

## DTAPI-TS Documentation



# Contents

<b>1</b>	<b>Welcome</b>	<b>1</b>
1.1	Summary	1
1.2	Installation	1
1.2.1	Windows	1
1.2.2	Linux	2
1.3	Usage	2
<b>2</b>	<b>Revision History</b>	<b>3</b>
<b>3</b>	<b>Module Index</b>	<b>11</b>
3.1	Modules	11
<b>4</b>	<b>Namespace Index</b>	<b>13</b>
4.1	Namespace List	13
<b>5</b>	<b>Hierarchical Index</b>	<b>15</b>
5.1	Class Hierarchy	15
<b>6</b>	<b>Class Index</b>	<b>19</b>
6.1	Class List	19
<b>7</b>	<b>Module Documentation</b>	<b>23</b>
7.1	Structured information from binary descriptors	23
7.1.1	Detailed Description	24
7.2	Structured information from binary tables	25
7.2.1	Detailed Description	25
<b>8</b>	<b>Namespace Documentation</b>	<b>27</b>
8.1	DtapiTs Namespace Reference	27
8.1.1	Detailed Description	39
8.1.2	Enumeration Type Documentation	39
8.1.2.1	DtAacObjType	39
8.1.2.2	DtAacProfile	40
8.1.2.3	DTAPITS_RESULT	40
8.1.2.4	DtAtscCcType	41
8.1.2.5	DtAudioMode	41
8.1.2.6	DtDeliverySystem	42
8.1.2.7	DtDvbT2MisoMode	42
8.1.2.8	DtFecOuter	42
8.1.2.9	DtGuardInterval	42
8.1.2.10	DtMpaLayer	42
8.1.2.11	DtMpaVersion	43
8.1.2.12	DtPolarization	43
8.1.2.13	DtRollOff	43
8.1.2.14	DtScrambling	43
8.1.2.15	DtServiceType	43
8.1.2.16	DtShBandwidth	44

8.1.2.17	DtShCodeRate	44
8.1.2.18	DtShModMode	44
8.1.2.19	DtShModType	44
8.1.2.20	DtStandardMode	45
8.1.2.21	DtStreamType	45
8.1.2.22	DtTableType	46
8.1.2.23	DtTr101290Bitmask	47
8.1.2.24	DtTr101290Indicator	48
8.1.2.25	DtTransmissionMode	49
8.1.2.26	DtVideoChromaFormat	49
8.1.2.27	DtWeFlag	50
<b>9</b>	<b>Class Documentation</b>	<b>51</b>
9.1	DtapiTs::DtPes::DataBuffer Class Reference	51
9.2	DtapiTs::DtAacEsInfo Class Reference	51
9.3	DtapiTs::DtAc3EsInfo Class Reference	52
9.4	DtapiTs::DtAc4EsInfo Class Reference	52
9.5	DtapiTs::DtAudioEsInfo Class Reference	52
9.5.1	Detailed Description	54
9.6	DtapiTs::DtAudioEsInfo2 Class Reference	54
9.7	DtapiTs::DtBitrate Class Reference	55
9.7.1	Detailed Description	55
9.8	DtapiTs::DtBitrateSettings Class Reference	55
9.8.1	Detailed Description	56
9.8.2	Constructor & Destructor Documentation	56
9.8.2.1	DtBitrateSettings	56
9.8.3	Member Data Documentation	56
9.8.3.1	m_NumAvgValues	56
9.9	DtapiTs::DtCallback1< TArg1 > Class Template Reference	56
9.9.1	Detailed Description	57
9.10	DtapiTs::DtCallback2< TArg1, TArg2 > Class Template Reference	57
9.10.1	Detailed Description	57
9.11	DtapiTs::DtCallback3< TArg1, TArg2, TArg3 > Class Template Reference	57
9.11.1	Detailed Description	58
9.12	DtapiTs::DtCaSystem Class Reference	58
9.12.1	Detailed Description	58
9.12.2	Member Data Documentation	58
9.12.2.1	m_CaSystemId	58
9.13	DtapiTs::DtDescDvbAc3 Class Reference	59
9.13.1	Detailed Description	59
9.13.2	Member Data Documentation	59
9.13.2.1	m_Asvc	59
9.13.2.2	m_Bsid	59
9.13.2.3	m_ComponentType	60
9.13.2.4	m_MainId	60
9.14	DtapiTs::DtDescDvbCDelivery Class Reference	60
9.14.1	Detailed Description	60
9.15	DtapiTs::DtDescDvbComponent Class Reference	60
9.15.1	Detailed Description	61
9.15.2	Member Data Documentation	61
9.15.2.1	m_ComponentTag	61
9.16	DtapiTs::DtDescDvbDataBroadcast Class Reference	61
9.16.1	Detailed Description	62
9.16.2	Member Data Documentation	62
9.16.2.1	m_ComponentTag	62
9.16.2.2	m_DataBroadcastId	62
9.16.2.3	m_SelectorBytes	62
9.17	DtapiTs::DtDescDvbDataBroadcastId Class Reference	62

9.17.1	Detailed Description	63
9.17.2	Member Data Documentation	63
9.17.2.1	m_DataBroadcastId	63
9.17.2.2	m_SelectorBytes	63
9.18	DtapiTs::DtDescDvbLinkage Class Reference	63
9.18.1	Detailed Description	64
9.18.2	Member Data Documentation	64
9.18.2.1	m_Event	64
9.18.2.2	m_ExtendedEvents	64
9.18.2.3	m_MobileHandOver	64
9.18.2.4	m_OrigNetworkId	64
9.18.2.5	m_TransportStreamId	64
9.19	DtapiTs::DtDescDvbLocalTimeOffset Class Reference	64
9.19.1	Detailed Description	65
9.20	DtapiTs::DtDescDvbMultilingualComponent Class Reference	65
9.20.1	Detailed Description	65
9.20.2	Member Data Documentation	65
9.20.2.1	m_ComponentTag	65
9.20.2.2	m_Descriptions	65
9.21	DtapiTs::DtDescDvbNetworkName Class Reference	66
9.21.1	Detailed Description	66
9.21.2	Member Data Documentation	66
9.21.2.1	m_NetworkName	66
9.22	DtapiTs::DtDescDvbSDelivery Class Reference	66
9.22.1	Detailed Description	67
9.22.2	Member Data Documentation	67
9.22.2.1	m_IsDvbS2	67
9.22.2.2	m_ModType	67
9.22.2.3	m_RollOff	67
9.22.2.4	m_WestEastFlag	67
9.23	DtapiTs::DtDescDvbService Class Reference	67
9.23.1	Detailed Description	68
9.23.2	Member Data Documentation	68
9.23.2.1	m_ServiceType	68
9.24	DtapiTs::DtDescDvbServiceList Class Reference	68
9.24.1	Detailed Description	69
9.25	DtapiTs::DtDescDvbSubtitling Class Reference	69
9.25.1	Detailed Description	69
9.26	DtapiTs::DtDescDvbTDelivery Class Reference	69
9.26.1	Detailed Description	70
9.26.2	Member Data Documentation	70
9.26.2.1	m_HierarchyInformation	70
9.27	DtapiTs::DtDescDvbTeletext Class Reference	70
9.27.1	Detailed Description	71
9.28	DtapiTs::DtDescMpegCa Class Reference	71
9.28.1	Detailed Description	71
9.28.2	Member Data Documentation	71
9.28.2.1	m_CaPid	71
9.28.2.2	m_CaSystemId	71
9.29	DtapiTs::DtDescMpegLanguage Class Reference	72
9.29.1	Detailed Description	72
9.29.2	Member Data Documentation	72
9.29.2.1	m_Codes	72
9.30	DtapiTs::DtDescMpegPrivDataIndicator Class Reference	72
9.30.1	Detailed Description	73
9.31	DtapiTs::DtDescMpegRegistration Class Reference	73
9.31.1	Detailed Description	73
9.32	DtapiTs::DtDescMpegVideoStream Class Reference	73

9.32.1	Detailed Description	74
9.32.2	Member Data Documentation	74
9.32.2.1	m_ChromaFormat	74
9.32.2.2	m_ConstrainedParameter	74
9.32.2.3	m_FrameRateCode	74
9.32.2.4	m_FrameRateExtension	74
9.32.2.5	m_Mpeg1Only	74
9.32.2.6	m_MultipleFrameRates	74
9.32.2.7	m_ProfileLevelIndication	75
9.33	DtapiTs::DtDescPrivLcn Class Reference	75
9.33.1	Detailed Description	75
9.34	DtapiTs::DtDescriptor Class Reference	75
9.34.1	Detailed Description	76
9.34.2	Member Data Documentation	76
9.34.2.1	m_DescriptorType	76
9.34.2.2	m_ExtendedTag	76
9.34.2.3	m_Pds	76
9.35	DtapiTs::DtDvbCNitInfo Class Reference	77
9.35.1	Detailed Description	77
9.35.2	Member Data Documentation	77
9.35.2.1	Constellation	77
9.36	DtapiTs::DtDvbShModInfo Class Reference	77
9.36.1	Detailed Description	78
9.36.2	Member Data Documentation	78
9.36.2.1	m_CommonMultiplier	78
9.36.2.2	m_CompleteInterleaver	78
9.36.2.3	m_NofLateTaps	78
9.36.2.4	m_NofSlices	78
9.36.2.5	m_NonLateIncrement	78
9.36.2.6	m_SliceDistance	78
9.37	DtapiTs::DtDvbShNitInfo Class Reference	79
9.37.1	Detailed Description	79
9.37.2	Member Data Documentation	79
9.37.2.1	m_DiversityMode	79
9.37.2.2	m_ModInfo	79
9.38	DtapiTs::DtDvbShOfdmInfo Class Reference	79
9.38.1	Detailed Description	80
9.38.2	Member Data Documentation	80
9.38.2.1	m_CommonFrequency	80
9.38.2.2	m_Constellation	80
9.38.2.3	m_Priority	80
9.39	DtapiTs::DtDvbShTdmInfo Class Reference	80
9.39.1	Detailed Description	80
9.39.2	Member Data Documentation	81
9.39.2.1	m_SymbolRate	81
9.40	DtapiTs::DtDvbSNitInfo Class Reference	81
9.40.1	Detailed Description	81
9.40.2	Member Data Documentation	81
9.40.2.1	InputStreamIdentifier	81
9.40.2.2	ModType	82
9.40.2.3	S2FieldsPresent	82
9.40.2.4	ScramblingSequenceIndex	82
9.41	DtapiTs::DtDvbT2CellInfo Class Reference	82
9.41.1	Detailed Description	82
9.42	DtapiTs::DtDvbT2NitInfo Class Reference	82
9.42.1	Detailed Description	83
9.42.2	Member Data Documentation	83
9.42.2.1	m_OtherFrequencyUsed	83

9.42.2.2	<a href="#">m_Plpld</a>	83
9.42.2.3	<a href="#">m_T2SystemId</a>	83
9.43	<a href="#">DtapiTs::DtDvbT2SubCellInfo Class Reference</a>	83
9.43.1	<a href="#">Detailed Description</a>	83
9.43.2	<a href="#">Member Data Documentation</a>	84
9.43.2.1	<a href="#">m_SubCellId</a>	84
9.44	<a href="#">DtapiTs::DtDvbTNitInfo Class Reference</a>	84
9.44.1	<a href="#">Detailed Description</a>	84
9.44.2	<a href="#">Member Data Documentation</a>	84
9.44.2.1	<a href="#">Bandwith</a>	84
9.44.2.2	<a href="#">CodeRateHpStream</a>	85
9.44.2.3	<a href="#">HierarchyInformation</a>	85
9.44.2.4	<a href="#">OtherFrequencyUsed</a>	85
9.44.2.5	<a href="#">TransmissionMode</a>	85
9.45	<a href="#">DtapiTs::DtEac3EsInfo Class Reference</a>	85
9.46	<a href="#">DtapiTs::DtEsInfoBase Class Reference</a>	85
9.47	<a href="#">DtapiTs::DtHeAacEsInfo Class Reference</a>	86
9.48	<a href="#">DtapiTs::DtJitterPoint Class Reference</a>	86
9.49	<a href="#">DtapiTs::DtDescPrivLcn::DtLogicalChannelNumber Struct Reference</a>	86
9.49.1	<a href="#">Member Data Documentation</a>	87
9.49.1.1	<a href="#">m_IsVisible</a>	87
9.50	<a href="#">DtapiTs::DtMpaEsInfo Class Reference</a>	87
9.51	<a href="#">DtapiTs::DtPcr Class Reference</a>	87
9.51.1	<a href="#">Detailed Description</a>	88
9.52	<a href="#">DtapiTs::DtPcrInfo Class Reference</a>	88
9.52.1	<a href="#">Detailed Description</a>	88
9.52.2	<a href="#">Member Data Documentation</a>	89
9.52.2.1	<a href="#">m_Df</a>	89
9.53	<a href="#">DtapiTs::DtPes Class Reference</a>	89
9.54	<a href="#">DtapiTs::DtPidInfo Class Reference</a>	90
9.54.1	<a href="#">Detailed Description</a>	91
9.54.2	<a href="#">Member Function Documentation</a>	91
9.54.2.1	<a href="#">GetDescription</a>	91
9.54.2.2	<a href="#">HasTableType</a>	91
9.54.3	<a href="#">Member Data Documentation</a>	91
9.54.3.1	<a href="#">m_SeenBefore</a>	92
9.54.3.2	<a href="#">m_TableTypeMask</a>	92
9.55	<a href="#">DtapiTs::DtTablePat::DtProgramMapping Struct Reference</a>	92
9.55.1	<a href="#">Detailed Description</a>	92
9.56	<a href="#">DtapiTs::DtPtsDts Class Reference</a>	92
9.56.1	<a href="#">Detailed Description</a>	93
9.57	<a href="#">DtapiTs::DtServiceComponentInfo Class Reference</a>	93
9.57.1	<a href="#">Detailed Description</a>	94
9.57.2	<a href="#">Member Data Documentation</a>	94
9.57.2.1	<a href="#">m_CaSystems</a>	94
9.57.2.2	<a href="#">m_Description</a>	94
9.57.2.3	<a href="#">m_HasPrivateDataDesc</a>	94
9.58	<a href="#">DtapiTs::DtServiceInfo Class Reference</a>	94
9.58.1	<a href="#">Detailed Description</a>	95
9.58.2	<a href="#">Member Function Documentation</a>	95
9.58.2.1	<a href="#">GetName</a>	95
9.58.2.2	<a href="#">InService</a>	96
9.58.3	<a href="#">Member Data Documentation</a>	96
9.58.3.1	<a href="#">m_CaSystems</a>	96
9.58.3.2	<a href="#">m_OrigServiceType</a>	96
9.58.3.3	<a href="#">m_ProgramNumber</a>	96
9.58.3.4	<a href="#">m_ServiceType</a>	96
9.59	<a href="#">DtapiTs::DtStructuredTable Class Reference</a>	96

9.59.1	Detailed Description	97
9.59.2	Member Function Documentation	97
9.59.2.1	DecodeFromTable	97
9.60	DtapiTs::DtSubTableId Class Reference	97
9.60.1	Detailed Description	98
9.60.2	Member Function Documentation	98
9.60.2.1	Matches	98
9.60.2.2	operator<	98
9.61	DtapiTs::DtTable Class Reference	98
9.61.1	Detailed Description	99
9.61.2	Constructor & Destructor Documentation	99
9.61.2.1	DtTable	99
9.61.3	Member Function Documentation	99
9.61.3.1	operator=	99
9.61.4	Member Data Documentation	99
9.61.4.1	m_Sections	99
9.61.4.2	m_Version	100
9.62	DtapiTs::DtTableBat Class Reference	100
9.62.1	Detailed Description	100
9.62.2	Member Function Documentation	100
9.62.2.1	FindTs	100
9.63	DtapiTs::DtTableBatInner Class Reference	101
9.63.1	Detailed Description	101
9.64	DtapiTs::DtTableCat Class Reference	101
9.64.1	Detailed Description	101
9.65	DtapiTs::DtTableNit Class Reference	102
9.65.1	Detailed Description	102
9.65.2	Member Function Documentation	102
9.65.2.1	FindTsLoop	102
9.65.3	Member Data Documentation	102
9.65.3.1	m_NetworkDescriptors	102
9.66	DtapiTs::DtTableNitInner Class Reference	103
9.66.1	Detailed Description	103
9.67	DtapiTs::DtTablePat Class Reference	103
9.67.1	Detailed Description	104
9.68	DtapiTs::DtTablePmt Class Reference	104
9.68.1	Detailed Description	104
9.69	DtapiTs::DtTablePmtInner Class Reference	104
9.69.1	Detailed Description	105
9.70	DtapiTs::DtTableSdt Class Reference	105
9.70.1	Detailed Description	105
9.70.2	Member Function Documentation	106
9.70.2.1	FindService	106
9.70.3	Member Data Documentation	106
9.70.3.1	m_TransportStreamId	106
9.71	DtapiTs::DtTableSdtInner Class Reference	106
9.71.1	Detailed Description	106
9.71.2	Member Data Documentation	106
9.71.2.1	m_EitPresentFollowing	106
9.71.2.2	m_EitSchedule	106
9.71.2.3	m_FreeCaMode	107
9.72	DtapiTs::DtTableSection Class Reference	107
9.72.1	Detailed Description	107
9.72.2	Constructor & Destructor Documentation	107
9.72.2.1	DtTableSection	107
9.72.3	Member Function Documentation	107
9.72.3.1	operator=	107
9.73	DtapiTs::DtTableTdt Class Reference	108



