

*Written by:*

**Aleksandar Andjelkovic**

Tel.: +381 (0)62 88 80 051

E-mail: aleksandar.andjelkovic@iwedia.com

*Approved by:*

**Aleksandar Andjelkovic**

Tel.: +381 (0)62 88 80 051

E-mail: aleksandar.andjelkovic@iwedia.com

*Viewed by:*

**Djordje Glisic**

Tel.: +381 (0)21 48 01 278

E-mail: philippe.poher@iwedia.com

**TDAL\_MP Technical Specifications**

SUMMARY

[1 Reference documents 3](#_Toc444162462)

[2 Documentation History 4](#_Toc444162463)

[3 Introduction 5](#_Toc444162464)

[4 Module Overview and Architecture 6](#_Toc444162465)

[4.1 Definitions, Acronyms and Abbreviations 6](#_Toc444162466)

[4.2 General Description 7](#_Toc444162467)

[4.2.1 Interactions with other TDAL modules 8](#_Toc444162468)

[4.3 Architecture 8](#_Toc444162469)

[4.4 TDAL\_MP API Description 9](#_Toc444162470)

[4.4.1 Generic Interface 9](#_Toc444162471)

[4.4.2 Control Interface 9](#_Toc444162472)

[4.4.3 Status Interface 9](#_Toc444162473)

[4.4.4 Window Interface 10](#_Toc444162474)

[4.4.5 Track selection Interface 10](#_Toc444162475)

[4.4.6 Memory Allocation Interface 10](#_Toc444162476)

[4.4.7 Memory Allocation Interface 10](#_Toc444162477)

[5 TDAL\_MP API Specification 12](#_Toc444162478)

[5.1 Type Specifications 12](#_Toc444162479)

[5.1.1 Generic 12](#_Toc444162480)

[5.1.2 Control 12](#_Toc444162481)

[5.1.3 Window 12](#_Toc444162482)

[5.1.4 Track 12](#_Toc444162483)

[5.2 TDAL\_KBD API Usage 14](#_Toc444162484)

[5.2.1 Event reception processing 14](#_Toc444162485)

[5.3 TDAL\_KBD Macro/Define/Enum/Typedef/Structure Definitions 15](#_Toc444162486)

[5.3.1 Define Definitions 15](#_Toc444162487)

[5.3.2 Enum Definitions 16](#_Toc444162488)

[5.3.3 Typedef Definitions 18](#_Toc444162489)

[5.3.4 Structure Definitions 21](#_Toc444162490)

[5.4 TDAL\_MP API Functions 24](#_Toc444162491)

[5.4.1 Generic Interface API 24](#_Toc444162492)

[6 TDAL\_KBD External Dependencies 43](#_Toc444162493)

# Reference documents

The following documents were used to write this specification:

|  |  |  |
| --- | --- | --- |
| **Document** | **Version** | **Title** |
| TDAL\_MP Technical Specifications | 1.0 | Initial version |
| TDAL\_MP Technical Specifications | 1.1 | Metadata parsing update |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Documentation History

|  |  |  |
| --- | --- | --- |
| **History** | | |
| Vers. | Date | Nature |
| 1.0 | 22/02/2016 | DJG: First version of TDAL\_MP |
| 1.1 | 22/09/2016 | Metadata parsing update |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

The purpose of this document is to specify the interface and to describe the internal design of the Thin Driver Adaptation Layer for Media Player [TDAL\_MP]. The aim of this module is to provide a complete interface between middlewares and the media player stage low level driver of a set-top-box.

# Module Overview and Architecture

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| CHAL | Comedia hardware Adaptation Layer, composed of TDALs driver, TKEL and TBOX module |
| TDAL | Thin Driver Adaptation Layer |
| TKEL | Thin Kernel Encapsulation Layer |
| TBOX | Tool Box |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## General Description

The main job of the TDAL\_MP is to control audio/video playback.

Module allows user to play, pause, stop or rewind or fast forward reproduction.

The client gives the key code array of button managed by the application.

### Interactions with other TDAL modules

## Architecture

The figure below shows the relation between the client module and the low level drivers:



## TDAL\_MP API Description

### Generic Interface

The generic interface allows clients to control the TDAL\_MP module.

