MFPlayer Library

mrfearless

Jan 21, 2025

CONTENTS:

CHAPTER

ONE

INTRODUCTION

1.1 Overview

MFPlayer Library consists of functions that wrap the MFPlay COM implementation of the IMFPMediaPlayer and IMFPMediaItem objects. MFPlay is a Microsoft Media Foundation API for creating media playback applications. Thus the MFPlayer Library functions hide the complexities of interacting with the COM objects.

The MFPlayer library and source code are free to use for anyone, and anyone can contribute to the MFPlayer Library project.

1.1.1 Download

The MFPlayer Library is available to download from the github repository at github.com/mrfearless/MFPlayer-Library

1.1.2 Installation

See the *Installation & Setup* section for more details.

1.1.3 Contribute

If you wish to contribute, please follow the *Contributing* guide for details on how to add or edit, the MFPlayer Library source or documentation.

1.1.4 Frequently Asked Questions

Please visit the *Frequently Asked Questions* section for details.

1.2 Installation & Setup

1.2.1 MASM for 32bit assembler

- Download and install the MASM32 package.
- Download the 32bit version of the MFPlayer Library: MFPlayer-x86.zip
- Copy the `MFPlayer.inc` file to `X:\MASM32\Include` folder overwriting any previous versions.
- Copy the `MFPlayer.lib` file to `X:\MASM32\Lib` folder overwriting any previous versions.

Note: `X` is the drive letter where the MASM32 package has been installed to.

Adding MFPlayer Library to your MASM project

You are now ready to begin using the MFPlayer library in your Masm projects. Simply add the following lines to your project:

```
include MFPlayer.inc
includelib MFPlayer.lib
```

1.2.2 UASM for 64bit assembler

- Download and install the UASM assembler. Ideally you will have a setup that mimics the MASM32 setup, where you create manually folders for `bin`, `include` and `lib`
- Download the 64bit version of the MFPlayer Library: MFPlayer-x64.zip
- Copy the `MFPlayer.inc` file to `X:\UASM\Include` folder overwriting any previous versions.
- Copy the `MFPlayer.lib` file to `X:\UASM\Lib\x64` folder overwriting any previous versions.

Note: `X` is the drive letter where the UASM package has been installed to.

Adding MFPlayer Library to your UASM project

You are now ready to begin using the MFPlayer library in your Uasm projects. Simply add the following lines to your project:

```
include MFPlayer.inc
includelib MFPlayer.lib
```

Note: See the following for more details on setting up UASM to work with RadASM and other details that may be useful in creating a development environment that mimics the MASM32 SDK: UASM-with-RadASM, UASM-SDK

Tip: UASM can be used as a x86 32 bit assembler as well.

1.3 Building

The MFPlayer Libraries (both x86 and x64 versions) come with RadASMs project to help you build the sources. However if you wish to build the libraries manually, here are the command line options you should use.

1.3.1 Building the MFPlayer Library x86

The MFPlayer Library x86 source consists of the following files:

- MFPlayer.inc
- MFPlayer.asm

Building with Microsoft MASM (ML.EXE):

```
ML.EXE /c /coff /Cp /nologo /I"X:\MASM32\Include" *.asm
```

Note: X is the drive letter where the MASM32 package has been installed to.

Linking with Microsoft Library Manager (LIB.EXE):

```
LIB *.obj /out:MFPlayer.lib
```

1.3.2 Building the MFPlayer Library x86 Debug Build

To build the MFPlayer Library x86 with debug information, supply the additional flag options /Zi /Zd on the command line for MASM (ML.EXE) like so:

```
ML.EXE /c /coff /Cp /Zi /Zd /nologo /I"X:\MASM32\Include" *.asm
```

Note: X is the drive letter where the MASM32 package has been installed to.

1.3.3 Building the MFPlayer Library x64

The MFPlayer Library x64 source consists of the following files:

- MFPlayer.inc
- MFPlayer.asm

Building with UASM (UASM64.EXE):

```
UASM64.EXE /c -win64 -Zp8 /win64 /D_WIN64 /Cp /nologo /W2 /I"X:\UASM\Include" *.asm
```

Note: X is the drive letter where the UASM assembler has been installed to.

Linking with Microsoft Library Manager (LIB.EXE):

```
LIB *.obj /out:MFPlayer.lib
```

1.3.4 Building the MFPlayer Library x64 Debug Build

To build the MFPlayer Library x64 with debug information, supply the additional flag options `/Zi /Zd` on the command line for UASM (`UASM64.EXE`) like so:

```
UASM64.EXE /c -win64 -Zp8 /Zi /Zd /win64 /D_WIN64 /Cp /nologo /W2 /I"X:\UASM\Include" *.
↪asm
```

Note: `X` is the drive letter where the MASM32 package has been installed to... note:: `X` is the drive letter where the UASM assembler has been installed to.

1.4 Contributing

To contribute to the MFPlayer Library source code or documentation you will need to have a Github account. Sign up at www.Github.com if you don't have a Github account already.

1.4.1 Requirements

MFPlayer Library uses RadASM IDE for the project.

- Install RadASM
- Install UASM
- Install UASM-for-RadASM
- Install UASM-SDK

MFPlayer Library uses readthedocs for hosting the documentation. This requires the installation of python, and some python extensions.

- Install Python
- Open a command prompt/terminal and type :
  - `pip install sphinx`
  - `pip install sphinx_rtd_theme`
  - `pip install recommonmark`

1.4.2 MFPlayer Library Github Repository

To work with the MFPlayer Library source and/or documentation files you will need to first clone or fork the MFPlayer Library repository using a git GUI client or using git commands in a command prompt/terminal. The MFPlayer Library repository is located at: https://github.com/mrfearless/MFPlayer-Library

The workflow for users is:

For users who are already authorized contributors:

- Clone the MFPlayer Library repository locally
- Make changes
- Commit changes back to the MFPlayer Library repository

For users who are not authorized contributors, but who wish to contribute:

- Fork the MFPlayer Library repository to your own github account.
- Clone that repository locally
- Make changes
- Commit changes back to your version hosted on your account
- Submit a pull request - which will generate a pending commit on the original MFPlayer Library repository. This pending commit can be reviewed by the Author and/or other authorized persons and accepted or rejected.
- At some later stage, if changes have occurred on the MFPlayer Library repository, and if you are to keep up to date with these changes, then instead of deleting your repo version and reforking the MFPlayer Library, you can rebase your version to sync with the latest changes. Some GUI git clients may offer this as an upstream, which will allow you to sync with the main repository.

1.4.3 Editing the MFPlayer Library source files

Once the requirements above have been met and the repository has been cloned/forked, you are ready to edit and make changes to the MFPlayer Library source files.

Make your edits to the source files, and then once finished you are ready to submit changes

1.4.4 Editing the MFPlayer Library documentation files

The MFPlayer Library documentation is stored in the **docs** folder. All the files in that folder and sub-folders are **.rst** files (reStructuredText) and are similar to markdown files. See the reStructuredText Primer for details of usage.

Once any changes you have made are saved, you can preview the changes locally by generating the html files:

- Open a command prompt/terminal and change to the MFPlayer Library **docs** directory (for example: `cd "c:\Github\MFPlayer-Library\docs"`)
- Generate the html files by typing `make html`

The sphinx builder will create a **build** folder inside the MFPlayer Library **docs** folder automatically (if it doesn't exist already) and will create an **html** folder under that. Inside this **html** folder is the html files including the **index.html** which you can open to locally preview your changes.

