DTAPI

Advanced Demodulator API

REFERENCE

May 2013

Table of Contents

struct DtComplexFloat

Structure describing a complex floating-point number.

```
Struct DtComplexFloat
{
   float  m_Re;                  // Real part
   float  m_Im;                  // Imaginary part
};
```

Members

m_Re
The real part of the complex floating-point number.

m_Im
The imaginary part of the complex floating-point number.

struct DtConstelPars

This structure specifies the parameters for a stream of constellation points. It is used in structure **DtStreamSelPars**.

```
struct DtConstelPars
{
   int  m_Period;              // Minimum time between callbacks in ms
   int  m_ConstellationType;   // Constellation per PLP (0) or carrier (1)
   int  m_Index;               // Index of the PLP or carrier
   int  m_MaxNumPoints;        // Maximum number of constellation points
};
```

Members

m_Period

The minimum time period in milliseconds between two calls of the constellation point callback function. This time period must be the same for all active constellation point streams.

m_ConstellationType

Specifies whether constellation points apply to a single carrier or to a PLP.

Value	Meaning
0	Constellation points for the selected PLP
1	Constellation points for the selected carrier

m_Index

Specifies the index of the PLP (if *m_ConstellationType* equals '0') or specifies the index of the carrier (if *m_ConstellationType* equals '1').

m_MaxNumPoints

Maximum number of constellation points that will be passed through the constellation point callback function.

struct DtDabEtiStreamSelPars

This structure specifies the selection parameters for a DAB Ensemble Transport Interface (ETI) stream.

```
struct DtDabEtiStreamSelPars
{
   // No parameters required
};
```

Members

DtDabEtiStreamSelPars structure has no members

Remark

All DAB sub-channels are selected and output in a DAB Ensemble Transport Interface (ETI) stream.

struct DtDabStreamSelPars

This structure specifies the selection parameters for a DAB sub-channel. It is used in structure **DtStreamSelPars** to select a specific stream.

```
struct DtDabStreamSelPars
{
  int  m_BitrateKbps;         // Bitrate in kbps
  int  m_ErrProtLevel;        // Error protection level
  int  m_ErrProtMode;         // Error protection mode
  int  m_ErrProtOption;       // Error protection option
  int  m_StartAddress;        // Start address in capacity units
  DtDabExtractionMode m_ExtractionMode;  // DAB data extraction mode
};
```

Members

m_BitrateKbps

Specifies the bitrate of the channel in kbps. The valid bitrate range for the UEP profile is: 32 ... 384 and the bitrate must be a multiple of 8. The valid bitrate range for the EEP profile is: 8 ... 2048 and the bitrate must be a multiple of 8.

m_ErrProtLevel

The valid range for UEP profile is: 1 ... 4. The valid range for EEP profile is: 1 ... 5.

m_ErrProtMode

Value	Meaning
DTAPI_DAB_UEP	Unequal Error Protection (UEP)
DTAPI_DAB_EEP	Equal Error Protection (EEP)

m_ErrProtOption

EEP protection level option: 0 or 1. Only meaningful for EEP profile.

m_StartAddress

Specifies the address of the first capacity unit (CU) of the sub-channel. The valid range is: 0 ... 863.

m_ExtractionMode

Value	Meaning
DAB_RAW	Raw DAB stream
DAB_EXTRACTION_AAC	AAC/DAB+ stream extraction
DAB_EXTRACTION_DMB	DMB stream extraction

struct DtDabFicStreamSelPars

This structure specifies the selection parameters for a DAB Fast Information Channel (FIC). It is used in structure **DtStreamSelPars** to select a specific stream.The parameters are not used for selection but are passed in the WriteStreamFunc() callback function to indicate the parameters of the Fast Information Block that is passed.

```
struct DtDabFicStreamSelPars
{
  int  m_CifIndex;          // Index of the CIF in the DAB frame
  int  m_FibIndex;          // Index of this FIB
};
```

