detailed Description

Main DtapiTs class that contains all data extracted from a transport stream.

Objects of this class can be easily copied

Examples:

example3.cpp.

9.80.2 Member Function Documentation

9.80.2.1 __int64 DtapiTs::DtTsData::GetNitFrequency ( )

Get the frequency as defined in any of the above delivery descriptors, otherwise -1.

9.80.3 Member Data Documentation

9.80.3.1 DtCaSystemList DtapiTs::DtTsData::m_CaSystems

List of conditional access systems that are valid for all components of all services in this stream See also DtServiceInfo::m_CaSystems and DtServiceComponentInfo::m_CaSystems.

9.80.3.2 DtDeliverySystem DtapiTs::DtTsData::m_DeliverySystem

Indicates which of the following nit info objects contains valid data.

9.80.3.3 int DtapiTs::DtTsData::m_ErrIndErrors

Number of packets dropped because the error indicator bit was set.

This value will only be increased when in sync.

9.80.3.4 int DtapiTs::DtTsData::m_NitTsRate

Transport stream rate as indicated by the NIT if available, otherwise -1.

9.80.3.5 int DtapiTs::DtTsData::m_PacketSize

Size of each packet.

This is only valid while m_InSync is true.

9.80.3.6 int DtapiTs::DtTsData::m_SyncByteErrors

Number of packets dropped because sync byte was incorrect.

Only increased when we're in sync.

9.80.3.7 bool DtapiTs::DtTsData::m_TmccDataValid

True if the ISDB-T information parsed from the TMCC data is valid.

9.81 DtapiTs::DtTsInfo Class Reference

Main DtapiTs class that is used for setting parameters, adding callbacks, passing in the transport stream buffer (via a DtTsInfoInput object) and finally reading back the results.

`#include <DTAPITS.h>`

Public Types

- typedef DtCallback2< int, const DtPcrJitter & > DtJitterCallback

  Type of a jitter callback function.
- typedef DtCallback3< const uint8_t *, int, DtTimestamp > DtPacketCallback

  Type of a packet callback function.
- typedef DtCallback3< int, const uint8_t *, int > DtPesCallback

  Type of a PES callback function.
- typedef DtCallback3< int, uint8_t, const DtTableSection * > DtSectionCallback

  Type of a section callback function.
- typedef DtCallback3 < DtSubTableId, const DtTable *, bool > DtTableCallback

  Type of a table callback function.
- typedef DtCallback2 < DtSubTableId, DtTimestamp > DtTableTimeoutCallback

  Type of a table timeout callback function.
- typedef DtCallback2 < DtTr101290Indicator, const DtTr101290Error & > **DtTr101290Callback**

Public Member Functions

- virtual void AddJitterCallback (DtJitterCallback Callback)=0

  Register a new callback function to be called when more jitter values are ready.
- virtual void AddJitterCallback (int Pid, DtJitterCallback Callback)=0

  Register a new callback function to be called when more jitter values are ready for PCRs on a specific pid.
- virtual void AddNewSectionCallback (DtSectionCallback Callback)=0

  Register a new callback function to be called when a new section arrives.
- virtual void AddNewSectionCallback (int Pid, uint8_t TableId, DtSectionCallback Callback)=0

  Register a new callback function to be called when a new section of a given table on a specific PID arrives.

- virtual void AddPacketCallback (DtPacketCallback Callback)=0

  Register a new callback function for any valid packet.
- virtual void AddPacketCallback (int Pid, DtPacketCallback Callback)=0

  Register a new callback function for valid packets on the given PID.
- virtual void AddPesPacketCallback (DtPesCallback Callback)=0

  Register a new callback function to be called when a PES packet is complete.
- virtual void AddPesPacketCallback (int Pid, DtPesCallback Callback)=0

  Register a new callback function to be called when a PES packet on the specified PID is complete.
- virtual void AddTableChangedCallback (DtTableCallback Callback)=0

  Register a new callback function to be called when any table is updated.
- virtual void AddTableChangedCallback (DtSubTableId Key, DtTableCallback Callback)=0

  Register a new callback function to be called when the table with the given TableId is updated.
- virtual void AddTableTimeoutCallback (DtSubTableId Key, DtTableTimeoutCallback Callback)=0

  Register a new callback function to be called when the table with the given TableId is not seen for a certain time.
- virtual void **AddTr101290ErrorCallback** (DtTr101290Bitmask, DtTr101290Callback Callback)=0
- virtual void **AddTr101290IndicatorCallback** (DtTr101290Bitmask, DtTr101290Callback Callback)=0
- virtual int GetFirstServiceNum (int Pid)=0

  Get the first service that references this Pid or -1.
- virtual DTAPI_RESULT GetIsdbtPars (DtIsdbtPars &Pars)=0

  Retrieve the ISDB-T parameters as they hvae been found in the TMCC data in the stream.
- virtual void Lock ()=0

  Acquire a lock that makes it safe to access all public data members.
- virtual void NewPacket (uint8_t *Buf, int BufLen, DtTimestamp Timestamp)=0

  Called when a new packet is available.
- virtual void NewTimestamp (DtTimestamp Timestamp)=0

  Called periodically to handle timeouts if case no packets are available.
- virtual void Reset ()

  Reset all data except for the TR 101 290 error status and count.
- virtual void **SetInSync** (bool Sync)=0
- virtual void SetJitterWindow (DtTimeDiff TimeDiff, int Multiplier)=0

  Change the window used for jitter calculations, default = 1s, 10.
- virtual void SetPidBitrateWindow (DtBitrateSettings Settings)=0

  Change the sliding window parameters used to compute the bitrate per PID.
- virtual void SetStandardMode (DtStandardMode NewMode)=0

  Change the standard mode.
- virtual void Unlock ()=0

  Release the lock that makes it safe to access all public data members.

Public Attributes

- bool m_CompletePes [DT_NUM_PIDS]

  Set to true if you want complete PES packets to be cached (and given to the PES callback).
- DtTsData m_Data

  The data extracted from the transport stream.
- DtTr101290 m_DvbErrs

  TR 101 290 error indicators.
- std::vector< std::string > m_PreferredLanguages

  List of languages that should be preferred.
- DtTimeDiff m_TableTimeoutCb [DT_NUM_TABLES]

  Duration before a table times out and the corresponding callback should be called.
- DtTimeDiff m_TableTimeoutEraseData [DT_NUM_TABLES]

  Duration before data from a table becomes so outdated it will be removed.
- bool m_UseTableCache [DT_NUM_TABLES]

  Used to enable/disable the table cache for certain tables.