Note: The **build** folder won't be included in any pull requests or commits, as its automatically ignored. You can safely ignore this folder, or delete it if you wish, as it will be built automatically each time you run `make html`

1.4.5 Submitting Changes

Once your happy with your changes, you can then commit to your locally cloned repository and submit a pull request on the MFPlayer Library on the Github website on your Github account. For authorized contributors you can just commit directly to the MFPlayer Library repository.

1.5 Frequently Asked Questions

1.5.1 What does MFPlayer Library do?

MFPlayer Library consists of functions that wrap the MFPlay COM implementation of the IMFPMediaPlayer and IMFPMediaItem objects. MFPlay is a Microsoft Media Foundation API for creating media playback applications. Thus the MFPlayer Library functions hide the complexities of interacting with the COM objects.

1.5.2 What do I need to do to install MFPlayer Library?

Please follow the *Installation & Setup* guide for details on how to install the MFPlayer Library.

1.5.3 How do I add to the MFPlayer Library documentation?

Please follow the *Contributing* guide for details on how to add to, or edit, the MFPlayer Library documentation.

1.5.4 How do I report a bug?

If you should encounter any bugs, please open a new issue on the MFPlayer Library Github repository.

CHAPTER

TWO

MFPLAYER FUNCTIONS

2.1 MFPMediaPlayer Functions

2.1.1 MFPMediaPlayer_ClearMediaItem

Clears the current media item.

```
MFPMediaPlayer_ClearMediaItem PROTO MediaPlayer:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

This method is currently not implemented. It still sends a notification to the Media Event callback as `MFP_EVENT_TYPE_MEDIAITEM_CLEARED`, if the callback function is specified when creating the Media Player (IMFPMediaPlayer) object during the *MFPMediaPlayer_Init* function (MFPCreateMediaPlayer call) Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_MEDIAITEM_CLEARED`

See Also

MFPMediaPlayer_SetMediaItem, *MFPMediaPlayer_CreateMediaItemA*, *MFPMediaPlayer_CreateMediaItemW*

2.1.2 MFPMediaPlayer_CreateMediaItemA

Creates a media item from a string.

```
MFPMediaPlayer_CreateMediaItemA PROTO pMediaPlayer:DWORD, lpszMediaItem:DWORD,␣
↪dwUserData:DWORD, ppMediaItem:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `lpszMediaItem` - A pointer to an ANSI string containing the media file to create the media item from.
- `dwUserData` - A DWORD value used as custom data.
- `ppMediaItem` - A pointer to the media item's IMFPMediaItem interface.

Returns

TRUE if successful or FALSE otherwise.

Notes

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_MEDIAITEM_CREATED`

See Also

MFPMediaPlayer_SetMediaItem, *MFPMediaPlayer_GetMediaItem*

2.1.3 MFPMediaPlayer_CreateMediaItemFromObject

Creates a media item from an object.

```
MFPMediaPlayer_CreateMediaItemFromObject PROTO MediaPlayer:DWORD, pIUnknownObj:DWORD,␣
↪dwUserData:DWORD, ppMediaItem:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pIUnknownObj` - A pointer to an IUnknown object.
- `dwUserData` - A dword value used as custom data.
- `ppMediaItem` - A pointer to the media item's IMFPMediaItem interface.

Returns

TRUE if successful or FALSE otherwise.

Notes

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_MEDIAITEM_CREATED`

See Also

MFPMediaPlayer_SetMediaItem, *MFPMediaPlayer_GetMediaItem*, *MFPMediaPlayer_CreateMediaItemA*, *MFPMediaPlayer_CreateMediaItemW*

2.1.4 MFPMediaPlayer_CreateMediaItemW

Creates a media item from a string.

```
MFPMediaPlayer_CreateMediaItemW PROTO pMediaPlayer:DWORD, lpszMediaItem:DWORD,␣
↪dwUserData:DWORD, ppMediaItem:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `lpszMediaItem` - A pointer to an Wide/Unicode string containing the media file to create the media item from.
- `dwUserData` - A DWORD value used as custom data.
- `ppMediaItem` - A pointer to the media item's IMFPMediaItem interface.

Returns

TRUE if successful or FALSE otherwise.

Notes

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_MEDIAITEM_CREATED`

See Also

MFPMediaPlayer_SetMediaItem, *MFPMediaPlayer_GetMediaItem*

2.1.5 MFPMediaPlayer_Free

Shuts down the MFPlay IMFPMediaPlayer COM object and frees any resources used by it.

```
MFPMediaPlayer_Free PROTO ppMediaPlayer:DWORD
```

Parameters

- `ppMediaPlayer` - pointer to a DWORD value that contains a pointer to the pMediaPlayer (IMFPMediaPlayer) object. The pMediaPlayer variable is returned from the *MFPMediaPlayer_Init* function.

Returns

None.

Notes

The pMediaPlayer variable, pointed to by the `ppMediaPlayer` parameter is set to 0 by this function.

See Also

MFPMediaPlayer_Init

2.1.6 MFPMediaPlayer_GetAspectRatioMode

Gets the current aspect-ratio correction mode.

```
MFPMediaPlayer_GetAspectRatioMode PROTO MediaPlayer:DWORD, pdwAspectRatioMode:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pdwAspectRatioMode` - A pointer to a DWORD variable that contains aspect ratio bitflags.

Returns

TRUE if successful or FALSE otherwise.

Notes

The variable pointed to by the `pdwAspectRatioMode` parameter can contain a combination of the following:

- `MFVideoARMode_None`
- `MFVideoARMode_PreservePicture`
- `MFVideoARMode_PreservePixel`
- `MFVideoARMode_NonLinearStretch`

See Also

MFPMediaPlayer_SetAspectRatioMode

2.1.7 MFPMediaPlayer_GetBalance

Gets the current audio balance.

```
MFPMediaPlayer_GetBalance PROTO MediaPlayer:DWORD, pdwBalance:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pdwBalance` - A pointer to a DWORD variable to store the balance level.

Returns

TRUE if successful or FALSE otherwise.

Notes

0 indicates balance, -100 indicates left, +100 indicates right.

See Also

MFPMediaPlayer_GetBalance, *MFPMediaPlayer_GetVolume*, *MFPMediaPlayer_GetMute*, *MFPMediaPlayer_SetMute*

2.1.8 MFPMediaPlayer_GetBorderColor

Gets the current color of the video border. The border color is used to letterbox the video.

```
MFPMediaPlayer_GetBorderColor PROTO MediaPlayer:DWORD, pColor:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pColor` - Pointer to a DWORD that stores the border color (COLORREF) value.

Returns

TRUE if successful or FALSE otherwise.

Notes

Default color is black.

See Also

MFPMediaPlayer_SetBorderColor

2.1.9 MFPMediaPlayer_GetDuration

Gets the playback duration of the current media item.

```
MFPMediaPlayer_GetDuration PROTO pMediaPlayer:DWORD, pdwMilliseconds:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pdwMilliseconds` - A pointer to a DWORD variable to store the duration of the media item, returned in milliseconds.

Returns

TRUE if successful or FALSE otherwise.

Notes

This function converts the nano second time value to milliseconds. If an error occurs the value of the milliseconds is set to -1.

