



## DTPreviewEngine Programmer's Interface Specification

Copyright 2003-2006 Drastic Technologies Ltd.

All Rights Reserved.

### Table of Contents

INSTALLATION.....	3
Manual Installation.....	3
CONSTANTS.....	4
For Play Mode Commands.....	4
For Time Code Returns.....	4
For Time Code Source Returns.....	5
COMMANDS.....	6
Open.....	6
Close.....	6
PlayModeSupported.....	6
PlayMode.....	6
Refresh.....	6
Play.....	6
PlayAtSpeed.....	6
PlayFromTo.....	7
FastForward.....	7
FastRewind.....	7
Pause.....	7
Seek.....	7
SeekRelative.....	7
SetTrack.....	7
Stop.....	7
Eject.....	8
Duration.....	8
LastFrame.....	8
CurState.....	8
CurSpeed.....	8
CurPosition.....	8
CurTrack.....	8
CurTcType.....	10
CurTC.....	10
CurUB.....	10

CurDATA.....	10
Duration.....	10
Tracks.....	10
VideoTBCSetup.....	11
VideoTBCVideo.....	11
VideoTBCHue.....	11
VideoTBCChroma.....	11
VideoTBCGamma.....	11
AudioOutputLevel.....	11
AudioBassLevel.....	11
AudioTrebleLevel.....	12
SourceMetaDataDWORD.....	13
SourceMetaDataSTR.....	14
GetCurExtendedData.....	16
TargetRect.....	17
TargetTop.....	17
TargetLeft.....	17
TargetWidth.....	17
TargetHeight.....	17
TargetAspectRatio.....	17
GetIn.....	18
SetIn.....	18
GetOut.....	18
SetOut.....	18
VideoLUT.....	18
SaveCurFrame.....	19
SourceFileName.....	19
SourceHeight.....	19
SourceWidth.....	19
SourceBitDepth.....	19
SourceFourCC.....	19
SourceRate.....	20
SourceScale.....	20
SourceBitRate.....	20
SourceFrameSize.....	21
SourceVideoChannels.....	21
SourceAudioChannels.....	21
SourceAudioFrequency.....	21
SourceAudioBitsPerSample.....	21

# ***INSTALLATION***

## **Manual Installation**

Default Installation Directory: C:\Program Files\Drastic\DrasticPreview\

- Install the latest version of DrasticPreview from <http://www.drasticpreview.com>. For convenience it should be installed to the default directory.
- Copy the DTPreviewEngine.dll from the DTPreviewEngine.zip file to the DrasticPreview installation directory.
- Open a command prompt (Start | Run | cmd) and go to the installation directory:
  - C:
  - cd “\Program Files\Drastic\DrasticPreview”
- Register the DTPreviewEngine.dll control.
  - Regsvr32 DTPreviewEngine.dll
- Unpack the example file to the directory of your choice and test the supplied executables.

## CONSTANTS

### For Play Mode Commands

Name	Hex	Decimal
#define PEPLAYMODE_NORMAL	0x0001	// 1
#define PEPLAYMODE_LOOP	0x0002	// 2
#define PEPLAYMODE_PALINDROME	0x0004	// 4
#define PEPLAYMODE_RAM	0x0010	// 16

### For Time Code Returns

```
//! Film 24 FPS time code
#define TC2_TCTYPE_FILM      0x00000001 // 1 - 24 fps
//! Non Drop Frame 30 FPS time code
#define TC2_TCTYPE_NDF      0x00000002 // 2 - NTSC Non Drop Frame
//! Drop Frame 29.97 FPS time code
#define TC2_TCTYPE_DF       0x00000004 // 4 - NTSC Drop Frame
//! PAL 25 FPS time code
#define TC2_TCTYPE_PAL      0x00000008 // 8 - PAL
//! Double PAL 50 FPS
#define TC2_TCTYPE_50       0x00000010 // 16 - PAL 720p (double rate)
//! 720p DROP 59.94 FPS
#define TC2_TCTYPE_5994     0x00000020 // 32 - NTSC 59.94fps 720p
//! 720p 60 FPS
#define TC2_TCTYPE_60       0x00000040 // 64 - NTSC 60fps 720p
//! 23.98 FILM for NTSC 23.98 FPS
#define TC2_TCTYPE_NTSCFILM 0x00000080 // 128 - NTSC FILM 23.98
//! Hundredths of a second HH:MM:SS:/100 100 FPS effective
#define TC2_TCTYPE_100      0x00000044 // 68 -
Hours:Minutes:Seconds:Hundreds
```

