MediaCmd API

Version 4.0

Drastic Technologies
523 The Queensway, Suite 102
Toronto, ON, M8Y 1J7
CANADA
(416) 255 5636
(Fax) 255 8780
http://www.drastictech.com

Contents

MediaCmd API....1
Version 4.0....1
Contents....2
Introduction....5
Components....10
References....11
Interface....12
JAVA....13
Channel Control....14
EnableChannels....14
GetMaxChannels....14
SetCurChannel....15
GetCurChannel....15
GetCurChannelName....15
Long vvwGetChannelName(long lChannel, char * szChanName)....15
GetCurChannelType....16
long vvwGetChannelType (long lChannel)....16
ShowConfigDialog....16
Network Specific....17
OpenChannel (vvwNet2 ActiveX)....17
CloseChannel (vvwNet2 ActiveX)....17
Connect....17
Disconnect....18
Transport Operations....19
Play....19
PlayAtSpeed....19
PlayFromTo....20
LoadClip....20
PlayClip....21
PlayClipFromTo....21
PlayAtMs....22
FastFoward....22
FastRewind....22
Pause....23
Seek....23
SeekRelative....23
Stop....24
Record....24
RecordAtMs....24
RecordFromTo....25
RecordStop (prepare record)....25
SetRecordPresets....26
Eject....26
Special Commands....27
Transfer....27

Status Operation....28
UpdateStatus....28
GetState....29
GetFlags....29
GetSpeed....30
GetPosition....31
GetLastMs....31
GetStart....31
GetEnd....31
GetClipName....32
GetFileName....32
GetCurTC....32
GetCurState....33
Media Operations....34
Clip Mode Operations....34
GetNextClip....34
GetClipInfo....35
GetClipInfoEx....36
CopyClip....36
VTR Mode Operations....37
EDLResetToStart....37
EDLGetEdit....37
Shared Operations....39
GetLastChangeMs....39
Insert....40
Blank....41
Delete....42
Trim....43
Settings....44
ValueSupported....44
ValueGet....44
ValueSet....45
ValueSet2....45
Settings Format....45
Base Settings....46
Get/SetClipMode....46
Get/SetTCType....46
Get/SetTCSource....46
Get/SetAutoMode....46
GetAvailablePresets....46
Audio Settings....47
Get/SetAudioInput....47
Get/SetAudioInputLevel....47
Get/SetAudioOutput....48
Get/SetAudioOutputLevel....48
GetAudioPeakRMS....48
Video Settings....48
Get/SetVideoInput....48
Get/SetVideoOutput....49

Get/SetVideoInputSetup....49
Get/SetVideoInputVideo....49
Get/SetVideoInputHue....50
Get/SetVideoInputChroma....50
Get/SetVideoTBCSetup....50
Get./SetVideoTBCVideo....50
Get/SetVideoTBCHue....50
Get/SetVideoTBCChroma....51
Get/SetVideoGenlock....51
Get/SetVideoGenlockSource – not implemented externally....51
Error Log....51
vvwGetErrorLogMs....51
vvwSetErrorLog....51
vvwGetErrorLength....51
vvwGetError....51
System Settings....51
Get/SetCompressionRate....51
Get/SetSuperImpose – not implemented externally....52
GetTotalTime....52
GetFreeTime....52
GetTotalStorage....52
GetFreeStorage....52
GetCurMs....52
GetChannelCapabilities....52
GetVVWVersion....52
GetMRVersion....52
GetVVWType....52
Utility Functions....53
FreeString....53
TCMaxFrame....53
TCToFrame....54
TCToString....55
VVWSpeedToPercentage....55
PercentageToVVWSpeed....55
Java Data Structure Definitions....56
ClipInfo....56
Data members 56
VTREditLine....56
GetValueMcmd....57
Data members 57
AvailabePresets....57
Appendix I – MediaCmd.h....58
Appendix II – VVWIF.h....227
Appendix III – VvwTypes.h....250
Appendix IV – VVWMediaCmd.bas....324
Appendix V – MEDIACMD.java....324

Introduction

The Drastic MediaCmd SDK is the mechanism by which all the elements of Drastic's DDRs communicate with one and other. This includes:

- Controlling Titan/VVW DDR Servers locally or remotely
- Controlling QuickClip locally
- Controlling QuickClip remotely with the network server option
- Controlling 9 pin serial VTRs and Servers via Sony, Odetics or VDCP protocol
- Receiving commands from 9 pin serial controller via Sony, Odetics or VDCP protocol
- Receiving commands from Drastic GUIs, servers and controllers
- Building HTML/Ajax status and control pages

MediaCmd is the communication method used within Drastic's DDR products. Any operation you see in a Drastic interface is available to your application through MediaCmd.

Overview

MediaCmd is a simple structure that supports a small, well defined set of commands for communicating transport, status and setup information between components in Drastic's DDR software. There are a number of fields in the stucture, but the important fields are:

* ctCmd – the primary command of this packet (Play, Pause, Stop, Record, etc)
* lSpeed – the transport speed for any play commands (integer where 65520 = normal forward play)
* dwPosition – the frame position for any play, pause or record commands
* dwStart – the starting frame for any play or record commands (inclusive)
* dwEnd – the ending frame for any play or record commands (exclusive)
* arbID – clip name, file name or other string/binary data for the command
* cfFlags – denotes which fields above are valid and their meaning

With the standard initialization of the structure, you can quickly build commands in this structure by changing a few members and sending it. The primary motion commands are ctPlay, ctPause, ctStop, ctRecStop, ctRecord, ctEject and ctTransfer. To get the current state (position, speed, start and end, current clip), the command ctGetState will return a filled in MediaCmd. For setup and less common status (e.g. video input, audio rms level, genlock) there is ctGetValue and ctSetValue. This is documented in the Low Level Header Docs.

Hopefully, you will not have to deal with the MediaCmd structure directly. The SDK includes a series of simple commands that should provide 99% of what your application needs. These functions are simply wrappers that create and send MediaCmd structures. The source for all these functions is provided in the SDK under SRC\General\vvwIF.cpp in case you need to modify or create new commands. The commands have slightly different names depending on which interface you use, but have the same root name, such as:

Play(), PlayFromTo(), Stop(), Pause(), Seek(), Record() and UpdateStatus(). Commands are also included for getting clip lists (GetNextClip()) and EDL elements from ::VTR_TC time code spaces (EDLResetToStart(), EDLGetEdit()). A selection of the most common settings are also included (SetVideoInput(), SetAudioInput(), SetVideoGenlock(), GetAudioPeakRMS(), etc). This interface is documented in the MediaCmd Documentation (previously called "VVW Interface Specification").

Installation

To properly work with the MediaCmd SDK, you should have a copy of the QuickClip software installed on your development system. Even if you target application will only use a part of the QuickClip software, it should all be installed for the development phase. Before proceeding with the SDK you should familarize your self with QuickClip's operation and toolset. All the elements available within QuickClip are the same elements available to your application through the SDK.

Once you have QuickClip installed, you should install the MediaCmd SDK. This will install the headers, libraries and source needed to control QuickClip from your application.

Choosing An Access Method

The SDK access method you should use depends on what you you would like your application to do, what programming language you are using and how involved you would like to/need to get in the low level MediaCmd structures. No matter which method you choose, the MediaCmd structure packets are exactly the same. Here are the main access method, with their pros and cons:

ActiveX – Direct On Server
Type: Microsoft ActiveX/COM access method
Pros: Easy to program, 1:1 relationship with QuickClip/XO interface.
Cons: Uses same config as QuickClip/XO. Requires a local copy of QuickClip.
Setup: Register VVW.DLL using RegSvr32.exe in the QuickClip installation directory.
Issues: Difficult to use when communicating via TCP/IP within the same machine. Can be overcome by using the default pipe communication system, but this requires changes for remote network control.

ActiveX – Network Remove
Type: Microsoft ActiveX/COM access method
Pros: Easy to program. No permanent config. Setup by user calls each time.
Cons: No permanent config. Setup by user calls each time.
Setup: Register vvwNet2.DLL using RegSvr32.exe.
Issues: Each connection must be setup through the ActiveX by the SDK caller

Ajax/HTTP – Standard Network
Type: Standard HTTP calls with XML Ajax returns
Pros: Easy to program. Realtime debug (firefox). Works with most browsers.
Cons: Can be slow in some instances, JavaScript somewhat limited.
Setup: Connect to server with you web browser
Issues: No permanent config beyond standard server

Java Direct – Network
Type: Network MediaCmd access via port 1234
Pros: Easy to program, some interface source included
Cons: Requires NDA, Drastic approval
Setup: Add package to project and call
Issues: Each connection must be setup by caller

Direct Link – no longer actively supported, use ActiveX
Type: Direct link to VVW.DLL
Pros: No ActiveX layer, code compatible with Linux, Irix, Mac OS-X.
Cons: Uses default config from QuickClip/XO, application must be run in QuickClip directory. Requires a local copy of QuickClip.
Setup: Link to vvw.lib, include vvw.h. Copy application into the QuickClip directory before running
Issues: Needs access to VVW.dll and all its support DLLs/D1Xs. Still needs to be setup by LocalConfig.exe or QuickClip/XO

Network DLL – no longer actively supported, use ActiveX
Type: Direct line to vvwNet2.dll
Pros: Consistant interface between local/remote and various OSs. Does not require a local copy of QuickClip.
Cons: Requires vvwNet2.dll and support dlls
Setup: Link to vvwNet2.lib, include vvwNet2.h. Copy dll set from SDK\bin directory with your application
Issues: Use the vvwOpenLocal function to avoid QuickClip configuration issues. Requires a few dlls to be added to you application installations. Does not run the client software automatically, so you application may need to start it, depending on what your application is doing.