9.81.1 Detailed Description

Main DtapiTs class that is used for setting parameters, adding callbacks, passing in the transport stream buffer (via a DtTsInfoInput object) and finally reading back the results.

All function calls are thread-safe. Every function automatically acquires/releases an internal lock when the functions starts/exits. You only have to worry about thread-safety when reading or setting any of the member variables if you use another thread to provide the data via a DtTsInfoInput object. For those cases, call Lock() before you access any member variable and Unlock() when you're done. You should try to make sure you don't take the lock for too long. One way to make sure of this is to call Lock(), copy m_Data and call Unlock() directly after. Now you're free to read all data from the copy while another thread can continue processing new packets.

Examples:

example1.cpp, and example3.cpp.

9.81.2 Member Typedef Documentation

9.81.2.1 typedef DtCallback2<int, const DtPcrJitter&> DtapiTs::DtTsInfo::DtJitterCallback

Type of a jitter callback function.

A jitter callback function is called as soon as a new vector with jitter offsets is ready.

9.81.2.2 typedef DtCallback3<const uint8_t*, int, DtTimestamp> DtapiTs::DtTsInfo::DtPacketCallback

Type of a packet callback function.

Packet callbacks are for every valid transport stream packet. A pid filter is optional. Parameters to the function are:

- Buf: Pointer to a buffer that contains the transport packet.
- BufLen: Size of the buffer with the transport packet. At least 188 bytes.
- Timestamp: Exact time the packet was completely received.

9.81.2.3 typedef DtCallback3<int, const uint8_t*, int> DtapiTs::DtTsInfo::DtPesCallback

Type of a PES callback function.

Pes callback functions are called whenever a PES packet is complete. A PES packet is complete when one of the following is true:

- A new PES packet starts in a transport stream packet with the same Pid as indicated by the unit start indicator.
- `m_CompletePes`[Pid]==false and at least 8192 bytes are read.
- `m_CompletePes`[Pid]==true and 8Mb is read. This situation should never occur for a valid transport stream, the PES packet should have been complete and a new one started by this time. The PES callback function gets the following parameters:
- Pid: Pid of this PES stream.
- Buf: Pointer to a buffer that contains the PES packet.
- BufLen: Size of the buffer with the PES packet.

9.81.2.4 typedef DtCallback3<int, uint8_t, const DtTableSection*> DtapiTs::DtTsInfo::DtSectionCallback

Type of a section callback function.

Section callbacks are called whenever a section of a table is completely received. At this point no checks have been made to see if a cached version is exactly the same, so you will receive duplicates. Parameters to the function are:

- Pid: The PID on which this section was received.
- TableId: Table identifier of this section.
- Section: Pointer to an object containing the actual data in the table section.

9.81.2.5 typedef DtCallback3<DtSubTableId, const DtTable*, bool> DtapiTs::DtTsInfo::DtTableCallback

Type of a table callback function.

Table callbacks are called whenever all sections of a table have been received. The `Changed` parameter indicates whether any part of the table has been changed. This makes it very easy to only filter out updates (ignore any call where `Changed==false`) or to look for repetition rates (don't use the `Change` parameter). The following parameters are passed to the callback function:

- TableId: Unique table identifier for the table.

  See Also

  DtSubTableId.

- Table: Object that links to all sections in this table which in turn contain the actual data.
- Changed: True if any data byte has been changed, false otherwise.

If you don't fill out DtSubTableId completely the fields set to -1 will be taken as "don't care". This means that you can set DtSubTableId::m_TableId to 2 to register a callback for all PMTs.

9.81.2.6 typedef DtCallback2<DtSubTableId, DtTimestamp> DtapiTs::DtTsInfo::DtTableTimeoutCallback

Type of a table timeout callback function.

Table timeout callbacks are called when the specified table has not been seen for a certain period (adjustable in m_TableTimeoutCb). The following parameters are passed to the callback function:

- TableId: Unique table identifier for the table.

  See Also

  DtSubTableId.

- Timestamp: Timestamp the timeout has been noticed.

9.81.3 Member Function Documentation

9.81.3.1 virtual void DtapiTs::DtTsInfo::AddJitterCallback (int *Pid,* DtJitterCallback *Callback*) [pure virtual]

Register a new callback function to be called when more jitter values are ready for PCRs on a specific pid.

9.81.3.2 virtual void DtapiTs::DtTsInfo::AddNewSectionCallback (int *Pid,* uint8_t *TableId,* DtSectionCallback *Callback*) [pure virtual]

Register a new callback function to be called when a new section of a given table on a specific PID arrives.

9.81.3.3 virtual void DtapiTs::DtTsInfo::AddPesPacketCallback (DtPesCallback *Callback*) [pure virtual]

Register a new callback function to be called when a PES packet is complete.

If m_CompletePes[Pid] is set to false (the default), the callback will be called with a maximum of 8192 bytes of data, if it's set to true the callback will be called only with complete PES packets.

9.81.3.4 virtual void DtapiTs::DtTsInfo::AddPesPacketCallback (int *Pid,* DtPesCallback *Callback*) [pure virtual]

Register a new callback function to be called when a PES packet on the specified PID is complete.

If m_CompletePes[Pid] is set to false (the default), the callback will be called with a maximum of 8192 bytes of data, if it's set to true the callback will be called only with complete PES packets.

9.81.3.5 virtual void DtapiTs::DtTsInfo::AddTableChangedCallback (DtSubTableId *Key,* DtTableCallback *Callback*) [pure virtual]

Register a new callback function to be called when the table with the given TableId is updated.

9.81.3.6 virtual void DtapiTs::DtTsInfo::AddTableTimeoutCallback (DtSubTableId *Key,* DtTableTimeoutCallback *Callback*) [pure virtual]

Register a new callback function to be called when the table with the given TableId is not seen for a certain time.

9.81.3.7 virtual DTAPI_RESULT DtapiTs::DtTsInfo::GetIsdbtPars (DtIsdbtPars & *Pars*) [pure virtual]

Retrieve the ISDB-T parameters as they hvae been found in the TMCC data in the stream.