9.73.1 Detailed Description	108
9.74 DtapiTs::DtTableTot Class Reference	108
9.74.1 Detailed Description	109
9.75 DtapiTs::DtTimeDiff Class Reference	109
9.75.1 Detailed Description	110
9.75.2 Member Function Documentation	110
9.75.2.1 operator<	110
9.76 DtapiTs::DtTimestamp Class Reference	110
9.76.1 Detailed Description	111
9.77 DtapiTs::DtTp Class Reference	111
9.78 DtapiTs::DtTr101290 Class Reference	112
9.78.1 Detailed Description	112
9.79 DtapiTs::DtTr101290Error Class Reference	112
9.79.1 Detailed Description	113
9.79.2 Member Data Documentation	113
9.79.2.1 m_ErrCount	113
9.79.2.2 m_IsSet	113
9.79.2.3 m_Latched	113
9.79.2.4 m_Time	113
9.80 DtapiTs::DtTsData Class Reference	113
9.80.1 Detailed Description	115
9.80.2 Member Function Documentation	115
9.80.2.1 GetNitFrequency	115
9.80.3 Member Data Documentation	115
9.80.3.1 m_CaSystems	115
9.80.3.2 m_DeliverySystem	115
9.80.3.3 m_ErrIndErrors	115
9.80.3.4 m_NitTsRate	115
9.80.3.5 m_PacketSize	115
9.80.3.6 m_SyncByteErrors	116
9.80.3.7 m_TmccDataValid	116
9.81 DtapiTs::DtTsInfo Class Reference	116
9.81.1 Detailed Description	118
9.81.2 Member Typedef Documentation	118
9.81.2.1 DtJitterCallback	118
9.81.2.2 DtPacketCallback	118
9.81.2.3 DtPesCallback	118
9.81.2.4 DtSectionCallback	119
9.81.2.5 DtTableCallback	119
9.81.2.6 DtTableTimeoutCallback	119
9.81.3 Member Function Documentation	119
9.81.3.1 AddJitterCallback	119
9.81.3.2 AddNewSectionCallback	119
9.81.3.3 AddPesPacketCallback	120
9.81.3.4 AddPesPacketCallback	120
9.81.3.5 AddTableChangedCallback	120
9.81.3.6 AddTableTimeoutCallback	120
9.81.3.7 GetIsdbtPars	120
9.81.3.8 Lock	120
9.81.3.9 NewPacket	120
9.81.3.10 NewTimestamp	120
9.81.3.11 Reset	121
9.81.3.12 SetJitterWindow	121
9.81.3.13 SetStandardMode	121
9.81.3.14 Unlock	121
9.81.4 Member Data Documentation	121
9.81.4.1 m_CompletePes	121
9.81.4.2 m_Data	121

9.81.4.3	<a href="#">m_PreferredLanguages</a>	121
9.81.4.4	<a href="#">m_TableTimeoutCb</a>	122
9.81.4.5	<a href="#">m_UseTableCache</a>	122
9.82	<a href="#">DtapiTs::DtTsInfoInput Class Reference</a>	122
9.82.1	<a href="#">Detailed Description</a>	122
9.82.2	<a href="#">Member Function Documentation</a>	123
9.82.2.1	<a href="#">NewData</a>	123
9.82.2.2	<a href="#">SetTsInfoObject</a>	123
9.83	<a href="#">DtapiTs::DtTsLib Class Reference</a>	123
9.83.1	<a href="#">Detailed Description</a>	123
9.83.2	<a href="#">Member Function Documentation</a>	124
9.83.2.1	<a href="#">CreateDtTsInfoInstance</a>	124
9.84	<a href="#">DtapiTs::DtTsPacketInput Class Reference</a>	124
9.84.1	<a href="#">Detailed Description</a>	124
9.84.2	<a href="#">Member Function Documentation</a>	124
9.84.2.1	<a href="#">CreateInstance</a>	124
9.85	<a href="#">DtapiTs::DtTsTimestampedPacketInput Class Reference</a>	125
9.85.1	<a href="#">Detailed Description</a>	125
9.85.2	<a href="#">Member Function Documentation</a>	125
9.85.2.1	<a href="#">CreateInstance</a>	125
9.86	<a href="#">DtapiTs::DtTsTransparentInput Class Reference</a>	126
9.86.1	<a href="#">Detailed Description</a>	126
9.86.2	<a href="#">Member Function Documentation</a>	126
9.86.2.1	<a href="#">CreateInstance</a>	126
9.87	<a href="#">DtapiTs::DtVideoAspectRatio Class Reference</a>	126
9.88	<a href="#">DtapiTs::DtVideoEsAvclInfo Class Reference</a>	127
9.89	<a href="#">DtapiTs::DtVideoEsInfo Class Reference</a>	127
9.89.1	<a href="#">Detailed Description</a>	129
9.90	<a href="#">DtapiTs::DtDescDvbLinkage::EventLinkage Struct Reference</a>	129
9.91	<a href="#">DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage Struct Reference</a>	130
9.91.1	<a href="#">Member Data Documentation</a>	130
9.91.1.1	<a href="#">m_LinkType</a>	130
9.91.1.2	<a href="#">m_TargetIdType</a>	130
9.91.1.3	<a href="#">m_TargetOrigNetworkId</a>	130
9.91.1.4	<a href="#">m_TargetServiceId</a>	130
9.91.1.5	<a href="#">m_TargetTsId</a>	130
9.92	<a href="#">DtapiTs::DtEsInfoBase::InfoField&lt; T &gt; Class Template Reference</a>	131
9.93	<a href="#">DtapiTs::DtDescMpegLanguage::LangCode Struct Reference</a>	131
9.94	<a href="#">DtapiTs::DtDescDvbLocalTimeOffset::LocalTimeOffset Struct Reference</a>	131
9.94.1	<a href="#">Member Data Documentation</a>	131
9.94.1.1	<a href="#">m_TimeOfChange</a>	131
9.95	<a href="#">DtapiTs::DtDescDvbLinkage::MobileHandOverInfo Struct Reference</a>	131
9.95.1	<a href="#">Member Data Documentation</a>	132
9.95.1.1	<a href="#">m_InitialServiceId</a>	132
9.95.1.2	<a href="#">m_NetworkId</a>	132
9.95.1.3	<a href="#">m_OrigType</a>	132
9.96	<a href="#">DtapiTs::DtDescDvbServiceList::ServiceListItem Struct Reference</a>	132
9.96.1	<a href="#">Member Data Documentation</a>	132
9.96.1.1	<a href="#">m_ServiceId</a>	132
9.96.1.2	<a href="#">m_ServiceType</a>	132
9.97	<a href="#">DtapiTs::DtDescDvbSubtitling::Subtitling Struct Reference</a>	132
9.98	<a href="#">DtapiTs::DtDescDvbTeletext::Teletext Struct Reference</a>	133
9.98.1	<a href="#">Member Data Documentation</a>	133
9.98.1.1	<a href="#">m_PageNum</a>	133
<b>10</b>	<b>Example Documentation</b>	<b>135</b>
10.1	<a href="#">example1.cpp</a>	135
10.2	<a href="#">example2.cpp</a>	137

---

10.3 <a href="#">example3.cpp</a> . . . . .	138
10.4 <a href="#">example4.cpp</a> . . . . .	139



# 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](mailto: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 settings 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
v1.11.2.52	2024.04.22	<ul style="list-style-type: none"><li>• Update for Apr2024 DTAPI changes</li></ul>
v1.11.1.51	2024.01.25	<ul style="list-style-type: none"><li>• Update for Jan2024 DTAPI changes</li></ul>
v1.11.0.50	2023.11.06	<ul style="list-style-type: none"><li>• Support for Visual Studio 2019/2022(V16/VC17)</li><li>• Update for Oct2023 DTAPI changes</li></ul>
v1.10.8.49	2023.08.25	<ul style="list-style-type: none"><li>• Update for Aug2023 DTAPI changes</li></ul>
v1.10.7.48	2023.06.14	<ul style="list-style-type: none"><li>• Support AVS3 streamtype detection</li><li>• Update for Jun2023 DTAPI changes</li></ul>
v1.10.6.47	2023.01.13	<ul style="list-style-type: none"><li>• Fix for DTAPITS.NET Can't retrieve complete PES packet (&gt;8Kb)</li></ul>

v1.10.5.46	2022.12.15	<ul style="list-style-type: none"> <li>• Update for Dec2022 DTAPI changes</li> </ul>
v1.10.4.45	2022.11.18	<ul style="list-style-type: none"> <li>• Add support for detecting MPEG-H</li> <li>• Add support for detecting SCTE-35</li> <li>• Fix for parsing error on JPEG-2000</li> <li>• Update for Nov2022 DTAPI changes</li> </ul>
v1.10.3.44	2022.06.08	<ul style="list-style-type: none"> <li>• Update for Jun2022 DTAPI changes</li> </ul>
v1.10.2.43	2022.02.15	<ul style="list-style-type: none"> <li>• Update for Feb2022 DTAPI changes</li> </ul>
v1.10.1.42	2021.12.09	<ul style="list-style-type: none"> <li>• Update for Dec2021 DTAPI changes</li> </ul>
v1.10.0.41	2021.09.13	<ul style="list-style-type: none"> <li>• Update for Sept2021 DTAPI changes</li> </ul>
v1.9.0.40	2021.07.02	<ul style="list-style-type: none"> <li>• Fixed error in Example 4</li> <li>• Update for July2021 DTAPI changes</li> </ul>
v1.8.0.39	2021.05.21	<ul style="list-style-type: none"> <li>• Update for May2021 DTAPI changes</li> </ul>
v1.7.0.38	2021.02.01	<ul style="list-style-type: none"> <li>• Added support for J2K detection and ES info</li> <li>• Added support for JPEG-XS detection</li> <li>• Update for Feb2021 DTAPI changes</li> </ul>



v1.6.12.37	2020.11.16	<ul style="list-style-type: none"><li>• Update TR 101 290 2020: 100ms for PCR-error and PCR repetition error</li><li>• Fix for crash on descriptor with invalid extended tag</li><li>• Update for Nov2020 DTAPI changes</li></ul>
v1.5.11.36	2020.08.21	<ul style="list-style-type: none"><li>• Update for Aug2020 DTAPI changes</li></ul>
v1.5.10.35	2020.07.07	<ul style="list-style-type: none"><li>• Update for July2020 DTAPI changes</li></ul>
v1.5.9.34	2020.05.01	<ul style="list-style-type: none"><li>• Update for May2020 DTAPI changes</li></ul>
v1.5.8.33	2020.03.11	<ul style="list-style-type: none"><li>• Update for Mar2020 DTAPI changes</li></ul>
v1.5.7.32	2020.02.17	<ul style="list-style-type: none"><li>• Update for Feb2020 DTAPI changes</li></ul>
v1.5.6.31	2020.01.08	<ul style="list-style-type: none"><li>• Update for Jan2020 DTAPI changes</li><li>• Fix for possible crash on illegal stream data</li></ul>
v1.5.5.30	2019.10.30	<ul style="list-style-type: none"><li>• Update for Oct2019 DTAPI changes</li></ul>
v1.5.4.29	2019.07.25	<ul style="list-style-type: none"><li>• Update for July2019 DTAPI changes</li></ul>
v1.5.3.28	2019.06.25	<ul style="list-style-type: none"><li>• Fix for potential HEVC parsing crash</li><li>• Fix for H264 Interlaced/Non interlaced signalling not correctly parsed</li></ul>

v1.5.2.27	2019.05.15	<ul style="list-style-type: none"><li>• Fix for potential crash when using DT_STANDARDMODE_ATSC mode</li><li>• Fix for HEVC Frame rate not always indicated</li><li>• Fix for H264 videoformat did not indicate Interlaced/Non interlaced signalling</li></ul>
v1.5.1.26	2019.02.15	<ul style="list-style-type: none"><li>• Fix for potential crash on older PC's that did not support AVX instruction set</li></ul>
v1.5.0.25	2019.01.24	<ul style="list-style-type: none"><li>• Added support for TR 101 290 priority 3 errors</li><li>• Added missing Linux support for Aac, Ac3, Ac4, Aes, Eac3, Tp, PtsDts, Pes, Pcr, Mpa</li><li>• Added support for HEVC chroma format and bit depth fields</li><li>• Fix for possible incorrect resolution for HEVC</li><li>• Fix for exception during HEVC header parse (seen on Ubuntu 14.04 linker v2.24)</li><li>• Fix for possible AAC header parse issue</li><li>• Fix problem with memory deallocation PES packed (Windows only)</li></ul>

v1.4.2.22	2018.06.13	<ul style="list-style-type: none"><li>• Added support for timestamped TS packets</li><li>• Added support for Teletext streams</li><li>• Added support for Visual Studio 2017 (VC15)</li><li>• Fix wrong m_Pid and m_TableId in P2_PCR_ACCURACY callback indication</li><li>• Fix for failing DecodeFromTable method of DtTableBat class</li><li>• Fix for extracted resolution error for HEVC due to conformance_window_flag</li><li>• Fix for DtapiTs::BitPtr::GetBits() Assertion</li><li>• Fix problem with DEBUG version and assertions in Linux</li></ul>
-----------	------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

v1.4.1.21	2017.07.18	<ul style="list-style-type: none"> <li>• Added support for AC-4 audio in a DVB-stream</li> </ul>
v1.3.3.19	2017.01.23	<ul style="list-style-type: none"> <li>• Fix for rare crash in TsInfoImpl::RemovePidRef</li> </ul>
v1.3.2.18	2016.08.24	<ul style="list-style-type: none"> <li>• Fix for incomplete packet extraction in corner cases involving stuffing tables</li> <li>• Add AddPesPacketCallback(int Pid, DtPesCallback Callback) to .NET wrapper</li> </ul>
v1.3.1.17	2016.06.06	<ul style="list-style-type: none"> <li>• Reported H.264 frame rate was double the actual frame rate in some cases</li> <li>• Detect SMPTE 302M AES3 audio</li> </ul>
v1.3.0.16	2016.02.24	<ul style="list-style-type: none"> <li>• Add support for Visual Studio 2015 (VC14)</li> </ul>
v1.2.5.15	2014.09.18	<ul style="list-style-type: none"> <li>• H.264 / AVC: adjust frame size for cropping</li> <li>• Add DtStructuredTable subclasses and DtDesc* classes to .NET wrapper</li> <li>• Fix for possible crash while parsing E-AC3 audio</li> <li>• DtTsData::m_Tables was not accesible via .NET wrapper</li> </ul>

v1.2.4.14	2014.06.25	<ul style="list-style-type: none"> <li>• Descriptor parsing could fail if there were extended descriptors</li> <li>• Parse HEVC streams</li> <li>• Add MSVC 2013 libraries</li> </ul>
v1.2.3.13	2014.03.03	<ul style="list-style-type: none"> <li>• Add TR 101 290 support to .NET wrapper</li> <li>• Add MSVC 2012 libraries</li> <li>• Add code for TS file analyzer as example</li> <li>• Add example on how to use callbacks</li> </ul>
v1.2.2.12	2013.08.14	<ul style="list-style-type: none"> <li>• 1 and 3 segments TMCC files were not recognized correctly</li> <li>• Fix infinite loop if a stream contains a DVB-T2 descriptor</li> <li>• DtTs*Input classes called Lock() to make sure the DtTsInfo was not changed but never unlocked the object again</li> </ul>
v1.2.0.10	2013.04.29	<ul style="list-style-type: none"> <li>• TR 101 290 support (priority 1 and 2).</li> <li>• Add support for more descriptors via new DtDesc* classes.</li> <li>• Compute DtTsData::m_NitTsRate also from a terrestrial delivery system descriptor.</li> <li>• DtVideoEsInfo::m_HorzSize and DtVideoEsInfo::m_VertSize could contain garbage in some rare cases, even though DtVideoEsInfo::m_Mask indicated those fields were valid.</li> <li>• Don't parse PAT/CAT when they are on the wrong PID.</li> </ul>
Copyright © 2012-2024 DekTec Digital Video BV. All rights reserved.		<ul style="list-style-type: none"> <li>• DtTablePat::DecodeFromSection swapped m_Pid and m_ServiceId.</li> </ul>

v1.1.0.7	2013.02.25	<ul style="list-style-type: none"><li>• Add new classes that help to extract descriptor information from raw table data.</li><li>• Changes to DtSubTableId to make it easier to filter only on certain values.</li><li>• Fix for small memory leak when delete-ing a DtTsInfo object.</li></ul>
v1.0.4.5	2012.12.06	<ul style="list-style-type: none"><li>• Crash in case a file had sections with more than 4093 bytes.</li></ul>
V1.0.3.3	2012.11.13	<ul style="list-style-type: none"><li>• 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.</li><li>• String conversion between different charsets was broken on Linux.</li></ul>
V1.0.2.2	2012.09.10	<ul style="list-style-type: none"><li>• First external release</li></ul>