Members

m_CifIndex
Index of the Common Interleaved Frame (CIF) in the DAB frame to which this Fast Information Block (FIB) is associated

m_FibIndex
Index of this Fast Information Block (FIB) in the group of FIBs that are associated to the same Common Interleaved Frame (CIF).

struct DtDvbC2StreamSelPars

This structure specifies the selection parameters for a DVB-C2 stream. It is used in structure **DtStreamSelPars** to select a specific stream.

```
struct DtDvbC2StreamSelPars
{
  int  m_DSliceId;          // Data slice ID
  int  m_PlpId;             // Data PLP ID
  int  m_CommonPlpId;       // Common PLP ID
};
```

Members

m_DSliceId

Unique identification of the data slice within the DVB-C2 stream. Valid values are: 0 … 255 and **DTAPI_DVBC2_DSLICE_ID_AUTO**. The latter value specifies automatic selection of the data slice. In this case the first data slice is selected.

m_PlpId

Unique identification of the data PLP within the DVB-C2 stream. The valid range is 0 … 255 and **DTAPI_DVBC2_PLP_ID_AUTO**. The latter value specifies automatic selection of the first data PLP.

m_CommonPlpId

Unique identification of the common PLP within the DVB-C2 stream. It will be combined with the selected data PLP. The valid values are: 0 … 255, **DTAPI_DVBC2_PLP_ID_NONE** and **DTAPI_DVBC2_PLP_ID_AUTO**.

The value **DTAPI_DVBC2_PLP_ID_NONE** indicates that no common PLP is selected. The value **DTAPI_DVBC2_PLP_ID_AUTO** indicates automatic selection of the common PLP.

struct DtDvbTStreamSelPars

This structure specifies the selection parameters for a DVB-T transport stream. It is used in structure **DtStreamSelPars**.

```
struct DtDvbTStreamSelPars
{
   // No parameters required
};
```

Members

Remarks

No additional parameters are required to select a DVB-T transport stream. Hierarchical DVB-T demodulation is not supported.

struct DtDvbT2StreamSelPars

This structure specifies the selection parameters for a DVB-T2 stream. It is used in structure **DtStreamSelPars** to select a specific stream.

```
struct DtDvbT2StreamSelPars
{
  int  m_PlpId;              // Data PLP ID
  int  m_CommonPlpId;        // Common PLP ID
};
```

Members

m_PlpId

Unique identification of the data PLP within the DVB-T2 stream. The valid range is 0 … 255 and **DTAPI_DVBT2_PLP_ID_AUTO**. The value **DTAPI_DVBT2_PLP_ID_AUTO** specifies automatic selection of the PLP. In this case the first PLP is selected.

m_CommonPlpId

Unique identification of the common PLP within the DVB-T2 stream. It will be combined with the selected data physical layer pipe. The valid values for *m_CommonPlpId* are: 0 … 255, **DTAPI_DVBT2_PLP_ID_NONE** and **DTAPI_DVBT2_PLP_ID_AUTO**.
The value **DTAPI_DVBT2_PLP_ID_NONE** specifies that no common PLP is used. The value **DTAPI_DVBT2_PLP_ID_AUTO** specifies automatic selection of the common PLP.

Remarks

Multiple streams (PLPs) from a DVB-T2 stream can be selected simultaneously. However, selecting one or more PLPs cannot be combined with the selection of a T2-MI stream.

struct DtImpRespPars

This structure specifies the parameters for an impulse-response stream. It is used in structure **DtStreamSelPars**.

```
struct DtImpRespPars
{
  int  m_Period;              // Minimum time between callbacks in ms
  int  m_Channel;             // Channel used for MISO
};
```

Members

m_Period

The minimum time period in milliseconds between two calls of the impulse-response callback function. This time period must be the same for all selected impulse-response and transfer-function streams.

m_Channel

Specifies the MISO channel: '0' for TX1 and '1' for TX2. TX1 should be selected in case of a SISO input signal.

struct DtIsdbtStreamSelPars

This structure specifies the selection parameters for an ISDB-T stream. It is used in structure **DtStreamSelPars** to select the ISDB-T stream.

```
struct DtIsdbtStreamSelPars
{
   // Empty
};
```