See Also

MFPMediaPlayer_GetPosition, *MFPMediaPlayer_SetPosition*

2.1.10 MFPMediaPlayer_GetIdealVideoSize

Gets the range of video sizes that can be displayed without significantly degrading performance or image quality.

```
MFPMediaPlayer_GetIdealVideoSize PROTO MediaPlayer:DWORD, pszMin:DWORD, pszMax:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pszMin` - Pointer to a SIZE variable to store the minimum size of video that is preferable. Can be NULL.
- `pszMax` - Pointer to a SIZE variable to store the maximum size of video that is preferable. Can be NULL.

Returns

TRUE if successful or FALSE otherwise.

Notes

At least one parameter must be non-NULL. Sizes are given in pixels.

See Also

MFPMediaPlayer_GetNativeVideoSize

2.1.11 MFPMediaPlayer_GetMediaItem

Gets a pointer to the current media item.

```
MFPMediaPlayer_GetMediaItem PROTO MediaPlayer:DWORD, ppMediaItem:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `ppMediaItem` - A pointer to the media item's IMFPMediaItem interface.

Returns

TRUE if successful or FALSE otherwise.

Notes

The caller must release the interface.

See Also

MFPMediaPlayer_SetMediaItem, *MFPMediaPlayer_CreateMediaItemA*, *MFPMediaPlayer_CreateMediaItemW*

2.1.12 MFPMediaPlayer_GetMute

Queries whether the audio is muted.

```
MFPMediaPlayer_GetMute PROTO MediaPlayer:DWORD, pbMute:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pbMute` - A pointer to a DWORD variable containing TRUE or FALSE.

Returns

TRUE if successful or FALSE otherwise.

Notes

The variable pointed to by the `pbMute` parameter will contain TRUE if the audio is muted, or FALSE otherwise.

See Also

MFPMediaPlayer_SetMute, *MFPMediaPlayer_GetVolume*, *MFPMediaPlayer_SetVolume*

2.1.13 MFPMediaPlayer_GetNativeVideoSize

Gets the size and aspect ratio of the video. These values are computed before any scaling is done to fit the video into the destination window.

```
MFPMediaPlayer_GetNativeVideoSize PROTO MediaPlayer:DWORD, pszVideo:DWORD,␣
↪pszARVideo:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pszVideo` - Pointer to a SIZE variable to store the width and height of the video, in pixels. Can be NULL.
- `pszARVideo` - Pointer to a SIZE variable to store the aspect ratio of the video. Can be NULL.

Returns

TRUE if successful or FALSE otherwise.

Notes

At least one parameter must be non-NULL.

See Also

MFPMediaPlayer_GetIdealVideoSize

2.1.14 MFPMediaPlayer_GetPosition

Gets the current playback position.

```
MFPMediaPlayer_GetPosition PROTO pMediaPlayer:DWORD, pdwMilliseconds:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pdwMilliseconds` - A pointer to a DWORD variable to store the position of the media item, returned in milliseconds.

Returns

TRUE if successful or FALSE otherwise.

Notes

This function converts the nano second time value to milliseconds. If an error occurs the value of the milliseconds is set to -1.

See Also

MFPMediaPlayer_SetPosition, *MFPMediaPlayer_GetDuration*

2.1.15 MFPMediaPlayer_GetRate

Gets the current playback rate.

```
MFPMediaPlayer_GetRate PROTO pMediaPlayer:DWORD, pdwRate:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pdwRate` - A pointer to a DWORD variable that store the rate value.

Returns

TRUE if successful or FALSE otherwise.

Notes

100 indicates normal playback speed, 50 indicates half speed, and 200 indicates twice speed, etc.

See Also

MFPMediaPlayer_SetRate, *MFPMediaPlayer_GetSupportedRates*

2.1.16 MFPMediaPlayer_GetState

Obtains the current state of the MFPlay Media Player.

```
MFPMediaPlayer_GetState PROTO pMediaPlayer:DWORD, pdwState:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pdwState` - pointer to a DWORD value to store the state of the Media Player.

Returns

TRUE if successful or FALSE otherwise. The DWORD pointer to by the `pdwState` parameter will contain one of following value:

- `MFP_MEDIAPLAYER_STATE_EMPTY`
- `MFP_MEDIAPLAYER_STATE_STOPPED`
- `MFP_MEDIAPLAYER_STATE_PLAYING`
- `MFP_MEDIAPLAYER_STATE_PAUSED`
- `MFP_MEDIAPLAYER_STATE_SHUTDOWN`

See Also

MFPMediaPlayer_GetPosition, *MFPMediaPlayer_GetDuration*

2.1.17 MFPMediaPlayer_GetSupportedRates

Gets the range of supported playback rates.

```
MFPMediaPlayer_GetSupportedRates PROTO pMediaPlayer:DWORD, bForwardDirection:DWORD,␣
↪pdwSlowestRate:DWORD, pdwFastestRate:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `bForwardDirection` - TRUE for forward playback or FALSE for reverse playback
- `pdwSlowestRate` - A pointer to a DWORD to store the slowest rate value.
- `pdwFastestRate` - A pointer to a DWORD to store the fastest rate value.

Returns

TRUE if successful or FALSE otherwise.

Notes

100 indicates normal playback speed, 50 indicates half speed, and 200 indicates twice speed, etc.

See Also

MFPMediaPlayer_GetRate, *MFPMediaPlayer_SetRate*

2.1.18 MFPMediaPlayer_GetVideoSourceRect

Gets the video source rectangle.

```
MFPMediaPlayer_GetVideoSourceRect PROTO MediaPlayer:DWORD, pnrcSource:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pnrcSource` - Pointer to an MFVideoNormalizedRect structure.

Returns

TRUE if successful or FALSE otherwise.

Notes

The upper-left corner of the video image is (0, 0). The lower-right corner of the video image is (1, 1) If the source rectangle is {0, 0, 1, 1}, the entire image is displayed. This is the default value.

See Also

MFPMediaPlayer_SetVideoSourceRect

2.1.19 MFPMediaPlayer_GetVideoWindow

Gets the window where the video is displayed.

```
MFPMediaPlayer_GetVideoWindow PROTO MediaPlayer:DWORD, phwndVideo:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `phwndVideo` - A pointer to a variable to hold the window handle.

Returns

TRUE if successful or FALSE otherwise.

Notes

The video window is specified when you first call *MFPMediaPlayer_Init* to create the MFPlay player object.

See Also

MFPMediaPlayer_Init, *MFPMediaPlayer_UpdateVideo*

2.1.20 MFPMediaPlayer_GetVolume

Gets the current audio volume.

```
MFPMediaPlayer_GetVolume PROTO pMediaPlayer:DWORD, pdwVolume:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pdwVolume` - A pointer to a DWORD variable to store the volume level.

Returns

TRUE if successful or FALSE otherwise.

Notes

0 indicates silence and 100 indicates full volume.

See Also

MFPMediaPlayer_SetVolume, *MFPMediaPlayer_GetMute*, *MFPMediaPlayer_SetMute*

2.1.21 MFPMediaPlayer_Init

Create and initialize the MFPlay IMFPMediaPlayer COM object with the video window handle and the callback function (if specified).

```
MFPMediaPlayer_Init PROTO hMFPWindow:DWORD, pCallback:DWORD, ppMediaPlayer:DWORD
```

Parameters

- `hMFPWindow` - handle to the window to use for the video output
- `pCallback` - Address of the MFPlay event notification callback function.
- `ppMediaPlayer` - pointer to a DWORD value to store the pMediaPlayer (IMFPMediaPlayer) object.

Returns

Stores a pMediaPlayer object in the DWORD pointer to by the `ppMediaPlayer` parameter, or 0 if an error occurred. Returns TRUE if successful or FALSE otherwise.

Notes

MFPMediaPlayer_Free should be called on program close to free up any resources.

See Also

MFPMediaPlayer_Free

2.1.22 MFPMediaPlayer_InsertEffect

Applies an audio or video effect to playback.