Network Direct
Type: Direct compile of network sources in your app or your dll.
Pros: No extra dlls. Easy to customize and modify. Lots of comands already written.
Cons: You app needs to handle setup and may need to run QuickClip.exe/VVWServer.exe/QCRun.exe.
Setup: Copy source files from vvwNet2 into you project, modify and compile
Issues: Does not run the client software automatically, so you application may need to start it, depending on what your application is doing.

Manual
Type: Use the structures and defines to write your own communication and control layer.
Pros: This is required if you are using an unsupported development platform like PHP.
Cons: Everything has to be built and tested from the ground up.
Setup: None.
Issues: Unless you absolutely have to, this method is not recommended.

SDK Structure

The location of the SDK directories will depend on the location you choose during the installation, but the directories within there will always be the same:

* \BIN – Copies of the minimum dll set from a QuickClip installation.
* \LIB – Libraries required to link the vvwNet2.dll, examples and your application
* \INC – Header files required to compile vvwNet2.dll, examples and your application
* \Src\vwwNet2 – The source to our vvwNet2.dll from QuickClip
* \Src\General – Useful source files that do not compile into examples directly. The most important would be vvwIF.cpp that is the code behind the SDK functions described below.
* \Sample – Broken down into sub directories based on access type
    - \ActiveX – Examples that use the ActiveX control
    - \Direct – Examples that link directly to DLLs
    - \Java – Java based examples
    - \HTTP – Ajax based examples (must use QuickClip HTTP server to run)

Main Documentation

*** PDF version of the MediaCmd Documentation
http://www.drastictech.com/manuals/mediacmd4api.pdf

*** Online wiki
http://www.drasticpreview.org/wakka.php?wakka=DrasticSoftwareSdks&v=131g

Low Level Header Documentation

*** Windows CHM help file version of the MediaCmd headers
http://www.drastictech.com/manuals/mediacmd4api.chm

*** Online version of the MediaCmd headers
http://www.drastictech.com/manuals/mediacmd/

HTTP XML AJAX Documentation

*** Wikki area for HTTP XML MediaCmd
http://www.drasticpreview.org/wakka.php?wakka=DrasticHttpCommands&v=7sv

Components

The VVW interfaces are designed to allow VWW customers, OEMs and Drastic component OEMS to create custom control solutions in the simplest manner possible. We have attempted to create a series of similar interfaces across as many development environments as possible.

Controllable Components:

VVW	- Contains all other components
vvwXXX	- Hardware driver components for VVW-1000 through 7000 series DDRs
vvvNET2	- Direct network sender to VVW or user component (TCP/IP)
vvwDSync	- House clock/SMPTE LTC, Local GPI Control

Controlling Components:

vvwCTL	- RS-422/Network Interpreter for VTR/DDR Emulation
vvwNET2	- Direct network interpreter from VVW or user component (TCP/IP)
vvwHTTPD	- XML and Web based network control

Interface Types:

ActiveX	- MediaCmdX	- Visual C++, Visual Basic, Borland C/Delphi, etc
Java (direct)	- MediaCmdInterface	- Visual J++, Sun Java, Symantec
Unix	- mediacmd.so	- GNU (Linux/BSD)
Unix	- mediacmd.c	- Shrouded Source
HTML	- mediacmd.xml	- XML I/O (w HTML components)
XML	- mediacmd.xml	- XML direct network connection

Do not use this interface type (being removed):

VVW	- VVW.lib/VVW.h	- Direct DLL Control

References

MediaCmd.h	- Internal media cmd structures on which this interface is based.
VVWIF.h	- VVW Direct Access (includes previous access method as well)
VVWX.tlb	- MediaCmdX type library
MediaCmd.xml	- XML command set mirroring MediaCmd.h
SimpleVB/2	- Simple control from Visual Basic
SimlpleMFC/2	- Simple control from Visual C++ with MFC
CntrlVVW_Java	- Simple control from Visual J++/WFC
CntrlVVW_Cmd	- Command line controller using the DLL methods
VTRControl	- Test app for VTR control
LocalConfig	- VVW Setup
X-extdev.prm	- VTR plug in for Adobe Premiere
QuickClipProDVI	- Software only DDR

Interface

The following is assumed:

Language/Platform	Name	Notes
ActiveX VB	Vbx	ActiveX Control
ActiveX C++	Pcx	Pointer to ActiveX Control
Java Instantiation	Mci	MediaCmd Interface – Network Only
DLL		No prefix
Unix		No prefix – Network Only
XML		Via HTTP or Direct

- All pointers for ActiveX C++, DLL and Unix must be valid and correctly sized
- Any BSTR returns are freed by the caller using our utility function FreeString()
- Any char * returns are freed by the caller using our utility function FreeString()
- For Unix the type VVWBOOL is defined as typedef long VVWBOOL
- For ActiveX/Dll, BOOL is the Windows.h BOOL definition
- szClipName has a maximum of 8 alpha numeric characters as per the Louth and Odetics specifications
- szFileName is DOS/Windows formatted in one of two forms
  - X:\Some Path\On The Drive\Media Files Name.ext
  - \\VVWSERVER\X\ Some Path\On The Drive\Media Files Name.ext
- Not all channels are equally capable. Check functions before assuming they will work as they do on other VVW systems or channels. Each VVW System and associated channel will be consistent. Please check you server documentation for its specific features.

Nomenclature:

szClipName. Sz8CharClipName, etc

Any character area identified by the words clip and name refers to a Louth/Odetics style clip name. These names may be up to 8 characters in length plus a NULL terminating character. This means they should be allocated in the following manner:

```
Char szClipName[9] = "";            // Include unused 1 char safety
Char pClipName = new char[9];
Char szpClipName = malloc(9);
```

SzFileName, sz260FileName, etc

Any character area identified by the words file and name refers to a win32 style drive/path/file/ext name. These areas may be up to 260 characters in length plus a terminating NULL character. This means they should be allocated in the following manner:

```
Char szClipName[261] = "";          // Include unused 1 char safety
Char pClipName = new char[261];
```

```
Char szpClipName = malloc(261);
```

Time code strings

Time code strings may be as small as one character and have a maximum length of 14 characters plus a terminating NULL. This means they should be allocated in the following manner:

```
Char szClipName[15] = "";                // Include unused 1 char safety
Char pClipName = new char[15];
Char szpClipName = malloc(15);
```

JAVA

To use the Java VVW interface begin by importing the VVW interface package. The Java VVW Interface is neatly packaged in the VVWInterface.zip file provided.

```
importdrastic.mCmdIF.*;
```

You must ensure that the VVWInterface.zip file is accessible by the JVM. If the VVWInterface.zip file is not in the same file structure as your current project you must tell the JVM where to look for it by setting your classpath to it's current location.

After you have successfully imported the VVW interface package begin by instantiating a new instance of the MediaCmdIF class.

```
MediaCmdIFmci = new MediaCmdIF();
```

The newly instantiated MediaCmdIF object will have access to all VVW interface methods. Some MediaCmdIF methods require the user to instantiate a new object of the method's parameter type.

```
MediaCmdIF.ClipInfoclipData= new MediaCmdIF.ClipInfo(clipName)
mci.GetClipInfo(clipData);
long lEnd= clipInfo.lEnd;
```

This allows the VVWInterface to return multiple values in the defined class structure.

Note: Applets using the VVW Interface must have an archive tag in the applet's HTML file specifying the location of the VVWInterface.zip file.

Channel Control

EnableChannels

ActiveX VB	vbx.EnableChannels (lInternal0_31 As Long, lInternal32_63 As Long, lExternal64_95 As Long, lExternal96_127 As Long, lNetwork128_159 As Long, lNetwork160_192 As Long) As Long
ActiveX C++	long pcx->EnableChannels (long lInternal0_31, long lInternal32_63, long lExternal64_95, long lExternal96_127, long lNetwork128_159, long lNetwork160_192)
Java	*Not Available*
Dll	long vvw EnableChannels (long lInternal0_31, long lInternal32_63, long lExternal64_95, long lExternal96_127, long lNetwork128_159, long lNetwork160_192)
Unix	*Not Available*
XML	*Not Available*

Enable or disable channels based on the bit array supplied. VVW can contain up to 256 channels per access point. Channels 193-255 are disabled by default. The remaining channels may be enabled (if the corresponding bit is set to 1) or disabled (if the corresponding bit is set to 0) with this call. The first 64 channels (0 through 63) are reserved for internal ddr channels. Then next 64 channels (64 through 127) are reserved for VTR or DDR devices controlled via serial, Odetics or Louth protocol. The remaining channels are for controlling other devices through the network. Please note that a network channel controls all the channels on the network server box, so disabling one network connection may disable more than one channel. Always call GetMaxChannels() after setting the bits to make sure all the channels you expect exist actually exist. This should be the first call made to the activex component.

