

# DRMRX Control API – The MediaReactor Interface

© Copyright 2001, Drastic Technologies Ltd.  
12 Drummond Street, Unit 3  
Toronto, Ontario, Canada  
M8V 1Y8  
(416) 255 5636  
(Fax) 255 8780

[www.drastictech.com](http://www.drastictech.com)

DRMRX Control API – The MediaReactor Interface .....	1
Introduction .....	2
Installation .....	3
Concepts .....	4
Examples .....	5
Reference .....	6
Function Group: Utility .....	6
Function Group: File Info/Preview.....	6
Function Group: Get Available Functions .....	7
Function Group: Set File Output.....	7
Function Group: Set Video Output.....	7
Function Group: Set Audio Output .....	8
Function Group: Set Input And Output Files .....	9
Function Group: Process.....	9
Function Group: Translation.....	9
Function Group: Advanced.....	11
Alphabetical Listing .....	11

## ***Introduction***

The MediaReactor activex interface provides a simple interface to the MediaReactor file handling functionality. This activex provides file information on source media files, RGBA and PCM audio reading for thumbnail and preview generation, as well as translation from supported media files and plug-ins to supported output types.

The supplied MediaReactor functionality is split into two independent sections:

1. PreviewXXX functions for getting source file information and generating preview images and sound.
2. Translation functions for enumerating available types, setting source and target files and initiating the translation.

The preview functions are found in the '*Function Group: **File Info/Preview***' section. All other '*Function Group:*' sections refer to file translation.

Please refer to the 'Installation' and 'Concepts' sections before continuing.

## ***Installation***

To facilitate application distribution, two installations are provided with the SDK:

- 1.DTMRX\_Setup####.exe (the # signs indicate the revision number)
- 2.DTMRX\_SetupNoMR.exe

The DTMRX\_Setup####.exe is the full installation of the MediaReactor activex component and should be used for all development installs as well as included in the installation of any application that will use any of the functionality provided by MediaReactor.

The DTMRX\_SetupNoMR.exe is provided to allow the installation of the activex without any of the MediaReactor functionality. This should be used for any application installation that does not require, or is not licensed for, the MediaReactor functionality. This allows distribution of applications with or without MediaReactor.

The default installation directory for MediaReactor is C:\Program Files\Drastic\MediaReactor. This should not be changed as all subsequent updates, plug-ins and utilities will also install to this default directory. Once installed, the activex may be used from any location.

## Concepts

### File Information And Preview

To find information and retrieve images and audio from media files, MediaReactor provides the Preview functions.

Pass the file name to the PreviewOpen() function. PreviewOpen() also allows setting of the output type for audio and video reads. If a particular size is needed, set the VidHorizontalSize and VidVerticalSize to the number of pixels required. Video frames are always returned as 32 bit RGBA. For audio settings, set the AudBitsPerSample, AudSampleRate and AudChannels to match you output or display. To uses the source file settings, set each of these values to -1.

Once the file has been opened, the video and audio information may be retrieved using the PreviewFileXXX functions described in the '*Function Group: File Info/Preview*' section. Video frames and PCM audio may also be read using the PreviewReadVideo and PreviewReadAudio functions.

Once finished, the file should be close with PreviewClose.

### MediaReactor Translation

## **Examples**

### **Visual Basic – DTMRX\_StepOne**

This is the most basic translation application possible. Once the input and output files are set, the DisplayOutputDialog member is called to setup the output type. Once the output type is set, the translate button starts the translation which is monitored by the Progress member in the timer.

### **Visual Basic – DTMRX\_StepTwo**

This example uses the PreviewXXX members to open a media file and display it's information.

### **Visual Basic – DTMRXTest**

This is an example of a user interface that does setup and validation of parameters for a translation. The DisplayOutputDialog and SetOutputPreset are also available.

### **Visual C - DTMXMFC**

This example uses Microsoft Visual C and MFC to create a translation interface.