```
MFPMediaPlayer_InsertEffect PROTO MediaPlayer:DWORD, pEffect:DWORD, bOptional:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pEffect` - Pointer to the IUnknown interface for one of the following:
  - A Media Foundation transform (MFT) that implements the effect. MFTs expose the IMFTransform interface.
  - An activation object that creates an MFT. Activation objects expose the IMFActivate interface.
- `bOptional` - Specifies whether the effect is optional.

Returns

TRUE if successful or FALSE otherwise.

Notes

The object specified in the `pEffect` parameter can implement either a video effect or an audio effect. The effect is applied to any media items set after the method is called. It is not applied to the current media item.

See Also

MFPMediaPlayer_RemoveEffect, *MFPMediaPlayer_RemoveAllEffects*

2.1.23 MFPMediaPlayer_Pause

Pauses playback of a media item that is currently set in the media player.

```
MFPMediaPlayer_Pause PROTO MediaPlayer:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

Media items have to be created and set before they can be played, paused, stepped or stopped. A media item is created with the *MFPMediaPlayer_CreateMediaItemA* or *MFPMediaPlayer_CreateMediaItemW* function and is set in the queue to play with the *MFPMediaPlayer_SetMediaItem* function.

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_PAUSE`

See Also

MFPMediaPlayer_CreateMediaItemA, *MFPMediaPlayer_CreateMediaItemW*, *MFPMediaPlayer_SetMediaItem*, *MFPMediaPlayer_Play*, *MFPMediaPlayer_Stop*, *MFPMediaPlayer_Toggle*

2.1.24 MFPMediaPlayer_Play

Begins playback of a media item that is currently set in the media player.

```
MFPMediaPlayer_Play PROTO MediaPlayer:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

Media items have to be created and set before they can be played, paused, stepped or stopped. A media item is created with the *MFPMediaPlayer_CreateMediaItemA* or *MFPMediaPlayer_CreateMediaItemW* function and is set in the queue to play with the *MFPMediaPlayer_SetMediaItem* function.

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_PLAY`

See Also

MFPMediaPlayer_CreateMediaItemA, *MFPMediaPlayer_CreateMediaItemW*, *MFPMediaPlayer_SetMediaItem*, *MFPMediaPlayer_Pause*, *MFPMediaPlayer_Stop*, *MFPMediaPlayer_Toggle*

2.1.25 MFPMediaPlayer_RemoveAllEffects

Removes all effects that were added with the MFPMediaPlayer_InsertEffect function.

```
MFPMediaPlayer_RemoveAllEffects PROTO MediaPlayer:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

The change applies to the next media item that is set on the player. The effect is not removed from the current media item.

See Also

MFPMediaPlayer_InsertEffect, *MFPMediaPlayer_RemoveEffect*

2.1.26 MFPMediaPlayer_RemoveEffect

Removes an effect that was added with the MFPMediaPlayer_InsertEffect function.

```
MFPMediaPlayer_RemoveEffect PROTO MediaPlayer:DWORD, pEffect:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pEffect` - Pointer to the IUnknown interface of the effect object. Use the same pointer that you passed to the *MFPMediaPlayer_InsertEffect* function.

Returns

TRUE if successful or FALSE otherwise.

Notes

The change applies to the next media item that is set on the player. The effect is not removed from the current media item.

See Also

MFPMediaPlayer_InsertEffect, *MFPMediaPlayer_RemoveAllEffects*

2.1.27 MFPMediaPlayer_SetAspectRatioMode

Specifies whether the aspect ratio of the video is preserved during playback.

```
MFPMediaPlayer_SetAspectRatioMode PROTO MediaPlayer:DWORD, dwAspectRatioMode:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `dwAspectRatioMode` - Bitwise flags for aspect ratio.

Returns

TRUE if successful or FALSE otherwise.

Notes

dwAspectRatioMode can contain a combination of the following:

- `MFVideoARMode_None`
- `MFVideoARMode_PreservePicture`
- `MFVideoARMode_PreservePixel`
- `MFVideoARMode_NonLinearStretch`

In practice only `MFVideoARMode_None` and (`MFVideoARMode_PreservePicture` or `MFVideoARMode_PreservePixel`) actually do anything.

See Also

MFPMediaPlayer_GetAspectRatioMode

2.1.28 MFPMediaPlayer_SetBalance

Sets the current audio balance.

```
MFPMediaPlayer_SetBalance PROTO MediaPlayer:DWORD, dwBalance:SDWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `dwBalance` - The balance level to set, -100 to +100.

Returns

TRUE if successful or FALSE otherwise.

Notes

0 indicates balance, -100 indicates left, +100 indicates right.

See Also

MFPMediaPlayer_GetBalance, *MFPMediaPlayer_GetVolume*, *MFPMediaPlayer_GetMute*, *MFPMediaPlayer_SetMute*

2.1.29 MFPMediaPlayer_SetBorderColor

Sets the color for the video border. The border color is used to letterbox the video.

```
MFPMediaPlayer_SetBorderColor PROTO MediaPlayer:DWORD, Color:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `Color` - Specifies the border color as a COLORREF value.

Returns

TRUE if successful or FALSE otherwise.

Notes

Default color is black.

See Also

MFPMediaPlayer_GetBorderColor

2.1.30 MFPMediaPlayer_SetMediaItem

Queues a media item for playback.

```
MFPMediaPlayer_SetMediaItem PROTO MediaPlayer:DWORD, pMediaItem:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pMediaItem` - A pointer to the media item (IMFPMediaItem) to queue for play.

Returns

TRUE if successful or FALSE otherwise.

Notes

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_MEDIAITEM_SET`

See Also

MFPMediaPlayer_CreateMediaItemA, *MFPMediaPlayer_CreateMediaItemW*, *MFPMediaPlayer_GetMediaItem*

2.1.31 MFPMediaPlayer_SetMute

Mutes or unmutes the audio.

```
MFPMediaPlayer_SetMute PROTO MediaPlayer:DWORD, bMute:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `bMute` - TRUE to mute the audio, or FALSE to unmute the audio.

Returns

TRUE if successful or FALSE otherwise.

Notes

Get volume level before setting mute, then restore that level apon unmute.

See Also

MFPMediaPlayer_GetMute, *MFPMediaPlayer_GetVolume*, *MFPMediaPlayer_SetVolume*

2.1.32 MFPMediaPlayer_SetPosition

Sets the playback position.

```
MFPMediaPlayer_SetPosition PROTO pMediaPlayer:DWORD, dwMilliseconds:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `dwMilliseconds` - The position to set in the media item, in milliseconds.

Returns

TRUE if successful or FALSE otherwise.

Notes

This function converts the milliseconds value to nano seconds.

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_POSITION_SET`

See Also

MFPMediaPlayer_GetPosition, *MFPMediaPlayer_GetDuration*

2.1.33 MFPMediaPlayer_SetRate

Sets the playback rate.

```
MFPMediaPlayer_SetRate PROTO MediaPlayer:DWORD, dwRate:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `dwRate` - A DWORD value of the rate to set.

Returns

TRUE if successful or FALSE otherwise.

Notes

100 indicates normal playback speed, 50 indicates half speed, and 200 indicates twice speed, etc.

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_RATE_SET`

See Also

MFPMediaPlayer_GetRate, *MFPMediaPlayer_GetSupportedRates*

2.1.34 MFPMediaPlayer_SetVideoSourceRect

Sets the video source rectangle. MFPlay clips the video to this rectangle and stretches the rectangle to fill the video window.