Members

DtIsdbtStreamSelPars structure has no members

Remark

All layers are selected and output the same stream

struct DtMeasurement

Structure describing a set of measurement values. It used to pass measurements from the advanced demodulator to the user application through user-supplied **DtWriteMeasFunc** callback.

```
struct DtMeasurement
{
   DtStreamType   m_MeasurementType;     // Type of measurement values
   __int64   m_TimeStamp;                // Timestamp in number of samples
   int   m_NumValues;                    // Number of measurement values
   DtComplexFloat*   m_pMeasurement;     // Measurement values
};
```

Members

m_MeasurementType

Type of measurement data.

Value	Meaning
STREAM_CONSTEL	Constellation points
STREAM_IMPRESP	Impulse response
STREAM_MER	MER
STREAM_SPECTRUM	Spectrum
STREAM_TF_ABS	Transfer function – Absolute value
STREAM_TF_PHASE	Transfer function – Phase
STREAM_TF_GROUPDELAY	Transfer function – Group delay

m_TimeStamp

Timestamp of measurement data. It is expressed in the number of I/Q samples processed since demodulator start up.

m_NumValues

The number of measurement values in *m_pMeasurement*.

m_pMeasurement

Pointer to a buffer of **DtComplexFloat** elements with length *m_NumValues*. The meaning of the **DtComplexFloat** elements depends on the type of the measurement data. The buffer is allocated and released by the advanced demodulator.

Stream Type	DtComplexFloat.m_Re	DtComplexFloat.m_Im
STREAM_CONSTEL	X-coordinate	Y-coordinate
STREAM_IMPRESP	Delay (μs)	Relative power (db)
STREAM_MER	Delay (μs)	MER (db)
STREAM_SPECTRUM	Frequency (MHz)	Relative power (db)
STREAM_TF_ABS	Frequency (MHz)	Relative power (db)
STREAM_TF_PHASE	Frequency (MHz)	Phase (degrees)
STREAM_TF_GROUPDELAY	Frequency (MHz)	Group delay (μs)

struct DtMerPars

This structure specifies the parameters for a MER stream. It is used in structure **DtStreamSelPars**.

```
struct DtMerPars
{
  int  m_Period;              // Minimum time between callbacks in ms
};
```

Members

m_Period
The minimum time period in milliseconds between two calls of the MER callback function.

m_ValueBool, m_ValueDouble, m_ValueInt, m_pValue
The value of the *DtPar.m_ValueType* determines which parameter is used.

struct DtSpectrumPars

This structure specifies the parameters for a spectrum stream. It is used in structure **DtStreamSelPars**.

```
struct DtSpectrumPars
{
   int  m_Period;              // Minimum time between callbacks in ms
   int  m_FftLength;           // FFT length, must be a power of tw
   int  m_AverageLength;       // Number of FFT blocks used for averaging
};
```

Members

m_Period
The minimum time period in milliseconds between two calls of the spectrum callback function.

m_FftLength
FFT length, must be equal or greater than 32 and a power of 2.

m_AverageLength
The number of FFT blocks on which the average is performed.

Remarks

The processing time for creating the spectrum plot data is proportional to *m_FftLength * m_AverageLength*.

struct DtStreamSelPars

This structure is used in **`DtAdvDemod::OpenStream()`** to specify the parameters for the stream to be opened. It is also used as parameter in the callback function to identify the associated stream.

```
struct DtStreamSelPars
{
 intptr_t  m_Id;                        // Unique stream identifier
 DtStreamType  m_StreamType;            // Stream type
 union {
   // Selection parameters for demodulated streams
   DtDabStreamSelPars     m_Dab;        // DAB stream
   DtDabEtiStreamSelPars  m_DabEti;     // DAB Ensemble Tranport Interface
   DtDabFicStreamSelPars  m_DabFic;     // DAB Fast Information Channel
   DtDvbC2StreamSelPars   m_DvbC2;      // DVB-C2 stream
   DtDvbTStreamSelPars    m_DvbT;       // DVB-T stream
   DtDvbT2StreamSelPars   m_DvbT2;      // DVB-T2 stream
   DtIsdbtStreamSelPars   m_Isdbt;      // ISDB-T stream
   DtT2MiStreamSelPars    m_T2Mi;       // T2-MI stream

   // Parameters for streams of measurement values
   DtConstelPars          m_Constel;    // Constellation points
   DtImpRespSelPars       m_ImpResp;    // Impulse response
   DtMerPars              m_Mer;        // MER
   DtSpectrumPars         m_Spectrum;   // Spectrum
   DtTransFuncPars        m_TransFunc;  // Transfer function
 } u;
};
```