This interface is used with the following functions:

* **TDAL\_MP\_Init(-)** in order to initialize the TDAL\_MP module.
* **TDAL\_MP\_Terminate(-)** in order to terminate the TDAL\_MP module.
* **TDAL\_MP\_RevisionGet(-)** in order to get version of TDAL\_MP.
* **TDAL\_MP\_Open(-)** in order to open media content.
* **TDAL\_MP\_Close(-)** in order to close media content.
* **TDAL\_MP\_CapabilityGet(-)** in order to get capabilities of TDAL\_MP instance.
* **TDAL\_MP\_PictureGetMetadata(-)** in order to getpicture metadata
* **TDAL\_MP\_AudioGetMetadata(-)** in order to get audio metadata
* **TDAL\_MP\_** **AVGetMetadata(-)** in order to get AV metadata
* **TDAL\_MP\_OutputSet(-)** in order to set output end for media content being played.

### Control Interface

The control interface allows clients to control the TDAL\_MP instance. This interface is used with the following functions:

* **TDAL\_MP\_Start (-)** in order to start the TDAL\_MP playback.
* **TDAL\_MP\_Stop (-)** in order to stop the TDAL\_MP playback.
* **TDAL\_MP\_PlayCtrlModeSet (-)** in order to initialize the TDAL\_MP module.
* **TDAL\_MP\_PosSet (-)** in order to set playback position for the TDAL\_MP instance.
* **TDAL\_MP\_PosGet (-)** in order to get playback position for the TDAL\_MP instance.
* **TDAL\_MP\_SpeedSet (-)** in order to set speed of playback for the TDAL\_MP instance.
* **TDAL\_MP\_SpeedGet (-)** in order to get speed of playback for the TDAL\_MP instance.

### Status Interface

The control interface allows clients to control output window and output end for the TDAL\_MP instance. This interface is used with the following functions:

* **TDAL\_MP\_StatusGet(-)** in order to get status of playback for the TDAL\_MP instance.
* **TDAL\_MP\_LengthGet (-)** in order to get total duration of playback for the TDAL\_MP instance.

### Window Interface

The control interface allows clients to control output window for output of the TDAL\_MP instance. This interface is used with the following functions:

* **TDAL\_MP\_ScaleWndSet(-)** in order to set output window for output of the TDAL\_MP instance.
* **TDAL\_MP\_ScaleWndGet(-)**in order to get output window for output of the TDAL\_MP instance.
* **TDAL\_MP\_ResolutionGet(-)** in order to get resolution for the TDAL\_MP instance.

### Track selection Interface

The control interface allows clients to control track for the TDAL\_MP instance. This interface is used with the following functions:

* **TDAL\_MP\_TracksGet (-)** in order to get list of tracks and currently selected track from the TDAL\_MP instance.
* **TDAL\_MP\_TracksRelease (-)** in order to free track list allocated by TDAL\_MP\_TracksGetfunction.
* **TDAL\_MP\_TrackSelect (-)** in order to select track for playback with the TDAL\_MP instance.

### Memory Allocation Interface

The control interface allows clients to control track for the TDAL\_MP instance. This interface is used with the following functions:

* **TDAL\_MP\_Malloc(-)** to allocate memory for the TDAL\_MP memory pool.
* **TDAL\_MP\_Free(-)** to dallocate memory for the TDAL\_MP memory pool.
* **TDAL\_MP\_GetAvailableSpace (-)** to dallocate memory for the TDAL\_MP memory pool.