## For Time Code Source Returns

```
//! Using absolute position
#define GS_TCSOURCE_ABS    0
//! Using LTC
#define GS_TCSOURCE_LTC    1
//! Using VITC
#define GS_TCSOURCE_VITC   2
```

## **COMMANDS**

### **Open**

HRESULT Open([in]BSTR bstrFileName,[in]long dwFlags,[out,retval]long \* pIRtn);  
Open a new file, stream or network source for preview.

### **Close**

HRESULT Close([out,retval] long \* pIRtn);  
Close the currently open stream or file.

### **PlayModeSupported**

HRESULT PlayModeSupported(long dwTestPlayMode, [out, retval] long \*pVal);  
Returns a bitwise array of supported playback modes (see PlayMode).

### **PlayMode**

HRESULT PlayMode([out, retval] long \*pVal);  
Returns the current play mode (1=Normal, 2=Loop, 4=Plaindrome, 16=Ram).

HRESULT PlayMode([in] long newVal);  
Sets the current play mode (1=Normal, 2=Loop, 4=Plaindrome, 16=Ram).

### **Refresh**

HRESULT Refresh([out,retval] long \* pIRtn);  
Refresh the current frame displays in the video window.

### **Play**

HRESULT Play([out,retval] long \* IRtn);  
Play the current stream and its normal rate.

### **PlayAtSpeed**

HRESULT PlayAtSpeed([in] double lSpeed, [out,retval] long \* IRtn);  
Play the current stream at a specific speed where 1.0 = normal play.

## **PlayFromTo**

HRESULT PlayFromTo([in] long dwFrom, [in] long dwTo, [in] long dwFlags, [in] double ddSpeed, [out,retval] long \* lpRtn);

Play the current stream from a start frame to an end frame (end is exclusive).

## **FastForward**

HRESULT FastForward([out,retval] long \* plRtn);

FastForward the current stream.

## **FastRewind**

HRESULT FastRewind([out,retval] long \* plRtn);

FastRewind the current stream.

## **Pause**

HRESULT Pause([out,retval] long \* plRtn);

Pause the current stream and display the current frame.

## **Seek**

HRESULT Seek([in] long dwFrame, [out,retval] long \* plRtn);

Seek the current stream to a particular frame.

## **SeekRelative**

HRESULT SeekRelative([in] long lOffset, [out,retval] long \* plRtn);

Seek the current stream to a new position relative to the current position.

## **SetTrack**

HRESULT SetTrack([in] long dwTrack, [out,retval] long \* plRtn);

Set the current track in the media or on the device.

## **Stop**

HRESULT Stop([out,retval] long \* lpRtn);

Stop the current stream and return to its beginning (absolute frame 0).

## **Eject**

HRESULT Eject([out,retval] long \* pIRtn);

Eject the current media or device. Only used for removable media. The 'eject' button in most interfaces is actually a file browser and would not use this command.

## **Duration**

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

Return the duration (total number of frames) of the media. This will be one frame more than the last possible display frame. If the duration is 60 frames, then the frames that will actually play will be 0..59 inclusive.

## **LastFrame**

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

Get last frame that will be displayed. This is normally one frame less than the duration/out point, but can be earlier. This is very useful for play lists, especially where the file length is actually changing during play back.