GetMaxChannels

ActiveX VB	vbx.GetMaxChannels () As Long
ActiveX C++	long pcx->GetMaxChannels ()
Java	long mci.GetMaxChannels ()
Dll	long vvwGetMaxChannels (long lChannel)
Unix	long GetMaxChannels ()
XML	*Not implemented*

Returns the maximum number of channels available for control. Channels start at 0 and end at max channels – 1. This return is one greater than the largest value available for SetCurChannel(), GetCurChannel() and the lChannel parameter for the DLL interface. This value will change if channels are being opened and closed via OpenChannel/CloseChannel, and every channel between 0 and GetMaxChannels() may not be active.

SetCurChannel

ActiveX VB	vbx.GetMaxChannels (lChannel As Long) As Long
ActiveX C++	long pcx->GetMaxChannels (long lChannel)
Java	long mci.SetMaxChannels (long lChannel)
Dll	*Not implemented*
Unix	long SetCurChannel (lChannel As long)
XML	Use 'setchannel=

Sets the channel to which all subsequent commands will be sent. This command does not exist in the Dll interface as the channel is sent on a per command basis.

Returns 0 or an error code

GetCurChannel

ActiveX VB	vbx.GetCurChannels () As Long
ActiveX C++	long pcx->GetCurChannels ()
Java	long mci.GetCurChannels ()
Dll	*Not implemented*
Unix	long GetCurChannel ()
XML	*Not required*

Get channel currently under control. This value will be between 0 and GetMaxChannels – 1.

GetCurChannelName

ActiveX VB	vbx.GetCurChannelName () As String
ActiveX C++	BSTR pcx->GetCurChannelName ()
Java	String mci.GetCurChannelName ()
Dll	Long vvwGetChannelName(long lChannel, char * szChanName)
Unix	long GetCurChannelName (char * szChanName)
XML	*Not required*

Get the name of the current channel. For unix and dll access, pass a null to get the channel name size, then pass in a pointer that points to a memory size of at least that many bytes (ANSI characters only).

GetCurChannelType

ActiveX VB	vbx.GetCurChannelType () As Long
ActiveX C++	long pcx->GetCurChannelType ()
Java	long mci.GetCurChannelType ()
Dll	long vvwGetChannelType (long lChannel)
Unix	long GetCurChannelType ()
XML	*Not supported*

Returns the basic type of the channel (VTR, Internal, User, House)

VVW_CHANTYPE_HOUSE	0x1	1
VVW_CHANTYPE_INTERNAL	0x2	2
VVW_CHANTYPE_VTR_DDR	0x4	4
VVW_CHANTYPE_UNKNOWN	0xFFFFFFFF	-1

ShowConfigDialog

ActiveX VB	vbx.ShowConfigDialog (hWnd as Long) As Long
ActiveX C++	long pcx->ShowConfigDialog (long hWnd)
Java	*Not Available*
Dll	long vvw ShowConfigDialog (long hWnd)
Unix	*Not Available*
XML	*Not Available*

Show the configuration dialog box for the current channel. If the channel does not have a configuration dialog, this function will return an error. It is not available in Java or unix as the dialog only shows up on the local machine, and cannot be seem through the network.

Network Specific

OpenChannel (vvwNet2 ActiveX)

ActiveX VB	vbx.OpenChannel (BSTR szAddress, long lPort, long lRemoteChannel) As Long
ActiveX C++	long pcx->OpenChannel (BSTR szAddress, long lPort, long lRemoteChannel)
Java	*Not implemented*
Dll	vvwOpenLocal(char * szServerAddress, long lPort, long lRemoteChannel)
Unix	vvwOpenLocal(char * szServerAddress, long lPort, long lRemoteChannel)
XML	*Not required*

Open a local channel to a remote (or localhost) server specified in szServerName (either the IP address or network name) using a specific port (default 1234) for a specific remote channel (default 0).

Returns local channel identifier if the connect was successful, else it returns -1. Please do not make any assumptions about what the local channel identifier will be, especially if you are opening and closing multiple channels. Closed identifiers may later get used for newly opened channels.

CloseChannel (vvwNet2 ActiveX)

ActiveX VB	vbx.OpenChannel (long lLocalChannel)
ActiveX C++	pcx->OpenChannel (long lLocalChannel)
Java	*Not implemented*
Dll	vvwClose (long lLocalChannel)
Unix	vvwClose (long lLocalChannel)
XML	*Not required*

Disconnect from a previously connected server using the local channel value returned by OpenChannel.

Connect

ActiveX VB	*Not implemented*
ActiveX C++	*Not implemented*
Java	boolean mci.Connect(String szServerAddress, int lPort)
Dll	*Not implemented*
Unix	Long Connect (char * szServerAddress, long lPort)
XML	*Not required*

Connect to a server specified in szServerName (either the IP address or network name) using a specific port (default 1234). For ActiveX and Dll access, please setup a persistent network connection in the VVW configuation utility.

Returns true if the connect was successful, else it returns false.

Disconnect

ActiveX VB	*Not implemented*
ActiveX C++	*Not implemented*
Java	boolean mci.Disconnect()
Dll	*Not implemented*
Unix	Long Disconnect()
XML	*Not required*

Disconnect from a previously connected server. For ActiveX and Dll access, please setup a persistent network connection in the VVW configuation utility.

Transport Operations

Play

ActiveX VB	vbx.Play () As Long
ActiveX C++	long pcx->Play ()
Java	long mci.Play ()
Dll	long vvwPlay (long lChannel)
Unix	long Play ()
XML	http://localhost/VVWXMLMediaCmd?Play

Play at normal speed.

Returns 0 if successful, else an error code.

PlayAtSpeed

ActiveX VB	vbx.PlayAtSpeed (lVVWSpeed As Long) As Long
ActiveX C++	long pcx->PlayAtSpeed (long lVVWSpeed)
Java	long mci.PlayAtSpeed (long lVVWSpeed)
Dll	long vvwPlayAtSpeed (long lChannel, long lVVWSpeed)
Unix	long PlayAtSpeed(long lVVWSpeed)
XML	http://localhost/VVWXMLMediaCmd?Play &speed=lVVWSpeed

Play at a particular VVW speed. VVW speeds use a base play speed of 65520. This means that play = 65520, reverse play = -65520, four times play = 262080, half play speed = 32760. Percentage play speeds may be converted to VVW speeds using the PercentageToVVWSpeed() function. For Speed calculations please see GetSpeed() below.

Returns 0 if successful, else an error code.

PlayFromTo

ActiveX VB	vbx.PlayFromTo (lFrom As Long, lTo As Long, fDeferred As Boolean) As Long
ActiveX C++	long pcx->PlayFromTo (long lFrom, long lTo, BOOL fDeferred)
Java	long mci.PlayFromTo (long lFrom, long lTo, boolean fDeferred)
Dll	long vvwPlayFromTo (long lChannel, long lFrom, BOOL lTo, bool fDeferred)
Unix	long PlayFromTo (long lFrom, long lTo, VVWBOOL fDeferred)
XML	http://localhost/VVWXMLMediaCmd?Play&start=lFrom &end=lTo

Play from a frame to another frame. As with editing systems, the 'from' point is included and will be displayed but the to point is NOT included and will not be displayed. This means that the last frame displayed will be lFrom – 1. The deferred flag allows

PlayFromTos to be stacked so that they will play back to back. The deferred flag in the status return should be false before another deferred command is added.

Returns 0 if successful, else an error code.

LoadClip

ActiveX VB	vbx.LoadClip (szClipName As String, lStartFrame As Long) As Long
ActiveX C++	long pcx-> LoadClip (BSTR szClipName, long lStartFrame)
Java	long mci. LoadClip (String szClipName, long lStartFrame)
Dll	long vvwLoadClip (long lChannel, char * szClipName, long lStartFrame)
Unix	long LoadClip (char * szClipName, long lStartFrame)
XML	http://localhost/VVWXMLMediaCmd?Pause &ClipID=szClipname

Clip Mode Only. Load a clip into the channel and display the lStartFrame.

Returns 0 if successful, else an error code.