## Chapter 3

# Module Index

### 3.1 Modules

Here is a list of all modules:

Structured information from binary descriptors . . . . .	<a href="#">23</a>
Structured information from binary tables . . . . .	<a href="#">25</a>





## Chapter 4

# Namespace Index

### 4.1 Namespace List

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

<a href="#">DtapiTs</a>	All DTAPITS code lives in the <a href="#">DtapiTs</a> namespace . . . . .	<a href="#">27</a>
-------------------------	---------------------------------------------------------------------------	--------------------



## Chapter 5

# Hierarchical Index

### 5.1 Class Hierarchy

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

DtapiTs::DtPes::DataBuffer . . . . .	51
DtapiTs::DtAudioEsInfo . . . . .	52
DtapiTs::DtBitrate . . . . .	55
DtapiTs::DtBitrateSettings . . . . .	55
DtapiTs::DtCallback1< TArg1 > . . . . .	56
DtapiTs::DtCallback2< TArg1, TArg2 > . . . . .	57
DtapiTs::DtCallback3< TArg1, TArg2, TArg3 > . . . . .	57
DtapiTs::DtCaSystem . . . . .	58
DtapiTs::DtDescDvbAc3 . . . . .	59
DtapiTs::DtDescDvbCDelivery . . . . .	60
DtapiTs::DtDescDvbComponent . . . . .	60
DtapiTs::DtDescDvbDataBroadcast . . . . .	61
DtapiTs::DtDescDvbDataBroadcastId . . . . .	62
DtapiTs::DtDescDvbLinkage . . . . .	63
DtapiTs::DtDescDvbLocalTimeOffset . . . . .	64
DtapiTs::DtDescDvbMultilingualComponent . . . . .	65
DtapiTs::DtDescDvbNetworkName . . . . .	66
DtapiTs::DtDescDvbSDelivery . . . . .	66
DtapiTs::DtDescDvbService . . . . .	67
DtapiTs::DtDescDvbServiceList . . . . .	68
DtapiTs::DtDescDvbSubtitling . . . . .	69
DtapiTs::DtDescDvbTDelivery . . . . .	69
DtapiTs::DtDescDvbTeletext . . . . .	70
DtapiTs::DtDescMpegCa . . . . .	71
DtapiTs::DtDescMpegLanguage . . . . .	72
DtapiTs::DtDescMpegPrivDataIndicator . . . . .	72
DtapiTs::DtDescMpegRegistration . . . . .	73
DtapiTs::DtDescMpegVideoStream . . . . .	73
DtapiTs::DtDescPrivLcn . . . . .	75
DtapiTs::DtDescriptor . . . . .	75
DtapiTs::DtDvbCNitInfo . . . . .	77
DtapiTs::DtDvbShModInfo . . . . .	77
DtapiTs::DtDvbShNitInfo . . . . .	79
DtapiTs::DtDvbShOfdmInfo . . . . .	79
DtapiTs::DtDvbShTdmInfo . . . . .	80
DtapiTs::DtDvbSNitInfo . . . . .	81
DtapiTs::DtDvbT2CellInfo . . . . .	82
DtapiTs::DtDvbT2NitInfo . . . . .	82

DtapiTs::DtDvbT2SubCellInfo . . . . .	83
DtapiTs::DtDvbTNitInfo . . . . .	84
DtapiTs::DtEsInfoBase . . . . .	85
DtapiTs::DtAudioEsInfo2 . . . . .	54
DtapiTs::DtAacEsInfo . . . . .	51
DtapiTs::DtAc3EsInfo . . . . .	52
DtapiTs::DtAc4EsInfo . . . . .	52
DtapiTs::DtEac3EsInfo . . . . .	85
DtapiTs::DtHeAacEsInfo . . . . .	86
DtapiTs::DtMpaEsInfo . . . . .	87
DtapiTs::DtJitterPoint . . . . .	86
DtapiTs::DtDescPrivLcn::DtLogicalChannelNumber . . . . .	86
DtapiTs::DtPcr . . . . .	87
DtapiTs::DtPcrInfo . . . . .	88
DtapiTs::DtPes . . . . .	89
DtapiTs::DtPidInfo . . . . .	90
DtapiTs::DtTablePat::DtProgramMapping . . . . .	92
DtapiTs::DtPtsDts . . . . .	92
DtapiTs::DtServiceComponentInfo . . . . .	93
DtapiTs::DtServiceInfo . . . . .	94
DtapiTs::DtStructuredTable . . . . .	96
DtapiTs::DtTableBat . . . . .	100
DtapiTs::DtTableCat . . . . .	101
DtapiTs::DtTableNit . . . . .	102
DtapiTs::DtTablePat . . . . .	103
DtapiTs::DtTablePmt . . . . .	104
DtapiTs::DtTableSdt . . . . .	105
DtapiTs::DtTableTdt . . . . .	108
DtapiTs::DtTableTot . . . . .	108
DtapiTs::DtSubTableId . . . . .	97
DtapiTs::DtTable . . . . .	98
DtapiTs::DtTableBatInner . . . . .	101
DtapiTs::DtTableNitInner . . . . .	103
DtapiTs::DtTablePmtInner . . . . .	104
DtapiTs::DtTableSdtInner . . . . .	106
DtapiTs::DtTableSection . . . . .	107
DtapiTs::DtTimeDiff . . . . .	109
DtapiTs::DtTimestamp . . . . .	110
DtapiTs::DtTp . . . . .	111
DtapiTs::DtTr101290 . . . . .	112
DtapiTs::DtTr101290Error . . . . .	112
DtapiTs::DtTsData . . . . .	113
DtapiTs::DtTsInfo . . . . .	116
DtapiTs::DtTsInfoInput . . . . .	122
DtapiTs::DtTsPacketInput . . . . .	124
DtapiTs::DtTsTimestampedPacketInput . . . . .	125
DtapiTs::DtTsTransparentInput . . . . .	126
DtapiTs::DtTsLib . . . . .	123
DtapiTs::DtVideoAspectRatio . . . . .	126
DtapiTs::DtVideoEsInfo . . . . .	127
DtapiTs::DtVideoEsAvclInfo . . . . .	127
DtapiTs::DtDescDvbLinkage::EventLinkage . . . . .	129
DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage . . . . .	130
DtapiTs::DtEsInfoBase::InfoField< T > . . . . .	131
DtapiTs::DtDescMpegLanguage::LangCode . . . . .	131
DtapiTs::DtDescDvbLocalTimeOffset::LocalTimeOffset . . . . .	131
DtapiTs::DtDescDvbLinkage::MobileHandOverInfo . . . . .	131

DtapiTs::DtDescDvbServiceList::ServiceListItem . . . . .	132
DtapiTs::DtDescDvbSubtitling::Subtitling . . . . .	132
DtapiTs::DtDescDvbTeletext::Teletext . . . . .	133



## Chapter 6

# Class Index

### 6.1 Class List

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

<a href="#">DtapiTs::DtPes::DataBuffer</a>	51
<a href="#">DtapiTs::DtAacEsInfo</a>	51
<a href="#">DtapiTs::DtAc3EsInfo</a>	52
<a href="#">DtapiTs::DtAc4EsInfo</a>	52
<a href="#">DtapiTs::DtAudioEsInfo</a>	
Information about an audio elementary stream extracted from PES packets	52
<a href="#">DtapiTs::DtAudioEsInfo2</a>	54
<a href="#">DtapiTs::DtBitrate</a>	
Some statistics about the bitrate of a PID, service or transport stream	55
<a href="#">DtapiTs::DtBitrateSettings</a>	
Settings for the bitrate measurement sliding window	55
<a href="#">DtapiTs::DtCallback1&lt; TArg1 &gt;</a>	
Helper class to store a function pointer to a function with one argument and a pointer to the associated data	56
<a href="#">DtapiTs::DtCallback2&lt; TArg1, TArg2 &gt;</a>	
Helper class to store a function pointer to a function with two arguments and a pointer to the associated data	57
<a href="#">DtapiTs::DtCallback3&lt; TArg1, TArg2, TArg3 &gt;</a>	
Helper class to store a function pointer to a function with 3 arguments and a pointer to the associated data	57
<a href="#">DtapiTs::DtCaSystem</a>	
Description of a conditional access system	58
<a href="#">DtapiTs::DtDescDvbAc3</a>	
Parsed information from the DVB AC-3 descriptor	59
<a href="#">DtapiTs::DtDescDvbCDelivery</a>	
Parsed information from the DVB cable delivery system descriptor	60
<a href="#">DtapiTs::DtDescDvbComponent</a>	
Parsed information from the DVB component descriptor	60
<a href="#">DtapiTs::DtDescDvbDataBroadcast</a>	
Parsed information from the DVB data broadcast descriptor	61
<a href="#">DtapiTs::DtDescDvbDataBroadcastId</a>	
Parsed information from the DVB data broadcast id descriptor	62
<a href="#">DtapiTs::DtDescDvbLinkage</a>	
Parsed information from the DVB linkage descriptor	63
<a href="#">DtapiTs::DtDescDvbLocalTimeOffset</a>	
Parsed information from the DVB local time offset descriptor	64
<a href="#">DtapiTs::DtDescDvbMultilingualComponent</a>	
Parsed information from the DVB multilingual component descriptor	65

<a href="#">DtapiTs::DtDescDvbNetworkName</a>	
Parsed information from the DVB network name descriptor . . . . .	66
<a href="#">DtapiTs::DtDescDvbSDelivery</a>	
Parsed information from the DVB satellite delivery system descriptor . . . . .	66
<a href="#">DtapiTs::DtDescDvbService</a>	
Parsed information from the DVB service descriptor . . . . .	67
<a href="#">DtapiTs::DtDescDvbServiceList</a>	
Parsed information from the DVB service list descriptor . . . . .	68
<a href="#">DtapiTs::DtDescDvbSubtitling</a>	
Parsed information from the DVB subtitling descriptor . . . . .	69
<a href="#">DtapiTs::DtDescDvbTDelivery</a>	
Parsed information from the DVB terrestrial delivery system descriptor . . . . .	69
<a href="#">DtapiTs::DtDescDvbTeletext</a>	
Parsed information from the DVB teletext descriptor . . . . .	70
<a href="#">DtapiTs::DtDescMpegCa</a>	
Parsed information from the conditional access descriptor . . . . .	71
<a href="#">DtapiTs::DtDescMpegLanguage</a>	
Parsed information from the ISO 639 language descriptor . . . . .	72
<a href="#">DtapiTs::DtDescMpegPrivDataIndicator</a>	
Parsed information from private data indicator descriptor . . . . .	72
<a href="#">DtapiTs::DtDescMpegRegistration</a>	
Parsed information from the registration descriptor . . . . .	73
<a href="#">DtapiTs::DtDescMpegVideoStream</a>	
Parsed information from the video stream descriptor . . . . .	73
<a href="#">DtapiTs::DtDescPrivLcn</a>	
Class that can be used to parse logical channel numbers from a (list of) descriptors . . . . .	75
<a href="#">DtapiTs::DtDescriptor</a>	
DtDescriptor represents a single descriptor within one of the tables . . . . .	75
<a href="#">DtapiTs::DtDvbCNitInfo</a>	
DVB-C delivery system information as extracted from the NIT . . . . .	77
<a href="#">DtapiTs::DtDvbShModInfo</a>	
DVB-SH modulation info . . . . .	77
<a href="#">DtapiTs::DtDvbShNitInfo</a>	
DVB-SH delivery system information as extracted from the NIT . . . . .	79
<a href="#">DtapiTs::DtDvbShOfdmInfo</a>	
DVB-SH OFDM modulation info . . . . .	79
<a href="#">DtapiTs::DtDvbShTdmInfo</a>	
DVB-SH TDM modulation info . . . . .	80
<a href="#">DtapiTs::DtDvbSNitInfo</a>	
DVB-S delivery system information as extracted from the NIT . . . . .	81
<a href="#">DtapiTs::DtDvbT2CellInfo</a>	
Information about a DVB-T2 cell . . . . .	82
<a href="#">DtapiTs::DtDvbT2NitInfo</a>	
DVB-T2 delivery system information as extracted from the NIT . . . . .	82
<a href="#">DtapiTs::DtDvbT2SubCellInfo</a>	
Information about a single DVB-T2 sub-cell . . . . .	83
<a href="#">DtapiTs::DtDvbTNitInfo</a>	
DVB-T delivery system information as extracted from the NIT . . . . .	84
<a href="#">DtapiTs::DtEac3EsInfo</a>	
. . . . .	85
<a href="#">DtapiTs::DtEsInfoBase</a>	
. . . . .	85
<a href="#">DtapiTs::DtHeAacEsInfo</a>	
. . . . .	86
<a href="#">DtapiTs::DtJitterPoint</a>	
. . . . .	86
<a href="#">DtapiTs::DtDescPrivLcn::DtLogicalChannelNumber</a>	
. . . . .	86
<a href="#">DtapiTs::DtMpaEsInfo</a>	
. . . . .	87
<a href="#">DtapiTs::DtPcr</a>	
Class representing PCR timestamp: . . . . .	87
<a href="#">DtapiTs::DtPcrInfo</a>	
Some statistics about the bitrate of a PID, service or transport stream . . . . .	88



<a href="#">DtapiTs::DtPes</a>	89
<a href="#">DtapiTs::DtPidInfo</a>	
Class that contains general information about one PID	90
<a href="#">DtapiTs::DtTablePat::DtProgramMapping</a>	
ServiceId to PMT Pid mapping for a single service	92
<a href="#">DtapiTs::DtPtsDts</a>	
Class representing PTS/DTS timestamp:	92
<a href="#">DtapiTs::DtServiceComponentInfo</a>	
Information about a service component	93
<a href="#">DtapiTs::DtServiceInfo</a>	
Information about a service	94
<a href="#">DtapiTs::DtStructuredTable</a>	
Base class for all structured table classes	96
<a href="#">DtapiTs::DtSubTableId</a>	
Unique identifier for each sub-table	97
<a href="#">DtapiTs::DtTable</a>	
Binary representation of an SI table	98
<a href="#">DtapiTs::DtTableBat</a>	
Structured version of the raw data contained in a Bouquet association table	100
<a href="#">DtapiTs::DtTableBatInner</a>	
This class holds all descriptors of one sub-loop of a BAT table	101
<a href="#">DtapiTs::DtTableCat</a>	
Class that can be used to parse all descriptors in the Conditional Access Table	101
<a href="#">DtapiTs::DtTableNit</a>	
Structured version of the raw data contained in a Network Information Table	102
<a href="#">DtapiTs::DtTableNitInner</a>	
This class holds all descriptors of one sub-loop of the NIT table	103
<a href="#">DtapiTs::DtTablePat</a>	
Structured version of the raw data contained in a Program Association Table	103
<a href="#">DtapiTs::DtTablePmt</a>	
Class that can be used to parse all descriptors in the Program Map Table	104
<a href="#">DtapiTs::DtTablePmtInner</a>	
This class holds all descriptors of compoment in a PMT table	104
<a href="#">DtapiTs::DtTableSdt</a>	
Structured version of the raw data contained in a Service description table	105
<a href="#">DtapiTs::DtTableSdtInner</a>	
This class holds all descriptors of one sub-loop of the SDT table	106
<a href="#">DtapiTs::DtTableSection</a>	
Binary representation of one section of an SI table	107
<a href="#">DtapiTs::DtTableTdt</a>	
Structured version of the raw data contained in a Time and Date section	108
<a href="#">DtapiTs::DtTableTot</a>	
Structured version of the raw data contained in a Time Offset Table	108
<a href="#">DtapiTs::DtTimeDiff</a>	
Difference between two timestamps	109
<a href="#">DtapiTs::DtTimestamp</a>	
Abstract timestamp, do not rely on the internal representation	110
<a href="#">DtapiTs::DtTp</a>	111
<a href="#">DtapiTs::DtTr101290</a>	
Base class for TR 101 290 support	112
<a href="#">DtapiTs::DtTr101290Error</a>	
Information about a single TR 101 290 indicator	112
<a href="#">DtapiTs::DtTsData</a>	
Main <a href="#">DtapiTs</a> class that contains all data extracted from a transport stream	113
<a href="#">DtapiTs::DtTsInfo</a>	
Main <a href="#">DtapiTs</a> class that is used for setting parameters, adding callbacks, passing in the transport stream buffer (via a <a href="#">DtTsInfoInput</a> object) and finally reading back the results	116