Parameters

out	*Pars*	ISDB-T parameters as found in the transport stream.

Returns

DTAPI_OK if there are valid parameters, DTAPI_E_∗ otherwise.

9.81.3.8 virtual void DtapiTs::DtTsInfo::Lock ( ) [pure virtual]

Acquire a lock that makes it safe to access all public data members.

To access any data member while data is processed from another thread you'll have to acquire the lock first. All DtTsInfoInput classes will acquire the lock before calling NewPacket/NewTimestamp.

9.81.3.9 virtual void DtapiTs::DtTsInfo::NewPacket (uint8_t ∗ *Buf,* int *BufLen,* DtTimestamp *Timestamp*) [pure virtual]

Called when a new packet is available.

You should not call this function directly but instead use one of the DtTsInfoInput subclasses to handle it for you.

9.81.3.10 virtual void DtapiTs::DtTsInfo::NewTimestamp (DtTimestamp *Timestamp*) [pure virtual]

Called periodically to handle timeouts if case no packets are available.

You should not call this function directly but instead use one of the DtTsInfoInput subclasses to handle it for you.

9.81.3.11 virtual void DtapiTs::DtTsInfo::Reset ( ) [virtual]

Reset all data except for the TR 101 290 error status and count.

If you want to reset those also call m_DvbErrs.ResetAll().

9.81.3.12 virtual void DtapiTs::DtTsInfo::SetJitterWindow (DtTimeDiff *TimeDiff,* int *Multiplier*) [pure virtual]

Change the window used for jitter calculations, default = 1s, 10.

The first value is the update frequency (how often the jitter values in DtPidInfo are updated and how often any jitter callback is called). The second window is a multiplier for the Timediff increasing the window used for the linear regression test.

9.81.3.13 virtual void DtapiTs::DtTsInfo::SetStandardMode (DtStandardMode *NewMode*) [pure virtual]

Change the standard mode.

Note

Since the way to parse the service information completely changes, this function will internally reset the state and also the TR 101 290 error status/count if the mode actually changes. It's recommended to set the mode only once and before starting the data input.

Examples:

example1.cpp, and example3.cpp.

9.81.3.14 virtual void DtapiTs::DtTsInfo::Unlock ( ) [pure virtual]

Release the lock that makes it safe to access all public data members.

Call this function after you're doing setting any parameters below or reading m_Data.

9.81.4 Member Data Documentation

9.81.4.1 bool DtapiTs::DtTsInfo::m_CompletePes[DT_NUM_PIDS]

Set to true if you want complete PES packets to be cached (and given to the PES callback).

The default is to cache only the first 8kb from each PES packet to save memory.

9.81.4.2 DtTsData DtapiTs::DtTsInfo::m_Data

The data extracted from the transport stream.

Examples:

example3.cpp.

9.81.4.3 std::vector<std::string> DtapiTs::DtTsInfo::m_PreferredLanguages

List of languages that should be preferred.

This variable is used when the transport stream offers one string in multiple languages (for example the network name or a service component description). The earlier in this list the higher the preference given. Members should

be strings of exactly 3 characters long. See ISO 639-2 for valid language codes. List of preferred languages in order.

9.81.4.4 DtTimeDiff DtapiTs::DtTsInfo::m_TableTimeoutCb[DT_NUM_TABLES]

Duration before a table times out and the corresponding callback should be called.

The parsed and raw data of this table will still be available until the timeout specified in m_TableTimeoutEraseData also runs out.

9.81.4.5 bool DtapiTs::DtTsInfo::m_UseTableCache[DT_NUM_TABLES]

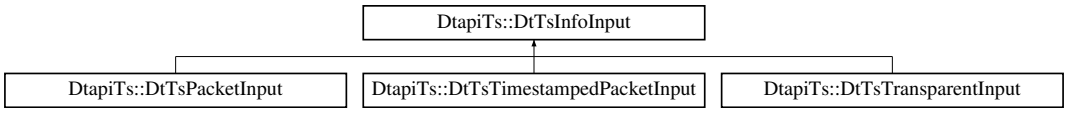
Used to enable/disable the table cache for certain tables.

9.82 DtapiTs::DtTsInfoInput Class Reference

Abstract base class for splitting (large) input buffers into timestamped packets.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTsInfoInput:



Public Member Functions

- virtual void NewData (uint8_t *Buf, int BufLen)=0

  Splits the input buffer in packets, computes/reads a timestamp for each packet and calls m_Info->NewPacket() for each valid packet.
- void SetTsInfoObject (DtTsInfo *Info)

  Set the DtTsInfo object that has to be called from now on when a new data is passed to NewData.

Static Public Member Functions

- static void RemoveInstance (DtTsInfoInput *Instance)

  Remvove an instance of a subclass of DtTsInfoInput.

Protected Attributes

- DtTsInfo * m_Info

  DtTsInfo object that all packets are forwarded to.

9.82.1 Detailed Description

Abstract base class for splitting (large) input buffers into timestamped packets.

Don't instantiate this class directly but use one of the provided sub-classes.

Examples:

example1.cpp.

9.82.2 Member Function Documentation

9.82.2.1 virtual void DtapiTs::DtTsInfoInput::NewData (uint8_t * *Buf,* int *BufLen*) [pure virtual]

Splits the input buffer in packets, computes/reads a timestamp for each packet and calls m_Info->NewPacket() for each valid packet.

Precondition

: m_Info != NULL

Examples:

example1.cpp, and example3.cpp.

9.82.2.2 void DtapiTs::DtTsInfoInput::SetTsInfoObject (DtTsInfo * *Info*)

Set the DtTsInfo object that has to be called from now on when a new data is passed to NewData.

9.83 DtapiTs::DtTsLib Class Reference

Main DtapiTs class used to create instances of the analysis classes.

`#include <DTAPITS.h>`

Public Member Functions

- DtTsLib (const char *CustId="!!CUST_ID!!", const char *LicStr="!!LIC_STR!!")

  DtTsLib constructor. The arguments have default values, do not change them!
- DtTsInfo * CreateDtTsInfoInstance ()

  Create a new DtTsInfo instance.