PlayClip

ActiveX VB	vbx.PlayClip (szClipName As String, fDeferred As Boolean) As Long
ActiveX C++	long pcx->PlayClip (BSTR szClipName, BOOL fDeferred)
Java	long mci.PlayClip (String szClipName, boolean fDeferred)
Dll	long vvwPlayClip (long lChannel, char * szClipName, BOOL fDeferred)
Unix	long PlayClip(char * szClipName, VVWBOOL fDeferred)
XML	http://localhost/VVWXMLMediaCmd?Play &ClipID=szClipname&Flags=Deferred

Clip Mode Only. Play the entire clip specified by clip name. If the deferred flag is true, clip playback will only occur once the currently playing clip has finished. If there is no currently playing clip, playback will occur immediately.

Returns 0 if successful, else an error code.

PlayClipFromTo

ActiveX VB	vbx.PlayClipFromTo (szClipName As String, lFrom AS Long, lTo As Long, fDeferred As Boolean) As Long
ActiveX C++	long pcx->PlayClipFromTo (BSTR szClipName, long lFrom, long lTo, BOOL fDeferred)
Java	long mci.PlayClipFromTo (String szClipName, long lFrom, long lTo, boolean fDeferred)
Dll	long vvwPlayClipFromTo (long lChannel, char * szClipName, long lFrom, long lTo, BOOL fDeferred)
Unix	long PlayClipFromTo (char * szClipName, long lFrom, long lTo, VVWBOOL fDeferred)
XML	http://localhost/VVWXMLMediaCmd?Play&start=lFrom &end=lTo&ClipID=szClipname

Clip Mode Only. Play the specified portion of the clip specified by clip name. If the deferred flag is true, clip playback will only occur once the currently playing clip has finished. If there is no clip currently playing, playback will occur immediately.

Returns 0 if successful, else an error code.

PlayAtMs

ActiveX VB	vbx.PlayAtMs (lMsStart As Long) As Long
ActiveX C++	long pcx->PlayAtMs (long lMsStart)
Java	long mci.PlayAtMs (long lMsStart)
Dll	long vvwPlayAtMs (long lMsStart)
Unix	long PlayAtMs (long lMsStart)
XML	*Not supported*

Play starting at a particular millisecond time. Use the CurState to get the last frame aligned millisecond time from the controlled channel. Add a the millisecond equivalent of a number of frames and playback will commence at that time. Be sure and leave enough time for the command to be received and processed.

FastFoward

ActiveX VB	vbx.FastForward () As Long
ActiveX C++	long pcx-> FastForward ()
Java	long mci. FastForward ()
Dll	long vvwFastForward (long lChannel)
Unix	long FastForward ()
XML	http://localhost/VVWXMLMediaCmd?Play &speed=655200

Set the channel into its fastest possible forward motion state.
Returns 0 if successful, else an error code.

FastRewind

ActiveX VB	vbx.FastRewind () As Long
ActiveX C++	long pcx-> FastRewind ()
Java	long mci. FastRewind ()
Dll	long vvwFastRewind (long lChannel)
Unix	long FastRewind ()
XML	http://localhost/VVWXMLMediaCmd?Play &speed=-655200

Set the channel into its fastest possible reverse motion state.
Returns 0 if successful, else an error code.

Pause

ActiveX VB	vbx.Pause () As Long
ActiveX C++	long pcx-> Pause ()
Java	long mci. Pause ()
Dll	long vvwPause (long lChannel)
Unix	long Pause ()
XML	http://localhost/VVWXMLMediaCmd?Pause

Stop playback and display the current frame.

Returns 0 if successful, else an error code.

Seek

ActiveX VB	vbx.Seek (lFrame As Long) As Long
ActiveX C++	long pcx-> Seek (long lFrame)
Java	long mci. Seek (long lFrame)
Dll	long vvwSeek (long lChannel, long lFrame)
Unix	long Seek (long lFrame)
XML	http://localhost/VVWXMLMediaCmd?Pause &position=lFrame

Seek to a particular frame and display it to the user. This call will return before the seek is complete. Once the Position return in the status reaches the lFrame, the seek is complete.

Returns 0 if successful, else an error code.

SeekRelative

ActiveX VB	vbx.SeekRelative (lFrameOffset As Long) As Long
ActiveX C++	long pcx-> SeekRelative (long lFrameOffset)
Java	long mci. SeekRelative (long lFrameOffset)
Dll	long vvwSeekRelative (long lChannel, long lFrameOffset)
Unix	long SeekRelative (long lFrameOffset)
XML	http://localhost/VVWXMLMediaCmd?Pause &position=lFrameOffset

Seek a certain number of frames from the current position. Positive offsets imply forward direction, negative offset imply reverse.

Stop

ActiveX VB	vbx.Stop () As Long
ActiveX C++	long pcx-> Stop ()
Java	long mci. Stop ()
Dll	long vvwStop (long lChannel)
Unix	long Stop ()
XML	http://localhost/VVWXMLMediaCmd?Stop

Stop the output of the controlled channel and display the input video (not supported on all devices). On unsupported devices stop will be the same as a pause.

Returns 0 if successful, else an error code.

Record

ActiveX VB	vbx.Record () As Long
ActiveX C++	long pcx-> Record ()
Java	long mci. Record ()
Dll	long vvwRecord (long lChannel)
Unix	long Record ()
XML	http://localhost/VVWXMLMediaCmd?Record

Start the channel recording. In clip mode a default clip name will be used with a duration set to infinity. The record will stop on any transport command or at the point that the disk is full.

Returns 0 if successful, else an error code.

RecordAtMs

ActiveX VB	vbx.RecordAtMs (lMsStart As Long) As Long
ActiveX C++	long pcx->RecordAtMs (long lMsStart)
Java	long mci.RecordAtMs (long lMsStart)
Dll	long vvwRecordAtMs (long lMsStart)
Unix	long RecordAtMs (long lMsStart)
XML	*Not supported*

Record starting at a particular millisecond time. Use the CurState to get the last frame aligned millisecond time from the controlled channel. Add a the millisecond equivalent of a number of frames and the record will commence at that time. Be sure and leave enough time for the command to be received and processed.

RecordFromTo

ActiveX VB	vbx.RecordFromTo (lFrom As Long, lTo As Long) As Long
ActiveX C++	long pcx-> RecordFromTo (long lFrom, long lTo)
Java	long mci. RecordFromTo (long lFrom, long lTo)
Dll	long vwwRecordFromTo (long lChannel, long lFrom, long lTo)
Unix	long RecordFromTo (long lFrom, long lTo)
XML	http://localhost/VVWXMLMediaCmd?Record &start=lStart&end=lEnd

Record from a frame value to a frame value. As with editing systems, the 'from' point is included and will be recorded but the to point is NOT included and will not be recorded. This means that the last frame recorded will be lFrom – 1.

Returns 0 if successful, else an error code.

RecordStop (prepare record)

ActiveX VB	Vbx.RecordStop (szClipName As String, lDuration As Long) As Long
ActiveX C++	Long pcx-> RecordStop (BSTR szClipName, long lDuration)
Java	Long mci. RecordStop (String szClipName, long lDuration)
Dll	Long vvwRecordStop (long lChannel, char * szClipName, long lDuration)
Unix	Long RecordStop (char * szClipName, long lDuration)
XML	http://localhost/VVWXMLMediaCmd?RecStop

Clip Mode Only. Set the clip name and length of time to record in frames. The record will not actually start until Record() is called. If the lDuration is set to –1 the record will continue until Stop() is called or the channel runs out of space.

Returns 0 if successful, else an error code.

SetRecordPresets

ActiveX VB	vbx.SetRecordPresets (lVidEdit As Long, lAudEdit As Long lInfEdit As Long) As Long
ActiveX C++	long pcx-> SetRecordPresets (long lVidEdit, long lAudEdit, long lInfEdit)
Java	long mci. SetRecordPresets (long lVidEdit, long lAudEdit, long lInfEdit)
Dll	long vvwSetRecordPresets (long lVidEdit, long lAudEdit, long lInfEdit)
Unix	long SetRecordPresets (long lVidEdit, long lAudEdit, long lInfEdit)
XML	http://localhost/VVWXMLMediaCmd?Record &videochannels=lVidEdit&audiochannels=lAudEdit &infochannels=lInfEdit

Set the channels to record. Using –1 values implies that the Preset should be set to all available channels. Record Presets will remain set until the user changes them.

Returns 0 if successful, else an error code.

Eject

ActiveX VB	vbx.Eject () As Long
ActiveX C++	long pcx-> Eject ()
Java	long mci. Eject ()
Dll	long vvwEject (long lChannel)
Unix	long Eject ()
XML	http://localhost/VVWXMLMediaCmd?Eject

Eject the current media if it is removable. Normally only used with VTRs.

Returns 0 if successful, else an error code.