<a href="#">DtapiTs::DtTsInfoInput</a>	Abstract base class for splitting (large) input buffers into timestamped packets . . . . .	122
<a href="#">DtapiTs::DtTsLib</a>	Main <a href="#">DtapiTs</a> class used to create instances of the analysis classes . . . . .	123
<a href="#">DtapiTs::DtTsPacketInput</a>	Input class that handles a transport stream without any timestamps . . . . .	124
<a href="#">DtapiTs::DtTsTimestampedPacketInput</a>	Input class that handles a transport stream packets with timestamps as delivered by DTAPI when a <a href="#">DtInpChannel</a> is set to <a href="#">DTAPI_RXMODE_TIMESTAMP32</a> . . . . .	125
<a href="#">DtapiTs::DtTsTransparentInput</a>	Input class that handles timestamped transparent packets as delivered by DTAPI when a <a href="#">DtInpChannel</a> is set to <a href="#">DTAPI_RXMODE_STTRP DTAPI_RXMODE_TIMESTAMP32</a> . . . . .	126
<a href="#">DtapiTs::DtVideoAspectRatio</a>		126
<a href="#">DtapiTs::DtVideoEsAvclInfo</a>		127
<a href="#">DtapiTs::DtVideoEsInfo</a>	Information about a video elementary stream extracted from PES packets . . . . .	127
<a href="#">DtapiTs::DtDescDvbLinkage::EventLinkage</a>		129
<a href="#">DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage</a>		130
<a href="#">DtapiTs::DtEsInfoBase::InfoField&lt; T &gt;</a>		131
<a href="#">DtapiTs::DtDescMpegLanguage::LangCode</a>		131
<a href="#">DtapiTs::DtDescDvbLocalTimeOffset::LocalTimeOffset</a>		131
<a href="#">DtapiTs::DtDescDvbLinkage::MobileHandOverInfo</a>		131
<a href="#">DtapiTs::DtDescDvbServiceList::ServiceListItem</a>		132
<a href="#">DtapiTs::DtDescDvbSubtitling::Subtitling</a>		132
<a href="#">DtapiTs::DtDescDvbTeletext::Teletext</a>		133

## 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](#)  
*Description 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 representation 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 representation 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 delivery 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 forward 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 coarse type of service, useful for quickly identifying the content type.*

- enum [DtShBandwidth](#) {  
[DT\\_SH\\_BANDWIDTH\\_8MHZ](#),  
[DT\\_SH\\_BANDWIDTH\\_7MHZ](#),  
[DT\\_SH\\_BANDWIDTH\\_6MHZ](#),  
[DT\\_SH\\_BANDWIDTH\\_5MHZ](#),  
[DT\\_SH\\_BANDWIDTH\\_1\\_7MHZ](#),  
[DT\\_SH\\_BANDWIDTH\\_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 [DtapTs](#) 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\\_MPEGH\\_AUDIO](#) = 0x0000002D,
  - [DT\\_STREAMTYPE\\_JPEGXS\\_VIDEO](#) = 0x00000032,
  - [DT\\_STREAMTYPE\\_AVS\\_VIDEO](#) = 0x00000042,
  - [DT\\_STREAMTYPE\\_AVS2\\_VIDEO](#) = 0x000000D2,
  - [DT\\_STREAMTYPE\\_DVB\\_AVS3\\_VIDEO](#) = 0x000000D4,
  - [DT\\_STREAMTYPE\\_AVS3\\_VIDEO](#) = 0x000000D6,
  - [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\\_SCTE\\_35](#) = 0x00000086,
  - [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 occurred.  
***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 delivery 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 forward 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 coarse 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\_BANDWIDTH\_8MHZ*** Bandwidth is 8Mhz.  
***DT\_SH\_BANDWIDTH\_7MHZ*** Bandwidth is 7Mhz.  
***DT\_SH\_BANDWIDTH\_6MHZ*** Bandwidth is 6Mhz.  
***DT\_SH\_BANDWIDTH\_5MHZ*** Bandwidth is 5Mhz.  
***DT\_SH\_BANDWIDTH\_1\_7MHZ*** Bandwidth is 1.7Mhz.  
***DT\_SH\_BANDWIDTH\_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*** HEVC/H.256 Video.

***DT\_STREAMTYPE\_MPEGH\_AUDIO*** MPEG-H Audio Stream.

***DT\_STREAMTYPE\_JPEGXS\_VIDEO*** JPEGXS video.

***DT\_STREAMTYPE\_AVS\_VIDEO*** NOT THE OFFICIAL STREAM-TYPE (YET)

***DT\_STREAMTYPE\_AVS2\_VIDEO*** NOT THE OFFICIAL STREAM-TYPE (YET)

***DT\_STREAMTYPE\_DVB\_AVS3\_VIDEO*** DVB AVS3 Video // Source: DVB bluebook A001r21 4.1.6.1.

***DT\_STREAMTYPE\_AVS3\_VIDEO*** AVS3 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\_SCTE\_35*** User private range, but commonly used for SCTE-35.

***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

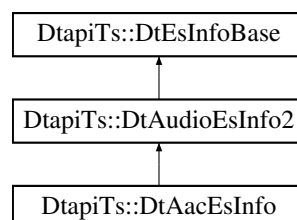
- **DataBuffer** (uint8\_t \*pData=NULL, int Size=0)
- **DataBuffer** (const [DataBuffer](#) &Oth)
- uint8\_t \* **Data** () const
- bool **IsOwnedByUs** () const
- [DataBuffer](#) & **operator=** (const [DataBuffer](#) &Oth)
- uint8\_t & **operator[]** (size\_t n) const
- int **Resize** (int Size)
- int **Size** () const

#### Protected Attributes

- uint8\_t \* **m\_pData**
- int **m\_Size**

### 9.2 DtapiTs::DtAacEsInfo Class Reference

Inheritance diagram for DtapiTs::DtAacEsInfo:



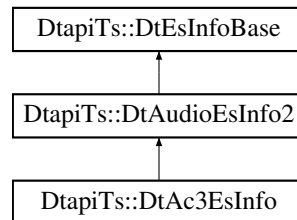
#### Public Member Functions

- virtual void **Clear** ()

## Additional Inherited Members

### 9.3 DtapiTs::DtAc3EsInfo Class Reference

Inheritance diagram for DtapiTs::DtAc3EsInfo:



## Public Member Functions

- virtual void **Clear** ()

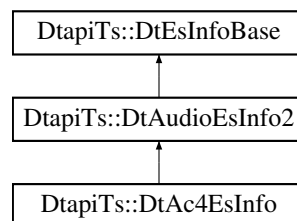
## Public Attributes

- [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](#)

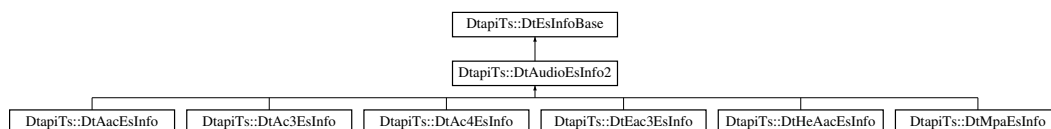
- static const unsigned int **c\_SamplerateField** = [SAMPLERATE\\_FIELD](#)
- static const unsigned int [CH\\_MODE\\_FIELD](#) = 0x00000004  
*Bit set in m\_Mask when m\_ChMode is valid.*
- static const unsigned int [COPYRIGHT\\_FIELD](#) = 0x00000008  
*Bit set in m\_Mask when m\_Copyrighted is valid.*
- 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

<i>TimeSliceTime,:</i>	The duration of each time slice.
<i>NumTime-Slices,:</i>	The number of timeslices in the window.
<i>NumAvgValues,:</i>	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 <DTAPIITS.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 <DTAPIITS.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

Description of a conditional access system.

```
#include <DTAPIITS.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

Description 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 identifier 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 eastern 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 meaningful when m\_IsDvbS2 is true.

#### 9.22.2.4 int DtapiTs::DtDescDvbSDelivery::m\_WestEastFlag

Indication of whether the satellite position is in the eastern 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 meaningful 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 <DTAPIITS.h>
```

### Public Attributes

- int [FeclInner](#)  
*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\\_Plpld](#)  
*Identifier for this data PLP, unique in the DVB-T2 system.*
- [int m\\_T2SystemId](#)  
*Identifier for the current system, unique in the DVB-T2 network.*
- [bool m\\_TfsArrangement](#)  
*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_Plpld`

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, unique in the DVB-T2 network.

## 9.43 DtapiTs::DtDvbT2SubCellInfo Class Reference

Information about a single DVB-T2 sub-cell.

```
#include <DTAPIITS.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 <DTAPIITS.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.

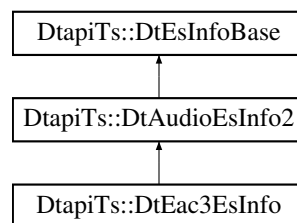
## 9.44.2.5 int DtapiTs::DtDvbTNitInfo::TransmissionMode

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`:



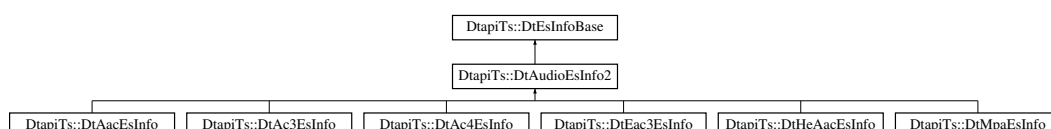
## Public Member Functions

- virtual void **Clear** ()

## Additional Inherited Members

## 9.46 DtapiTs::DtEsInfoBase Class Reference

Inheritance diagram for `DtapiTs::DtEsInfoBase`:



## Classes

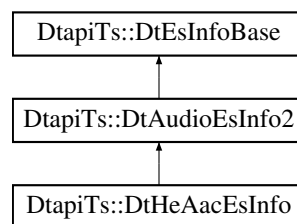
- class [InfoField](#)

## 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

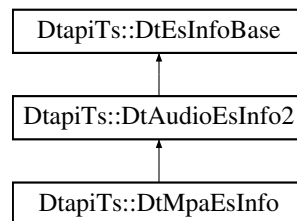
Should this channel be visible to the end-user?

#### 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

<i>Type</i> ,:	The table type to test for.
----------------	-----------------------------

Returns

: True if the table has occurred 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=PTS\_DTS\_INVALID, \_\_int64 DtsFlat=PTS\_DTS\_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 if 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**

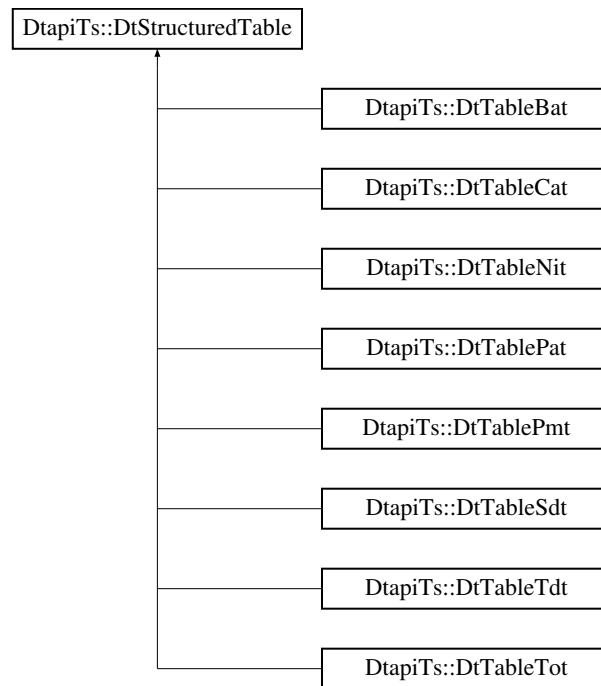
Simplified service type based on m\_StreamType in the various PIDs.

**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 then 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 then 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 meaningful 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 representation of an SI table.

```
#include <DTAPIITS.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 representation 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

<i>Other,:</i>	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

<i>Other,:</i>	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.

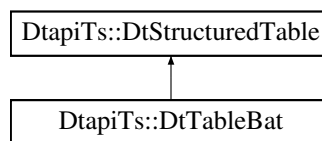
This should increment on every update.

## 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

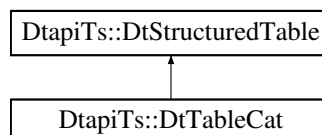
This class holds all descriptors of one sub-loop of a BAT table.

## 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

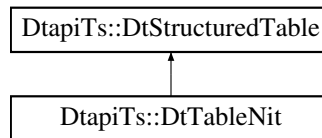
Class that can be used to parse all descriptors in the Conditional Access Table.

## 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

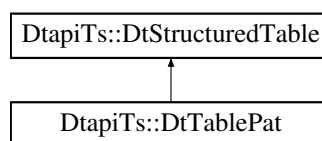
This class holds all descriptors of one sub-loop of the NIT table.

## 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

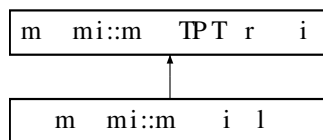
Structured version of the raw data contained in a Program Association Table.

## 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

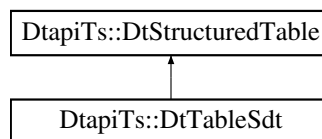
This class holds all descriptors of compoment in a PMT table.

## 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 SvclId )

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 representation 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 existing [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 representation 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 existing [DtTableSection](#) object.

#### Parameters

<i>Other,:</i>	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

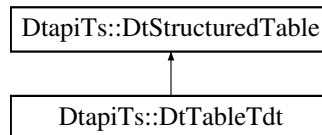
<i>Other,:</i>	the object to copy.
----------------	---------------------

## 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

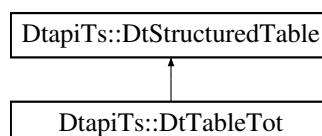
Structured version of the raw data contained in a Time and Date section.

## 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](#) objects can 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 occurred.*
- 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 occurred last.*

- `double m_Time`

*Duration for this error.*

- `DtTimestamp m_Timestamp`

*Timestamp this error occurred 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 occurred.

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 <DTAPIITS.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 terrestrial 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 have 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 have been found in the TMCC data in the stream.

#### Parameters

out	<i>Pars</i>	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]

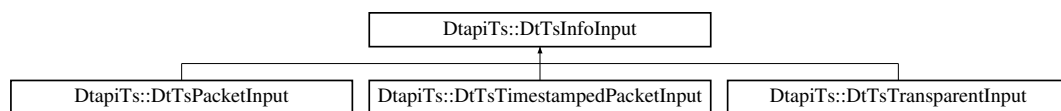
Used to enable/disable the table cache for certain tables.

## 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)  
*Remove 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 function returns NULL if the [DtapiTs](#) header file does not belong to the object file.

Examples:

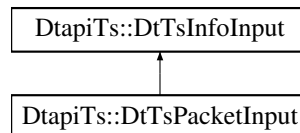
[example3.cpp](#).