## **CurState**

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

Return the current state of the player (0=stop, 1=pause, 2=play, 5=eject).

## **CurSpeed**

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

Return the current speed of the player (1.0=normal play, -1.0=reverse play, 0.0=pause).

## **CurPosition**

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

Return the current position (absolute 0..end of stream position). To get the current time code position, please see CurTC below.

## **CurTrack**

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

Return the current track being played (1..Tracks inclusive)



## CurTcType

HRESULT CurTcType([in] long dwTcSource, [out, retval] long \*pVal);

Return the current time code type for the LTC or VITC time code. The dwTcSource must be either GS\_TCSOURCE\_ABS or GS\_TCSOURCE\_LTC or GS\_TCSOURCE\_VITC (defined above). It will return TC2\_TCTYPE\_FILM, TC2\_TCTYPE\_NDF, TC2\_TCTYPE\_DF, TC2\_TCTYPE\_PAL, TC2\_TCTYPE\_50, TC2\_TCTYPE\_5994, TC2\_TCTYPE\_60, TC2\_TCTYPE\_NTSCFILM or TC2\_TCTYPE\_100 (also defined above).

## CurTC

HRESULT CurTC([in] long dwTcSource, [out, retval] long \*pVal);

Return the current VITC, LTC or CTL time code. The dwTcSource must be either GS\_TCSOURCE\_ABS or GS\_TCSOURCE\_LTC or GS\_TCSOURCE\_VITC (defined above).

## CurUB

HRESULT CurUB([in] long dwTcSource, [out, retval] long \*pVal);

Return the current VITC or LTC user bits. The dwTcSource must be either GS\_TCSOURCE\_LTC or GS\_TCSOURCE\_VITC (defined above).

## CurDATA

HRESULT CurDATA([in] long dwSource, [in] long dwElement, [out, retval] long \*pVal);

Return the current data elements of the frame. This is a custom requester that should only be used if you know the data type available and that it is supported by the SDK. It is NOT normally used.

## Duration

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

Return the duration (total number of frames) of the media. The frame number of the duration will be one beyond the last physical frame in the file. So if the file is 60 frames long, the available frames will be 0..59 inclusive. To play them all, the in would be 0 and the out would be 60.

## Tracks

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

Return the total number of tracks available

## **VideoTBCSetup**

HRESULT VideoTBCSetup([out, retval] long \*pVal);  
HRESULT VideoTBCSetup([in] long newVal);  
Video TBC Setup (Brightness, range 0..65535, -1=unsupported)

## **VideoTBCVideo**

HRESULT VideoTBCVideo([out, retval] long \*pVal);  
HRESULT VideoTBCVideo([in] long newVal);  
Video TBC Video (Contrast, range 0..65535, -1=unsupported)

## **VideoTBCHue**

HRESULT VideoTBCHue([out, retval] long \*pVal);  
HRESULT VideoTBCHue([in] long newVal);  
Video TBC Hue (Color, range 0..35999, -1=unsupported)

## **VideoTBCChroma**

HRESULT VideoTBCChroma([out, retval] long \*pVal);  
HRESULT VideoTBCChroma([in] long newVal);  
Video TBC Chroma (Saturation, range 0..65535, -1=unsupported)

## **VideoTBCGamma**

HRESULT VideoTBCGamma([out, retval] double \*pVal);  
HRESULT VideoTBCGamma([in] double newVal);  
Video TBC Gamma correction (-1.0 if not supported)

## **AudioOutputLevel**

HRESULT AudioOutputLevel([out, retval] long \*pVal);  
HRESULT AudioOutputLevel([in] long newVal);  
Audio Output Level (range 0..65535, -1=unsupported)

## **AudioBassLevel**

HRESULT AudioBassLevel([out, retval] long \*pVal);  
HRESULT AudioBassLevel([in] long newVal);  
Audio Bass Level (range 0..65535, -1=unsupported)