Special Commands

Please note: Not all the following commands are supported on all channels. Special restrictions may apply

Transfer

ActiveX VB	vbx.Transfer (lTargetChannel As Long, lPosition As Long, lStart As Long, lEnd As Long, lVidEdit As Long, lAudEdit As Long, lInfEdit As Long, szClipName As String, fToTape As Boolean) As Long
ActiveX C++	long pcx-> Transfer (long lTargetChannel, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOL fToTape)
Java	long mci. Transfer (long lTargetChannel, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, Boolean fToTape)
Dll	long vvwTransfer (long lChannel, long lTargetChannel, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOL fToTape)
Unix	long Transfer (long lTargetChannel, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, VVWBOOL fToTape)
XML	http://localhost/VVWXMLMediaCmd?Transfer &channel=lTargetChannel&position=lPosition &start=lStart&end=lEnd&videochannels=lVidEdit &audiochannels=lAudEdit&infochannels=lInfEdit &Flags=Invert

Transfer media from one channel to another. Only supported by VTR channels. Currently only implemented for VTR to internal channels or internal channels to VTR channels. To record media from a VTR, the fToTape should be false, to record media onto a VTR the fToTape should be true. The start and end point are from the playback device. The edit will occur at the current timecode location on the recorder.

Returns 0 if successful, else an error code.

Status Operation

UpdateStatus

ActiveX VB	vbx.UpdateStatus () As Long
ActiveX C++	long pcx-> UpdateStatus ()
Java	long mci. UpdateStatus ()
Dll	long vvwUpdateStatus (long lChannel)
Unix	long UpdateStatus ()
XML	http://localhost/VVWXMLGetStatus?

Retrieve the current status from the controlled device. The status is automatically updated by the interface, but this call ensures that the status is current when you are checking it.

Returns 0 if successful, else an error code.

VVWXMLGetStatus returns XML with a MediaCmd root element, for example:

```
  <?xml version="1.0" ?>
- <MediaCmd>
- <!--  Drastic MEDIACMD xml structure version 1,0
  -->
  <CmdID Value="-98238205" />
  <StructSize Value="336" />
  <Channel Value="-1" />
  <Cmd Value="1" UseClipID="1">Pause</Cmd>
  <Speed Value="0">0</Speed>
  <CmdAlt Value="2083947" TimeMs="1" />
  <Position Value="102" TcType="non-drop-frame"
UsingFrameCount="1">00:00:03:12</Position>
  <Start Value="0" TcType="non-drop-frame"
UsingFrameCount="1">00:00:00:00</Start>
  <End Value="2592000" TcType="non-drop-frame"
UsingFrameCount="1">24:00:00:00</End>
  <ClipID>::VTR_TC</ClipID>
  </MediaCmd>
```

GetState

ActiveX VB	vbx.GetState () As Long
ActiveX C++	long pcx-> GetState ()
Java	long mci. GetState ()
Dll	long vvwGetState (long lChannel)
Unix	long GetState ()
XML	http://localhost/VVWXMLGetStatus?

Returns the current state

- ctStop 0 // Stop all action
- ctPause 1 // Pause, Seek
- ctPlay 2 // Play at specified speed (includes pause)
- ctRecord 3 // Record at specified speed
- ctRecStop 4 // Stop ready for recording
- ctEject 5 // Eject the current media
- ctError 17 // An error has occurred
- ctAbort 19 // Abort any queued commands

XML: See <MediaCmd> root element, <Cmd> sub-element (value)

GetFlags

ActiveX VB	vbx. GetFlags () As Long
ActiveX C++	long pcx-> GetFlags ()
Java	long mci. GetFlags ()
Dll	long vvwGetFlags (long lChannel)
Unix	long GetFlags ()
XML	Not required

Returns the current flags

- cfDeferred = 1, // 0x00000001 This is a delayed
- cfOverideDeferred = 1 << 30, // 0x40000000 Overide all previous deferred commands
- cfTimeMs = 1 << 1, // 0x00000002 Use Millisecond time for delayed time, not fields
- cfTimeTarget = 1 << 2, // 0x00000004 Delayed time is offset from current time code
- cfTimeHouseClock = 1 << 3, // 0x00000008 Delayed time is based on absolute (real) time
- cfUseSpeed = 1 << 4, // 0x00000010 Set the new speed
- cfUsePresets = 1 << 5, // 0x00000020 Use video and audio edit presets
- cfUsePosition = 1 << 6, // 0x00000040 Use the position setting
- cfUsePositionOffset = 1 << 7, // 0x00000080 Position is an offset
- cfUseStart = 1 << 8, // 0x00000100 Start a new timecode
- cfUseStartOffset = 1 << 9, // 0x00000200 Start is an offset from current tc
- cfUseEnd = 1 << 10, // 0x00000400 End command as specified
- cfUseEndOffset = 1 << 11, // 0x00000800 End is and offset from current tc
- cfUseAllIDs = 1 << 12, // 0x00001000 Use all clip IDs

- cfUseClipID = 1 << 13, // 0x00002000 Use new clip ID, otherwise use last or none
- cfNoClipFiles = 1 << 14, // 0x00004000 Use new clip ID, otherwise use last or none
- cfNoTCSpaces = 1 << 15, // 0x00008000 Use new clip ID, otherwise use last or none
- cfUseCmdAlt = 1 << 16, // 0x00010000 Use the dwCmdAlt
- cfIsShuttle = 1 << 17, // 0x00020000 Use speed in play for shuttle
- cfFields = 1 << 20, // 0x00100000 Position, start and end are fields, not frames
- cfRipple = 1 << 21, // 0x00200000 Ripple for insert or delete
- cfLoop = 1 << 22, // 0x00400000 Loop the clip or in out
- cfTrigger = 1 << 23, // 0x00800000 Trigger using dsync class
- cfPreview = 1 << 24, // 0x01000000 Preview set (EE, non rt play)
- cfInvert = 1 << 28, // 0x10000000 Invert a transfer
- cfTest = 1 << 29, // 0x20000000 See if the command exists
- cfNoReturn = 1 << 31, // 0x80000000 No return mediacmd is required

GetSpeed

ActiveX VB	vbx. GetSpeed () As Long
ActiveX C++	long pcx-> GetSpeed ()
Java	long mci. GetSpeed ()
Dll	long vvwGetSpeed (long lChannel)
Unix	long GetSpeed ()
XML	http://localhost/VVWXMLGetStatus?

Returns the current VVW speed if the cfUseSpeed flag is set, otherwise pause or full play speed. VVW speeds are based on 65520 as the play speed. To translate to decimal number where 1.0 represents play, use the following formula:

D1Speed = ((double)VVWSpeed / 65520.0)

For percentages, where 100.0 represents play speed, use the following formula:

Dpercent = (((double)VVWSpeed * 100.0) / 65520.0)
= ((double)VVWSpeed / 655.2)

XML: See <MediaCmd> root element, <Speed> sub-element

Typical VVW speeds (note speeds are linear):

Pause	0%	0
Play	100%	65520
Half Play	50%	32760
Reverse Play	-100%	-65520
Reverse Double Play	-200%	131040
10 Time Forward Play	1000%	655200
Max Forward Play	90000%	5896800
Max Reverse Play	-90000%	-5896800

GetPosition

ActiveX VB	vbx. GetPosition () As Long
ActiveX C++	long pcx-> GetPosition ()
Java	long mci. GetPosition ()
Dll	long vvwGetPosition (long lChannel)
Unix	long GetPosition ()
XML	http://localhost/VVWXMLGetStatus?

Returns the current position if the cfUsePosition flag is set, otherwise invalid.
XML: See <MediaCmd> root element, <Position> sub-element (value)

GetLastMs

ActiveX VB	vbx. GetLastMs () As Long
ActiveX C++	long pcx-> GetLastMs ()
Java	long mci. GetLastMs ()
Dll	long vvwGetLastMs (long lChannel)
Unix	long GetLastMs ()
XML	http://localhost/VVWXMLGetStatus?

Returns the millisecond time the last status occurred (time of the last vertical blank).
XML: See <MediaCmd> root element, <CmdAlt> sub-element

GetStart

ActiveX VB	vbx. GetStart () As Long
ActiveX C++	long pcx-> GetStart ()
Java	long mci. GetStart ()
Dll	long vvwGetStart (long lChannel)
Unix	long GetStart ()
XML	http://localhost/VVWXMLGetStatus?

Returns the current start or in point if the cfUseStart flag is set.
XML: See <MediaCmd> root element, <Start> sub-element

GetEnd

ActiveX VB	vbx. GetEnd () As Long
ActiveX C++	long pcx-> GetEnd ()
Java	long mci. GetEnd ()
Dll	long vvwGetEnd (long lChannel)
Unix	long GetEnd ()
XML	http://localhost/VVWXMLGetStatus?

Return the current end point or out point if cfUseEnd is set.
XML: See <MediaCmd> root element, <End> sub-element

GetClipName

ActiveX VB	vbx. GetClipName () As String
ActiveX C++	BSTR pcx-> GetClipName ()
Java	String mci. GetClipName ()
Dll	long vvwGetClipName (long lChannel, char * sz8CharClipName)
Unix	Char * GetClipName ()
XML	http://localhost/VVWXMLGetStatus?