## Reference

### Function Group: **Utility**

HRESULT CheckSystem([in] long hWnd, [in] BOOL fDisplayErrors)

- Check the operating system for common installation errors. Display fill cause message boxes to be used for each error

### Function Group: **File Info/Preview**

HRESULT PreviewOpen([in] BSTR FileName, [in] long VidHorizontalSize, [in] long VidVerticalSize, [in] long AudBitsPerSample, [in] long AudSampleRate, [in] long AudChannels)

- PreviewOpen to get source file info and read uncompressed frames or samples

HRESULT PreviewReadVideo([in] long Frame, [out] long \* pRGBABuffer, [in,out] long \* BufferSize)

- PreviewReadVideo returns resized video frames as RGBA data

HRESULT PreviewReadAudio([in] long StartSample, [out] long \* pBuffer, [in,out] long \* BufferLength)

- PreviewReadAudio returns converted audio samples as PCM data

HRESULT PreviewClose()

- PreviewClose closes the current preview file

HRESULT PreviewFileHorizontal([out, retval] long \*pVal)

- PreviewFileHorizontal returns the source files original number of horizontal pixels

HRESULT PreviewFileVertical([out, retval] long \*pVal)

- PreviewFileVertical returns the source files original number of vertical pixels

HRESULT PreviewFileVideoChannels([out, retval] long \*pVal)

- PreviewFileVideoChannels returns the source files video channels as a bit array

HRESULT PreviewFileBitDepth([out, retval] long \*pVal)

- PreviewFileBitDepth returns the number of bits per pixel in the source file's video

HRESULT PreviewFileAudioChannels([out, retval] long \*pVal)

- PreviewFileAudioChannels returns the source files audio channels as a bit array

HRESULT PreviewFileInfoChannels([out, retval] long \*pVal)

- PreviewFileInfoChannels returns the source files information channels as a bit array

HRESULT PreviewFileCodec([out, retval] long \*pVal)

- PreviewFileCodec return the codec used by the source file

HRESULT PreviewFileFrameRate([out, retval] double \*pVal)

- PreviewFileFrameRate returns the frame rate of the source file

HRESULT PreviewFileFrames([out, retval] long \*pVal)

PreviewFileFrames returns the number of frames available in the source file

HRESULT PreviewFileSampleRate([out, retval] long \*pVal)

- PreviewFileSampleRate returns the samples rate of the source file's audio

HRESULT PreviewFileSamples([out, retval] long \*pVal)

- PreviewFileSamples returns the number of samples in the source file

HRESULT PreviewFileBitsPerSample([out, retval] long \*pVal)

- PreviewFileBitsPerSample returns the number of bits per sample in the source file's audio

## Function Group: **Get Available Functions**

HRESULT CanTranslate([out, retval] BOOL \*pVal)

- If true, the control can translate from one file type to another

HRESULT CanPreviewRead([out, retval] BOOL \*pVal)

- If true, the control can read pcm audio and RGBA frames via the preview interface

HRESULT CanPreviewWrite([out, retval] BOOL \*pVal)

- If true, the control can write pcm audio and RGBA frames via the preview interface

HRESULT infoSupportedFileTypes([in] long FileCount, [out] BSTR \* FileExt, [out] BSTR \* ShortDesc, [out] BSTR \* LongDesc, [out] long \* plFlags, [out] long \* pFileID)

- Retrieve the supported output file types. FileIDs start at 0 and increment until an error return indicates the last type has been returned

HRESULT infoSupportedCodecTypes([in] long FileID, [in] long CodecCount, [out] BSTR \* ShortDesc, [out] BSTR \* LongDesc, [out] long \* plFlags, [out] long \* pCodecID)

- Retrieve the supported output codec types. FileIDs start at 0 and increment until an error return indicates the last type has been returned

HRESULT infoSupportedProcessTypes([in] long ProcessCount, [out] BSTR \* ShortDesc, [out] BSTR \* LongDesc, [out] long \* plFlags, [out] long \* pProcessID)

- Retrieve the supported processing types. FileIDs start at 0 and increment until an error return indicates the last type has been returned

## Function Group: **Set File Output**

HRESULT DisplayOutputDialog([in] long hwndParent)

- DisplayOutputDialog displays a dialog which allows the user to setup the output parameters for the translation

HRESULT SetOutputPreset([in] long lTargetFileType, [in] BOOL fPAL25Fps)

- Set the output to one of the standard supported types

## Function Group: **Set Video Output**

HRESULT VOutFileType([in] long pVal)

HRESULT VOutFileType([out, retval] long \*pVal)

- Get/Set the main output file type using types returned by infoSupportedFileTypes

HRESULT VOutCodecType([in] long pVal)

HRESULT VOutCodecType([out, retval] long \*pVal)  
•Get/Set the main output codec type using types returned by infoSupportedCodecTypes

HRESULT VOutHorizontalXPixels([in] long pVal)  
HRESULT VOutHorizontalXPixels([out, retval] long \*pVal)  
•Video Output Horizontal Pixels