## 9.84 DtapiTs::DtTsPacketInput Class Reference

Input class that handles a transport stream without any timestamps.

```
#include <DTAPIITS.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

<i>Info</i> ,:	The <a href="#">DtTsInfo</a> object that will be used for the actual analysis. May be <code>NULL</code> , but in that case you have to provide a valid object by calling <code>SetTsInfoObject</code> before using <code>NewData</code> .
<i>Bitrate</i> ,:	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:

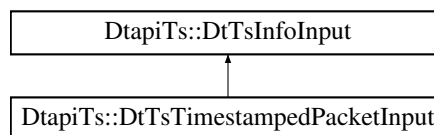
[example3.cpp](#).

## 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 ClickFreq)  
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 ClickFreq ) [static]`

Create a new [DtTsTimestampedPacketInput](#) object.

## Parameters

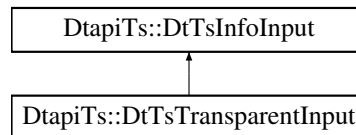
<i>Info</i> ,:	The <a href="#">DtTsInfo</a> object that will be used for the actual analysis. May be <code>NULL</code> , but in that case you have to provide a valid object by calling <code>SetTsInfoObject</code> before using <code>NewData</code> .
<i>ClickFreq</i> ,:	The hardware clock frequency. Use <code>DtDevice::GetRefClkFreq</code> 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 <DTAPIITS.h>
```

Inheritance diagram for DtapiTs::DtTsTransparentInput:



### Static Public Member Functions

- static [DtTsTransparentInput](#) \* [CreateInstance](#) ([DtTsInfo](#) \*Info, int ClickFreq)  
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 ClickFreq )  
[static]

Create a new [DtTsTransparentInput](#) object.

##### Parameters

<i>Info,:</i>	The <a href="#">DtTsInfo</a> 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.
<i>ClickFreq,:</i>	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**

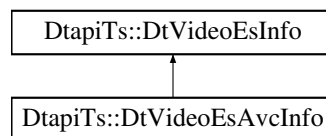
*Horizontal.*

### 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**

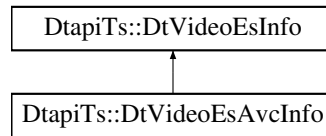
### Additional Inherited Members

## 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.*

### 9.99.1 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_FIFLOAD (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 purely 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 modifying 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 *****/
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_HorizSize, 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 occurred 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

- AddJitterCallback
  - DtapiTs::DtTsInfo, [119](#)
- AddNewSectionCallback
  - DtapiTs::DtTsInfo, [119](#)
- AddPesPacketCallback
  - DtapiTs::DtTsInfo, [119](#), [120](#)
- AddTableChangedCallback
  - DtapiTs::DtTsInfo, [120](#)
- AddTableTimeoutCallback
  - DtapiTs::DtTsInfo, [120](#)
- Bandwith
  - DtapiTs::DtDvbTNitInfo, [84](#)
- CodeRateHpStream
  - DtapiTs::DtDvbTNitInfo, [84](#)
- Constellation
  - DtapiTs::DtDvbCNitInfo, [77](#)
- CreateDtTsInfoInstance
  - DtapiTs::DtTsLib, [124](#)
- CreateInstance
  - DtapiTs::DtTsPacketInput, [124](#)
  - DtapiTs::DtTsTimestampedPacketInput, [125](#)
  - DtapiTs::DtTsTransparentInput, [126](#)
- DT\_AACOBJTYPE\_AAC\_LC
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_AAC\_MAIN
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_AAC\_SCALABLE
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_AAC\_SSR
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_ALGO\_SYNTHESIS\_AND\_AUDIO\_FXE
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_ALS
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_CELP
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_DST
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_ER\_AAC\_ELD
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_ER\_AAC\_LC
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_ER\_AAC\_LD
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_ER\_AAC\_LTP
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_ER\_AAC\_SCALABLE
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_ER\_BASC
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_ER\_CELP
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_ER\_HILN
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_ER\_HVXC
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_ER\_PARAMETRIC
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_ER\_TWINVQ
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_ESCAPE
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_GENERAL\_MIDI
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_HVXC
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_LAYER1
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_LAYER2
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_LAYER3
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_LTP
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_MAIN\_SYNTHETIC
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_MPEG\_SURROUND
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_NULL
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_PS
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_RESV1
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_RESV2
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_RESV3
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_SBR
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_SLS
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_SLS\_NON\_CORE
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_SMR\_MAIN

- DtapiTs, [40](#)
- DT\_AACOBJTYPE\_SMR\_SIMPLE
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_SSC
  - DtapiTs, [40](#)
- DT\_AACOBJTYPE\_TTSI
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_TWINVQ
  - DtapiTs, [39](#)
- DT\_AACOBJTYPE\_WAVETABLE\_SYNTHETIC
  - DtapiTs, [39](#)
- DT\_AACPROFILE\_LOW\_COMPLEXITY
  - DtapiTs, [40](#)
- DT\_AACPROFILE\_MAIN
  - DtapiTs, [40](#)
- DT\_AACPROFILE\_SCALABLE\_SAMPLING\_RATE
  - DtapiTs, [40](#)
- DT\_AACPROFILE\_UNKNOWN
  - DtapiTs, [40](#)
- DT\_AUDIOMODE\_AAC\_CF
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AAC\_CF\_LF\_RF
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AAC\_CF\_LF\_RF\_RS
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AAC\_CF\_RF\_LF\_LR\_RR
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AAC\_CF\_RF\_LF\_LR\_RR\_FLF
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AAC\_CR\_RF\_LF\_ROF\_LOF\_LR\_-RR\_FLF
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AAC\_LF\_RF
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AC3\_CENTER
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AC3\_CH1CH2
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AC3\_LCR
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AC3\_LCR\_SL\_SR
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AC3\_LCRS
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AC3\_LR
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AC3\_LR\_SL\_SR
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_AC3\_LRS
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_DUAL
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_JOINT\_STEREO
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_MONO
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_STEREO
  - DtapiTs, [41](#)
- DT\_AUDIOMODE\_UNKNOWN
  - DtapiTs, [41](#)
- DT\_CCTYPE\_EIA608
  - DtapiTs, [41](#)
- DT\_CCTYPE\_EIA708
  - DtapiTs, [41](#)
- DT\_CHROMAFORMAT\_420
  - DtapiTs, [49](#)
- DT\_CHROMAFORMAT\_422
  - DtapiTs, [49](#)
- DT\_CHROMAFORMAT\_424
  - DtapiTs, [50](#)
- DT\_CHROMAFORMAT\_444
  - DtapiTs, [50](#)
- DT\_CHROMAFORMAT\_INVALID
  - DtapiTs, [49](#)
- DT\_CHROMAFORMAT\_MONO
  - DtapiTs, [50](#)
- DT\_DELIVERYSYSTEM\_CABLE
  - DtapiTs, [42](#)
- DT\_DELIVERYSYSTEM\_INVALID
  - DtapiTs, [42](#)
- DT\_DELIVERYSYSTEM\_SATELLITE
  - DtapiTs, [42](#)
- DT\_DELIVERYSYSTEM\_SH
  - DtapiTs, [42](#)
- DT\_DELIVERYSYSTEM\_T2
  - DtapiTs, [42](#)
- DT\_DELIVERYSYSTEM\_TERRESTRIAL
  - DtapiTs, [42](#)
- DT\_ERR\_B\_ALL
  - DtapiTs, [48](#)
- DT\_ERR\_B\_P1
  - DtapiTs, [48](#)
- DT\_ERR\_B\_P2
  - DtapiTs, [48](#)
- DT\_ERR\_B\_P3
  - DtapiTs, [48](#)
- DT\_ERR\_P1\_CONTINUITY\_COUNTER
  - DtapiTs, [48](#)
- DT\_ERR\_P1\_PAT\_2
  - DtapiTs, [48](#)
- DT\_ERR\_P1\_PID
  - DtapiTs, [48](#)
- DT\_ERR\_P1\_PMT\_2
  - DtapiTs, [48](#)
- DT\_ERR\_P1\_SYNC\_BYTE
  - DtapiTs, [48](#)
- DT\_ERR\_P1\_TS\_SYNC\_LOSS
  - DtapiTs, [48](#)
- DT\_ERR\_P2\_CAT
  - DtapiTs, [48](#)
- DT\_ERR\_P2\_CRC
  - DtapiTs, [48](#)
- DT\_ERR\_P2\_PCR\_ACCURACY
  - DtapiTs, [48](#)
- DT\_ERR\_P2\_PCR\_DISC\_IND
  - DtapiTs, [48](#)

DT_ERR_P2_PCR_REPETITION	DT_POLAR_LIN_VERT
DtapiTs, <a href="#">48</a>	DtapiTs, <a href="#">43</a>
DT_ERR_P2_PTS	DT_SCRAMBLING_EVEN
DtapiTs, <a href="#">48</a>	DtapiTs, <a href="#">43</a>
DT_ERR_P2_TRANSPORT	DT_SCRAMBLING_NONE
DtapiTs, <a href="#">48</a>	DtapiTs, <a href="#">43</a>
DT_ERR_P3_BUFFER	DT_SCRAMBLING_ODD
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">43</a>
DT_ERR_P3_DATA_DELAY	DT_SCRAMBLING_RESERVED
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">43</a>
DT_ERR_P3_EIT_ACTUAL	DT_SERVICETYPE_RADIO
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">43</a>
DT_ERR_P3_EIT_OTHER	DT_SERVICETYPE_TELEVISION
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">43</a>
DT_ERR_P3_EIT_PF	DT_SERVICETYPE_UNKNOWN
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">43</a>
DT_ERR_P3_EMPTY_BUFFER	DT_SH_BANDWIDTH_1_7MHZ
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">44</a>
DT_ERR_P3_NIT_ACTUAL	DT_SH_BANDWIDTH_5MHZ
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">44</a>
DT_ERR_P3_NIT_OTHER	DT_SH_BANDWIDTH_6MHZ
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">44</a>
DT_ERR_P3_RST	DT_SH_BANDWIDTH_7MHZ
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">44</a>
DT_ERR_P3_SDT_ACTUAL	DT_SH_BANDWIDTH_8MHZ
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">44</a>
DT_ERR_P3_SDT_OTHER	DT_SH_BANDWIDTH_UNK
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">44</a>
DT_ERR_P3_SI_REPETITION	DT_SH_CODERATE_1_2
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">44</a>
DT_ERR_P3_TDT	DT_SH_CODERATE_1_2_COMPL
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">44</a>
DT_ERR_P3_UNREFERENCED_PID	DT_SH_CODERATE_1_3
DtapiTs, <a href="#">49</a>	DtapiTs, <a href="#">44</a>
DT_FECOUTER_NONE	DT_SH_CODERATE_1_3_COMPL
DtapiTs, <a href="#">42</a>	DtapiTs, <a href="#">44</a>
DT_FECOUTER_RS_204_188	DT_SH_CODERATE_1_4
DtapiTs, <a href="#">42</a>	DtapiTs, <a href="#">44</a>
DT_FECOUTER_UNK	DT_SH_CODERATE_1_5
DtapiTs, <a href="#">42</a>	DtapiTs, <a href="#">44</a>
DT_MPA_LAYER_1	DT_SH_CODERATE_2_3
DtapiTs, <a href="#">42</a>	DtapiTs, <a href="#">44</a>
DT_MPA_LAYER_2	DT_SH_CODERATE_2_3_COMPL
DtapiTs, <a href="#">42</a>	DtapiTs, <a href="#">44</a>
DT_MPA_LAYER_3	DT_SH_CODERATE_2_5
DtapiTs, <a href="#">42</a>	DtapiTs, <a href="#">44</a>
DT_MPA_VERSION_1	DT_SH_CODERATE_2_5_COMPL
DtapiTs, <a href="#">43</a>	DtapiTs, <a href="#">44</a>
DT_MPA_VERSION_2	DT_SH_CODERATE_2_7
DtapiTs, <a href="#">43</a>	DtapiTs, <a href="#">44</a>
DT_MPA_VERSION_2_5	DT_SH_CODERATE_2_9
DtapiTs, <a href="#">43</a>	DtapiTs, <a href="#">44</a>
DT_POLAR_CIRC_LEFT	DT_SH_CODERATE_RESERVED
DtapiTs, <a href="#">43</a>	DtapiTs, <a href="#">44</a>
DT_POLAR_CIRC_RIGHT	DT_SH_GUARDINTERVAL_1_16
DtapiTs, <a href="#">43</a>	DtapiTs, <a href="#">42</a>
DT_POLAR_LIN_HOR	DT_SH_GUARDINTERVAL_1_32
DtapiTs, <a href="#">43</a>	DtapiTs, <a href="#">42</a>

- DT\_SH\_GUARDINTERVAL\_1\_4  
DtapiTs, [42](#)
- DT\_SH\_GUARDINTERVAL\_1\_8  
DtapiTs, [42](#)
- DT\_SH\_MOD\_16APSK  
DtapiTs, [44](#)
- DT\_SH\_MOD\_8PSK  
DtapiTs, [44](#)
- DT\_SH\_MOD\_OFDM  
DtapiTs, [44](#)
- DT\_SH\_MOD\_QPSK  
DtapiTs, [44](#)
- DT\_SH\_MOD\_TDM  
DtapiTs, [44](#)
- DT\_SH\_MOD\_UNK  
DtapiTs, [44](#)
- DT\_SH\_ROLLOFF\_15  
DtapiTs, [43](#)
- DT\_SH\_ROLLOFF\_25  
DtapiTs, [43](#)
- DT\_SH\_ROLLOFF\_35  
DtapiTs, [43](#)
- DT\_SH\_ROLLOFF\_UNK  
DtapiTs, [43](#)
- DT\_SH\_TRANSMODE\_1K  
DtapiTs, [49](#)
- DT\_SH\_TRANSMODE\_2K  
DtapiTs, [49](#)
- DT\_SH\_TRANSMODE\_4K  
DtapiTs, [49](#)
- DT\_SH\_TRANSMODE\_8K  
DtapiTs, [49](#)
- DT\_STANDARDMODE\_ATSC  
DtapiTs, [45](#)
- DT\_STANDARDMODE\_DVB  
DtapiTs, [45](#)
- DT\_STANDARDMODE\_DVB\_RCS  
DtapiTs, [45](#)
- DT\_STANDARDMODE\_UNK  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_13818\_6\_SDP  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_13818\_6\_TA  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_13818\_6\_TB  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_13818\_6\_TC  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_13818\_6\_TD  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_AAC  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_ATSC\_AC3  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_ATSC\_DIGICYPH2  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_ATSC\_EAC3  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_AUX  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_AVC\_VIDEO  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_AVS2\_VIDEO  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_AVS3\_VIDEO  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_AVS\_VIDEO  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_DSMCC  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_DVB\_AC3  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_DVB\_AC4  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_DVB\_AVS3\_VIDEO  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_DVB\_DATA\_CAROUSEL  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_DVB\_EAC3  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_DVB\_INT  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_DVB\_MPE  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_DVB\_TELETEXT  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_H\_222\_1  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_HEAAC  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_HEVC\_VIDEO  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_INVALID  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_IPMP\_STREAM  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_IPMP\_STREAM\_MP2  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_J2K\_VIDEO  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_JPEGXS\_VIDEO  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_METADATA\_DC  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_METADATA\_OC  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_METADATA\_PES  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_METADATA\_SDP  
DtapiTs, [46](#)
- DT\_STREAMTYPE\_METADATA\_SECT  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_MHEG  
DtapiTs, [45](#)
- DT\_STREAMTYPE\_MPEG1\_AUDIO  
DtapiTs, [45](#)