Only supported in clip Mode. Returns the current clip name, if any. For dll access, the memory must be at least 9 bytes long (8 character bytes + NULL) and is always ANSI.
XML: See <MediaCmd> root element, <CmdID> sub-element

GetFileName

ActiveX VB	Vbx. GetFileName () As String
ActiveX C++	BSTR pcx-> GetFileName ()
Java	String mci. GetFileName ()
Dll	Long GetFileName (long lChannel, char * sz260CharFileName)
Unix	Char * GetFileName ()
XML	Not supported

Returns the current file name, if any. For dll access, the memory must be at lease 261 bytes long (260 bytes max path + NULL) and is always ANSI.

GetCurTC

ActiveX VB	Vbx. GetCurTC () As String
ActiveX C++	BSTR pcx-> GetCurTC ()
Java	String mci. GetCurTC ()
Dll	Long GetCurTC (long lChannel, char * sz14ByteTC)
Unix	Char * GetCurTC ()
XML	http://localhost/VVWXMLGetStatus?

Returns the current time code as a string (e.g. "00:01:00:00"). For dll access, the memory must always be at least 15 bytes long (14 byte time code plus id + NULL) and is always ANSI.
XML: See <MediaCmd> root element, <Positon> sub-element (text)

GetCurState

ActiveX VB	Vbx. GetCurState () As String
ActiveX C++	BSTR pcx-> GetCurState ()
Java	String mci. GetCurState ()
Dll	Long GetCurState (long lChannel, char * sz14ByteState)
Unix	Char * GetCurState ()
XML	http://localhost/VVWXMLGetStatus?

Returns the current state as a string (e.g. "Play"). For dll access, the memory must always be at least 15 bytes long (14 byte state + NULL) and is always ANSI.
XML: See <MediaCmd> root element, <Cmd> sub-element (text)

Media Operations

Clip Mode Operations

GetNextClip

ActiveX VB	Vbx. GetNextClip (szLastClip As String) As String
ActiveX C++	BSTR pcx-> GetNextClip (BSTR szLastClip)
Java	String mci. GetNextClip (String szLastClip)
Dll	Char * vvwGetNextClip (long lChannel, char * sz8CharLastClipCurClip)
Unix	Char * GetNextClip (char * sz8CharLastClipCurClip)
XML	http://localhost/VVWXMLNextClip?

Clip Mode Only. Returns the next clip identifier. To get the first clip, szLastClip should be an empty string. Once the last clip available has been returned, GetNextClip will return an error or NULL for unix/dll access. Please note: For unix/dll access, the sz8CharLastClipCurClip memory area is used for the new clip. The previous clip name is therefore lost and the memory is not allocated by the vvw.

Returns 0 if successful, else an error code.

VVWXMLNextClip returns XML with a ClipInfo root element, for example:

```
  <?xml version="1.0" ?>
- <ClipInfo>
- <!--  Drastic ClipInfo xml structure version 1,0
  -->
  <ClipID>::Test</ClipID>
  <FileName>::Test</FileName>
  <Start Value="0" TcType="non-drop-frame">00:00:00:00</Start>
  <End Value="0" TcType="non-drop-frame">02:00:00:00</End>
  </ClipInfo>
```

GetClipInfo

ActiveX VB	Vbx. GetClipInfo (szClipName As String, ByRef lStart As Long, ByRef lEnd As Long, ByRef lVidEdit As Long, ByRef lAudEdit As Long, ByRef lInfEdit As Long, szFileName As String) As Long
ActiveX C++	Long pcx-> GetClipInfo (BSTR szLastClip, long * lStart, long * lEnd, long * lVidEdit, long * lAudEdit, long * lInfEdit, BSTR * szFileName)
Java	Long mci. GetClipInfo (mci.ClipInfo clipData)
Dll	Long vvwGetClipInfo (long lChannel, char * sz8CharClipName, long * lStart, long * lEnd, long * lVidEdit, long * lAudEdit, long * lInfEdit, char * sz260CharFileName)
Unix	Long GetClipInfo (char * sz8CharClipName, long * lStart, long * lEnd, long * lVidEdit, long * lAudEdit, long * lInfEdit, char * sz260CharFileName)
XML	http://localhost/VVWXMLNextClip?

Retuns the basic information from szClip. The information is located in lStart, lEnd, lVidEdit, lAudEdit and szFileName as the in point, out point, number of video channels, number of audio channels, and the file name respectively.

Returns 0 if successful, else an error code.

Java: This method requires the user to instantiate a new object of type ClipInfo. The sz8CharClipName, lStart, lEnd, lVidEdit, lAudEdit, lInfEdit, and sz260CharFileName values are returned in the oject's instance variables.

XML: returns <ClipInfo> root element, <ClipID>, <FileName>, <Start>, <End> sub elements

GetClipInfoEx

ActiveX VB	vbx. GetClipInfoEx (szClipName As String, ByRef lCreation As Long, ByRef lLastModification As Long, ByReg lFileSize As Long, ByRef lDiskFragments) As Long
ActiveX C++	Long pcx-> GetClipInfoEx (BSTR szClipName, long * lCreation, long * lLastModification, long * lFileSize, long * lDiskFragments)
Java	*Not Implemented*
Dll	Long vvwGetNextClipEx (long lChannel, char * szClipName, long * lCreation, long * lLastModification, long * lFileSize, long * lDiskFragments)
Unix	Long GetNextClipEx (char * szClipName, long * lCreation, long * lLastModification, long * lFileSize, long * lDiskFragments)
XML	Not supported

Retuns the extended information from szClip. The information is located in lStart, lEnd, lVidEdit, lAudEdit and szFileName as time of creation, last modified date, the file size, and the number of fragments in the file respectively.

Returns 0 if successful, else an error code.

CopyClip

ActiveX VB	vbx. CopyClip (szSourceClip As String, szDestClip As String, lStart As Long, lEnd As Long) As long
ActiveX C++	Long pcx-> CopyClip (BSTR szSourceClip, BSTR szDestClip, long lStart, long lEnd)
Java	Long mci. CopyClip (String szSourceClip, String szDestClip, long lStart, long lEnd)
Dll	Long vvwCopyClip (long lChannel, char * szSourceClip, char * szDestClip, long lStart, long lEnd)
Unix	Long CopyClip (char * szSourceClip, char * szDestClip, long lStart, long lEnd)
XML	Not supported

Create a virtual copy of a clip, changing the in and out points if necessary. To use the whole clip, set lStart to 0 and the end to –1.

Returns 0 if successful, else an error code.

VTR Mode Operations

EDLResetToStart

ActiveX VB	Vbx. EDLResetToStart () As Long
ActiveX C++	Long pcx-> EDLResetToStart ()
Java	Long mci. EDLResetToStart ()
Dll	Long vvwEDLResetToStart (long lChannel)
Unix	Long EDLResetToStart ()
XML	Not supported

Reset the edl returns in VTR mode to the first element of the list.

EDLGetEdit

ActiveX VB	Vbx. EDLGetEdit (ByRef lRecordIn As Long, ByRef lPlayIn As Long, ByRef lPlayOut As Long, ByRef lVidEdit As Long, ByRef lAudEdit As Long, ByRef lInfEdit As Long, ByRef szClipName As String, ByRef szFileName As Long) As Long
ActiveX C++	Long pcx-> EDLGetEdit (long * lRecordIn, long * lPlayIn, long * lPlayOut, long * lVidEdit, long * lAudEdit, long * lInfEdit, BSTR * szClipName, BSTR * szFileName)
Java	long mci. EDLGetEdit (VTREditLine editInfo)
Dll	long vvwEDLGetEdit (long lChannel, long * lRecordIn, long * lPlayIn, long * lPlayOut, long * lVidEdit, long * lAudEdit, long * lInfEdit, char * sz8CharClipName, char * sz260CharFileName)
Unix	Long EDLGetEdit (long * lRecordIn, long * lPlayIn, long * lPlayOut, long * lVidEdit, long * lAudEdit, long * lInfEdit, char * szClipName, char * szFileName)
XML	http://localhost/VVWXMLEDLInfo?position=0 &videochannels=0&audiochannels=0&infochannels=0

Returns an edit line from the VTR space of an internal channel. The function will continue to return the next edit in the timecode space until the last edit is returned, after which an error will be returned. To reset to the start of the Edl use EDLResetToStart.

Returns 0 if successfule else an Error code.

Java: This method requires the user to instantiate a new object of type VTREditLine. The lRecordIn, lPlayIn, lPlayOut, lVidEdit, lAudEdit, lInfEdit, szClipName, and szFileName values are returned in the oject's instance variables of the same name.