```
MFPMediaPlayer_SetVideoSourceRect PROTO MediaPlayer:DWORD, pnrcSource:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `pnrcSource` - Pointer to an MFVideoNormalizedRect structure.

Returns

TRUE if successful or FALSE otherwise.

Notes

The upper-left corner of the video image is (0, 0). The lower-right corner of the video image is (1, 1) If the source rectangle is {0, 0, 1, 1}, the entire image is displayed. This is the default value.

See Also

MFPMediaPlayer_GetVideoSourceRect

2.1.35 MFPMediaPlayer_SetVolume

Sets the audio volume.

```
MFPMediaPlayer_SetVolume PROTO MediaPlayer:DWORD, dwVolume:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.
- `dwVolume` - The volume level to set, 0 - 100.

Returns

TRUE if successful or FALSE otherwise.

Notes

0 indicates silence and 100 indicates full volume.

See Also

MFPMediaPlayer_GetVolume, *MFPMediaPlayer_GetMute*, *MFPMediaPlayer_SetMute*

2.1.36 MFPMediaPlayer_Shutdown

Shuts down the MFPlay player object and releases any resources the object is using.

```
MFPMediaPlayer_Shutdown PROTO MediaPlayer:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

The player object automatically shuts itself down when its reference count reaches zero. You can use the Shutdown method to shut down the player before all of the references have been released.

See Also

MFPMediaPlayer_Init, *MFPMediaPlayer_Free*

2.1.37 MFPMediaPlayer_Step

Steps a frame of a media item that is currently set in the media player.

```
MFPMediaPlayer_Step PROTO MediaPlayer:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

Media items have to be created and set before they can be played, paused, stepped or stopped. A media item is created with the *MFPMediaPlayer_CreateMediaItemA* or *MFPMediaPlayer_CreateMediaItemW* function and is set in the queue to play with the *MFPMediaPlayer_SetMediaItem* function.

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_FRAME_STEP`

See Also

MFPMediaPlayer_CreateMediaItemA, *MFPMediaPlayer_CreateMediaItemW*, *MFPMediaPlayer_SetMediaItem*, *MFPMediaPlayer_Play*, *MFPMediaPlayer_Pause*, *MFPMediaPlayer_Stop*, *MFPMediaPlayer_Toggle*

2.1.38 MFPMediaPlayer_Stop

Stops playback of a media item that is currently set in the media player.

```
MFPMediaPlayer_Stop PROTO MediaPlayer:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

Media items have to be created and set before they can be played, paused, stepped or stopped. A media item is created with the *MFPMediaPlayer_CreateMediaItemA* or *MFPMediaPlayer_CreateMediaItemW* function and is set in the queue to play with the *MFPMediaPlayer_SetMediaItem* function.

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_STOP`

See Also

MFPMediaPlayer_CreateMediaItemA, *MFPMediaPlayer_CreateMediaItemW*, *MFPMediaPlayer_SetMediaItem*, *MFPMediaPlayer_Play*, *MFPMediaPlayer_Pause*, *MFPMediaPlayer_Toggle*

2.1.39 MFPMediaPlayer_Toggle

Toggles between play and pause for a media item that is currently set in the media player.

```
MFPMediaPlayer_Toggle PROTO MediaPlayer:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

Media items have to be created and set before they can be played, paused, stepped or stopped. A media item is created with the *MFPMediaPlayer_CreateMediaItemA* or *MFPMediaPlayer_CreateMediaItemW* function and is set in the queue to play with the *MFPMediaPlayer_SetMediaItem* function.

Sends a notification to the Media Event callback function as `MFP_EVENT_TYPE_PAUSE` or `MFP_EVENT_TYPE_PLAY`

See Also

MFPMediaPlayer_CreateMediaItemA, *MFPMediaPlayer_CreateMediaItemW*, *MFPMediaPlayer_SetMediaItem*, *MFPMediaPlayer_Play*, *MFPMediaPlayer_Pause*, *MFPMediaPlayer_Stop*, *MFPMediaPlayer_Step*

2.1.40 MFPMediaPlayer_UpdateVideo

Updates the video frame.

```
MFPMediaPlayer_UpdateVideo PROTO MediaPlayer:DWORD
```

Parameters

- `MediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

Is supposed to allow painting in `WM_PAINT` or `WM_SIZE` when video is not rendering, but tests show that this isn't the case. At a best guess the window that is used to render is sub-classed and doesn't pass back the chain of events for `WM_PAINT` once rendering starts. My workaround for this to allow a custom draw of the video window when not playing is to hide the window when stopped and show it when playing, this allows the background of the parent window to show any custom painting, including a fake video screen with a logo.

See Also

MFPMediaPlayer_GetVideoWindow

Function	Description
MFPMediaPlayer_ClearMediaItem	
MFPMediaPlayer_CreateMediaItemA	

continues on next page

Table 1 – continued from previous page

MFPMediaPlayer_CreateMediaItemFromObject	
MFPMediaPlayer_CreateMediaItemW	
MFPMediaPlayer_Free	
MFPMediaPlayer_GetAspectRatioMode	
MFPMediaPlayer_GetBalance	
MFPMediaPlayer_GetBorderColor	
MFPMediaPlayer_GetDuration	
MFPMediaPlayer_GetIdealVideoSize	
MFPMediaPlayer_GetMediaItem	
MFPMediaPlayer_GetMute	
MFPMediaPlayer_GetNativeVideoSize	
MFPMediaPlayer_GetPosition	
MFPMediaPlayer_GetRate	
MFPMediaPlayer_GetState	
MFPMediaPlayer_GetSupportedRates	
MFPMediaPlayer_GetVideoSourceRect	
MFPMediaPlayer_GetVideoWindow	
MFPMediaPlayer_GetVolume	
MFPMediaPlayer_Init	
MFPMediaPlayer_InsertEffect	
MFPMediaPlayer_Pause	
MFPMediaPlayer_Play	
MFPMediaPlayer_RemoveAllEffects	
MFPMediaPlayer_RemoveEffect	
MFPMediaPlayer_SetAspectRatioMode	
MFPMediaPlayer_SetBalance	
MFPMediaPlayer_SetBorderColor	
MFPMediaPlayer_SetMediaItem	
MFPMediaPlayer_SetMute	
MFPMediaPlayer_SetPosition	
MFPMediaPlayer_SetRate	
MFPMediaPlayer_SetVideoSourceRect	
MFPMediaPlayer_SetVolume	
MFPMediaPlayer_Shutdown	
MFPMediaPlayer_Step	
MFPMediaPlayer_Stop	
MFPMediaPlayer_Toggle	
MFPMediaPlayer_UpdateVideo	

2.2 MFPMediaItem Functions

2.2.1 MFPMediaItem_AddRef

Increments the reference count for an interface pointer to a COM object. You should call this method whenever you make a copy of an interface pointer.

```
MFPMediaItem_AddRef PROTO pMediaItem:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

A COM object uses a per-interface reference-counting mechanism to ensure that the object doesn't outlive references to it. You use AddRef to stabilize a copy of an interface pointer. It can also be called when the life of a cloned pointer must extend beyond the lifetime of the original pointer. The cloned pointer must be released by calling IUnknown_Release on it.