# TDAL\_MP API Specification

## Type Specifications

### Generic

**tTDAL\_MP\_TraceId** represents trace level for TBOX logger.

**tTDAL\_MP\_Capabilities** holds information about capabilities for the TDAL\_MP instance.

**tTDAL\_MP\_StreamType** is enumeration that represents stream types.

**tTDAL\_MP\_ContentType** is enumeration that represents audio, video, or subtitle endecoding standards.

**tTDAL\_MP\_Status** enum representing status of the playback.

**tTDAL\_MP\_PlayCtrlMode** represents playback modes.

**tTDAL\_MP\_Event** represents client notification event.

**tTDAL\_MP\_SubtitleType** is enumeration defining subtitle types.

**tTDAL\_MP\_SeekMode** represents supported seek modes.

**tTDAL\_MP\_Handle** is a representation of TDAL\_MP instance.

**tTDAL\_MP\_InputDataRead** is the callback to implement by the client for reading input data.

**tTDAL\_MP\_InputDataSeek** is the callback to implement by the client for seeking input data.

**tTDAL\_MP\_InputDataLength** is the callback to implement by the client for calculating length of input data.

**tTDAL\_MP\_EvtNotify** is the callback to implement by the client to receive event notification from TDAL\_MP instace.

### Control

**tTDAL\_MP\_InitParams** contains information for initialization of TDAL\_MP instance.

**tTDAL\_MP\_OpenParams** contains information for opening media content with TDAL\_MP instance.

**tTDAL\_MP\_StartParams** contains information for starting playback of media content using TDAL\_MP instance.

### Window

**tTDAL\_MP\_WndPosition** defines coordinates.

**tTDAL\_MP\_Wnd** defines position and size of the window.

### Track

**tTDAL\_MP\_ISO639LangCode** is the language code by ISO639 standard.

**tTDAL\_MP\_AudioTrack** contains information about audio track.

**tTDAL\_MP\_SubtitleTrack** contains information about subtitle track.

**tTDAL\_MP\_VideoTrack** contains information about video track.

**tTDAL\_MP\_TrackList** is the list of tracks for specific type (audio,video or subtitle).

## TDAL\_MP API Usage

### Open play pause and stop AV playback



### Fast forward, rewind and event on reaching end of stream



## TDAL\_MP Macro/Define/Enum/Typedef/Structure Definitions

### Define Definitions

### Enum Definitions

#### tTDAL\_MP\_Error;

An enumeration of the different possible error value handled by TDAL\_MP.

typedef enum {

eTDAL\_MP\_NO\_ERROR,

eTDAL\_MP\_NOT\_INIT\_ERROR,

eTDAL\_MP\_BAD\_PARAMETER\_ERROR,

eTDAL\_MP\_NOT\_SUPPORTED\_ERROR,

eTDAL\_MP\_NO\_MEMORY\_ERROR,

eTDAL\_MP\_NB\_HANDLE\_MAX\_ERROR, /\* max session handle reached \*/

eTDAL\_MP\_NOT\_DONE\_ERROR

}tTDAL\_MP\_Error;

#### tTDAL\_MP\_TraceId

An enumeration that represents trace level for TBOX logger.

typedef enum {

eTDAL\_MP\_TRACE\_GENERIC\_ID = 0x045F0000,

eTDAL\_MP\_TRACE\_CONTROL\_ID = 0x045F0100,

eTDAL\_MP\_TRACE\_STATUS\_ID = 0x045F0200,

eTDAL\_MP\_TRACE\_TRACKS\_ID = 0x045F0300,

eTDAL\_MP\_TRACE\_WINDOW\_ID = 0x045F0400

}tTDAL\_MP\_TraceId;

#### tTDAL\_MP\_Capabilities

An enumeration that holds information about capabilities for the TDAL\_MP instance.

typedef enum {

eTDAL\_MP\_CAPS\_BASIC = 0x00000000, /\* always basic ops (Play, Stop) \*/

eTDAL\_MP\_CAPS\_POS = 0x00000001, /\* supports PosSet \*/

eTDAL\_MP\_CAPS\_SCALE = 0x00000002, /\* can scale the video \*/

eTDAL\_MP\_CAPS\_SPEED = 0x00000004, /\* supports changing playback speed \*/

eTDAL\_MP\_CAPS\_TRACK = 0x00000008 /\* supports changing of tracks \*/

}tTDAL\_MP\_Capabilities;

#### tTDAL\_MP\_StreamType

An enumeration that represents stream types.

typedef enum {

eTDAL\_MP\_STREAM\_TYPE\_UNKNOWN = -1,

eTDAL\_MP\_STREAM\_TYPE\_VIDEO,

eTDAL\_MP\_STREAM\_TYPE\_AUDIO,

eTDAL\_MP\_STREAM\_TYPE\_DATA,

eTDAL\_MP\_STREAM\_TYPE\_SUBTITLE

}tTDAL\_MP\_StreamType;

#### tTDAL\_MP\_ContentType

An enumeration that represents audio, video, or subtitle endecoding standards.

typedef enum {

eTDAL\_MP\_STREAM\_CONTENT\_INVALID, /\* None : Invalid type \*/

eTDAL\_MP\_STREAM\_CONTENT\_MP1V, /\* Video : MPEG1 \*/

eTDAL\_MP\_STREAM\_CONTENT\_MP2V, /\* Video : MPEG2 \*/

eTDAL\_MP\_STREAM\_CONTENT\_VC1, /\* Video : VC-1 Decode SimpleMain/Advanced profiles \*/