DT_STREAMTYPE_MPEG1_VIDEO DtapiTs, <a href="#">45</a>	DT_TABLETYPE_DVB_BAT DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_MPEG2_AUDIO DtapiTs, <a href="#">45</a>	DT_TABLETYPE_DVB_DIT DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_MPEG2_VIDEO DtapiTs, <a href="#">45</a>	DT_TABLETYPE_DVB_EITACT DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_MPEG4_PES DtapiTs, <a href="#">45</a>	DT_TABLETYPE_DVB_EITACTS DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_MPEG4_SECT DtapiTs, <a href="#">45</a>	DT_TABLETYPE_DVB_EITOTH DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_MPEG4_VIDEO DtapiTs, <a href="#">45</a>	DT_TABLETYPE_DVB_EITOTHS DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_MPEGH_AUDIO DtapiTs, <a href="#">46</a>	DT_TABLETYPE_DVB_INT DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_PRIV_DATA DtapiTs, <a href="#">45</a>	DT_TABLETYPE_DVB_NITACT DtapiTs, <a href="#">46</a>
DT_STREAMTYPE_PRIV_SECTIONS DtapiTs, <a href="#">45</a>	DT_TABLETYPE_DVB_NITOTH DtapiTs, <a href="#">46</a>
DT_STREAMTYPE_SCTE_35 DtapiTs, <a href="#">46</a>	DT_TABLETYPE_DVB_RCS_CMT DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_SCTE_ISOCHR_DATA DtapiTs, <a href="#">46</a>	DT_TABLETYPE_DVB_RCS_FCT DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_SCTE_SUBTITLE DtapiTs, <a href="#">46</a>	DT_TABLETYPE_DVB_RCS_LL_FEC_PDT DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_SMPTE_AES3 DtapiTs, <a href="#">46</a>	DT_TABLETYPE_DVB_RCS_PCRPP DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_UNKNOWN DtapiTs, <a href="#">45</a>	DT_TABLETYPE_DVB_RCS_RMT DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_VC1_VIDEO DtapiTs, <a href="#">46</a>	DT_TABLETYPE_DVB_RCS_SCT DtapiTs, <a href="#">47</a>
DT_STREAMTYPE_WM9_AUDIO DtapiTs, <a href="#">46</a>	DT_TABLETYPE_DVB_RCS_SPT DtapiTs, <a href="#">47</a>
DT_T2MISO_MISO DtapiTs, <a href="#">42</a>	DT_TABLETYPE_DVB_RCS_TBTP DtapiTs, <a href="#">47</a>
DT_T2MISO_SISO DtapiTs, <a href="#">42</a>	DT_TABLETYPE_DVB_RCS_TCT DtapiTs, <a href="#">47</a>
DT_T2MISO_UNK DtapiTs, <a href="#">42</a>	DT_TABLETYPE_DVB_RCS_TIM DtapiTs, <a href="#">47</a>
DT_TABLETYPE_ATSC_CVCT DtapiTs, <a href="#">47</a>	DT_TABLETYPE_DVB_RCS_TMST DtapiTs, <a href="#">47</a>
DT_TABLETYPE_ATSC_DCCSCT DtapiTs, <a href="#">47</a>	DT_TABLETYPE_DVB_RNT DtapiTs, <a href="#">47</a>
DT_TABLETYPE_ATSC_DCCT DtapiTs, <a href="#">47</a>	DT_TABLETYPE_DVB_RST DtapiTs, <a href="#">47</a>
DT_TABLETYPE_ATSC_EIT DtapiTs, <a href="#">47</a>	DT_TABLETYPE_DVB_SDTACT DtapiTs, <a href="#">47</a>
DT_TABLETYPE_ATSC_ETT DtapiTs, <a href="#">47</a>	DT_TABLETYPE_DVB_SDTOTH DtapiTs, <a href="#">47</a>
DT_TABLETYPE_ATSC_MGT DtapiTs, <a href="#">47</a>	DT_TABLETYPE_DVB_SIT DtapiTs, <a href="#">47</a>
DT_TABLETYPE_ATSC_RRT DtapiTs, <a href="#">47</a>	DT_TABLETYPE_DVB_ST DtapiTs, <a href="#">47</a>
DT_TABLETYPE_ATSC_STT DtapiTs, <a href="#">47</a>	DT_TABLETYPE_DVB_TDT DtapiTs, <a href="#">47</a>
DT_TABLETYPE_ATSC_TVCT DtapiTs, <a href="#">47</a>	DT_TABLETYPE_DVB_TOT DtapiTs, <a href="#">47</a>
DT_TABLETYPE_CAT DtapiTs, <a href="#">46</a>	DT_TABLETYPE_ECM DtapiTs, <a href="#">46</a>

- DT\_TABLETYPE\_EMM
  - DtapiTs, [46](#)
- DT\_TABLETYPE\_PAT
  - DtapiTs, [46](#)
- DT\_TABLETYPE\_PMT
  - DtapiTs, [46](#)
- DT\_TABLETYPE\_TSDT
  - DtapiTs, [46](#)
- DT\_TABLETYPE\_UNKNOWN
  - DtapiTs, [46](#)
- DT\_WEFLAG\_EAST
  - DtapiTs, [50](#)
- DT\_WEFLAG\_WEST
  - DtapiTs, [50](#)
- DTAPITS\_E\_CRC\_MISMATCH
  - DtapiTs, [40](#)
- DTAPITS\_E\_DESC\_NOT\_FOUND
  - DtapiTs, [41](#)
- DTAPITS\_E\_DESC\_TOO\_SHORT
  - DtapiTs, [41](#)
- DTAPITS\_E\_INVALID\_BUF
  - DtapiTs, [40](#)
- DTAPITS\_E\_INVALID\_DESC
  - DtapiTs, [40](#)
- DTAPITS\_E\_INVALID\_DESC\_LEN
  - DtapiTs, [40](#)
- DTAPITS\_E\_INVALID\_FIELD
  - DtapiTs, [41](#)
- DTAPITS\_E\_INVALID\_PDS
  - DtapiTs, [40](#)
- DTAPITS\_E\_INVALID\_TAG
  - DtapiTs, [40](#)
- DTAPITS\_E\_MISSING\_DATA
  - DtapiTs, [40](#)
- DTAPITS\_E\_NOT\_SUPPORTED
  - DtapiTs, [41](#)
- DTAPITS\_E\_PARSE\_ERROR
  - DtapiTs, [41](#)
- DTAPITS\_E\_TABLEID\_MISMATCH
  - DtapiTs, [40](#)
- DTAPITS\_OK
  - DtapiTs, [40](#)
- DTAPITS\_RESULT
  - DtapiTs, [40](#)
- DecodeFromTable
  - DtapiTs::DtStructuredTable, [97](#)
- DtAacObjType
  - DtapiTs, [39](#)
- DtAacProfile
  - DtapiTs, [40](#)
- DtAtscCcType
  - DtapiTs, [41](#)
- DtAudioMode
  - DtapiTs, [41](#)
- DtBitrateSettings
  - DtapiTs::DtBitrateSettings, [56](#)
- DtDeliverySystem
  - DtapiTs, [41](#)
- DtDvbT2MisoMode
  - DtapiTs, [42](#)
- DtFecOuter
  - DtapiTs, [42](#)
- DtGuardInterval
  - DtapiTs, [42](#)
- DtJitterCallback
  - DtapiTs::DtTsInfo, [118](#)
- DtMpaLayer
  - DtapiTs, [42](#)
- DtMpaVersion
  - DtapiTs, [42](#)
- DtPacketCallback
  - DtapiTs::DtTsInfo, [118](#)
- DtPesCallback
  - DtapiTs::DtTsInfo, [118](#)
- DtPolarization
  - DtapiTs, [43](#)
- DtRollOff
  - DtapiTs, [43](#)
- DtScrambling
  - DtapiTs, [43](#)
- DtSectionCallback
  - DtapiTs::DtTsInfo, [118](#)
- DtServiceType
  - DtapiTs, [43](#)
- DtShBandwidth
  - DtapiTs, [43](#)
- DtShCodeRate
  - DtapiTs, [44](#)
- DtShModMode
  - DtapiTs, [44](#)
- DtShModType
  - DtapiTs, [44](#)
- DtStandardMode
  - DtapiTs, [44](#)
- DtStreamType
  - DtapiTs, [45](#)
- DtTable
  - DtapiTs::DtTable, [99](#)
- DtTableCallback
  - DtapiTs::DtTsInfo, [119](#)
- DtTableSection
  - DtapiTs::DtTableSection, [107](#)
- DtTableTimeoutCallback
  - DtapiTs::DtTsInfo, [119](#)
- DtTableType
  - DtapiTs, [46](#)
- DtTr101290Bitmask
  - DtapiTs, [47](#)
- DtTr101290Indicator
  - DtapiTs, [48](#)
- DtTransmissionMode
  - DtapiTs, [49](#)
- DtVideoChromaFormat
  - DtapiTs, [49](#)
- DtWeFlag
  - DtapiTs, [50](#)

- DtapiTs, 27
- DT\_AACOBJTYPE\_AAC\_LC, 39
  - DT\_AACOBJTYPE\_AAC\_MAIN, 39
  - DT\_AACOBJTYPE\_AAC\_SCALABLE, 39
  - DT\_AACOBJTYPE\_AAC\_SSR, 39
  - DT\_AACOBJTYPE\_ALGO\_SYNTHESIS\_AND\_AUDIO\_FXE, 39
  - DT\_AACOBJTYPE\_ALS, 40
  - DT\_AACOBJTYPE\_CELP, 39
  - DT\_AACOBJTYPE\_DST, 40
  - DT\_AACOBJTYPE\_ER\_AAC\_ELD, 40
  - DT\_AACOBJTYPE\_ER\_AAC\_LC, 39
  - DT\_AACOBJTYPE\_ER\_AAC\_LD, 39
  - DT\_AACOBJTYPE\_ER\_AAC\_LTP, 39
  - DT\_AACOBJTYPE\_ER\_AAC\_SCALABLE, 39
  - DT\_AACOBJTYPE\_ER\_BASC, 39
  - DT\_AACOBJTYPE\_ER\_CELP, 40
  - DT\_AACOBJTYPE\_ER\_HILN, 40
  - DT\_AACOBJTYPE\_ER\_HVXC, 40
  - DT\_AACOBJTYPE\_ER\_PARAMETRIC, 40
  - DT\_AACOBJTYPE\_ER\_TWINVQ, 39
  - DT\_AACOBJTYPE\_ESCAPE, 40
  - DT\_AACOBJTYPE\_GENERAL\_MIDI, 39
  - DT\_AACOBJTYPE\_HVXC, 39
  - DT\_AACOBJTYPE\_LAYER1, 40
  - DT\_AACOBJTYPE\_LAYER2, 40
  - DT\_AACOBJTYPE\_LAYER3, 40
  - DT\_AACOBJTYPE\_LTP, 39
  - DT\_AACOBJTYPE\_MAIN\_SYNTHETIC, 39
  - DT\_AACOBJTYPE\_MPEG\_SURROUND, 40
  - DT\_AACOBJTYPE\_NULL, 39
  - DT\_AACOBJTYPE\_PS, 40
  - DT\_AACOBJTYPE\_RESV1, 39
  - DT\_AACOBJTYPE\_RESV2, 39
  - DT\_AACOBJTYPE\_RESV3, 39
  - DT\_AACOBJTYPE\_SBR, 39
  - DT\_AACOBJTYPE\_SLS, 40
  - DT\_AACOBJTYPE\_SLS\_NON\_CORE, 40
  - DT\_AACOBJTYPE\_SMR\_MAIN, 40
  - DT\_AACOBJTYPE\_SMR\_SIMPLE, 40
  - DT\_AACOBJTYPE\_SSC, 40
  - DT\_AACOBJTYPE\_TTSI, 39
  - DT\_AACOBJTYPE\_TWINVQ, 39
  - DT\_AACOBJTYPE\_WAVETABLE\_SYNTHETIC, 39
  - DT\_AACPROFILE\_LOW\_COMPLEXITY, 40
  - DT\_AACPROFILE\_MAIN, 40
  - DT\_AACPROFILE\_SCALABLE\_SAMPLING\_RATE, 40
  - DT\_AACPROFILE\_UNKNOWN, 40
  - DT\_AUDIOMODE\_AAC\_CF, 41
  - DT\_AUDIOMODE\_AAC\_CF\_LF\_RF, 41
  - DT\_AUDIOMODE\_AAC\_CF\_LF\_RF\_RS, 41
  - DT\_AUDIOMODE\_AAC\_CF\_RF\_LF\_LR\_RS, 41
  - DT\_AUDIOMODE\_AAC\_CF\_RF\_LF\_LR\_RS\_FLF, 41
  - DT\_AUDIOMODE\_AAC\_CR\_RF\_LF\_ROF\_LOF\_LR\_RS\_FLF, 41
  - DT\_AUDIOMODE\_AAC\_LF\_RF, 41
  - DT\_AUDIOMODE\_AC3\_CENTER, 41
  - DT\_AUDIOMODE\_AC3\_CH1CH2, 41
  - DT\_AUDIOMODE\_AC3\_LCR, 41
  - DT\_AUDIOMODE\_AC3\_LCR\_SL\_SR, 41
  - DT\_AUDIOMODE\_AC3\_LCRS, 41
  - DT\_AUDIOMODE\_AC3\_LR, 41
  - DT\_AUDIOMODE\_AC3\_LR\_SL\_SR, 41
  - DT\_AUDIOMODE\_AC3\_LRS, 41
  - DT\_AUDIOMODE\_DUAL, 41
  - DT\_AUDIOMODE\_JOINT\_STEREO, 41
  - DT\_AUDIOMODE\_MONO, 41
  - DT\_AUDIOMODE\_STEREO, 41
  - DT\_AUDIOMODE\_UNKNOWN, 41
  - DT\_CCTYPE\_EIA608, 41
  - DT\_CCTYPE\_EIA708, 41
  - DT\_CHROMAFORMAT\_420, 49
  - DT\_CHROMAFORMAT\_422, 49
  - DT\_CHROMAFORMAT\_424, 50
  - DT\_CHROMAFORMAT\_444, 50
  - DT\_CHROMAFORMAT\_INVALID, 49
  - DT\_CHROMAFORMAT\_MONO, 50
  - DT\_DELIVERYSYSTEM\_CABLE, 42
  - DT\_DELIVERYSYSTEM\_INVALID, 42
  - DT\_DELIVERYSYSTEM\_SATELLITE, 42
  - DT\_DELIVERYSYSTEM\_SH, 42
  - DT\_DELIVERYSYSTEM\_T2, 42
  - DT\_DELIVERYSYSTEM\_TERRESTRIAL, 42
  - DT\_ERR\_B\_ALL, 48
  - DT\_ERR\_B\_P1, 48
  - DT\_ERR\_B\_P2, 48
  - DT\_ERR\_B\_P3, 48
  - DT\_ERR\_P1\_CONTINUITY\_COUNTER, 48
  - DT\_ERR\_P1\_PAT\_2, 48
  - DT\_ERR\_P1\_PID, 48
  - DT\_ERR\_P1\_PMT\_2, 48
  - DT\_ERR\_P1\_SYNC\_BYTE, 48
  - DT\_ERR\_P1\_TS\_SYNC\_LOSS, 48
  - DT\_ERR\_P2\_CAT, 48
  - DT\_ERR\_P2\_CRC, 48
  - DT\_ERR\_P2\_PCR\_ACCURACY, 48
  - DT\_ERR\_P2\_PCR\_DISC\_IND, 48
  - DT\_ERR\_P2\_PCR\_REPETITION, 48
  - DT\_ERR\_P2\_PTS, 48
  - DT\_ERR\_P2\_TRANSPORT, 48
  - DT\_ERR\_P3\_BUFFER, 49
  - DT\_ERR\_P3\_DATA\_DELAY, 49
  - DT\_ERR\_P3\_EIT\_ACTUAL, 49
  - DT\_ERR\_P3\_EIT\_OTHER, 49
  - DT\_ERR\_P3\_EIT\_PF, 49
  - DT\_ERR\_P3\_EMPTY\_BUFFER, 49
  - DT\_ERR\_P3\_NIT\_ACTUAL, 49
  - DT\_ERR\_P3\_NIT\_OTHER, 49
  - DT\_ERR\_P3\_RST, 49
  - DT\_ERR\_P3\_SDT\_ACTUAL, 49
  - DT\_ERR\_P3\_SDT\_OTHER, 49
  - DT\_ERR\_P3\_SI\_REPETITION, 49
  - DT\_ERR\_P3\_TDT, 49