The internal reference counter that AddRef maintains should be a 32-bit unsigned integer.

See Also

MFPMediaItem_Release, *MFPMediaItem_QueryInterface*

2.2.2 MFPMediaItem_GetCharacteristics

Gets various flags that describe the media item.

```
MFPMediaItem_GetCharacteristics PROTO pMediaItem:DWORD, pCharacteristics:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `pCharacteristics` - a pointer to a DWORD variable to receive the bitflags of the characteristics of the media item.

Returns

TRUE if successful or FALSE otherwise.

Notes

The variable pointed to by the `pCharacteristics` parameter can contain a combination of the following values:

- `MFP_MEDIAITEM_IS_LIVE`
- `MFP_MEDIAITEM_CAN_SEEK`
- `MFP_MEDIAITEM_CAN_PAUSE`
- `MFP_MEDIAITEM_HAS_SLOW_SEEK`

See Also

MFPMediaItem_GetPresentationAttribute, *MFPMediaItem_GetStreamAttribute*

2.2.3 MFPMediaItem_GetDuration

Gets the duration of the media item.

```
MFPMediaItem_GetDuration PROTO pMediaItem:DWORD, pdwMilliseconds:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.

- `pdwMilliseconds` - A pointer to a DWORD variable to store the duration of the media item, returned in milliseconds.

Returns

TRUE if successful or FALSE otherwise.

Notes

This function converts the nano second time value to milliseconds. If an error occurs the value of the milliseconds is set to -1.

See Also

MFPMediaItem_GetStartStopPosition, *MFPMediaItem_SetStartStopPosition*, *MFPMediaPlayer_GetPosition*, *MFPMediaPlayer_SetPosition*, *MFPMediaPlayer_GetDuration*

2.2.4 MFPMediaItem_GetMediaPlayer

Gets a pointer to the MFPlay player object (IMFPMediaPlayer) that created the media item.

```
MFPMediaItem_GetMediaPlayer PROTO pMediaItem:DWORD, ppMediaPlayer:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `ppMediaPlayer` - pointer to a DWORD variable to store the pMediaPlayer object (IMFPMediaPlayer).

Returns

TRUE if successful or FALSE otherwise.

Notes

N/A

See Also

MFPMediaPlayer_Init

2.2.5 MFPMediaItem_GetMetadata

Gets a property store that contains metadata for the source, such as author or title.

```
MFPMediaItem_GetMetadata PROTO pMediaItem:DWORD, ppMetadataStore:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `ppMetadataStore` - A pointer to a DWORD variable to store the pMetadataStore object (IPropertyStore)

Returns

TRUE if successful or FALSE otherwise.

Notes

TRUE if successful or FALSE otherwise.

The caller must release the interface.

See Also

MFPMediaItem_GetCharacteristics, *MFPMediaItem_GetPresentationAttribute*, *MFPMediaItem_GetStreamAttribute*

2.2.6 MFPMediaItem_GetNumberOfStreams

Gets the number of streams (audio, video, and other) in the media item.

```
MFPMediaItem_GetNumberOfStreams PROTO pMediaItem:DWORD, pdwStreamCount:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `pdwStreamCount` - A pointer to a DWORD variable used to store the steam count of the media item.

Returns

TRUE if successful or FALSE otherwise.

Notes

N/A

See Also

MFPMediaItem_GetStreamSelection, *MFPMediaItem_SetStreamSelection*

2.2.7 MFPMediaItem_GetObject

Gets the object that was used to create the media item.

```
MFPMediaItem_GetObject PROTO pMediaItem:DWORD, ppIUnknown:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `ppIUnknown` - A pointer to a DWORD value used to store the object's IUnknown interface.

Returns

TRUE if successful or FALSE otherwise.

Notes

The caller must release the interface.

The object pointer is set if the application uses CreateMediaItemFromObject to create the media item.

See Also

MFPMediaPlayer_CreateMediaItemFromObject

2.2.8 MFPMediaItem_GetPresentationAttribute

Queries the media item for a presentation attribute.

```
MFPMediaItem_GetPresentationAttribute PROTO pMediaItem:DWORD, guidMFAttribute:DWORD,␣
↪pvValue:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `guidMFAttribute` - GUID that identifies the attribute value to query.
- `pvValue` - Pointer to a PROPVARIANT that receives the value.

Returns

TRUE if successful or FALSE otherwise.

Notes

Call PropVariantClear to free the memory allocated and stored in variable pointed to by the `pvValue` parameter.

Includelib mfuuid.lib to reference the already defined and exported property GUIDs in that library.

See Also

MFPMediaItem_GetStreamAttribute, *MFPMediaItem_GetCharacteristics*

2.2.9 MFPMediaItem_GetStartStopPosition

Gets the start and stop times for the media item.

```
MFPMediaItem_GetStartStopPosition PROTO pMediaItem:DWORD, pdwStartValue:DWORD,␣
↪pdwStopValue:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `pdwStartValue` - A pointer to a DWORD variable to store the start position of the media item, returned in milliseconds.
- `pdwStopValue` - A pointer to a DWORD variable to store the stop position of the media item, returned in milliseconds.

Returns

TRUE if successful or FALSE otherwise.

Notes

This function converts the nano second time values to milliseconds. If an error occurs the value of the milliseconds is set to -1.

See Also

MFPMediaItem_SetStartStopPosition, *MFPMediaItem_GetDuration*, *MFPMediaPlayer_GetPosition*, *MFPMediaPlayer_SetPosition*, *MFPMediaPlayer_GetDuration*

2.2.10 MFPMediaItem_GetStreamAttribute

Queries the media item for a stream attribute.

```
MFPMediaItem_GetStreamAttribute PROTO pMediaItem:DWORD, dwStreamIndex:DWORD,␣
↪guidMFAttribute:DWORD, pvValue:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `dwStreamIndex` - Zero-based index of the stream.
- `guidMFAttribute` - GUID that identifies the attribute value to query.
- `pvValue` - Pointer to a PROPVARIANT that receives the value.

Returns

TRUE if successful or FALSE otherwise.

Notes

Call PropVariantClear to free the memory allocated and stored in variable pointed to by the `pvValue` parameter.

Includelib mfuuid.lib to reference the already defined and exported property GUIDs in that library.

See Also

MFPMediaItem_GetPresentationAttribute, *MFPMediaItem_GetCharacteristics*

2.2.11 MFPMediaItem_GetStreamSelection

Queries whether a stream is selected to play.

```
MFPMediaItem_GetStreamSelection PROTO pMediaItem:DWORD, dwStreamIndex:DWORD,␣
↪pbEnabled:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `dwStreamIndex` - A zero based index of the stream.
- `pbEnabled` - A pointer to a DWORD that contains TRUE or FALSE.

Returns

TRUE if successful or FALSE otherwise.

Notes

To select or deselect a stream, call *MFPMediaItem_SetStreamSelection*.

See Also

MFPMediaItem_SetStreamSelection, *MFPMediaItem_GetNumberOfStreams*

2.2.12 MFPMediaItem_GetURLA

Gets the URL that was used to create the media item. This is the Ansi version of MFPMediaItem_GetURL. For the Wide/Unicode version see MFPMediaItem_GetURLW

```
MFPMediaItem_GetURLA PROTO pMediaItem:DWORD, ppszURL:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `ppszURL` - pointed to a DWORD that holds a pointer to the URL string.

Returns

TRUE if successful or FALSE otherwise.

Notes

Use GlobalFree on the `ppszURL` once no longer required.

See Also

MFPMediaItem_GetURLW

2.2.13 MFPMediaItem_GetURLW

Gets the URL that was used to create the media item. This is the Wide/Unicode version of MFPMediaItem_GetURL. For the Ansi version see MFPMediaItem_GetURLA

```
MFPMediaItem_GetURLW PROTO pMediaItem:DWORD, ppszURL:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `ppszURL` - pointed to a DWORD that holds a pointer to the URL string.