## **AudioTrebleLevel**

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

HRESULT AudioTrebleLevel([in] long newVal);

Audio Treble Level (range 0..65535, -1=unsupported)

## SourceMetaDataDWORD

HRESULT SourceMetaDataDWORD([in] long dwMetaDataElement, [out, retval] long \*pVal);

Return source metadata information that are numeric (DWORDs or longs). The currently supported string meta data types are:

```
enum {  
    vwwiNumericStart = 0x1000,  
    vwwiTimeCode,  
    vwwiUserBits,  
    vwwiVITCTimeCode,  
    vwwiVITCUserBits,  
    vwwiVITCLine3,  
    vwwiPosterFrame,  
    vwwiAFrame,  
    vwwiAspectRatio,  
    vwwiOriginalRate,  
    vwwiOriginalScale,  
    vwwiConversions,  
    vwwiVersionNumber,  
    vwwiFileSize,  
    vwwiFileDate,  
    vwwiFileTime,  
    vwwiSequenceNumber,  
    vwwiTotalStreams,  
    vwwiTotalLength,  
    vwwiFilmManufacturerCode,  
    vwwiFilmTypeCode,  
    vwwiWhitePoint,  
    vwwiBlackPoint,  
    vwwiBlackGain,  
    vwwiBreakPoint,  
    vwwiGamma1000,  
    vwwiTagNumber,  
    vwwiFlags,  
    vwwiTimeCodeType,  
    vwwiLTCTimeCodeType,  
    vwwiVITCTimeCodeType,  
  
    vwwiVideoWidth = 0x10000,  
    vwwiVideoHeight,  
    vwwiVideoPlanes,  
    vwwiVideoBitCount,  
    vwwiVideoCompression,  
    vwwiVideoSizeImage,
```

```

    vwwiVideoXPelsPerMeter,
    vwwiVideoYPelsPeMeter,
    vwwiVideoClrUsed,
    vwwiVideoClrImportant,
    vwwiVideoReserved,
    vwwiVideoFccType,
    vwwiVideoFccHandler,
    vwwiVideoFlags,
    vwwiVideoCaps,
    vwwiVideoPriority,
    vwwiVideoLanguage,
    vwwiVideoScale,
    vwwiVideoRate,
    vwwiVideoStart,
    vwwiVideoLength,
    vwwiVideoInitialFrames,
    vwwiVideoSuggestedBufferSize,
    vwwiVideoQuality,
    vwwiVideoSampleSize,
    vwwiVideoEditCount,
    vwwiVideoFormatChangeCount,
    vwwiVideoPitch,
    vwwiVideoDrFlags,
    vwwiVideoFileType,
    vwwiVideoResDrastic,
}

```

*Please see <http://www.drasticpreview.org> for more information.*

## **SourceMetaDataSTR**

HRESULT SourceMetaDataSTR([in] long dwMetaDataElement, [out, retval] BSTR \*pVal);

Return source metadata information that are string data. The currently supported string meta data types are:

```

enum {
    vwwiFileName,
    vwwiNativeLocator,
    vwwiUniversalName,
    vwwiIP,
    vwwiSourceLocator,

    vwwiChannel,
    vwwiChannelName,
    vwwiChannelDescription,
}