- DT\_ERR\_P3\_UNREFERENCED\_PID, 49
- DT\_FECOUTER\_NONE, 42
- DT\_FECOUTER\_RS\_204\_188, 42
- DT\_FECOUTER\_UNK, 42
- DT\_MPA\_LAYER\_1, 42
- DT\_MPA\_LAYER\_2, 42
- DT\_MPA\_LAYER\_3, 42
- DT\_MPA\_VERSION\_1, 43
- DT\_MPA\_VERSION\_2, 43
- DT\_MPA\_VERSION\_2\_5, 43
- DT\_POLAR\_CIRC\_LEFT, 43
- DT\_POLAR\_CIRC\_RIGHT, 43
- DT\_POLAR\_LIN\_HOR, 43
- DT\_POLAR\_LIN\_VERT, 43
- DT\_SCRAMBLING\_EVEN, 43
- DT\_SCRAMBLING\_NONE, 43
- DT\_SCRAMBLING\_ODD, 43
- DT\_SCRAMBLING\_RESERVED, 43
- DT\_SERVICETYPE\_RADIO, 43
- DT\_SERVICETYPE\_TELEVISION, 43
- DT\_SERVICETYPE\_UNKNOWN, 43
- DT\_SH\_BANDWIDTH\_1\_7MHZ, 44
- DT\_SH\_BANDWIDTH\_5MHZ, 44
- DT\_SH\_BANDWIDTH\_6MHZ, 44
- DT\_SH\_BANDWIDTH\_7MHZ, 44
- DT\_SH\_BANDWIDTH\_8MHZ, 44
- DT\_SH\_BANDWIDTH\_UNK, 44
- DT\_SH\_CODERATE\_1\_2, 44
- DT\_SH\_CODERATE\_1\_2\_COMPL, 44
- DT\_SH\_CODERATE\_1\_3, 44
- DT\_SH\_CODERATE\_1\_3\_COMPL, 44
- DT\_SH\_CODERATE\_1\_4, 44
- DT\_SH\_CODERATE\_1\_5, 44
- DT\_SH\_CODERATE\_2\_3, 44
- DT\_SH\_CODERATE\_2\_3\_COMPL, 44
- DT\_SH\_CODERATE\_2\_5, 44
- DT\_SH\_CODERATE\_2\_5\_COMPL, 44
- DT\_SH\_CODERATE\_2\_7, 44
- DT\_SH\_CODERATE\_2\_9, 44
- DT\_SH\_CODERATE\_RESERVED, 44
- DT\_SH\_GUARDINTERVAL\_1\_16, 42
- DT\_SH\_GUARDINTERVAL\_1\_32, 42
- DT\_SH\_GUARDINTERVAL\_1\_4, 42
- DT\_SH\_GUARDINTERVAL\_1\_8, 42
- DT\_SH\_MOD\_16APSK, 44
- DT\_SH\_MOD\_8PSK, 44
- DT\_SH\_MOD\_OFDM, 44
- DT\_SH\_MOD\_QPSK, 44
- DT\_SH\_MOD\_TDM, 44
- DT\_SH\_MOD\_UNK, 44
- DT\_SH\_ROLLOFF\_15, 43
- DT\_SH\_ROLLOFF\_25, 43
- DT\_SH\_ROLLOFF\_35, 43
- DT\_SH\_ROLLOFF\_UNK, 43
- DT\_SH\_TRANSMODE\_1K, 49
- DT\_SH\_TRANSMODE\_2K, 49
- DT\_SH\_TRANSMODE\_4K, 49
- DT\_SH\_TRANSMODE\_8K, 49
- DT\_STANDARDMODE\_ATSC, 45
- DT\_STANDARDMODE\_DVB, 45
- DT\_STANDARDMODE\_DVB\_RCS, 45
- DT\_STANDARDMODE\_UNK, 45
- DT\_STREAMTYPE\_13818\_6\_SDP, 45
- DT\_STREAMTYPE\_13818\_6\_TA, 45
- DT\_STREAMTYPE\_13818\_6\_TB, 45
- DT\_STREAMTYPE\_13818\_6\_TC, 45
- DT\_STREAMTYPE\_13818\_6\_TD, 45
- DT\_STREAMTYPE\_AAC, 45
- DT\_STREAMTYPE\_ATSC\_AC3, 46
- DT\_STREAMTYPE\_ATSC\_DIGICYPH2, 46
- DT\_STREAMTYPE\_ATSC\_EAC3, 46
- DT\_STREAMTYPE\_AUX, 45
- DT\_STREAMTYPE\_AVC\_VIDEO, 46
- DT\_STREAMTYPE\_AVS2\_VIDEO, 46
- DT\_STREAMTYPE\_AVS3\_VIDEO, 46
- DT\_STREAMTYPE\_AVS\_VIDEO, 46
- DT\_STREAMTYPE\_DSMCC, 45
- DT\_STREAMTYPE\_DVB\_AC3, 46
- DT\_STREAMTYPE\_DVB\_AC4, 46
- DT\_STREAMTYPE\_DVB\_AVS3\_VIDEO, 46
- DT\_STREAMTYPE\_DVB\_DATA\_CAROUSEL, 46
- DT\_STREAMTYPE\_DVB\_EAC3, 46
- DT\_STREAMTYPE\_DVB\_INT, 46
- DT\_STREAMTYPE\_DVB\_MPE, 46
- DT\_STREAMTYPE\_DVB\_TELETEXT, 46
- DT\_STREAMTYPE\_H\_222\_1, 45
- DT\_STREAMTYPE\_HEAAC, 45
- DT\_STREAMTYPE\_HEVC\_VIDEO, 46
- DT\_STREAMTYPE\_INVALID, 45
- DT\_STREAMTYPE\_IPMP\_STREAM, 46
- DT\_STREAMTYPE\_IPMP\_STREAM\_MP2, 46
- DT\_STREAMTYPE\_J2K\_VIDEO, 46
- DT\_STREAMTYPE\_JPEGXS\_VIDEO, 46
- DT\_STREAMTYPE\_METADATA\_DC, 45
- DT\_STREAMTYPE\_METADATA\_OC, 45
- DT\_STREAMTYPE\_METADATA\_PES, 45
- DT\_STREAMTYPE\_METADATA\_SDP, 46
- DT\_STREAMTYPE\_METADATA\_SECT, 45
- DT\_STREAMTYPE\_MHEG, 45
- DT\_STREAMTYPE\_MPEG1\_AUDIO, 45
- DT\_STREAMTYPE\_MPEG1\_VIDEO, 45
- DT\_STREAMTYPE\_MPEG2\_AUDIO, 45
- DT\_STREAMTYPE\_MPEG2\_VIDEO, 45
- DT\_STREAMTYPE\_MPEG4\_PES, 45
- DT\_STREAMTYPE\_MPEG4\_SECT, 45
- DT\_STREAMTYPE\_MPEG4\_VIDEO, 45
- DT\_STREAMTYPE\_MPEGH\_AUDIO, 46
- DT\_STREAMTYPE\_PRIV\_DATA, 45
- DT\_STREAMTYPE\_PRIV\_SECTIONS, 45
- DT\_STREAMTYPE\_SCTE\_35, 46
- DT\_STREAMTYPE\_SCTE\_ISOCHR\_DATA, 46
- DT\_STREAMTYPE\_SCTE\_SUBTITLE, 46
- DT\_STREAMTYPE\_SMPTE\_AES3, 46
- DT\_STREAMTYPE\_UNKNOWN, 45
- DT\_STREAMTYPE\_VC1\_VIDEO, 46
- DT\_STREAMTYPE\_WM9\_AUDIO, 46



- DT\_T2MISO\_MISO, [42](#)
- DT\_T2MISO\_SISO, [42](#)
- DT\_T2MISO\_UNK, [42](#)
- DT\_TABLETYPE\_ATSC\_CVCT, [47](#)
- DT\_TABLETYPE\_ATSC\_DCCSCT, [47](#)
- DT\_TABLETYPE\_ATSC\_DCCT, [47](#)
- DT\_TABLETYPE\_ATSC\_EIT, [47](#)
- DT\_TABLETYPE\_ATSC\_ETT, [47](#)
- DT\_TABLETYPE\_ATSC\_MGT, [47](#)
- DT\_TABLETYPE\_ATSC\_RRT, [47](#)
- DT\_TABLETYPE\_ATSC\_STT, [47](#)
- DT\_TABLETYPE\_ATSC\_TVCT, [47](#)
- DT\_TABLETYPE\_CAT, [46](#)
- DT\_TABLETYPE\_DVB\_BAT, [47](#)
- DT\_TABLETYPE\_DVB\_DIT, [47](#)
- DT\_TABLETYPE\_DVB\_EITACT, [47](#)
- DT\_TABLETYPE\_DVB\_EITACTS, [47](#)
- DT\_TABLETYPE\_DVB\_EITOTH, [47](#)
- DT\_TABLETYPE\_DVB\_EITOTHS, [47](#)
- DT\_TABLETYPE\_DVB\_INT, [47](#)
- DT\_TABLETYPE\_DVB\_NITACT, [46](#)
- DT\_TABLETYPE\_DVB\_NITOTH, [46](#)
- DT\_TABLETYPE\_DVB\_RCS\_CMT, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_FCT, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_LL\_FEC\_PDT, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_PCRPP, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_RMT, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_SCT, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_SPT, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_TBTP, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_TCT, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_TIM, [47](#)
- DT\_TABLETYPE\_DVB\_RCS\_TMST, [47](#)
- DT\_TABLETYPE\_DVB\_RNT, [47](#)
- DT\_TABLETYPE\_DVB\_RST, [47](#)
- DT\_TABLETYPE\_DVB\_SDTACT, [47](#)
- DT\_TABLETYPE\_DVB\_SDTOTH, [47](#)
- DT\_TABLETYPE\_DVB\_SIT, [47](#)
- DT\_TABLETYPE\_DVB\_ST, [47](#)
- DT\_TABLETYPE\_DVB\_TDT, [47](#)
- DT\_TABLETYPE\_DVB\_TOT, [47](#)
- DT\_TABLETYPE\_ECM, [46](#)
- DT\_TABLETYPE\_EMM, [46](#)
- DT\_TABLETYPE\_PAT, [46](#)
- DT\_TABLETYPE\_PMT, [46](#)
- DT\_TABLETYPE\_TSMT, [46](#)
- DT\_TABLETYPE\_UNKNOWN, [46](#)
- DT\_WEFLAG\_EAST, [50](#)
- DT\_WEFLAG\_WEST, [50](#)
- DTAPITS\_E\_CRC\_MISMATCH, [40](#)
- DTAPITS\_E\_DESC\_NOT\_FOUND, [41](#)
- DTAPITS\_E\_DESC\_TOO\_SHORT, [41](#)
- DTAPITS\_E\_INVALID\_BUF, [40](#)
- DTAPITS\_E\_INVALID\_DESC, [40](#)
- DTAPITS\_E\_INVALID\_DESC\_LEN, [40](#)
- DTAPITS\_E\_INVALID\_FIELD, [41](#)
- DTAPITS\_E\_INVALID\_PDS, [40](#)
- DTAPITS\_E\_INVALID\_TAG, [40](#)
- DTAPITS\_E\_MISSING\_DATA, [40](#)
- DTAPITS\_E\_NOT\_SUPPORTED, [41](#)
- DTAPITS\_E\_PARSE\_ERROR, [41](#)
- DTAPITS\_E\_TABLEID\_MISMATCH, [40](#)
- DTAPITS\_OK, [40](#)
- DTAPITS\_RESULT, [40](#)
- DtAacObjType, [39](#)
- DtAacProfile, [40](#)
- DtAtscCcType, [41](#)
- DtAudioMode, [41](#)
- DtDeliverySystem, [41](#)
- DtDvbT2MisoMode, [42](#)
- DtFecOuter, [42](#)
- DtGuardInterval, [42](#)
- DtMpaLayer, [42](#)
- DtMpaVersion, [42](#)
- DtPolarization, [43](#)
- DtRollOff, [43](#)
- DtScrambling, [43](#)
- DtServiceType, [43](#)
- DtShBandwidth, [43](#)
- DtShCodeRate, [44](#)
- DtShModMode, [44](#)
- DtShModType, [44](#)
- DtStandardMode, [44](#)
- DtStreamType, [45](#)
- DtTableType, [46](#)
- DtTr101290Bitmask, [47](#)
- DtTr101290Indicator, [48](#)
- DtTransmissionMode, [49](#)
- DtVideoChromaFormat, [49](#)
- DtWeFlag, [50](#)
- DtapiTs::DtAacEsInfo, [51](#)
- DtapiTs::DtAc3EsInfo, [52](#)
- DtapiTs::DtAc4EsInfo, [52](#)
- DtapiTs::DtAudioEsInfo, [52](#)
- DtapiTs::DtAudioEsInfo2, [54](#)
- DtapiTs::DtBitrate, [55](#)
- DtapiTs::DtBitrateSettings, [55](#)
  - DtBitrateSettings, [56](#)
  - m\_NumAvgValues, [56](#)
- DtapiTs::DtCaSystem, [58](#)
  - m\_CaSystemId, [58](#)
- DtapiTs::DtCallback1< TArg1 >, [56](#)
- DtapiTs::DtCallback2< TArg1, TArg2 >, [57](#)
- DtapiTs::DtCallback3< TArg1, TArg2, TArg3 >, [57](#)
- DtapiTs::DtDescDvbAc3, [59](#)
  - m\_Asvc, [59](#)
  - m\_Bsid, [59](#)
  - m\_ComponentType, [59](#)
  - m\_MainId, [60](#)
- DtapiTs::DtDescDvbCDelivery, [60](#)
- DtapiTs::DtDescDvbComponent, [60](#)
  - m\_ComponentTag, [61](#)
- DtapiTs::DtDescDvbDataBroadcast, [61](#)
  - m\_ComponentTag, [62](#)
  - m\_DataBroadcastId, [62](#)
  - m\_SelectorBytes, [62](#)

- DtapiTs::DtDescDvbDataBroadcastId, 62
  - m\_DataBroadcastId, 63
  - m\_SelectorBytes, 63
- DtapiTs::DtDescDvbLinkage, 63
  - m\_Event, 64
  - m\_ExtendedEvents, 64
  - m\_MobileHandOver, 64
  - m\_OrigNetworkId, 64
  - m\_TransportStreamId, 64
- DtapiTs::DtDescDvbLinkage::EventLinkage, 129
- DtapiTs::DtDescDvbLinkage::ExtendedEventLinkage, 130
  - m\_LinkType, 130
  - m\_TargetIdType, 130
  - m\_TargetOrigNetworkId, 130
  - m\_TargetServiceId, 130
  - m\_TargetTsId, 130
- DtapiTs::DtDescDvbLinkage::MobileHandOverInfo, 131
  - m\_InitialServiceId, 132
  - m\_NetworkId, 132
  - m\_OrigType, 132
- DtapiTs::DtDescDvbLocalTimeOffset, 64
- DtapiTs::DtDescDvbLocalTimeOffset::LocalTimeOffset, 131
  - m\_TimeOfChange, 131
- DtapiTs::DtDescDvbMultilingualComponent, 65
  - m\_ComponentTag, 65
  - m\_Descriptions, 65
- DtapiTs::DtDescDvbNetworkName, 66
  - m\_NetworkName, 66
- DtapiTs::DtDescDvbSDelivery, 66
  - m\_IsDvbS2, 67
  - m\_ModType, 67
  - m\_RollOff, 67
  - m\_WestEastFlag, 67
- DtapiTs::DtDescDvbService, 67
  - m\_ServiceType, 68
- DtapiTs::DtDescDvbServiceList, 68
- DtapiTs::DtDescDvbServiceList::ServiceListItem, 132
  - m\_ServiceId, 132
  - m\_ServiceType, 132
- DtapiTs::DtDescDvbSubtitling, 69
- DtapiTs::DtDescDvbSubtitling::Subtitling, 132
- DtapiTs::DtDescDvbTDelivery, 69
  - m\_HierarchyInformation, 70
- DtapiTs::DtDescDvbTeletext, 70
- DtapiTs::DtDescDvbTeletext::Teletext, 133
  - m\_PageNum, 133
- DtapiTs::DtDescMpegCa, 71
  - m\_CaPid, 71
  - m\_CaSystemId, 71
- DtapiTs::DtDescMpegLanguage, 72
  - m\_Codes, 72
- DtapiTs::DtDescMpegLanguage::LangCode, 131
- DtapiTs::DtDescMpegPrivDataIndicator, 72
- DtapiTs::DtDescMpegRegistration, 73
- DtapiTs::DtDescMpegVideoStream, 73
  - m\_ChromaFormat, 74
  - m\_ConstrainedParameter, 74
  - m\_FrameRateCode, 74
  - m\_FrameRateExtension, 74
  - m\_Mpeg1Only, 74
  - m\_MultipleFrameRates, 74
  - m\_ProfileLevelIndication, 74
- DtapiTs::DtDescPrivLcn, 75
- DtapiTs::DtDescPrivLcn::DtLogicalChannelNumber, 86
  - m\_IsVisible, 87
- DtapiTs::DtDescriptor, 75
  - m\_DescriptorType, 76
  - m\_ExtendedTag, 76
  - m\_Pds, 76
- DtapiTs::DtDvbCNitInfo, 77
  - Constellation, 77
- DtapiTs::DtDvbSNitInfo, 81
  - InputStreamIdentifier, 81
  - ModType, 81
  - S2FieldsPresent, 82
  - ScramblingSequenceIndex, 82
- DtapiTs::DtDvbShModInfo, 77
  - m\_CommonMultiplier, 78
  - m\_CompleteInterleaver, 78
  - m\_NofLateTaps, 78
  - m\_NofSlices, 78
  - m\_NonLateIncrement, 78
  - m\_SliceDistance, 78
- DtapiTs::DtDvbShNitInfo, 79
  - m\_DiversityMode, 79
  - m\_ModInfo, 79
- DtapiTs::DtDvbShOfdmInfo, 79
  - m\_CommonFrequency, 80
  - m\_Constellation, 80
  - m\_Priority, 80
- DtapiTs::DtDvbShTdmInfo, 80
  - m\_SymbolRate, 81
- DtapiTs::DtDvbT2CellInfo, 82
- DtapiTs::DtDvbT2NitInfo, 82
  - m\_OtherFrequencyUsed, 83
  - m\_Plpld, 83
  - m\_T2SystemId, 83
- DtapiTs::DtDvbT2SubCellInfo, 83
  - m\_SubCellId, 84
- DtapiTs::DtDvbTNitInfo, 84
  - Bandwidth, 84
  - CodeRateHpStream, 84
  - HierarchyInformation, 85
  - OtherFrequencyUsed, 85
  - TransmissionMode, 85
- DtapiTs::DtEac3EsInfo, 85
- DtapiTs::DtEsInfoBase, 85
- DtapiTs::DtEsInfoBase::InfoField< T >, 131
- DtapiTs::DtHeAacEsInfo, 86
- DtapiTs::DtJitterPoint, 86
- DtapiTs::DtMpaEsInfo, 87
- DtapiTs::DtPcr, 87
- DtapiTs::DtPcrInfo, 88
  - m\_Df, 89