Returns

TRUE if successful or FALSE otherwise.

Notes

Use GlobalFree on the `ppszURL` once no longer required.

See Also

MFPMediaItem_GetURLA

2.2.14 MFPMediaItem_GetUserData

Gets the application-defined value stored in the media item.

```
MFPMediaItem_GetUserData PROTO pMediaItem:DWORD, pdwUserData:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `pdwUserData` - A pointer to a DWORD used to store user data.

Returns

TRUE if successful or FALSE otherwise.

Notes

You can assign this value when you first create the media item, by specifying it in the `dwUserData` parameter of the *MFPMediaPlayer_CreateMediaItemA*, *MFPMediaPlayer_CreateMediaItemW* or *MFPMediaPlayer_CreateMediaItemFromObject* method. To update the value, call *MFPMediaItem_SetUserData*.

See Also

MFPMediaItem_SetUserData, *MFPMediaPlayer_CreateMediaItemA*, *MFPMediaPlayer_CreateMediaItemW*, *MFPMediaPlayer_CreateMediaItemFromObject*

2.2.15 MFPMediaItem_HasAudio

Queries whether the media item contains an audio stream.

```
MFPMediaItem_HasAudio PROTO pMediaItem:DWORD, pbHasAudio:DWORD, pbSelected:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `pbHasAudio` - A pointer to a DWORD that contains TRUE or FALSE.
- `pbSelected` - A pointer to a DWORD that contains TRUE or FALSE.

Returns

TRUE if successful or FALSE otherwise.

Notes

To select or deselect streams before playback starts, call *MFPMediaItem_SetStreamSelection*.

See Also

MFPMediaItem_HasVideo, *MFPMediaItem_IsProtected*, *MFPMediaItem_SetStreamSelection*

2.2.16 MFPMediaItem_HasVideo

Queries whether the media item contains a video stream.

```
MFPMediaItem_HasVideo PROTO pMediaItem:DWORD, pbHasVideo:DWORD, pbSelected:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `pbHasVideo` - A pointer to a DWORD that contains TRUE or FALSE.
- `pbSelected` - A pointer to a DWORD that contains TRUE or FALSE.

Returns

TRUE if successful or FALSE otherwise.

Notes

To select or deselect streams before playback starts, call *MFPMediaItem_SetStreamSelection*.

See Also

MFPMediaItem_HasAudio, *MFPMediaItem_IsProtected*, *MFPMediaItem_SetStreamSelection*

2.2.17 MFPMediaItem_IsProtected

Queries whether the media item contains protected content.

```
MFPMediaItem_IsProtected PROTO pMediaItem:DWORD, pbProtected:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `pbProtected` - A pointer to a DWORD that contains TRUE or FALSE.

Returns

TRUE if successful or FALSE otherwise.

Notes

If media item contains protected content any attempt to play it will cause a playback error.

See Also

MFPMediaItem_HasAudio, *MFPMediaItem_HasVideo*

2.2.18 MFPMediaItem_QueryInterface

Queries a COM object for a pointer to one of its interface; identifying the interface by a reference to its interface identifier (IID). If the COM object implements the interface, then it returns a pointer to that interface after calling IUnknown::AddRef on it.

```
MFPMediaItem_QueryInterface PROTO pMediaItem:DWORD, riid:DWORD, ppvObject:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `riid` - A reference to the interface identifier (IID) of the interface being queried for.
- `ppvObject` - The address of a pointer to an interface with the IID specified in the `riid` parameter. Because you pass the address of an interface pointer the method can overwrite that address with the pointer to the interface being queried for. Upon successful return, `ppvObject` (the dereferenced address) contains a pointer to the requested interface. If the object doesn't support the interface, the method sets `ppvObject` (the dereferenced address) to nullptr.

Returns

TRUE if successful or FALSE otherwise.

Notes

For any given COM object (also known as a COM component), a specific query for the IUnknown interface on any of the object's interfaces must always return the same pointer value. This enables a client to determine whether two pointers point to the same component by calling QueryInterface with IID_IUnknown and comparing the results. It is specifically not the case that queries for interfaces other than IUnknown (even the same interface through the same pointer) must return the same pointer value.

See Also

MFPMediaItem_AddRef , *MFPMediaItem_Release*

2.2.19 MFPMediaItem_Release

Decrements the reference count for an interface on a COM object.

```
MFPMediaItem_Release PROTO pMediaItem:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

When the reference count on an object reaches zero, Release must cause the interface pointer to free itself. When the released pointer is the only (formerly) outstanding reference to an object (whether the object supports single or multiple interfaces), the implementation must free the object.

See Also

MFPMediaItem_AddRef , *MFPMediaItem_QueryInterface*

2.2.20 MFPMediaItem_SetStartStopPosition

Sets the start and stop times for the media item.

```
MFPMediaItem_SetStartStopPosition PROTO pMediaItem:DWORD, dwStartValue:DWORD,␣
↪dwStopValue:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `dwStartValue` - The start position to set in the media item, in milliseconds
- `dwStopValue` - The stop position to set in the media item, in milliseconds.

Returns

TRUE if successful or FALSE otherwise.

Notes

This function converts the milliseconds values to nano seconds.

See Also

MFPMediaItem_GetStartStopPosition, *MFPMediaItem_GetDuration*, *MFPMediaPlayer_GetPosition*, *MFPMediaPlayer_SetPosition*, *MFPMediaPlayer_GetDuration*

2.2.21 MFPMediaItem_SetStreamSelection

Selects or deselects a stream.

```
MFPMediaItem_SetStreamSelection PROTO pMediaItem:DWORD, dwStreamIndex:DWORD,␣
↪bEnabled:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `dwStreamIndex` - A zero based index of the stream.
- `bEnabled` - TRUE or FALSE.

Returns

TRUE if successful or FALSE otherwise.

Notes

You can use this method to change which streams are selected. The change goes into effect the next time that *MFPMediaPlayer_SetMediaItem* is called with this media item. If the media item is already set on the player, the change does not happen unless you call *MFPMediaPlayer_SetMediaItem* again with this media item.

See Also

MFPMediaItem_GetStreamSelection, *MFPMediaItem_GetNumberOfStreams*, *MFPMediaPlayer_SetMediaItem*

2.2.22 MFPMediaItem_SetStreamSink

Sets a media sink for the media item. A media sink is an object that consumes the data from one or more streams.

```
MFPMediaItem_SetStreamSink PROTO pMediaItem:DWORD, dwStreamIndex:DWORD, pMediaSink:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `dwStreamIndex` - Zero-based index of a stream on the media source. The media sink will receive the data from this stream.
- `pMediaSink` - IUnknown pointer that specifies the media sink. Pass in one of the following:
  - A pointer to a stream sink. Every media sink contains one or more stream sinks. Each stream sink receives the data from one stream. The stream sink must expose the IMFStreamSink interface.
  - A pointer to an activation object that creates the media sink. The activation object must expose the IMFActivate interface. The media item uses the first stream sink on the media sink (that is, the stream sink at index 0).
  - NULL. If you set `pMediaSink` to NULL, the default media sink for the stream type is used.

Returns

TRUE if successful or FALSE otherwise.

Notes

By default, the MFPlay player object renders audio streams to the Streaming Audio Renderer (SAR) and video streams to the Enhanced Video Renderer (EVR).

Call this method before calling *MFPMediaPlayer_SetMediaItem*. Calling this method after *MFPMediaPlayer_SetMediaItem* has no effect, unless you stop playback and call *MFPMediaPlayer_SetMediaItem* again.

To reset the media item to use the default media sink, set `pMediaSink` to NULL

See Also

MFPMediaPlayer_SetMediaItem

2.2.23 MFPMediaItem_SetUserData

Stores an application-defined value in the media item.