Members

m_Id

Uniquely identifies the stream. The user can use any integer value or pointer, as long as the value is unique for each stream.

m_StreamType

Classifies the type of the stream.

Value	Meaning
Stream types – Demodulated data	
STREAM_DAB	DAB stream
STREAM_DABETI	DAB Ensemble Transport Interface steam
STREAM_DABFIC	DAB Fast Information Channel stream
STREAM_DVBC2	DVB-C2 stream
STREAM_DVBT	DVB-T stream
STREAM_DVBT2	DVB-T2 stream
STREAM_ISDBT	ISDB-T stream
STREAM_T2MI	T2-MI stream

Stream types – Measurement values	
STREAM_CONSTEL	Constellation points
STREAM_IMPRESP	Impulse response
STREAM_MER	MER
STREAM_SPECTRUM	Spectrum
STREAM_TF_ABS	Transfer function absolute
STREAM_TF_PHASE	Transfer function phase
STREAM_TF_GROUPDELAY	Transfer function group delay

u.m_Constel
Structure used if *m_Type* equals **STREAM_CONSTEL**. See **DtConstelPars** for the members.

u.m_Dab
Structure used if *m_Type* equals **STREAM_DAB**. See **DtDabStreamSelPars** for the members.

u.m_DabEti
Structure used if *m_Type* equals **STREAM_DABETI**. See **DtDabEtiStreamSelPars** for the members.

u.m_DabFic
Structure used if *m_Type* equals **STREAM_DABFIC**. See **DtDabFicStreamSelPars** for the members.

u.m_DvbC2
Structure used if *m_Type* equals **STREAM_DVBC2**. See **DtDvbC2StreamSelPars** for the members.

u.m_DvbT
Structure used if *m_Type* equals **STREAM_DVBT**. See **DtDvbTStreamSelPars** for the members.

u.m_DvbT2
Structure used if *m_Type* equals **STREAM_DVBT2**. See **DtDvbT2StreamSelPars** for the members.

u.m_ImpResp
Structure used if *m_Type* equals **STREAM_IMPRESP**. See **DtImpRespPars** for the members.

u.m_Isdbt
Structure used if *m_Type* equals **STREAM_ISDBT**. See **DtIsdbtStreamSelPars** for the members.

u.m_Mer
Structure used if *m_Type* equals **STREAM_MER**. See **DtMerPars** for the members.

u.m_Spectrum
Structure used if *m_Type* equals **STREAM_SPECTRUM**. See **DtSpectrumPars** for the members.

u.m_T2Mi
Structure used if case *m_Type* equals **STREAM_T2MI**. See **DtT2MiSelPars** for the members.

u.m_TransFunc

Structure used if *m_Type* equals **STREAM_TF_ABS**, **STREAM_TF_PHASE** or **STREAM_TF_GROUPDELAY**. See **DtTransFuncPars** for the members.

struct DtT2MiStreamSelPars

This structure specifies the selection parameters for a T2-MI transport stream containing a complete DVB-T2 stream. It is used in structure **DtStreamSelPars** to specify a selected stream.

```
struct DtT2MiStreamSelPars
{
   int  m_T2MiOutPid;          // T2-MI output PID
   int  m_T2MiTsRate;          // T2-MI transport stream rate
};
```

Members

m_T2MiOutPid
Specifies the PID carrying the T2-MI packet data. The valid range is 0 … 8190.

m_T2MiTsRate
Specifies the T2-MI transport-stream rate in bits per second. If set to '-1' a variable bitrate transport stream is created, else null packets are added to reach the specified rate. The maximum rate is 72 Mbps.
In case the specified transport-stream rate is too low, T2-MI overflows occur. The number of overflows can be retrieved using the **DTAPI_STAT_T2MI_OVFS** statistic.