VVWXMLEDLInfo returns XML with a <MediaCmd> root element, for example:

```
<?xml version="1.0" ?>
- <MediaCmd>
  - <!--  Drastic MEDIACMD xml structure version 1,0
  -->
  <CmdID Value="-98238205" />
  <StructSize Value="336" />
  <Channel Value="0" />
```

```
  <Cmd Value="14" UseClipID="1">GetValue</Cmd>
  <VideoChannels Value="1" />
  <AudioChannels Value="0" />
  <InfoChannels Value="0" />
  <CmdAlt Value="93" />
  <Position Value="5" TcType="non-drop-frame">00:00:00:05</Position>
  <Start Value="0" TcType="non-drop-frame">00:00:00:00</Start>
  <End Value="5" TcType="non-drop-frame">00:00:00:05</End>
  <FileName>V:\Drastic Base Media\avi_er001_720x486_YUY2.avi</FileName>
</MediaCmd>
```

Shared Operations

GetLastChangeMs

ActiveX VB	vbx. GetLastChangeMs () As Long
ActiveX C++	long pcx-> GetLastChangeMs ()
Java	long mci. GetLastChangeMs ()
Dll	long vvwGetLastChangeMs (long lChannel)
Unix	long GetLastChangeMs ()
XML	Not supported

Returns the millisecond time of the last change in the current mode (clip or vtr).

Insert

ActiveX VB	vbx. Insert (szClipName As String, szFileName As String, lPosition As Long, lStart As Long, lEnd As Long, lVidEdit As Long, lAudEdit As Long, lInfEdit As Long, fRipple As Boolean) As Long
ActiveX C++	long pcx-> Insert (BSTR szClipName, BSTR szFileName, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOl fRipple)
Java	long mci. Insert (String szClipName, String szFileName, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, Boolean fRipple)
Dll	long vvwInsert (long lChannel, char * szClipName, char * szFileName, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOl fRipple)
Unix	long Insert (char * szClipName, char * szFileName, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, VVWBOOl fRipple)
XML	http://localhost/VVWXMLMediaCmd?Insert &ClipID=szClipName&position=lPosition &start=lStart&end=lEnd&videochannels=lVidEdit &audiochannels=lAudEdit&infochannels=lInfEdit &Flags=Ripple

Internal Channels Only. Do not use yet.

Blank

ActiveX VB	vbx. Blank (szClipName As String, lStart As Long, lEnd As Long, lVidEdit As Long, lAudEdit As Long, lInfEdit As Long, fRipple As Boolean) As Long
ActiveX C++	long pcx-> Blank (BSTR szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOl fRipple)
Java	long mci. Blank (String szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, Boolean fRipple)
Dll	long vvwBlank (long lChannel, char * szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOl fRipple)
Unix	long Blank (char * szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, VVWBOOl fRipple)
XML	http://localhost/VVWXMLMediaCmd?Blank &ClipID=szClipName&position=lPosition &start=lStart&end=lEnd&videochannels=lVidEdit &audiochannels=lAudEdit&infochannels=lInfEdit &Flags=Ripple

Internal Channels Only. Do not use yet.

Delete

ActiveX VB	vbx. Delete (szClipName As String, lStart As Long, lEnd As Long, lVidEdit As Long, lAudEdit As Long, lInfEdit As Long, fRipple As Boolean) As Long
ActiveX C++	long pcx-> Delete (BSTR szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOl fRipple)
Java	long mci. Delete (String szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, Boolean fRipple)
Dll	long vvwDelete (long lChannel, char * szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOl fRipple)
Unix	long Delete (char * szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, VVWBOOl fRipple)
XML	http://localhost/VVWXMLMediaCmd?Delete &ClipID=szClipName&position=lPosition &start=lStart&end=lEnd&videochannels=lVidEdit &audiochannels=lAudEdit&infochannels=lInfEdit &Flags=Ripple

Internal Channels Only. Do not use yet.

Trim

ActiveX VB	vbx. Trim (lPosition As Long, lStartOffset As Long, lEndOffset As Long, lVidEdit As Long, lAudEdit As Long, lInfEdit As Long, fRipple As Boolean) As Long
ActiveX C++	long pcx-> Trim (long lPosition, long lStartOffset, long lEndOffset, long lVidEdit, long lAudEdit, long lInfEdit, BOOl fRipple)
Java	long mci. Trim (long lPosition, long lStartOffset, long lEndOffset, long lVidEdit, long lAudEdit, long lInfEdit, Boolean fRipple)
Dll	long vvwTrim (long lChannel, long lPosition, long lStartOffset, long lEndOffset, long lVidEdit, long lAudEdit, long lInfEdit, BOOl fRipple)
Unix	long Trim (long lPosition, long lStartOffset, long lEndOffset, long lVidEdit, long lAudEdit, long lInfEdit, VVWBOOl fRipple)
XML	http://localhost/VVWXMLMediaCmd?Trim &position=lPosition&start=lStartOffset &end=lEndOffset&videochannels=lVidEdit &audiochannels=lAudEdit&infochannels=lInfEdit &Flags=Ripple

Internal Channels Only. Do not use yet.

Settings

The 'Value' commands allow settings to be changed on a particular channel. The most common settings have been made into functions, but all settings use ValueSupported, ValueMin, ValueMax, ValueGet and ValueSet. Most applications will only require the functions below. If an extended settings is required, please see the MediaCmd reference.

ValueSupported

ActiveX VB	vbx. ValueSupported (lValueType As Long) As Long
ActiveX C++	long pcx-> ValueSupported (long lValueType)
Java	long mci. ValueSupported (long lValueType)
Dll	long vvwValueSupported (long lChannel, long lValueType)
Unix	long ValueSupported (long lValueType)
XML	http://localhost/ValueSupported&cmdalt=valuetype &position=lValueType

Returns the supported attributes of a get/set value (gsClipMode, gsTcSource, etc) or –1 for not supported.

ValueGet

ActiveX VB	vbx. ValueGet (lValueType As Long, ByRef lMin As Long, ByRef lMax As Long) As Long
ActiveX C++	long pcx-> ValueGet (long lValueType, long * plMin, long * plMax)
Java	long mci. ValueGet (GetValueMcmd mCmdValues)
Dll	long vvwValueGet (long lChannel, long lValueType, long * plMin, long * plMax)
Unix	long ValueGet (long lValueType, long * plMin, long * plMax)
XML	http://localhost/GetValue&cmdalt=valuetype &position=lValueType

Returns the current setting for a get/set value.

Java: This method requires the user to instantiate a new object of type GetValueMcmd. The plMin and plMax values are returned in the oject's instance variables: plMin and plMax.

ValueSet

ActiveX VB	vbx. ValueSet (lValueType As Long, lSetting As Long) As Long
ActiveX C++	long pcx-> ValueSet (long lValueType, long lSetting)
Java	long mci. ValueSet (long lValueType, long lSetting)
Dll	long vvwValueSet (long lChannel, long lValueType, long lSetting)
Unix	long ValueSet (long lValueType, long lSetting)
XML	http://localhost/SetValue&cmdalt=valuetype &position=lSetting

Sets the get/set value to setting.

ValueSet2

ActiveX VB	vbx. ValueSet2 (lValueType As Long, lSetting As Long, lStart As Long, lEnd As Long, lVidChannel As Long, lAudChannel As Long, lInfChannel As Long) As Long
ActiveX C++	long pcx-> ValueSet2 (long lValueType, long lSetting, long lStart, long lEnd, long lVidChannel, long lAudChannel, long lInfChannel)
Java	long mci. ValueSet2 (long lValueType, long lSetting, long lStart, long lEnd, long lVidChannel, long lAudChannel, long lInfChannel)
Dll	long vvwValueSet2 (long lChannel, long lValueType, long lSetting, long lStart, long lEnd, long lVidChannel, long lAudChannel, long lInfChannel)
Unix	long ValueSet2 (long lValueType, long lSetting, long lStart, long lEnd, long lVidChannel, long lAudChannel, long lInfChannel)
XML	Not supported – see ValueSet

Sets the get/set value to setting with extended parameters. Please set unused parameters to NULL.