Static Public Member Functions

- static void GetDtapiTsVersion (int &Major, int &Minor, int &BugFix, int &Build)

  Returns the DtapiTs version of the build object code.
- static void GetDtapiVersion (int &Major, int &Minor, int &BugFix, int &Build)

  Returns the Dtapi version of the build object code.
- static bool IsDtapiTsVersionOk ()

  Returns true if the DtapiTs header file belongs to the build DtapiTs object code.
- static void RemoveDtTsInfoInstance (DtTsInfo *Instance)

  Cleanup a DtTsInfo instance.

9.83.1 Detailed Description

Main DtapiTs class used to create instances of the analysis classes.

Examples:

example1.cpp, and example3.cpp.

9.83.2 Member Function Documentation

9.83.2.1 DtTsInfo* DtapiTs::DtTsLib::CreateDtTsInfoInstance ( ) [inline]

Create a new DtTsInfo instance.

You don't have to keep the DtTsLib object in scope once you've created a DtTsInfo instance. This funtion returns NULL if the DtapiTs header file does not belong to the object file.

Examples:

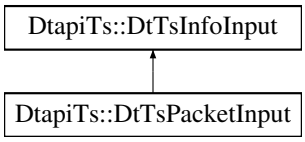
example3.cpp.

9.84 DtapiTs::DtTsPacketInput Class Reference

Input class that handles a transport stream without any timestamps.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTsPacketInput:



Public Member Functions

- virtual void SetBitRate (int Bitrate)=0
  Change the bitrate for this object.

Static Public Member Functions

- static DtTsPacketInput * CreateInstance (DtTsInfo *Info, int Bitrate)
  Create a new DtTsPacketInput object.

Additional Inherited Members

9.84.1 Detailed Description

Input class that handles a transport stream without any timestamps.

Timestamps for each packet are generated based on the given bit-rate.

Examples:

example3.cpp.

9.84.2 Member Function Documentation

9.84.2.1 static DtTsPacketInput* DtapiTs::DtTsPacketInput::CreateInstance (DtTsInfo * *Info,* int *Bitrate*) [static]

Create a new DtTsPacketInput object.

Parameters

Info,:	The DtTsInfo object that will be used for the actual analysis. May be NULL, but in that case you have to provide a valid object by calling SetTsInfoObject before using NewData.
Bitrate,:	The bitrate in bits/s based on a 188-byte packet size. If you have the bitrate of a file with 204-byte packets, multiply that value by (188/204).

Examples:

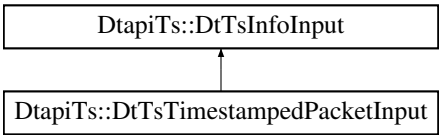
example3.cpp.

9.85 DtapiTs::DtTsTimestampedPacketInput Class Reference

Input class that handles a transport stream packets with timestamps as delivered by DTAPI when a DtInpChannel is set to DTAPI_RXMODE_TIMESTAMP32.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTsTimestampedPacketInput:



Static Public Member Functions

- static DtTsTimestampedPacketInput * CreateInstance (DtTsInfo *Info, int ClckFreq)

  Create a new DtTsTimestampedPacketInput object.

Additional Inherited Members

9.85.1 Detailed Description

Input class that handles a transport stream packets with timestamps as delivered by DTAPI when a DtInpChannel is set to DTAPI_RXMODE_TIMESTAMP32.

A 32-bit timestamp is added in front of the transport stream packet.

9.85.2 Member Function Documentation

9.85.2.1 static DtTsTimestampedPacketInput* DtapiTs::DtTsTimestampedPacketInput::CreateInstance (DtTsInfo * *Info,* int *ClckFreq*) [static]

Create a new DtTsTimestampedPacketInput object.

Parameters

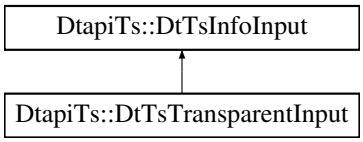
Info,:	The DtTsInfo object that will be used for the actual analysis. May be NULL, but in that case you have to provide a valid object by calling SetTsInfoObject before using NewData.
ClckFreq,:	The hardware clock frequency. Use DtDevice::GetRefClkFreq to determine this.

9.86 DtapiTs::DtTsTransparentInput Class Reference

Input class that handles timestamped transparent packets as delivered by DTAPI when a DtInpChannel is set to DTAPI_RXMODE_STTRP|DTAPI_RXMODE_TIMESTAMP32.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtTsTransparentInput:



Static Public Member Functions

- static DtTsTransparentInput * CreateInstance (DtTsInfo *Info, int ClckFreq)

  Create a new DtTsTransparentInput object.

Additional Inherited Members

9.86.1 Detailed Description

Input class that handles timestamped transparent packets as delivered by DTAPI when a DtInpChannel is set to DTAPI_RXMODE_STTRP|DTAPI_RXMODE_TIMESTAMP32.

9.86.2 Member Function Documentation

9.86.2.1 static DtTsTransparentInput* DtapiTs::DtTsTransparentInput::CreateInstance (DtTsInfo * *Info,* int *ClckFreq*) [static]

Create a new DtTsTransparentInput object.

Parameters

Info,:	The DtTsInfo object that will be used for the actual analysis. May be NULL, but in that case you have to provide a valid object by calling SetTsInfoObject before using NewData.
ClckFreq,:	The hardware clock frequency. Use DtDevice::GetRefClkFreq to determine this.

Examples:

example1.cpp.

9.87 DtapiTs::DtVideoAspectRatio Class Reference

Public Member Functions

- **DtVideoAspectRatio** (int x=1, int y=1)
- **DtVideoAspectRatio** (const DtVideoAspectRatio &Oth)
- bool **operator!=** (const DtVideoAspectRatio &Oth) const
- DtVideoAspectRatio **operator*** (const DtVideoAspectRatio &Oth) const
- DtVideoAspectRatio & **operator*=** (const DtVideoAspectRatio &Oth)
- DtVideoAspectRatio & operator= (const DtVideoAspectRatio &Oth)

Vertical.

- bool **operator==** (const DtVideoAspectRatio &Oth) const

Public Attributes

- int **m_X**
- int m_Y

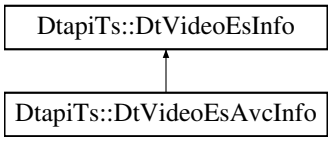
  Horizontal.