Remarks

T2-MI transport stream selection cannot be combined with DVB-T2 stream (PLP) selection.

struct DtTransFuncPars

This structure specifies the parameters for a transfer-function stream. It is used in structure **DtStreamSelPars**.

```
struct DtTransFuncPars
{
   int  m_Period;              // Minimum time between callbacks in ms
   int  m_Channel;             // Channel used for MISO
};
```

Members

m_Period

The minimum time period in milliseconds between two calls of the transfer function callback function. This time period must be the same for all selected impulse-response and transfer-function streams.

m_Channel

Specifies the MISO channel; *'0'* for TX1 and *'1'* for TX2. TX1 should be selected in case of SISO input signal.

DtOutputRateChangedFunc

Prototype of a callback function to be supplied by the user with **`DtAdvDemod::RegisterCallback`**. The advanced demodulator invokes this callback function when the bitrate of the stream has changed.

```
void  DtOutputRateChangedFunc(
  [in] void*  pOpaque,                  // Opaque pointer
  [in] DtStreamSelPars&  StreamSel,     // Stream selection parameters
  [in] int  Bitrate                     // New bitrate in bps
);
```

Parameters

pOpaque

The opaque pointer that was specified in **`DtAdvDemod::RegisterCallback()`**.

StreamSel

The stream selection parameters that were passed in **`DtAdvDemod::OpenStream()`**. This parameter can be used to identify the stream when multiple data streams are generated simultaneously.

Bitrate

New bitrate of the selected stream in bits per second.

Remarks

The callback function may not block and the amount of processing should be kept as low as possible to avoid stalling the advanced demodulator. In case significant processing time is required the data should be written to a temporary buffer and be processed in another thread.

DtReadIqFunc

Prototype of a callback function to be supplied by the user if he wants *virtual* I/Q input, this is supplying I/Q samples from a source other than a receiver device (e.g. from file). The advanced demodulator calls this function to obtain new I/Q samples.

```
void  DtReadIqFunc(
  [in] void*  pOpaque,              // Opaque pointer
  [in] unsigned char*  pIqBuf,      // Buffer to write I/Q samples to
  [in] int  IqBufSize,              // Size of I/Q sample buffer in bytes
 [out] int&  IqLength               // Number of bytes written in pIqBuf
);
```

Parameters

pOpaque

The opaque pointer that was specified in **`DtAdvDemod::AttachVirtual()`**.

pIqBuf

Pointer to a buffer – allocated by the advanced demodulator – into which the user can write his I/Q samples. The I/Q samples shall be signed 16-bit integer in I, Q order.

IqBufSize

Size of *pIqBuf* in number of bytes. This is the maximum number of bytes that may be written into *pIqBuf*.

IqLength

Output argument that is to be set to the actual number of bytes written in *pIqBuf*.

DtWriteMeasFunc

Prototype of a callback function to be supplied by the user with **DtAdvDemod::RegisterCallback**. The advanced demodulator calls this function when new measurement values are available. The user can process the measurements any way he likes, e.g. plot in a GUI or write to a file. If multiple streams of measurement values are used, they share a single callback function. The user must demultiplex the different measurement values using the stream selection parameters.

```
void  DtWriteMeasFunc(
  [in] void*  pOpaque,                  // Opaque pointer
  [in] DtStreamSelPars&  StreamSel,     // Stream selection parameters
  [in] DtMeasurement*  pMeasurement     // Buffer with measurement values
);
```

Parameters

pOpaque

The opaque pointer that was specified in **DtAdvDemod::RegisterCallback()**.

StreamSel

The stream selection parameters that were passed in **DtAdvDemod::OpenStream()**. This parameter can be used to identify the measurement values when multiple streams of measurement values are generated simultaneously.

pMeasurement

Pointer to a data buffer containing the measurement values.