eTDAL\_MP\_STREAM\_CONTENT\_H264, /\* Video : H264 \*/

eTDAL\_MP\_STREAM\_CONTENT\_MPEG4P2, /\* Video : MPEG4 Part II \*/

eTDAL\_MP\_STREAM\_CONTENT\_VMW, /\* Video : MPEG4 Part II \*/

eTDAL\_MP\_STREAM\_CONTENT\_MP1A, /\* Audio : MPEG1 Layer I \*/

eTDAL\_MP\_STREAM\_CONTENT\_MP2A, /\* Audio : MPEG1 Layer II \*/

eTDAL\_MP\_STREAM\_CONTENT\_MP3A, /\* Audio : MPEG1 Layer III \*/

eTDAL\_MP\_STREAM\_CONTENT\_AC3, /\* Audio : AC3 \*/

eTDAL\_MP\_STREAM\_CONTENT\_WMA, /\* Audio : AC3 \*/

eTDAL\_MP\_STREAM\_CONTENT\_AAC, /\* Audio : Decode ADTS - AAC \*/

eTDAL\_MP\_STREAM\_CONTENT\_HEAAC, /\* Audio : Decoder LOAS/LATM-AAC \*/

eTDAL\_MP\_STREAM\_CONTENT\_WMA, /\* Audio : Windows Media Audio\*/

eTDAL\_MP\_STREAM\_CONTENT\_DDPLUS, /\* Audio : Windows Media Audio \*/

eTDAL\_MP\_STREAM\_CONTENT\_DTS, /\* Audio : Digital Theater Systems \*/

eTDAL\_MP\_STREAM\_CONTENT\_PCM, /\* Audio : Pulse-code modulation \*/

eTDAL\_MP\_STREAM\_CONTENT\_TTXT, /\* Teletext : Teletext \*/

eTDAL\_MP\_STREAM\_CONTENT\_SUBT, /\* Subtitle : Subtitle \*/

eTDAL\_MP\_STREAM\_CONTENT\_OTHER /\* Misc : Non identified \*/} tTDAL\_MP\_ContentType;

#### tTDAL\_MP\_Status

An enumumeration that represents status of the playback.

typedef enum {

eTDAL\_MP\_STATUS\_UNKNOWN = 0x00000000, /\* unknown status \*/

eTDAL\_MP\_STATUS\_PLAY = 0x00000001, /\* playing \*/

eTDAL\_MP\_STATUS\_STOP = 0x00000002, /\* stopped \*/

eTDAL\_MP\_STATUS\_FINISHED = 0x00000003 /\* playback is finished \*/

} tTDAL\_MP\_Status;

#### tTDAL\_MP\_PlayCtrlMode

An enumeration that represents playback modes.

typedef enum {

eTDAL\_MP\_PLAY\_CTRL\_MODE\_NORMAL = 0x00000000, /\* normal playback \*/

eTDAL\_MP\_PLAY\_CTRL\_MODE\_REWIND = 0x00000001, /\* reverse playback \*/

eTDAL\_MP\_PLAY\_CTRL\_MODE\_LOOPING = 0x00000002 /\* automatically restart

playback when

end-of-stream is

reached (gapless). \*/

} tTDAL\_MP\_PlayCtrlMode;

#### tTDAL\_MP\_SubtitleType

An enumeration defining subtitle types.

typedef enum {

eTDAL\_MP\_SUBTITLES\_NORMAL = 0,

eTDAL\_MP\_SUBTITLES\_HH

} tTDAL\_MP\_SubtitleType;

#### tTDAL\_MP\_Event

An enumeration that represents client notification event.

typedef enum {

eTDAL\_MP\_EVENT\_PLAYED, /\* play control done \*/

eTDAL\_MP\_EVENT\_STOPPED, /\* stop control done \*/

eTDAL\_MP\_EVENT\_FINISHED, /\* rendering of media finished \*/

eTDAL\_MP\_EVENT\_REWINDED, /\* rewinding of media finished \*/

eTDAL\_MP\_EVENT\_LOOP, /\* looping in media done \*/

eTDAL\_MP\_EVENT\_NEW\_FRAME, /\* a new frame has been presented \*/

eTDAL\_MP\_EVENT\_UNDERFLOW, /\* Input buffer has reached low level threshold \*/

eTDAL\_MP\_EVENT\_NO\_MEMORY, /\* Not enough memory for rendering process \*/

eTDAL\_MP\_EVENT\_STREAM\_ERROR /\* Bad incoming data stream has been detected \*/

} tTDAL\_MP\_Event;

#### tTDAL\_MP\_SeekMode

An enumeration that represents supported seek modes.