```

```
vvwiTitle,
vvwiSubject,
vvwiCategory,           // <-- 10
vvwiKeywords,
vvwiRatings,
vvwiComments,
vvwiOwner,
vvwiEditor,
vvwiSupplier,
vvwiSource,
vvwiProject,
vvwiStatus,
vvwiAuthor,             // <-- 20
vvwiRevisionNumber,
vvwiProduced,
vvwiAlbum,
vvwiArtist,
vvwiComposer,
vvwiCopyright,
vvwiCreationData,
vvwiDescription,
vvwiDirector,
vvwiDisclaimer,        // <-- 30
vvwiEncodedBy,
vvwiFullName,
vvwiGenre,
vvwiHostComputer,
vvwiInformation,
vvwiMake,
vvwiModel,
vvwiOriginalArtist,
vvwiOriginalFormat,
vvwiPerformers,        // <-- 40
vvwiProducer,
vvwiProduct,
vvwiSoftware,
vvwiSpecialPlaybackRequirements,
vvwiTrack,
vvwiWarning,
vvwiURLLink,
vvwiEditData1,
vvwiEditData2,
vvwiEditData3,         // <-- 50
vvwiEditData4,
vvwiEditData5,
vvwiEditData6,
vvwiEditData7,
```

```

    vvwiEditData8,
    vvwiEditData9,
    vvwiVersionString,
    vvwiManufacturer,
    vvwiLanguage,
    vvwiFormat, // <-- 60
    vvwiInputDevice,
    vvwiDeviceModelNum,
    vvwiDeviceSerialNum,
    vvwiReel,
    vvwiShot,
    vvwiTake,
    vvwiSlateInfo,
    vvwiFrameAttribute,
    vvwiEpisode,
    vvwiScene, // <-- 70
    vvwiDailyRoll,
    vvwiCamRoll,
    vvwiSoundRoll,
    vvwiLabRoll,
    vvwiKeyNumberPrefix,
    vvwiInkNumberPrefix,
}

```

*Please see <http://www.drasticpreview.org> for more information.*

## **GetCurExtendedData**

HRESULT GetCurExtendedData([in,out] VARIANT \*pvData, [in,out] long \*plSize);

Get current extended frame data. This is custom frame data, embedded with each video frame. It is completely dependent on the source file and should not be used unless you are VERY familiar with the format of the data.

## TargetRect

HRESULT TargetRect([in] long dwX, [in] long dwY, [in] long dwWidth, [in] long dwHeight, [out,retval] long \* pIRtn);

Set the target rectangle for the video window. On older version of DrasticPreview, this had to be either full size (of the original video image), ½ or ¼ sized. More recent SDKs support arbitrary sizing of the output video window.

**For Visual Basic Users:** This call must be made with lines and pixels. To make sure you are using pixels for your form, make sure your ScaleMode is set to “3 – pixels” before calling TargetRect with the .Width/.Height or .ScaleWidth/.ScaleHeight.

## TargetTop

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

Target frame top position (Y absolute coordinate)

## TargetLeft

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

Target frame left position (X absolute coordinate)

## TargetWidth

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

Target frame width in pixels (CX delta value).

## TargetHeight

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

Target frame height in lines (CY delta value).

## TargetAspectRatio

HRESULT TargetAspectRatio([in] double valOne, [in] double valTwo);

Set target aspect ratio. This is to set the aspect ratio of your output display. It defaults to 4:3, which is the aspect ratio of most computer monitors, projectors and televisions. Some projectors and monitors have a 16:9 aspect ratio which can be compensated for here.



## **GetIn**

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

Returns the current in point (see SetIn). This is the first frame to be displayed in a preview or loop. It is included and displayed, unlike the out point.

## **SetIn**

HRESULT In([in] long newVal);

Set the in (or start) point. This is the first frame to be displayed in a preview or loop. It is included and displayed, unlike the out point.

## **GetOut**

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

Returns the current out (or end) point. This is one frame AFTER the last frame to display. The out point is never included, so a file with an outpoint set at frame 60 (2:00 NDF) will display 60 frames from 0..59 inclusive. It will never reach 60.

## **SetOut**

HRESULT Out([in] long newVal);

Set the out (or end) point. This is one frame AFTER the last frame to display. The out point is never included, so a file with an outpoint set at frame 60 (2:00 NDF) will display 60 frames from 0..59 inclusive. It will never reach 60.

## **VideoLUT**

HRESULT VideoLUT([in] long dwLUT, [in] long dwType);

Set video look up table (LUT). This is currently only used for 10 bit linear/logarithmic frame types such as DPX, Cineon and 10 bit per component RGB QuickTime Movie files.