Remarks

The callback function may not block and the amount of processing should be kept as low as possible to avoid stalling the advanced demodulator. In case significant processing time is required the data should be written to a temporary buffer and be processed in another thread.

DtWriteStreamFunc

Prototype of a callback function to be supplied by the user with **DtAdvDemod::RegisterCallback**. The advanced demodulator calls this function when new demodulated stream data is available. The user can process the data any way he likes, e.g. analyse the stream in real time, write to a file, etc.

```
void  DtWriteStreamFunc(
  [in] void*  pOpaque,                  // Opaque pointer
  [in] DtStreamSelPars&  StreamSel,     // Stream selection parameters
  [in] const unsigned char*  pData,     // Demodulated data
  [in] int  Length                      // Size in bytes of demodulated data
);
```

Parameters

pOpaque
The opaque pointer that was specified in **DtAdvDemod::RegisterCallback()**.

StreamSel
The corresponding stream selection parameters, passed in **DtAdvDemod::OpenStream()**.

StreamSel
The stream selection parameters that were passed in **DtAdvDemod::OpenStream()**. This parameter can be used to identify the stream when multiple data streams are generated simultaneously.

pData
Pointer to a buffer containing the demodulated data.

Length
Number of bytes available in the demodulated data buffer.

Remarks

The callback function may not block and the amount of processing should be kept as low as possible to avoid stalling the advanced demodulator. In case significant processing time is required the data should be written to a temporary buffer and be processed in another thread.

DtAdvDemod

Class representing an advanced demodulator. **DtAdvDemod** can be considered a specialized input channel.

```
class DtAdvDemod;
```

Class **DtAdvDemod** is closely related to **DtInpChannel**. The following common methods are documented in the DTAPI documentation.

```
DtAdvDemod::AttachToPort()
DtAdvDemod::ClearFlags()
DtAdvDemod::Detach()
DtAdvDemod::GetDemodControl()
DtAdvDemod::GetDescriptor()
DtAdvDemod::GetFlags()
DtAdvDemod::GetIoConfig()
DtAdvDemod::GetPars()
DtAdvDemod::GetRxControl()
DtAdvDemod::GetStatistics()
DtAdvDemod::GetSupportedPars()
DtAdvDemod::GetTunerFrequency()
DtAdvDemod::LedControl()
DtAdvDemod::Reset()
DtAdvDemod::SetAntPower()
DtAdvDemod::SetDemodControl()
DtAdvDemod::SetIoConfig()
DtAdvDemod::SetPars()
DtAdvDemod::SetRxControl()
DtAdvDemod::SetTunerFrequency()
DtAdvDemod::Tune()
```

DtAdvDemod::AttachVirtual

Set up a *virtual* I/Q input channel that lets the user supply I/Q samples through a callback function (*pReadIqFunc*), instead of DTAPI reading the data from a physical receiver device. A DekTec device has to be specified (*pDtDvc*), but this device is used only for checking the **RX_ADV** license.

```
DTAPI_RESULT DtAdvDemod::AttachVirtual(
  [in] DtDevice*  pDtDvc,          // DekTec device with RX_ADV license
  [in] DtReadIqFunc* pReadIqFunc,  // Callback for supplying I/Q samples
  [in] void*  pOpaque              // Opaque pointer passed to callback
);
```

Parameters

pDtDvc

DekTec device containing the **RX_ADV** license. The **DtDevice** object must be attached to the device hardware. The device is used only for checking licenses.

pReadIqFunc

Pointer to the user-provided callback function that will supply I/Q samples to the advanced demodulator.

pOpaque

Opaque pointer that is passed to the callback function.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	The virtual I/Q input channel has been set up successfully
DTAPI_E_ATTACHED	The advanced demodulator is already attached
DTAPI_E_DEVICE	The **DtDevice** pointer is not valid or the **DtDevice** object is not attached to the device hardware

Remarks

Virtual I/Q input channels enable usage of the advanced demodulator functions with I/Q samples from file, from non-DekTec I/Q sampling hardware or from computed I/Q samples.