Protected Member Functions

- void **Normalize** ()

9.88 DtapiTs::DtVideoEsAvcInfo Class Reference

Inheritance diagram for DtapiTs::DtVideoEsAvcInfo:



Public Attributes

- struct {
  int **m_CpbRemovalDelayLength**
  int **m_DpbOutputDelayLength**
  } **HrdParameters**

- int **m_SeqParameterSetId**
- struct {
  bool **m_NalHrdParametersPresentFlag**
  bool **m_PicStructPresentFlag**
  bool **m_VclHrdParametersPresentFlag**
  } **VuiParameters**

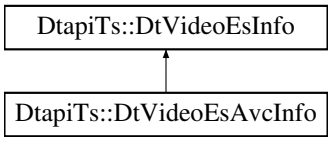
Additional Inherited Members

9.89 DtapiTs::DtVideoEsInfo Class Reference

Information about a video elementary stream extracted from PES packets.

`#include <DTAPITS.h>`

Inheritance diagram for DtapiTs::DtVideoEsInfo:



Public Member Functions

- DtVideoEsInfo ()

  Create a new DtVideoEsInfo object.

Public Attributes

- double m_AspectRatioHorz

  Horizontal ratio.
- double m_AspectRatioVert

  Vertical ratio.
- DtAtscCcType m_AtscCcType

  ATSC closed-caption type (EIA-608 or EIA-708)
- int m_BitDepthChroma

  number of bits for chroma samples
- int m_BitDepthLuma

  number of bits for luma samples
- DtVideoChromaFormat m_ChromaFormat

  Video chrominance format.
- DtVideoAspectRatio m_Dar

  Display aspect ratio.
- double m_FrameRate

  Video frame rate.
- int m_HorzSize

  Horizontal video size.
- int m_HorzSizeDisplay

  Horizontal video display size.
- bool m_IsInterlaced

  Video is coded with interlaced syntax.
- unsigned int m_Mask

  Indicates which fields are valid.
- DtVideoAspectRatio m_Par

  Pixel aspect ratio.
- DtVideoAspectRatio m_Sar

  Storage aspect ratio.
- int m_VertSize

  Vertical video size.
- int m_VertSizeDisplay

  Vertical video display size.

Static Public Attributes

- static const unsigned int ASPECT_RATIO_FIELD = 0x00000008

  Bit set in m_Mask when m_AspectRatioHorz and m_AspectRatioVert are valid.
- static const unsigned int ATSC_CC_TYPE_FIELD = 0x00000100

  Bit set in m_Mask when m_AtscCcType is valid.
- static const unsigned int BIT_DEPTH_FIELD = 0x00000400

  Bit set in m_Mask when m_BitDepthLuma and m_BitDepthChroma are valid.
- static const unsigned int **c_AspectRatioField** = ASPECT_RATIO_FIELD
- static const unsigned int **c_AtscCcTypeField** = ATSC_CC_TYPE_FIELD
- static const unsigned int **c_ChromaFormatField** = CHROMA_FORMAT_FIELD
- static const unsigned int **c_DisplaySizeField** = DISPLAY_SIZE_FIELD
- static const unsigned int **c_ExtSizeField** = EXT_SIZE_FIELD
- static const unsigned int **c_FrameRateField** = FRAME_RATE_FIELD
- static const unsigned int **c_InterlacedField** = INTERLACED_FIELD
- static const unsigned int **c_SizeField** = SIZE_FIELD
- static const unsigned int CHROMA_FORMAT_FIELD = 0x00000004

  Bit set in m_Mask when m_ChromaFormat is valid.
- static const unsigned int DAR_FIELD = 0x00000040

  Bit set in m_Mask when m_Dar is valid.
- static const unsigned int DISPLAY_SIZE_FIELD = 0x00000002

  Bit set in m_Mask when m_HorzSizeDisplay and m_VertSizeDisplay are valid.
- static const unsigned int EXT_SIZE_FIELD = 0x80000000

  Internal value. Do not use.
- static const unsigned int FRAME_RATE_FIELD = 0x00000080

  Bit set in m_Mask when m_FrameRate is valid.
- static const unsigned int INTERLACED_FIELD = 0x00000200

  Bit set in m_Mask when m_IsInterlaced is valid.
- static const unsigned int PAR_FIELD = 0x00000010

  Bit set in m_Mask when m_Par is valid.
- static const unsigned int SAR_FIELD = 0x00000020

  Bit set in m_Mask when m_Sar is valid.
- static const unsigned int SIZE_FIELD = 0x00000001

  Bit set in m_Mask when m_HorzSize and m_VertSize are valid.

9.89.1 Detailed Description

Information about a video elementary stream extracted from PES packets.

9.90 DtapiTs::DtDescDvbLinkage::EventLinkage Struct Reference

Public Attributes

- bool m_EventSimulcast

  True if both events are being simulcast.
- int m_TargetEventId

  Identifies the target event.
- bool m_TargetListed

  True if the target service is listed in the SDT.

9.91 DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage Struct Reference

Public Attributes

- bool m_EventSimulcast

  True if both events are being simulcast.
- int m_LinkType

  The type of the target service: 0 for SD, 1 for HD and 2 for 3D.
- int m_TargetEventId

  Identifies the target event.
- int m_TargetIdType

  Identifies which way to use to match the target service.
- bool m_TargetListed

  True if the target service is listed in the SDT.
- int m_TargetOrigNetworkId

  The network id of the alternate stream or -1 if not present.
- int m_TargetServiceId

  The service id of the alternate information service or -1 if not present.
- int m_TargetTsId

  The target transportstream id or -1 if not present.
- int m_UserDefinedId

  A user-defined ID or -1 if not present.

9.91.1 Member Data Documentation

9.91.1.1 int DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage::m_LinkType

The type of the target service: 0 for SD, 1 for HD and 2 for 3D.

9.91.1.2 int DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage::m_TargetIdType

Identifies which way to use to match the target service.

9.91.1.3 int DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage::m_TargetOrigNetworkId

The network id of the alternate stream or -1 if not present.

9.91.1.4 int DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage::m_TargetServiceId

The service id of the alternate information service or -1 if not present.

9.91.1.5 int DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage::m_TargetTsId

The target transportstream id or -1 if not present.