typedef enum {

eTDAL\_MP\_SEEK\_MODE\_SET = 0, /\* Beginning of file \*/

eTDAL\_MP\_SEEK\_MODE\_CUR, /\* Current position of the stream pointer \*/

eTDAL\_MP\_SEEK\_MODE\_END /\* End of file \*/

}tTDAL\_MP\_SeekMode;

#### tTDAL\_MP\_Output

typedef enum {

eTDAL\_MP\_OUTPUT\_MAIN = 0x00000001, /\* Main display destination \*/

eTDAL\_MP\_OUTPUT\_AUX = 0x00000002 /\* Auxiliary display destination (e.g. VCR or 2nd TV) \*/

} tTDAL\_MP\_Output;

### Typedef Definitions

#### tTDAL\_MP\_Handle

A representation of TDAL\_MP instance.

typedef uint32\_t tTDAL\_MP\_Handle;

#### tTDAL\_MP\_InputDataRead

The callback to implement by the client for reading input data.

typedef uint32\_t (\*tTDAL\_MP\_InputDataRead)(

tTDAL\_MP\_Handle Handle,

uint8\_t \*pucBuffer,

uint32\_t ulSize

);

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| \*pucBuffer | uint8\_t | In | Pointer to buffer |
| ulSize | uint32\_t | In | Size of buffer |

#### tTDAL\_MP\_InputDataSeek

The callback to implement by the client for seeking input data.

typedef uint32\_t (\*tTDAL\_MP\_InputDataSeek)(

tTDAL\_MP\_Handle Handle,

int32\_t llOffset,

tTDAL\_MP\_SeekMode eMode

);

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| llOffset | uint32\_t | In | Offset in bytes for seek |
| eMode | tTDAL\_MP\_SeekMode | In | Seek mode |

#### tTDAL\_MP\_InputDataLength

The callback to implement by the client for calculating length of input data.

typedef uint32\_t (\*tTDAL\_MP\_InputDataLength)(tTDAL\_MP\_Handle Handle);

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |

#### tTDAL\_MP\_EvtNotify

The callback to implement by the client to receive event notification from TDAL\_MP instace.

typedef uint32\_t (\*tTDAL\_MP\_InputDataSeek)(

tTDAL\_MP\_Handle Handle,

tTDAL\_MP\_Event eEvt,

const void \*pCtx

);

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In |  |
| eEvt | tTDAL\_MP\_Event | In |  |
| \*pCtx | const void | In |  |

### Structure Definitions

#### tTDAL\_MP\_InitParams

An structure that contains information for initialization of TDAL\_MP instance.

typedef struct {

uint32\_t uiDummy;

}tTDAL\_MP\_InitParams;

#### tTDAL\_MP\_OpenParams

An structure that contains information for opening media content with TDAL\_MP instance.

typedef struct {

uint32\_t dummy;

tTDAL\_MP\_InputDataRead InputDataRead;

tTDAL\_MP\_InputDataSeek InputDataSeek;

tTDAL\_MP\_InputDataLength InputDataLength;

tTDAL\_MP\_EvtNotify EvtNotify;

void \*pCtx;

tTDAL\_MP\_Output eDefaultOutput;

uint8\_t \*pPrivDataBuf;

uint32\_t uiPrivDataSize;

char\* fileExtension;

/\* Stream media info \*/

bool bIsStreamed;

tTDAL\_MP\_ContentType eStreamVidType,

tTDAL\_MP\_ContentType eStreamAudType;

} tTDAL\_MP\_OpenParams;

#### tTDAL\_MP\_StartParams

An structure that contains information for starting playback of media content using TDAL\_MP instance.

typedef struct {

uint32\_t uiTime; /\* Start time position in the stream (in millisecond) \*/

int32\_t iSpeed; /\* Start speed. Values below 100 reduce playback speed

while values over 100 increase it. Specifying a value

of 0 has the effect of putting the playback in pause

mode without stopping the provider \*/

} tTDAL\_MP\_StartParams;

#### tTDAL\_MP\_WndPosition

An structure that defines coordinates.

typedef struct {

int16\_t usLeftSide; /\* in pixels from the left \*/

int16\_t usTopSide; /\* in lines from the top \*/

uint16\_t usRightSide; /\* in pixels from the left \*/

uint16\_t usBottomSide; /\* in lines from the top \*/

} tTDAL\_MP\_StartParams;

#### tTDAL\_MP\_Wnd

An structure that defines position and size of the window.