## SaveCurFrame

HRESULT SaveCurFrame([in] BSTR bstrFileName, [in] long dwParam, [out,retval] long \* pIRtn);

Save the current frame as a still image. This is not currently implemented, but will eventually support JPG, BMP as well as YUV or DPX for some files.

## SourceFileName

HRESULT SourceFileName([out, retval] BSTR \*pVal);

The final file name used for the source fil. This may be altered from the file name that was used to open the file. For instance, opening a sequence of stills with V:\Test\NotTest\_0002.dpx would return V:\Test\NotTest\_\*.dpx.

## SourceHeight

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

Source video media's height in lines.

## SourceWidth

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

Source video media's width in pixels.

## SourceBitDepth

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

Source video media's bit depth. Normally 1,4,8,12,15,16,24,32,48 or 64. This is for all components in 1 pixel. For instance, an MPEG file may have 8 bit precision (per component), but would still have 12 (4:2:0) or 16 (4:2:2) bits per pixel (bit depth). In RGB a 24 bit depth TGA image would still have three 8 bit components (R:8,G:8,B:8).

## SourceFourCC

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

Source video media's fourcc compression code. FourCC stands for Four Character Code. This is the most basic description of a codec in video for windows, QuickTime and other file formats. Here it is extended to all supported types for compatibility. It is usually four normal characters (human readable), but can also be 0 (RGB), 1..4 (RGB Bit Fields) or other non displayable values.

## SourceRate

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

Source video frame rate 'rate' value ( $\text{FPS} = \text{SourceRate} / \text{SourceScale}$ ). Around 30 is generally NTSC, 25 is PAL and 24/23.9 is FILM. HD can use any of these frame rates.

Typical Values:	Rate	Scale	FPS
	30	1	30
	30000	1001	29.97...
	2970	100	29.97
	25	1	25
	24	1	24
	24000	1001	23.98

## SourceScale

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

Source video frame rate 'scale' value ( $\text{FPS} = \text{SourceRate} / \text{SourceScale}$ ). Around 30 is generally NTSC, 25 is PAL and 24/23.9 is FILM. HD can use any of these frame rates.

Typical Values:	Rate	Scale	FPS
	30	1	30
	30000	1001	29.97...
	2970	100	29.97
	25	1	25
	24	1	24
	24000	1001	23.98

## SourceBitRate

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

Source video media's bit rate in kilobytes per second. Multiply by 1024 to get bytes per second or 8096 to get bits per second. This is the overall bit rate for any streams in the main file/sequence. It is for reference only and should not be considered canonical.

## **SourceFrameSize**

HRESULT SourceFrameSize([in] long dwFrame, [out, retval] long \*pVal);  
Source video media's frame size in bytes for the requested frame (dwFrame) or current frame (if dwFrame == -1).

## **SourceVideoChannels**

HRESULT SourceVideoChannels([out, retval] long \*pVal);  
Source video total channels. This is an absolute figure (eg 6 for six channels) and not a bit wise figure (which would be hex 3f for six channels).

## **SourceAudioChannels**

HRESULT SourceAudioChannels([out, retval] long \*pVal);  
Source audio total channels. This is an absolute figure (eg 6 for six channels) and not a bit wise figure (which would be hex 3f for six channels).

## **SourceAudioFrequency**

HRESULT SourceAudioFrequency([out, retval] long \*pVal);  
Source audio media frequency

## **SourceAudioBitsPerSample**

HRESULT SourceAudioBitsPerSample([out, retval] long \*pVal);  
Source audio media bits per sample. Depending on the compression type, this may be complete accurate, rough or in accurate. Mostly correct for RGB types, but YcbCr types are normally marked as 16 bits per pixel, which is somewhat correct for 8 bits per component, but incorrect for 10 bits per component.