9.92 DtapiTs::DtEsInfoBase::InfoField< T > Class Template Reference

Public Member Functions

- **InfoField** (T &Value)
- void **Clear** ()
- bool **IsValid** () const
- **operator T** () const
- InfoField< T > & **operator=** (const T &Value)

9.93 DtapiTs::DtDescMpegLanguage::LangCode Struct Reference

Public Attributes

- int m_AudioType

  The type of audio stream.
- std::string m_LangCode

  3 character ISO 639 language code.

9.94 DtapiTs::DtDescDvbLocalTimeOffset::LocalTimeOffset Struct Reference

Public Attributes

- std::string m_CountryCode

  3-character country code as specified in ISO 3166
- int m_CountryRegionId

  Zone identifier within the country.
- int m_LocalTimeOffset

  Offset from UTC in minutes.
- int m_NextTimeOffset

  The new time offset from UTC in minutes.
- __int64 m_TimeOfChange

  The date and tiem in MJD and UTC when the time change takes place.

9.94.1 Member Data Documentation

9.94.1.1 __int64 DtapiTs::DtDescDvbLocalTimeOffset::LocalTimeOffset::m_TimeOfChange

The date and tiem in MJD and UTC when the time change takes place.

9.95 DtapiTs::DtDescDvbLinkage::MobileHandOverInfo Struct Reference

Public Attributes

- int m_InitialServiceId

  Service identifier for which the hand-over is valid.
- int m_NetworkId

  The network id of the terrestrial network that supports the indicated service.
- int m_OrigType

Flag that specified the table in which the link is originated.

- int m_Type

  The type of hand-over.

9.95.1 Member Data Documentation

9.95.1.1 int DtapiTs::DtDescDvbLinkage::MobileHandOverInfo::m_InitialServiceId

Service identifier for which the hand-over is valid.

9.95.1.2 int DtapiTs::DtDescDvbLinkage::MobileHandOverInfo::m_NetworkId

The network id of the terrestrial network that supports the indicated service.

9.95.1.3 int DtapiTs::DtDescDvbLinkage::MobileHandOverInfo::m_OrigType

Flag that specified the table in which the link is originated.

0 for NIT and 1 for SDT.

9.96 DtapiTs::DtDescDvbServiceList::ServiceListItem Struct Reference

Public Attributes

- int m_ServiceId

  Service identifier.

- int m_ServiceType

  Service type field as defined by table 87 in ETSI EN 300 468.

9.96.1 Member Data Documentation

9.96.1.1 int DtapiTs::DtDescDvbServiceList::ServiceListItem::m_ServiceId

Service identifier.

Usually corresponds with a service id in the PAT and PMT tables. If m_ServiceType is 0x04, 0x18 or 0x1B this is not the case.

9.96.1.2 int DtapiTs::DtDescDvbServiceList::ServiceListItem::m_ServiceType

Service type field as defined by table 87 in ETSI EN 300 468.

9.97 DtapiTs::DtDescDvbSubtitling::Subtitling Struct Reference

Public Attributes

- int m_AncPageId

  Ancillary page identifier.

- int m_CompositionPageId

  Composition page identifier.

- std::string m_LangCode

  ISO-639 language code of these subtitles.
- int m_SubtitlingPage

  The content and intended display of the subtitle.

9.98 DtapiTs::DtDescDvbTeletext::Teletext Struct Reference

Public Attributes

- std::string m_LangCode

  ISO-639 language code of this teletext.
- int m_MagazineNum

  The magazine number as defined in EN 300 706.
- uint8_t m_PageNum

  An 8-bit field consisting of two 4-bit hex digits that together make up the page number.
- int m_Type

  The type fo the teletext page.

9.98.1 Member Data Documentation

9.98.1.1 uint8_t DtapiTs::DtDescDvbTeletext::Teletext::m_PageNum

An 8-bit field consisting of two 4-bit hex digits that together make up the page number.