typedef struct {

tTDAL\_MP\_WndPosition stOutWnd; /\* "virtual" output window \*/

tTDAL\_MP\_WndPosition stDispWnd; /\* "real" display window \*/

} tTDAL\_MP\_StartParams;

#### tTDAL\_MP\_ISO639LangCode

An structure that represents the language code by ISO639 standard.

typedef struct {

tTDAL\_MP\_WndPosition stOutWnd; /\* "virtual" output window \*/

tTDAL\_MP\_WndPosition stDispWnd; /\* "real" display window \*/

} tTDAL\_MP\_StartParams;

#### tTDAL\_MP\_AudioTrack

An structure that contains information about audio track.

typedef struct {

tTDAL\_MP\_ContentType eType; /\* Type of audio \*/

tTDAL\_MP\_ISO639LangCode stLang; /\* ISO language of the audio track. \*/

uint16\_t usIdx; /\* Index of the audio track in the list. \*/

} tTDAL\_MP\_AudioTrack;

#### tTDAL\_MP\_SubtitleTrack

An structure that contains information about subtitle track.

typedef struct {

tTDAL\_MP\_SubtitleType eType; /\* Type of subtitle. \*/

tTDAL\_MP\_ISO639LangCode stLang; /\* ISO language of the track \*/

uint16\_t usCompositionPageID; /\* Composition page ID of the track \*/

uint16\_t usAncillaryPageID; /\* Ancillary page ID of the track \*/

uint16\_t usIdx; /\* Index of the track in the list \*/

} tTDAL\_MP\_SubtitleTrack;

#### tTDAL\_MP\_VideoTrack

An structure that contains information about video track.

typedef struct {

tTDAL\_MP\_ContentType eType; /\* Type of video \*/

uint16\_t usIdx; /\* Index of the video track in the list. \*/

} tTDAL\_MP\_VideoTrack;

#### tTDAL\_MP\_TrackList

An structure that represents the list of tracks for specific type (audio,video or subtitle).

typedef struct {

uint16\_t usTrackCount; /\* Number of tracks in the list \*/

void \*pTrackList; /\* Pointer to the track list that must be cast in the

track video, audio or subtitle type structure \*/

} tTDAL\_MP\_VideoTrack;

## TDAL\_MP API Functions

### Generic Interface API

#### TDAL\_MP\_Init(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_Init(tTDAL\_MP\_InitParams \*pstInitParams);

*Description:*

This function initialises the TDAL\_MP module.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR if initialization succeed.

It returns eTDAL\_MP\_NOT\_DONE\_ERROR otherwise.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| \*pstInitParams | tTDAL\_MP\_InitParams | In |  |

*Comments:*

This function must be called before any other functions of the TDAL\_MP API.

*See Also:*

TDAL\_MP\_Terminate(-).

#### TDAL\_MP\_Terminate(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_Terminate (void);

*Description:*

This function terminates the MP module.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR if termination succeed.

It could eTDAL\_MP\_NOT\_INIT\_ERROR if a previous initialization has not been done.

It returns eTDAL\_MP\_NOT\_DONE\_ERROR otherwise.

*Parameters:*

None.

*Comments:*

This function frees resources and terminates TDAL\_MP module. Any function of TDAL\_MP API should not be called after TDAL\_MP\_Terminate(-) call.

*See Also:*

TDAL\_MP\_Init(-).

#### TDAL\_MP\_RevisionGet(-)

*Prototype:*

char \*TDAL\_MP\_RevisionGet(void);

*Description:*

This function returns version of TDAL\_MP module.

*Return value:*

It returns string.

*Parameters:*

None.

*Comments:*

*See Also:*

#### TDAL\_MP\_Open(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_Open(

tTDAL\_MP\_OpenParams \*pstParams,

tTDAL\_MP\_Handle \*pHandle

);

*Description:*

This function is called in case of opening media content.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR if opening is succeed.

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if any parameter is NULL.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| \*pstParams | tTDAL\_MP\_OpenParams | In | Pointer to structure with callback function, media informations and pointer to buffer |
| \*pHandle | tTDAL\_MP\_Handle | Out | It will be setted in this function |

*Comments:*

This function calls before TDAL\_MP\_Start(-).

*See Also:*

TDAL\_MP\_Close(-)

#### TDAL\_MP\_Close(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_Close(tTDAL\_MP\_Handle Handle);

*Description:*

This function calls in order to close media content.

*Return value:*

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if parameter Handle is NULL.