DtAdvDemod::CloseStream

Closes a stream.

```
DTAPI_RESULT DtAdvDemod::CloseStream(
  [in] intptr_t  Id          // Identifier associated with the open stream
);
```

Parameters

Id

Identifies the stream that is to be closed. This is the value of the identifier that was specified in the stream selection parameters (**DtStreamSelPars.m_Id**).

Result

DTAPI_RESULT	Meaning
DTAPI_OK	Stream is closed successfully
DTAPI_E_INVALID_MODE	Demodulator is not active
DTAPI_E_NOT_ATTACHED	Advanced demodulator object is not attached
DTAPI_E_NOT_FOUND	Stream is not open

DtAdvDemod::GetStreamSelection

Returns all open streams.

```
DTAPI_RESULT DtAdvDemod::GetStreamSelection(
 [out] std::vector<StreamSelPars> &  StreamSelList  // Open streams
);
```

Parameters

StreamSelList

A vector containing all open streams.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	Open streams are returned successfully
DTAPI_E_INVALID_MODE	Demodulator is not active
DTAPI_E_NOT_ATTACHED	Advanced demodulator object is not attached

DtAdvDemod::GetTsRateBps

Get the transport-stream rate of the stream with a given ID.

```
DTAPI_RESULT DtAdvDemod::GetTsRateBps(
  [in] intptr_t  Id          // Identifies the selected stream
 [out] int&  TsRate          // Transport-stream rate in bits per second
);
```

Parameters

Id

Identifies the selected stream. This is the value of the identifier that was specified in the stream selection parameters (**DtStreamSelPars.m_Id**).

TsRate

The transport stream rate, expressed in bits per second.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	Transport-stream rate has been read successfully
DTAPI_E_NOT_ATTACHED	Advanced demodulator object is not attached
DTAPI_E_NOT_FOUND	Stream is not open
DTAPI_E_INVALID_MODE	Demodulator is not active

DtAdvDemod::OpenStream

Opens the specified stream.

```
DTAPI_RESULT DtAdvDemod::OpenStream(
  [in] DtStreamSelPars  StreamSel      // Stream selection
);
```

Parameters

StreamSel

Specifies a stream to open.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	Stream is opened successfully
DTAPI_E_IN_USE	Stream identification (*StreamSel.m_Id*) is not unique
DTAPI_E_INVALID_ARG	Invalid stream selection parameter
DTAPI_E_INVALID_MODE	Demodulator is not active or demodulator does not match with stream selection (e.g. select a DVB-T stream while DVB-C2 demodulation is active)
DTAPI_E_NOT_ATTACHED	Advanced demodulator object is not attached

DtAdvDemod::RegisterCallback

Register a callback function for handling demodulator data.

```
// Overload #1 -
// To be used for registering write measurement data callback function
DTAPI_RESULT DtAdvDemod::RegisterCallback(
  [in] DtWriteMeasFunc* pCallback,     // Callback function
  [in] void*  pOpaque                  // Opaque pointer for the callback
);
// Overload #2 -
// To be used for registering output bitrate changed callback function
DTAPI_RESULT DtAdvDemod::RegisterCallback(
  [in] DtOutputRateChangedFunc * pCallback, // Callback function
  [in] void*  pOpaque                  // Opaque pointer for the callback
);
// Overload #3 - To be used for registering stream-data callback function
DTAPI_RESULT DtAdvDemod::RegisterCallback(
  [in] DtWriteStreamFunc* pCallback,   // Callback function
  [in] void*  pOpaque                  // Opaque pointer for the callback
);
```

Parameters

pCallback

Pointer to a callback function for handling the measurement-data, bitrate and stream-data. Use **NULL** to unregister the callback.

pOpaque

Opaque pointer that is passed to the callback function.

Result

DTAPI_RESULT	Meaning
DTAPI_OK	Callback has been registered or unregistered successfully
DTAPI_E_NOT_ATTACHED	Advanced demodulator object is not attached