Chapter 10

Example Documentation

10.1 example1.cpp

Small example program that attaches to a DekTec input port, reads the available stream and feeds it to DTAPI-TS to get the network and program names. It'll keep reading data from the input port until you press any key to exit the application. Every 5 seconds the network name and all program names will be printed to the console.

```
#include "DTAPITS.h"
#include <stdio.h>
#include <time.h>
#include <Windows.h>
#include <conio.h>

using namespace DtapiTs;
using namespace Dtapi;

// Global buffer size. This is the maximum amount of data we read at a time.
#define BUF_SIZE  (4*1024*1024)
// Minimum fifoload before we read anything.
#define MIN_FIFOLOAD (8*1024)

// General define which calls a DTAPI function and checks the result. If it's
      not
// DTAPI_OK, an error message will be printed to the console and the
      application
// will exit. Please note that there is a memory leak (DtTsInfo* Info and
// DtTsInfoInput* InfoInput are never cleaned up in case of an error) but since
// the memory will be freed when the application exits it's no problem in this
      case.
// In a real application you'll want better error handling but as example this
// will do. This define is here to reduce the boilerplate in the rest of the
      code.
#define MUST_SUCCEED(expr) \
    if ((dr = (expr)) != DTAPI_OK) \
    { \
        fprintf(stderr, "ERROR: Function call failed (%s), error = %s\n",
      #expr, \

      DtapiResult2Str(dr)); \
       MyInputChan.Detach(0); \
       Dta2145Dev.Detach(); \
       return 1; \
    }

// A global buffer we'll use for reading data from the input port before
      feeding it
// to the DtTsInfoInput object.
static char  GlobalBuffer[BUF_SIZE];

// c++ main function
int main()
{
    printf("DtapiTs example program.\n");
    printf("Used DtapiTs version: %d.%d.%d.%d\n", DTAPITS_VERSION_MAJOR,
                  DTAPITS_VERSION_MINOR, DTAPITS_VERSION_BUGFIX,
      DTAPITS_VERSION_BUILD);
    printf("Used DTAPI version: %d.%d.%d.%d\n", DTAPI_VERSION_MAJOR,
                         DTAPI_VERSION_MINOR, DTAPI_VERSION_BUGFIX,
      DTAPI_VERSION_BUILD);
```

```
DTAPI_RESULT  dr;               // Result variable used by the MUST_SUCCEED
   macro
DtDevice  Dta2145Dev;           // The Dtapi device instance
DtInpChannel  MyInputChan;      // The input channel object

printf("Connecting to card and port\n");

// This example uses a hard-coded card type. In a real program you should
   use
// the available Dtapi functions to scan for all available cards. After
   this scan
// you can present let the user chose which card and port to attach to.
MUST_SUCCEED(Dta2145Dev.AttachToType(2145))
MUST_SUCCEED(MyInputChan.AttachToPort(&Dta2145Dev, 1))

// Transparant packets with a timestamp are the recommended input type for
   DtapiTs
MUST_SUCCEED(MyInputChan.SetRxMode(DTAPI_RXMODE_STTRP |
  DTAPI_RXMODE_TIMESTAMP32))

// Get the clock frequency the card uses internally. This is important
   since it
// defines the relation between the timestamp values and the wall clock
   time.
int RefClkFreqHz;
MUST_SUCCEED(Dta2145Dev.GetRefClkFreq(RefClkFreqHz))

// Create our DtapiTs main object and a corresponding input object.
DtTsLib  Lib;
DtTsInfo*  Info = Lib.CreateDtTsInfoInstance();
Info->SetStandardMode(DT_STANDARDMODE_DVB
  );

DtTsInfoInput*  InfoInput =
  DtTsTransparentInput::CreateInstance(Info,
  RefClkFreqHz);

// Start receiving data
MUST_SUCCEED(MyInputChan.SetRxControl(DTAPI_RXCTRL_RCV))

// First wait until the card has locked to the signal
printf("Waiting for lock: |");

while (!kbhit())
{
    int PacketSize, NumInv, ClkDet, AsiLock, RateOk, AsiInv;
    MUST_SUCCEED(MyInputChan.GetStatus(PacketSize, NumInv, ClkDet, AsiLock,
   RateOk,

    AsiInv));
    if (AsiLock==DTAPI_ASI_INLOCK && ClkDet==DTAPI_CLKDET_OK &&
                                                        PacketSize!=
  DTAPI_PCKSIZE_INV)
        break;
    static const char  WaitingSymbols[] = "/-\\|";
    static int  CurSymbol = 0;
    printf("\b%c", WaitingSymbols[CurSymbol]);
    CurSymbol = (CurSymbol + 1) % 4;
    // Wait 100ms before trying again
    Sleep(100);
}

printf("\nInput signal detected.\n");

// Keep track of the last time we printed anything
time_t  LastTime = time(NULL);
while (!kbhit())
{
    // Get the amount of data available in the input buffer
    int  FifoLoad;
    MUST_SUCCEED(MyInputChan.GetFifoLoad(FifoLoad))

    // Make sure there is some data available, if not wait a bit and try
   again
    if (FifoLoad < MIN_FIFOLOAD)
    {
        Sleep(20);
        continue;
    }

    int  BytesToRead = min(FifoLoad, BUF_SIZE);
    // Make sure we only read multiples of 4 bytes. This is a precondition
   of
    // DtInpChannel::Read
    BytesToRead &= ~3;
    MUST_SUCCEED(MyInputChan.Read(GlobalBuffer, BytesToRead))
    // Feed the data to the input object
```

```
        InfoInput->NewData((uint8_t*)GlobalBuffer, BytesToRead);

        // Display network name and program names every 5 s.
        time_t  CurTime = time(NULL);
        if (CurTime - LastTime > 5)
        {
            LastTime = CurTime;
            printf("\nNetwork name: %ls\n", Info->m_Data.m_NetworkName.c_str())
      ;
            DtTsData::DtSvcInfoMap::iterator it=Info->m_Data.m_ServiceInfo.
      begin();
            for (; it != Info->m_Data.m_ServiceInfo.end(); ++it)
            {
                DtServiceInfo  &Program = it->second;
                printf("Program %d. Name=%ls, bitrate=%d\n", Program.
      m_ProgramNumber,
                                          Program.GetName().c_str(),
      Program.m_AvgBitrate);
            }
        }
    }

    // Clean up the DtapiTs Info and InfoInput objects.
    DtTsInfoInput::RemoveInstance(InfoInput);
    Lib.RemoveDtTsInfoInstance(Info);

    return 0;
}
```

10.2 example2.cpp

Example with two threads: one for reading from the input and one for processing the data.This example builds on example1.cpp, see that for code on how to obtain the DtTsInfo and DtTsInfoInput objects that are used here. Creating the actual threads is left as exercise for the reader, this example is purerly here to demonstrate how to interact with the DTAPI-TS library in a thread-safe way.

```
#include "DTAPITS.h"

bool ExitThread1 = false;
bool ExitThread2 = false;

static char  Thread1GlobalBuffer[BUF_SIZE];

void Thread1Mainloop(DtInpChannel& MyInputChan, DtTsInfoInput* InfoInput)
{
    while (!ExitThread1)
    {
        int  FifoLoad;
        if (MyInputChan.GetFifoLoad(FifoLoad) != DTAPI_OK)
        {
            //TODO: error handling
            continue;
        }

        int  BytesToRead = min(FifoLoad, BUF_SIZE);
        // Make sure we only read multiples of 4 bytes. This is a precondition
       of
        // DtInpChannel::Read
        BytesToRead &= ~3;
        if (MyInputChan.Read(GlobalBuffer, BytesToRead) != DTAPI_OK)
        {
            //TODO: error handling
            continue;
        }
        // Feed the data to the input object. We don't have to do anything
       special here,
        // DtapiTs locks/unlocks before modyfing any data members of the
       DtTsInfo object.
        InfoInput->NewData((uint8_t*)GlobalBuffer, BytesToRead);
    }
}

void Thread2Mainloop(DtTsInfo* Info)
{
    while (!ExitThread2)
    {
        // Lock the DtTsInfo object
        Info->Lock();
        // Create a copy of the data
        DtTsData  Data = Info->m_Data;
```

```
        // Unlock the DtTsInfo object
        Info->Unlock();

        // TODO: here you can use the Data variable without worrying about
       thread-safety.
        // If you want to change any settings like the table timeouts, you'll
       have to
        // do in between the Lock() and Unlock() calls.
        Sleep(1000);
    }
}
```