It returns eTDAL\_MP\_NO\_ERROR if termination succeed.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |

*Comments:*

*See Also:*

TDAL\_MP\_Open(-)

#### TDAL\_MP\_CapabilityGet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_CapabilityGet(

tTDAL\_MP\_Handle Handle,

tTDAL\_MP\_Capabilities \*pstCaps

);

*Description:*

This function sets forwarded tTDAL\_MP\_Capabilities depending on the implemented capabilities of TDAL\_MP.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| \*pstCaps | tTDAL\_MP\_Capabilities | In | Pointer to variable which should set. |

*Comments:*

*See Also:*

...

#### TDAL\_MP\_OutputSet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_ OutputSet(

tTDAL\_MP\_Handle Handle,

uint32\_t uiOutputMask

);

*Description:*

This function sets output end for media content being played.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| uiOutputMask | uint32\_t | In | Mask whic describes output |

*Comments:*

*See Also:*

### Control Interface API

#### TDAL\_MP\_Start(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_Start(

tTDAL\_MP\_Handle Handle,

tTDAL\_MP\_StartParams \*pstParams

);

*Description:*

This function starts playback of opened media content.

*Return value:*

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if any argument is NULL.

It returns eTDAL\_MP\_NOT\_DONE\_ERROR if playback is not started.

Otherwise, it returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| \*pstParams | tTDAL\_MP\_StartParams | In | Pointer to structure with playback parameters |

*Comments:*

*See Also:*

TDAL\_MP\_Stop(-)

#### TDAL\_MP\_Stop(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_Stop(tTDAL\_MP\_Handle Handle);

*Description:*

This function stops playback of opened media content.

*Return value:*

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if argument is NULL.

Otherwise, it returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |

*Comments:*

*See Also:*

TDAL\_MP\_Start(-)

#### TDAL\_MP\_PlayCtrlModeSet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_PlayCtrlModeSet(

tTDAL\_MP\_Handle Handle,

uint32\_t uiMode

);

*Description:*

This function initializes the TDAL\_MP module.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| uiMode | uint32\_t | In | Mode of TDAL\_MP |

*Comments:*

*See Also:*

#### TDAL\_MP\_PosSet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_PosSet(

tTDAL\_MP\_Handle Handle,

uint32\_t uiSec

);

*Description:*

This function sets playback position for the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if any parameter is NULL or position uiSec is incorrect.

Otherwise, it returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| uiSec | uint32\_t | In | Position in seconds |

*Comments:*

Playback continues from the desired postion.

*See Also:*

TDAL\_MP\_PosGet(-)

#### TDAL\_MP\_PosGet

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_PosGet(

tTDAL\_MP\_Handle Handle,

uint32\_t \*puiSec

);

*Description:*

This function gets playback position for the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if any parameter is NULL.

Otherwise, it returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| \*puiSec | uint32\_t | Out | Pointer to variable which will be setted |

*Comments:*

*See Also:*

TDAL\_MP\_PosSet(-)

#### TDAL\_MP\_SpeedSet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_SpeedSet(

tTDAL\_MP\_Handle Handle,

uint32\_t uiSpeed

);

*Description:*

This function sets playback speed for the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if any parameter is NULL or speed uiSpeed is incorrect.

Otherwise, it returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| uiSpeed | uint32\_t | In | Playback speed |

*Comments:*

Playback continues with desired speed.

*See Also:*

TDAL\_MP\_SpeedGet(-)

#### TDAL\_MP\_SpeedGet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_SpeedGet(

tTDAL\_MP\_Handle Handle,

uint32\_t \*puiSpeed

);

*Description:*

This function gets playback speed for the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if any parameter is NULL.

Otherwise, it returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| \*puiSpeed | uint32\_t | Out | Pointer to variable which will be setted |

*Comments:*

*See Also:*

TDAL\_MP\_SpeedSet(-)

### Status Interface API

#### TDAL\_MP\_StatusGet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_StatusGet(

tTDAL\_MP\_Handle Handle,

tTDAL\_MP\_Status \*pstStatus

);

*Description:*

This function gets status of playback for the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| \*pstStatus | tTDAL\_MP\_Status | Out | Pointer to variable which will be setted |

*Comments:*

It get status of playback (is it playing, stopped, finished).