- DtapiTs::DtPes, 89
- DtapiTs::DtPes::DataBuffer, 51
- DtapiTs::DtPidInfo, 90
  - GetDescription, 91
  - HasTableType, 91
  - m\_SeenBefore, 91
  - m\_TableTypeMask, 92
- DtapiTs::DtPtsDts, 92
- DtapiTs::DtServiceComponentInfo, 93
  - m\_CaSystems, 94
  - m\_Description, 94
  - m\_HasPrivateDataDesc, 94
- DtapiTs::DtServiceInfo, 94
  - GetName, 95
  - InService, 96
  - m\_CaSystems, 96
  - m\_OrigServiceType, 96
  - m\_ProgramNumber, 96
  - m\_ServiceType, 96
- DtapiTs::DtStructuredTable, 96
  - DecodeFromTable, 97
- DtapiTs::DtSubTableId, 97
  - Matches, 98
  - operator<, 98
- DtapiTs::DtTable, 98
  - DtTable, 99
  - m\_Sections, 99
  - m\_Version, 100
  - operator=, 99
- DtapiTs::DtTableBat, 100
  - FindTs, 100
- DtapiTs::DtTableBatInner, 101
- DtapiTs::DtTableCat, 101
- DtapiTs::DtTableNit, 102
  - FindTsLoop, 102
  - m\_NetworkDescriptors, 102
- DtapiTs::DtTableNitInner, 103
- DtapiTs::DtTablePat, 103
- DtapiTs::DtTablePat::DtProgramMapping, 92
- DtapiTs::DtTablePmt, 104
- DtapiTs::DtTablePmtInner, 104
- DtapiTs::DtTableSdt, 105
  - FindService, 106
  - m\_TransportStreamId, 106
- DtapiTs::DtTableSdtInner, 106
  - m\_EitPresentFollowing, 106
  - m\_EitSchedule, 106
  - m\_FreeCaMode, 106
- DtapiTs::DtTableSection, 107
  - DtTableSection, 107
  - operator=, 107
- DtapiTs::DtTableTdt, 108
- DtapiTs::DtTableTot, 108
- DtapiTs::DtTimeDiff, 109
  - operator<, 110
- DtapiTs::DtTimestamp, 110
- DtapiTs::DtTp, 111
- DtapiTs::DtTr101290, 112
- DtapiTs::DtTr101290Error, 112
  - m\_ErrCount, 113
  - m\_IsSet, 113
  - m\_Latched, 113
  - m\_Time, 113
- DtapiTs::DtTsData, 113
  - GetNitFrequency, 115
  - m\_CaSystems, 115
  - m\_DeliverySystem, 115
  - m\_ErrIndErrors, 115
  - m\_NitTsRate, 115
  - m\_PacketSize, 115
  - m\_SyncByteErrors, 115
  - m\_TmccDataValid, 116
- DtapiTs::DtTsInfo, 116
  - AddJitterCallback, 119
  - AddNewSectionCallback, 119
  - AddPesPacketCallback, 119, 120
  - AddTableChangedCallback, 120
  - AddTableTimeoutCallback, 120
  - DtJitterCallback, 118
  - DtPacketCallback, 118
  - DtPesCallback, 118
  - DtSectionCallback, 118
  - DtTableCallback, 119
  - DtTableTimeoutCallback, 119
  - GetIsdbtPars, 120
  - Lock, 120
  - m\_CompletePes, 121
  - m\_Data, 121
  - m\_PreferredLanguages, 121
  - m\_TableTimeoutCb, 122
  - m\_UseTableCache, 122
  - NewPacket, 120
  - NewTimestamp, 120
  - Reset, 120
  - SetJitterWindow, 121
  - SetStandardMode, 121
  - Unlock, 121
- DtapiTs::DtTsInfoInput, 122
  - NewData, 123
  - SetTsInfoObject, 123
- DtapiTs::DtTsLib, 123
  - CreateDtTsInfoInstance, 124
- DtapiTs::DtTsPacketInput, 124
  - CreateInstance, 124
- DtapiTs::DtTsTimestampedPacketInput, 125
  - CreateInstance, 125
- DtapiTs::DtTsTransparentInput, 126
  - CreateInstance, 126
- DtapiTs::DtVideoAspectRatio, 126
- DtapiTs::DtVideoEsAvclInfo, 127
- DtapiTs::DtVideoEsInfo, 127
- FindService
  - DtapiTs::DtTableSdt, 106
- FindTs
  - DtapiTs::DtTableBat, 100
- FindTsLoop

- DtapiTs::DtTableNit, 102
- GetDescription
  - DtapiTs::DtPidInfo, 91
- GetIsdbtPars
  - DtapiTs::DtTsInfo, 120
- GetName
  - DtapiTs::DtServiceInfo, 95
- GetNitFrequency
  - DtapiTs::DtTsData, 115
- HasTableType
  - DtapiTs::DtPidInfo, 91
- HierarchyInformation
  - DtapiTs::DtDvbTNitInfo, 85
- InService
  - DtapiTs::DtServiceInfo, 96
- InputStreamIdentifier
  - DtapiTs::DtDvbSNitInfo, 81
- Lock
  - DtapiTs::DtTsInfo, 120
- m\_Asvc
  - DtapiTs::DtDescDvbAc3, 59
- m\_Bsid
  - DtapiTs::DtDescDvbAc3, 59
- m\_CaPid
  - DtapiTs::DtDescMpegCa, 71
- m\_CaSystemId
  - DtapiTs::DtCaSystem, 58
  - DtapiTs::DtDescMpegCa, 71
- m\_CaSystems
  - DtapiTs::DtServiceComponentInfo, 94
  - DtapiTs::DtServiceInfo, 96
  - DtapiTs::DtTsData, 115
- m\_ChromaFormat
  - DtapiTs::DtDescMpegVideoStream, 74
- m\_Codes
  - DtapiTs::DtDescMpegLanguage, 72
- m\_CommonFrequency
  - DtapiTs::DtDvbShOfdmInfo, 80
- m\_CommonMultiplier
  - DtapiTs::DtDvbShModInfo, 78
- m\_CompleteInterleaver
  - DtapiTs::DtDvbShModInfo, 78
- m\_CompletePes
  - DtapiTs::DtTsInfo, 121
- m\_ComponentTag
  - DtapiTs::DtDescDvbComponent, 61
  - DtapiTs::DtDescDvbDataBroadcast, 62
  - DtapiTs::DtDescDvbMultilingualComponent, 65
- m\_ComponentType
  - DtapiTs::DtDescDvbAc3, 59
- m\_Constellation
  - DtapiTs::DtDvbShOfdmInfo, 80
- m\_ConstrainedParameter
  - DtapiTs::DtDescMpegVideoStream, 74
- m\_Data
  - DtapiTs::DtTsInfo, 121
- m\_DataBroadcastId
  - DtapiTs::DtDescDvbDataBroadcast, 62
  - DtapiTs::DtDescDvbDataBroadcastId, 63
- m\_DeliverySystem
  - DtapiTs::DtTsData, 115
- m\_Description
  - DtapiTs::DtServiceComponentInfo, 94
- m\_Descriptions
  - DtapiTs::DtDescDvbMultilingualComponent, 65
- m\_DescriptorType
  - DtapiTs::DtDescriptor, 76
- m\_Df
  - DtapiTs::DtPcrInfo, 89
- m\_DiversityMode
  - DtapiTs::DtDvbShNitInfo, 79
- m\_EitPresentFollowing
  - DtapiTs::DtTableSdtInner, 106
- m\_EitSchedule
  - DtapiTs::DtTableSdtInner, 106
- m\_ErrCount
  - DtapiTs::DtTr101290Error, 113
- m\_ErrIndErrors
  - DtapiTs::DtTsData, 115
- m\_Event
  - DtapiTs::DtDescDvbLinkage, 64
- m\_ExtendedEvents
  - DtapiTs::DtDescDvbLinkage, 64
- m\_ExtendedTag
  - DtapiTs::DtDescriptor, 76
- m\_FrameRateCode
  - DtapiTs::DtDescMpegVideoStream, 74
- m\_FrameRateExtension
  - DtapiTs::DtDescMpegVideoStream, 74
- m\_FreeCaMode
  - DtapiTs::DtTableSdtInner, 106
- m\_HasPrivateDataDesc
  - DtapiTs::DtServiceComponentInfo, 94
- m\_HierarchyInformation
  - DtapiTs::DtDescDvbTDelivery, 70
- m\_InitialServiceId
  - DtapiTs::DtDescDvbLinkage::MobileHandOverInfo, 132
- m\_IsDvbS2
  - DtapiTs::DtDescDvbSDelivery, 67
- m\_IsSet
  - DtapiTs::DtTr101290Error, 113
- m\_IsVisible
  - DtapiTs::DtDescPrivLcn::DtLogicalChannel-  
Number, 87
- m\_Latched
  - DtapiTs::DtTr101290Error, 113
- m\_LinkType
  - DtapiTs::DtDescDvbLinkage::ExtendedEvent-  
Linkage, 130
- m\_MainId
  - DtapiTs::DtDescDvbAc3, 60

- m\_MobileHandOver
  - DtapiTs::DtDescDvbLinkage, [64](#)
- m\_ModInfo
  - DtapiTs::DtDvbShNitInfo, [79](#)
- m\_ModType
  - DtapiTs::DtDescDvbSDelivery, [67](#)
- m\_Mpeg1Only
  - DtapiTs::DtDescMpegVideoStream, [74](#)
- m\_MultipleFrameRates
  - DtapiTs::DtDescMpegVideoStream, [74](#)
- m\_NetworkDescriptors
  - DtapiTs::DtTableNit, [102](#)
- m\_NetworkId
  - DtapiTs::DtDescDvbLinkage::MobileHandOverInfo, [132](#)
- m\_NetworkName
  - DtapiTs::DtDescDvbNetworkName, [66](#)
- m\_NitTsRate
  - DtapiTs::DtTsData, [115](#)
- m\_NofLateTaps
  - DtapiTs::DtDvbShModInfo, [78](#)
- m\_NofSlices
  - DtapiTs::DtDvbShModInfo, [78](#)
- m\_NonLateIncrement
  - DtapiTs::DtDvbShModInfo, [78](#)
- m\_NumAvgValues
  - DtapiTs::DtBitrateSettings, [56](#)
- m\_OrigNetworkId
  - DtapiTs::DtDescDvbLinkage, [64](#)
- m\_OrigServiceType
  - DtapiTs::DtServiceInfo, [96](#)
- m\_OrigType
  - DtapiTs::DtDescDvbLinkage::MobileHandOverInfo, [132](#)
- m\_OtherFrequencyUsed
  - DtapiTs::DtDvbT2NitInfo, [83](#)
- m\_PacketSize
  - DtapiTs::DtTsData, [115](#)
- m\_PageNum
  - DtapiTs::DtDescDvbTeletext::Teletext, [133](#)
- m\_Pds
  - DtapiTs::DtDescriptor, [76](#)
- m\_Plpld
  - DtapiTs::DtDvbT2NitInfo, [83](#)
- m\_PreferedLanguages
  - DtapiTs::DtTsInfo, [121](#)
- m\_Priority
  - DtapiTs::DtDvbShOfdmInfo, [80](#)
- m\_ProfileLevelIndication
  - DtapiTs::DtDescMpegVideoStream, [74](#)
- m\_ProgramNumber
  - DtapiTs::DtServiceInfo, [96](#)
- m\_RollOff
  - DtapiTs::DtDescDvbSDelivery, [67](#)
- m\_Sections
  - DtapiTs::DtTable, [99](#)
- m\_SeenBefore
  - DtapiTs::DtPidInfo, [91](#)
- m\_SelectorBytes
  - DtapiTs::DtDescDvbDataBroadcast, [62](#)
  - DtapiTs::DtDescDvbDataBroadcastId, [63](#)
- m\_ServiceId
  - DtapiTs::DtDescDvbServiceList::ServiceListItem, [132](#)
- m\_ServiceType
  - DtapiTs::DtDescDvbService, [68](#)
  - DtapiTs::DtDescDvbServiceList::ServiceListItem, [132](#)
  - DtapiTs::DtServiceInfo, [96](#)
- m\_SliceDistance
  - DtapiTs::DtDvbShModInfo, [78](#)
- m\_SubCellId
  - DtapiTs::DtDvbT2SubCellInfo, [84](#)
- m\_SymbolRate
  - DtapiTs::DtDvbShTdmInfo, [81](#)
- m\_SyncByteErrors
  - DtapiTs::DtTsData, [115](#)
- m\_T2SystemId
  - DtapiTs::DtDvbT2NitInfo, [83](#)
- m\_TableTimeoutCb
  - DtapiTs::DtTsInfo, [122](#)
- m\_TableTypeMask
  - DtapiTs::DtPidInfo, [92](#)
- m\_TargetIdType
  - DtapiTs::DtDescDvbLinkage::ExtendedEvent-Linkage, [130](#)
- m\_TargetOrigNetworkId
  - DtapiTs::DtDescDvbLinkage::ExtendedEvent-Linkage, [130](#)
- m\_TargetServiceId
  - DtapiTs::DtDescDvbLinkage::ExtendedEvent-Linkage, [130](#)
- m\_TargetTsId
  - DtapiTs::DtDescDvbLinkage::ExtendedEvent-Linkage, [130](#)
- m\_Time
  - DtapiTs::DtTr101290Error, [113](#)
- m\_TimeOfChange
  - DtapiTs::DtDescDvbLocalTimeOffset::LocalTime-Offset, [131](#)
- m\_TmccDataValid
  - DtapiTs::DtTsData, [116](#)
- m\_TransportStreamId
  - DtapiTs::DtDescDvbLinkage, [64](#)
  - DtapiTs::DtTableSdt, [106](#)
- m\_UseTableCache
  - DtapiTs::DtTsInfo, [122](#)
- m\_Version
  - DtapiTs::DtTable, [100](#)
- m\_WestEastFlag
  - DtapiTs::DtDescDvbSDelivery, [67](#)
- Matches
  - DtapiTs::DtSubTableId, [98](#)
- ModType
  - DtapiTs::DtDvbSNitInfo, [81](#)
- NewData

- DtapiTs::DtTsInfoInput, [123](#)
- NewPacket
  - DtapiTs::DtTsInfo, [120](#)
- NewTimestamp
  - DtapiTs::DtTsInfo, [120](#)
- operator<
  - DtapiTs::DtSubTableId, [98](#)
  - DtapiTs::DtTimeDiff, [110](#)
- operator=
  - DtapiTs::DtTable, [99](#)
  - DtapiTs::DtTableSection, [107](#)
- OtherFrequencyUsed
  - DtapiTs::DtDvbTNitInfo, [85](#)
- Reset
  - DtapiTs::DtTsInfo, [120](#)
- S2FieldsPresent
  - DtapiTs::DtDvbSNitInfo, [82](#)
- ScramblingSequenceIndex
  - DtapiTs::DtDvbSNitInfo, [82](#)
- SetJitterWindow
  - DtapiTs::DtTsInfo, [121](#)
- SetStandardMode
  - DtapiTs::DtTsInfo, [121](#)
- SetTsInfoObject
  - DtapiTs::DtTsInfoInput, [123](#)
- Structured information from binary descriptors, [23](#)
- Structured information from binary tables, [25](#)
- TransmissionMode
  - DtapiTs::DtDvbTNitInfo, [85](#)
- Unlock
  - DtapiTs::DtTsInfo, [121](#)