10.3 example3.cpp

This is a complete demo program that analyzes a given transport-stream file and prints basic information about the PIDs and services it finds.

```
//#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#* TsAnalyzer.cpp *#*#*#*#*#*#*#*#*#*# (C)
       2012 DekTec
//
// Basic transport-stream file analyzer using DTAPI-TS
//
// Usage: TsAnalyzer file [bitrate]

#include "DTAPITS.h"
#include <cstdio>
#include <cstdlib>

using namespace DtapiTs;

int  main(int argc, char *argv[])
{
    if (argc < 2)
    {
        fprintf(stderr, "First argument must be the file to analyze.\n");
        return 1;
    }
    FILE* f = fopen(argv[1], "rb");
    if (f == NULL)
    {
        fprintf(stderr, "Failed to open file.\n");
        return 2;
    }
    int  Bitrate = 10000000; // 10Mbps is default bitrate if not specified
       otherwise
    if (argc >= 3)
        Bitrate = atoi(argv[2]);

    //+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ Create instance of DTAPI-TS
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=
    DtTsLib  Lib;

    //+=+=+=+=+=+=+ Create instance of DtTsInfo (will hold analysis results)
       +=+=+=+=+=+=+
    DtTsInfo*  Info = Lib.CreateDtTsInfoInstance(
      );
    Info->SetStandardMode(DT_STANDARDMODE_DVB
      );

    //+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ Create instance of TS packet source
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=
    DtTsPacketInput*  Input = DtTsPacketInput::CreateInstance
      (Info, Bitrate);

    //=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ Read file and feed it to DTAPI-TS
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=+
    int  BufSize = 1024*1024;
    uint8_t*  Buf = new uint8_t[BufSize];
    size_t  NumRead;
    size_t  TotalRead = 0;
    while ((NumRead = fread(Buf, 1, BufSize, f)) != 0)
    {
        TotalRead += NumRead;
        Input->NewData(Buf, NumRead);
    }
    printf("Bytes read from file: %d\n", TotalRead);
    delete [] Buf;

    //+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ Analysis is done now.
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
    DtTsData&  Data = Info->m_Data;
```

```
    for (int i=0; i<DT_NUM_PIDS; i++)
    {
        // We're not interested in the null pid
        if (i == 0x1FFF)
            continue;
        // Pid i is not part of this file
        if (Data.m_PidInfo[i] == NULL)
            continue;

        DtPidInfo  *Pid = Data.m_PidInfo[i];
        // Pid->m_StreamType contains a value from enum DtStreamType, like
        // DT_STREAMTYPE_MPEG2_VIDEO or DT_STREAMTYPE_AAC.
        printf("Pid %d: %ls\n", i, Pid->GetDescription().c_str())
      ;
        printf("  Avg bitrate for this pid: %d bps\n", Pid->m_Bitrate.
      m_Avg);

        if (Pid->m_Pcrs.m_AvgRate > 0)
        {
            // There are PCRs in this pid, so print some statistics
            printf("  Contains PCRs:\n");
            printf("    TS rate according to PCRs: %d\n", Pid->m_Pcrs.
      m_TsRate);
        }

        if (Pid->m_AudioEs != NULL)
        {
            // There is an audio elementary stream in this pid, print some
       information
            printf("  Audio stream\n");
            if ((Pid->m_AudioEs->m_Mask &
      DtAudioEsInfo::c_BitrateField) != 0)
                printf("    Audio bitrate: %d bps\n", Pid->m_AudioEs->
      m_Bitrate);
            if ((Pid->m_AudioEs->m_Mask &
      DtAudioEsInfo::c_SamplerateField) != 0)
                printf("    Audio sampling rate: %d Hz\n", Pid->m_AudioEs
      ->m_Samplerate);
        }
        if (Pid->m_VideoEs != NULL)
        {
            // There is a video elementary stream in this pid, print some
       information
            printf("  Video stream\n");
            if ((Pid->m_VideoEs->m_Mask &
      DtVideoEsInfo::c_SizeField) != 0)
                printf("    Resolution: %dx%d\n", Pid->m_VideoEs->
      m_HorzSize, Pid->m_VideoEs->m_VertSize);
            if ((Pid->m_VideoEs->m_Mask &
      DtVideoEsInfo::c_FrameRateField) != 0)
                printf("    Frame rate: %.2f\n", Pid->m_VideoEs->
      m_FrameRate);
        }
        printf("\n");
    }

    //+=+=+=+=+=+=+=+=+=+=+=+=+ Print service names and bitrates
       +=+=+=+=+=+=+=+=+=+=+=+
    printf("\n%d services:\n", Data.m_ServiceInfo.size());
    DtTsData::DtSvcInfoMap::iterator  It;
    for (It=Data.m_ServiceInfo.begin(); It!=Data.m_ServiceInfo
      .end(); ++It)
    {
        int  ServiceNo = It->first;
        DtServiceInfo&  SvcInfo = It->second;
        printf("\nService %d: %ls\n", ServiceNo, SvcInfo.GetName().c_str
      ());
        printf("  Average bitrate: %d\n", SvcInfo.m_AvgBitrate);
        if (!SvcInfo.m_ProviderName.empty())
            printf("  Provider: %ls\n", SvcInfo.m_ProviderName.
      c_str());
    }

    return 0;
}
```

10.4 example4.cpp

Example code on how to work with the TR 101 290 callbacks.

```
void  PcrRepetitionErrorCb(void*, DtTr101290Indicator Ind, const
```

```
      DtTr101290Error& Err)
{
    printf("A PCR repetition error has occured for the %d time.\n", Err.
      m_ErrCount);
    printf("Error message: %ls\n", Err.m_Msg.c_str());
}

int  main()
{
    //+=+=+=+=+=+=+=+=+=+=+=+=+=+ Create instance of DTAPI-TS
       +=+=+=+=+=+=+=+=+=+=+=+=+=+=
    DtTsLib  Lib;

    //+=+=+=+=+=+=+ Create instance of DtTsInfo (will hold analysis results)
       +=+=+=+=+=+=+
    DtTsInfo*  Info = Lib.CreateDtTsInfoInstance();

    //=+=+=+=+=+=+=+=+=+ Set custom callback for pcr repetition errors
       +=+=+=+=+=+=+=+=+=+
    Info->AddTr101290ErrorCallback(DT_ERR_B_P2_PCR_REPETITION,
      PcrRepetitionErrorCb);

    // TODO: read data from file or hardware and feed it to the analyzer.
    // Your custom callback function will be called every time an PCR
       repetition
    // error occurs.

    return 0;
}
```

Index