*See Also:*

#### TDAL\_MP\_LengthGet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_LengthGet(

tTDAL\_MP\_Handle Handle,

uint32\_t \*puiSec

);

*Description:*

This function gets total duration in seconds of playback for the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| \*puiSec | uint32\_t | Out | Pointer to variable which will be setted |

*Comments:*

*See Also:*

### Window Interface API

#### TDAL\_MP\_ScaleWndSet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_ScaleWndSet(

tTDAL\_MP\_Handle Handle,

tTDAL\_MP\_Output eOutput,

tTDAL\_MP\_Wnd \*pstWnd

);

*Description:*

This function sets output window for output of the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| eOutput | tTDAL\_MP\_Output | In | Output destination |
| \*pstWnd | tTDAL\_MP\_Wnd | In | Window position |

*Comments:*

*See Also:*

#### TDAL\_MP\_ScaleWndGet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_ScaleWndGet(

tTDAL\_MP\_Handle Handle,

tTDAL\_MP\_Output eOutput,

tTDAL\_MP\_Wnd \*pstWnd

);

*Description:*

This function gets output window for output of the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| eOutput | tTDAL\_MP\_Output | In | Output destination |
| \*pstWnd | tTDAL\_MP\_Wnd | Out | Pointer to variable to set windows position |

*Comments:*

*See Also:*

#### TDAL\_MP\_ResolutionGet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_ResolutionGet(

tTDAL\_MP\_Handle Handle,

uint16\_t \*pusWidth,

uint16\_t \*pusHeight

);

*Description:*

This function gets resolution for the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| \*pusWidth | uint16\_t | In | Pointer to variable to get width |
| \*pusHeight | uint16\_t | In | Pointer to variable to get height |

*Comments:*

*See Also:*

### Track selection Interface API

#### TDAL\_MP\_TracksGet(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_TracksGet(

tTDAL\_MP\_Handle Handle,

tTDAL\_MP\_StreamType eType,

tTDAL\_MP\_TrackList \*\*ppstList,

uint16\_t \*pusSelectedIdx

);

*Description:*

This function gets the list of tracks and currently selected track from the TDAL\_MP instance.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| eType | tTDAL\_MP\_StreamType | In | Type of media content in tracklist |
| \*\*ppstList | tTDAL\_MP\_TrackList | Out | Pointer to tracklist variable to get tracklist |
| \*pusSelectedIdx | uint16\_t | Out | Pointer to variable to set currently selected track |

*Comments:*

*See Also:*

TDAL\_MP\_TracksRelease(-)

#### TDAL\_MP\_TracksRelease(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_TracksRelease(

tTDAL\_MP\_Handle Handle,

tTDAL\_MP\_TrackList \*pstList,

);

*Description:*

This function to frees track list allocated by TDAL\_MP\_TracksGetfunction.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| \*pstList | tTDAL\_MP\_TrackList | In | Pointer to tracklist |

*Comments:*

*See Also:*

#### TDAL\_MP\_TrackSelect(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_TrackSelect(

tTDAL\_MP\_Handle Handle,

tTDAL\_MP\_StreamType eType,

uint16\_t usSelectedIdx

);

*Description:*

This function selects track for playback.

*Return value:*

It returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| Handle | tTDAL\_MP\_Handle | In | Handle of TDAL\_MP instance |
| eType | tTDAL\_MP\_StreamType | In | Type of media content |
| usSelectedIdx | uint16\_t | In | Track index in tracklist |

*Comments:*

*See Also:*

### Memory Allocation Interface API

#### TDAL\_MP\_Malloc(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_Malloc(

uint32\_t size,

void \*\*p

);

*Description:*

This function allocates memory for the TDAL\_MP memory pool.

*Return value:*

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if size is 0 or p is NULL.

It returns eTDAL\_MP\_NO\_MEMORY\_ERROR if it could not allocate memory.

Otherwise, it returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | **Description** |
| size | uint32\_t | In | Memory size |
| \*\*p | void | Out | Pointer to allocated memory |

*Comments:*

*See Also:*

TDAL\_MP\_Free(-)

#### TDAL\_MP\_Free(-)

*Prototype:*

tTDAL\_MP\_Error TDAL\_MP\_Free(void \* p);

*Description:*

This function dallocates memory for the TDAL\_MP memory pool.

*Return value:*

It returns eTDAL\_MP\_BAD\_PARAMETER\_ERROR if p is NULL.

Otherwise, it returns eTDAL\_MP\_NO\_ERROR.

*Parameters:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Type** | **Dir** | | **Description** |
| \*p | void | In | Pointer to memory to dallocate | |

*Comments:*

*See Also:*

TDAL\_MP\_Malloc(-)

# TDAL\_MP External Dependencies