HRESULT VOutVerticalYPixels([in] long pVal)  
HRESULT VOutVerticalYPixels([out, retval] long \*pVal)  
•Video Output Vertical Pixels

HRESULT VOutBitDepth([in] long pVal)  
HRESULT VOutBitDepth([out, retval] long \*pVal)  
•Video Output Bit Depth

HRESULT VOutQuality([in] long pVal)  
HRESULT VOutQuality([out, retval] long \*pVal)  
•Video Output Quality

HRESULT VOutDataRate([in] long pVal)  
HRESULT VOutDataRate([out, retval] long \*pVal)  
•Video Output Data Rate

HRESULT VOutKeyFrame([in] long pVal)  
HRESULT VOutKeyFrame([out, retval] long \*pVal)  
•Video Output Key Frame

## Function Group: **Set Audio Output**

HRESULT AOutFileType([in] long pVal)  
HRESULT AOutFileType([out, retval] long \*pVal)  
•Get/Set the separate audio output file type using types returned by infoSupportedFileTypes

HRESULT AOutCodecType([in] long pVal)  
HRESULT AOutCodecType([out, retval] long \*pVal)  
•Get/Set the audio output codec type using types returned by infoSupportedCodecTypes

HRESULT AOutFrequency([in] long pVal)  
HRESULT AOutFrequency([out, retval] long \*pVal)  
•Audio Output Frequency

HRESULT AOutSampleSize([in] long pVal)  
HRESULT AOutSampleSize([out, retval] long \*pVal)  
•Audio Output Sample Size

HRESULT AOutChannels([in] long pVal)  
HRESULT AOutChannels([out, retval] long \*pVal)  
•Audio Output Channels

HRESULT AOutQuality([in] long pVal)  
HRESULT AOutQuality([out, retval] long \*pVal)  
•Audio Output Quality Setting

HRESULT AOutSeparateAudio([in] BOOL newVal)  
HRESULT AOutSeparateAudio([out, retval] BOOL \*pVal)  
•Set/Get if a separate audio file is going to be generated

HRESULT AOutDualMono([in] BOOL newVal)  
HRESULT AOutDualMono([out, retval] BOOL \*pVal)  
•Get/Set the separate audio output for mono or stereo output"

## Function Group: **Set Input And Output Files**

HRESULT AddInput([in] BSTR FileName, [in] long StartFrame, [in] long EndFrame, [in] long AuxFrame, [in] long Flags)  
•Add an input file to be translated. This may be a audio video stream or seperate audio and video files added individually using the flags variable

HRESULT AddOutput([in] BSTR FileName, [in] long Flags)  
•Add an output file to write to. Output of interleaved stream is normal, but seperate audio and video streams may be created using the flags variable

## Function Group: **Process**

HRESULT ProcessEnable([in] BOOL newVal)  
HRESULT ProcessEnable([out, retval] BOOL \*pVal)  
•Get/Set if the current process type is enabled

HRESULT ProcessType([in] long newVal)  
HRESULT ProcessType([out, retval] long \*pVal)  
•Get/Set the current process type (one of the values returned by infoSupportedProcessTypes)

HRESULT ProcessP1([in] long newVal)  
HRESULT ProcessP1([out, retval] long \*pVal)  
HRESULT ProcessP2([in] long newVal)  
HRESULT ProcessP2([out, retval] long \*pVal)  
HRESULT ProcessP3([in] long newVal)  
HRESULT ProcessP3([out, retval] long \*pVal)  
•Get/Set the P1 process property (threshold), P2 process property (edge detection) or P3 process property

## Function Group: **Translation**

HRESULT NewTranslation()  
•Initialize for a new translation. If a translation is already running, this call will fail

HRESULT Translate()  
•Start the actual translation. Use the Progress property to monitor translation process

HRESULT TranslationActive([out, retval] BOOL \*pVal)

- Returns TRUE if a translation is currently active

HRESULT Progress([out, retval] long \*pVal)

- Current file translation progress (0 to 1000)

## Function Group: **Advanced**

HRESULT AdvancedSetFileOption([in] long IOID, [in] long Option, [in,out] long \* Setting)

- AdvancedSetFileOption is not normally required. Sets odd file settings for particular file types

HRESULT AdvancedSetCodecOption([in] long FileID, [in] long CodecID, [in] long OptionID, [in,out] long \*Setting)

- AdvancedSetCodecOption is not normally required. Sets odd codec settings for particular codec types

## Alphabetical Listing