```
MFPMediaItem_SetUserData PROTO pMediaItem:DWORD, dwUserData:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.
- `dwUserData` - The application-defined value.

Returns

TRUE if successful or FALSE otherwise.

Notes

You can assign this value when you first create the media item, by specifying it in the `dwUserData` parameter of the *MFPMediaPlayer_CreateMediaItemA*, *MFPMediaPlayer_CreateMediaItemW* or *MFPMediaPlayer_CreateMediaItemFromObject* method. To update the value, call *MFPMediaItem_SetUserData*.

See Also

MFPMediaItem_GetUserData, *MFPMediaPlayer_CreateMediaItemA*, *MFPMediaPlayer_CreateMediaItemW*, *MFPMediaPlayer_CreateMediaItemFromObject*

Function	Description
MFPMediaItem_AddRef	
MFPMediaItem_GetCharacteristics	
MFPMediaItem_GetDuration	
MFPMediaItem_GetMediaPlayer	
MFPMediaItem_GetMetadata	
MFPMediaItem_GetNumberOfStreams	
MFPMediaItem_GetObject	
MFPMediaItem_GetPresentationAttribute	
MFPMediaItem_GetStartStopPosition	
MFPMediaItem_GetStreamAttribute	
MFPMediaItem_GetStreamSelection	
MFPMediaItem_GetURLA	
MFPMediaItem_GetURLW	
MFPMediaItem_GetUserData	
MFPMediaItem_HasAudio	
MFPMediaItem_HasVideo	
MFPMediaItem_IsProtected	
MFPMediaItem_QueryInterface	
MFPMediaItem_Release	
MFPMediaItem_SetStartStopPosition	
MFPMediaItem_SetStreamSelection	
MFPMediaItem_SetStreamSink	
MFPMediaItem_SetUserData	

2.3 Misc Functions

2.3.1 IMFPMPCallback_AddRefProc

AddRef method of IMFPMPCallback - a IMFPMediaPlayerCallback object.

```
IMFPMPCallback_AddRefProc PROTO pThis:DWORD
```

Parameters

- `pThis` - pointer to this IMFPMediaPlayerCallback object.

Returns

0

Notes

Used in *MFPMediaPlayer_Init* for the callback function address.

See Also

IMFPMPCallback_QueryInterfaceProc, *IMFPMPCallback_ReleaseProc*

2.3.2 IMFPMPCallback_QueryInterfaceProc

QueryInterface method of IMFPMPCallback - a IMFPMediaPlayerCallback object.

```
IMFPMPCallback_QueryInterfaceProc PROTO pThis:DWORD, riid:DWORD, ppvObject:DWORD
```

Parameters

- `pThis` - pointer to this IMFPMediaPlayerCallback object.
- `riid` - A reference to the interface identifier (IID) of the interface being queried for.
- `ppvObject` - The address of a pointer to an interface with the IID specified in the riid parameter. Because you pass the address of an interface pointer the method can overwrite that address with the pointer to the interface being queried for. Upon successful return, `ppvObject` (the dereferenced address) contains a pointer to the requested interface. If the object doesn't support the interface, the method sets `ppvObject` (the dereferenced address) to nullptr.

Returns

`E_NOINTERFACE`

Notes

Used in *MFPMediaPlayer_Init* for the callback function address.

See Also

IMFPMPCallback_AddRefProc, *IMFPMPCallback_ReleaseProc*

2.3.3 IMFPMPCallback_ReleaseProc

Release method of IMFPMPCallback - a IMFPMediaPlayerCallback object.

```
IMFPMPCallback_ReleaseProc PROTO pThis:DWORD
```

Parameters

- `pThis` - pointer to this IMFPMediaPlayerCallback object.

Returns

0

Notes

Used in *MFPMediaPlayer_Init* for the callback function address.

See Also

IMFPMPCallback_QueryInterfaceProc, *IMFPMPCallback_AddRefProc*

2.3.4 IMFPMediaItemInit

Initializes the IMFPMediaItem object and its methods.

```
IMFPMediaItemInit PROTO pMediaItem:DWORD
```

Parameters

- `pMediaItem` - A pointer to the Media Item (IMFPMediaItem) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

Used to assign pointers to the various IMFPMediaItem methods to variables which are prototyped to the match the method parameters. This allows us to use Invoke to call the methods directly.

Probably easier to call the IMFPMedia functions directly instead of using this function.

See Also

IMFPMediaPlayerInit

2.3.5 IMFPMediaPlayerInit

Initializes the IMFPMediaPlayer object and its methods.

```
IMFPMediaPlayerInit PROTO pMediaPlayer:DWORD
```

Parameters

- `pMediaPlayer` - A pointer to the Media Player (IMFPMediaPlayer) object.

Returns

TRUE if successful or FALSE otherwise.

Notes

Used to assign pointers to the various IMFPMediaPlayer methods to variables which are prototyped to the match the method parameters. This allows us to use Invoke to call the methods directly. This function is called by the *MFPMediaPlayer_Init* function.

See Also

MFPMediaPlayer_Init, *IMFPMediaItemInit*

2.3.6 MFPConvertMSTimeToTimeStringA

Converts a milliseconds value to a time string, which shows hours, minutes, seconds and milliseconds.

```
MFPConvertMSTimeToTimeStringA PROTO dwMilliseconds:DWORD, lpszTime:DWORD,␣
↪dwTimeFormat:DWORD
```

Parameters

- `dwMilliseconds` - milliseconds value to convert.
- `lpszTime` - pointer to string buffer to store the converted time.

- `dwTimeFormat` - 0 to include milliseconds, 1 to exclude them.

Returns

TRUE if successful or FALSE otherwise

Notes

Ensure the string buffer pointed to by the `lpszTime` parameter is at least 16 bytes long.

See Also

2.3.7 MFPConvertMSTimeToTimeStringW

Converts a milliseconds value to a time string, which shows hours, minutes, seconds and milliseconds.

```
MFPConvertMSTimeToTimeStringW PROTO dwMilliseconds:DWORD, lpszTime:DWORD,␣
↪dwTimeFormat:DWORD
```

Parameters

- `dwMilliseconds` - milliseconds value to convert.
- `lpszTime` - pointer to string buffer to store the converted time.
- `dwTimeFormat` - 0 to include milliseconds, 1 to exclude them.

Returns

TRUE if successful or FALSE otherwise

Notes

Ensure the string buffer pointed to by the `lpszTime` parameter is at least 32 bytes long.

See Also

Function	Description
IMFPMediaItemInit	
IMFPMediaPlayerInit	
IMFPMPCallback_AddRefProc	
IMFPMPCallback_QueryInterfaceProc	
IMFPMPCallback_ReleaseProc	
MFPConvertMSTimeToTimeStringA	
MFPConvertMSTimeToTimeStringW	

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search