Settings Format

All the settings, except where noted, have the following format:

ActiveX VB	Vbx. Get*XXX* () As Long
	Vbx. Set*XXX* (lSetting As Long) As Long
ActiveX C++	Long pcx-> Get*XXX* ()
	Long pcx-> Set*XXX* (long lSetting)
Java	Long mci. Get*XXX* ()
	Long mci. Set*XXX* (long lSetting)
Dll	Long vvwGet*XXX* ()
	Long vvwSet*XXX* (long lSetting)
Unix	Long Get*XXX* ()
	Long Set*XXX* (long lSetting)

Base Settings

Get/SetClipMode

Calls ValueXXX with gsClipMode. If equal to 1 then the channel is in clip mode, if 0 the channel is in VTR mode.
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=clipmode&position=0)

Get/SetTCType

Calls ValueXXX with gsTcType (drop frame, non drop frame, pal).

```
#define TC2_TCTYPE_MASK          0x000000FF
#define TC2_TCTYPE_FILM          0x00000001    // 24 fps
#define TC2_TCTYPE_NDF           0x00000002    // NTSC Non Drop Frame
#define TC2_TCTYPE_DF            0x00000004    // NTSC Drop Frame
#define TC2_TCTYPE_PAL           0x00000008    // PAL
#define TC2_TCTYPE_50            0x00000010    // PAL (double rate)
#define TC2_TCTYPE_5994          0x00000020    // NTSC 59.94fps 720p
#define TC2_TCTYPE_60            0x00000040    // NTSC 60fps 720p
#define TC2_TCTYPE_NTSCFILM      0x00000080    // NTSC FILM 23.97
```

(XML: localhost/XMLMediaCmd?SetValue&cmdalt=tctype&position=2)

Get/SetTCSource

Calls ValueXXX with gsTcSource (VITC, LTC, Control, Clip).

```
//! For cmdGetSetValue::gsTcSource - Using LTC
#define GS_TCSOURCE_LTC          1
//! For cmdGetSetValue::gsTcSource - Using VITC
#define GS_TCSOURCE_VITC         2
//! For cmdGetSetValue::gsTcSource - Using CTL
#define GS_TCSOURCE_CTL          4
//! For cmdGetSetValue::gsTcSource - Using absolute clip
#define GS_TCSOURCE_CLIP         7
```

(XML: localhost/XMLMediaCmd?GetValue&cmdalt=tcsource)

Get/SetAutoMode

Calls ValueXXX with gsAutoMode. Required for play lists, deferred commands and auto edit commands on VTRs.

GetAvailablePresets

ADD FUNCTIONS lVidEdit, lAudEdit, lInfEdit

Returns the supported audio, video and info presets for a channel.

Java: Values for audio, video and info presets are returned in variable members of the class mci.AvailablePresets.

Audio Settings

Get/SetAudioInput

ADD FUNCTION lAudIn

Get the current audio input.

Java: Must indicate the channel(s) to get/set. This is achieved by sending a long value representing the channel(s) you wish to get/set. For example if you wish to set audio channels 1 and 4 set the audChannels paramater to 9 (1001). You may also use the predefined audio channel defenitions in the MEDIACMD structure (audChanX, where x is the audio channel - 1).

//! Audio in/out unbalanced (RCA connector) high impedence at -10db (cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_UNBALANCED_10 0x001
//! Audio in/out unbalanced (RCA connector) high impedence at -4db (cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_UNBALANCED_4 0x002
//! Audio in/out balanced (XLR connector) 600ohm impedence at -10db (cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_BALANCED_10 0x010
//! Audio in/out balanced (XLR connector) 600ohm impedence at +4db (cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_BALANCED_4 0x020
//! Audio in/out digital single wire (cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_SPDIF 0x100
//! Audio in/out digital balanced with clock (cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_AES_EBU 0x200
//! Audio in/out embedded in SDI or HD-SDI video signal (cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_EMBEDDED 0x400
//! No audio in/out available, or cannot be configured (cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_NONE 0
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsAudInSelect&position=lSetting &videochannels=0&audiochannels=lAudChannels&infochannels=0)

Get/SetAudioInputLevel

Get the current audio input level

Java: Must indicate the channel(s) to get/set. This is achieved by sending a long value representing the channel(s) you wish to get/set. For example if you wish to set audio channels 1 and 4 set the audChannels paramater to 9 (1001). You may also use the predefined audio channel defenitions in the MEDIACMD structure (audChanX, where x is the audio channel - 1).

Note: There are two get methods, one method will only return the current level.

The second method will return the current level as well as the max and min values for the audio channel in the GetValueMcmd object.

(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsAudInputLevel&position=lSetting &videochannels=0&audiochannels=lAudChannels&infochannels=0)

Get/SetAudioOutput

Get the current audio Output – See Get/SetAudioInput

Java: Must indicate the channel(s) to get/set. This is achieved by sending a long value representing the channel(s) you wish to get/set. For example if you wish to set audio channels 1 and 4 set the audChannels paramater to 9 (1001). You may also use the predefined audio channel defenitions in the MEDIACMD structure (audChanX, where x is the audio channel - 1).

Note: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the audio channel in the GetValueMcmd object.

(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsAudOutSelect&position=lSetting &videochannels=0&audiochannels=lAudChannels&infochannels=0)

Get/SetAudioOutputLevel

Get the current audio output level.

Java: Must indicate the channel(s) to get/set. This is achieved by sending a long value representing the channel(s) you wish to get/set. For example if you wish to set audio channels 1 and 4 set the audChannels paramater to 9 (1001). You may also use the predefined audio channel defenitions in the MEDIACMD structure (audChanX, where x is the audio channel - 1).

Note: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the audio channel in the GetValueMcmd object.

(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsAudOutputLevel&position=lSetting &videochannels=0&audiochannels=lAudChannels&infochannels=0)

GetAudioPeakRMS

Returns the RMS and Peak audio levels of the input (stop/record) or output (play/pause).

(XML: localhost/XMLMediaCmd?GetValue&cmdalt=gsAudWavePeakRMS)

Video Settings

Get/SetVideoInput

Get the current video input.

```
//! Standard NTSC or PAL composite video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPOSITE                    0x001
//! SVHS/S-Video four wire NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_SVIDEO                       0x002
```

//! Secondary NTSC or PAL video (often monitor selection)
(cmdGetSetValue::gsVidInSelect cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPOSITE_2 0x004
//! BetaCam level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV 0x010
//! Panasonic M2 level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV_M2 0x020
//! SMPTE standard level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV_SMPTE 0x040
//! RGB at video standard rate (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_RGB 0x080
//! D1 Serial Digital or HDSDI video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_D1_SERIAL 0x100
//! D1 Serial Parallel video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_D1_PARALLEL 0x200
//! SDTI/SDI including high speed transfer video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_SDTI 0x400
//! No video available or no configurable settings (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_NONE 0
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidInSelect
&position=lSetting)

Get/SetVideoOutput

Get the current video output. See Get/SetVideoInput for settings.

Get/SetVideoInputSetup

Get the current video input's 'Setup' TBC setting.
Java: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the Setup in the GetValueMcmd object.
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidInSetup&position=lSetting)

Get/SetVideoInputVideo

Get the current video input's 'Video' TBC setting.
Java: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the Video in the GetValueMcmd object.

(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidInVideo
&position=lSetting)

Get/SetVideoInputHue

Get the current video input's 'Hue' TBC setting.
Java: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the Hue in the GetValueMcmd object.
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidInHue
&position=lSetting)

Get/SetVideoInputChroma

Get the current video input's 'Chroma' TBC setting.
Java: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the Chroma in the GetValueMcmd object.
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidInChroma
&position=lSetting)

Get/SetVideoTBCSetup

Get the current global TBC's 'Setup' setting.
Java: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the Setup in the GetValueMcmd object.
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidSetup&position=lSetting)

Get./SetVideoTBCVideo

Get the current global TBC's 'Video' setting.
Java: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the Videoin the GetValueMcmd object.
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidVideo&position=lSetting)

Get/SetVideoTBCHue

Get the current global TBC's 'Hue' setting.
Java: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the Hue in the GetValueMcmd object.
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidHue&position=lSetting)

Get/SetVideoTBCChroma

Get the current global TBC's 'Chroma' setting.
Java: There are two get methods, one method will only return the current level. The second method will return the current level as well as the max and min values for the Chroma in the GetValueMcmd object.
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidChroma &position=lSetting)

Get/SetVideoGenlock

Turn the house/reference lock on or off

Get/SetVideoGenlockSource – not implemented externally

Set the source to input or external genlock

Error Log

vvwGetErrorLogMs

Get the ms time the last error was added to the error log

vvwSetErrorLog

Set the error log pointer to the message you want

vvwGetErrorLength

Get the length of the current error string

vvwGetError

Get the current error. Sets pointer to the next one automatically

System Settings

Get/SetCompressionRate

Get or set the current compression rate.

Get/SetSuperImpose – not implemented externally

Enable or disable the time code super impose on the main output.

GetTotalTime

Returns the total number of frames of storage available at current compression rate if the storage space was empty.

GetFreeTime

Returns the remaining number of frames of storage available at current compression rate.

GetTotalStorage

Returns the total storage connected in megabytes.

GetFreeStorage

Returns the amount of available storage for recording in megabytes.

GetCurMs

Get the current millisecond time from the controlled channel.

GetChannelCapabilities

Get the available commands for a channel.

GetVVWVersion

Returns the version string of the VVW subsystem.

GetMRVersion

Returns the version string of the MediaReactor subsystem.

GetVVWType

Returns the type string of the VVW channel.

Utility Functions

FreeString

ActiveX VB	*Not implemented*
ActiveX C++	*Not implemented*
Java	*Not implemented*
Dll	Void vvwFreeString (char * szString)
Unix	Void FreeString (char * szString)
XML	*Not supported*

Free a string value returned by the channel.

TCMaxFrame

ActiveX VB	vbx. TCMaxFrame (lFlags As Long) As Long
ActiveX C++	long pcx-> TCMaxFrame (long lFlags)
Java	long mci. TCMaxFrame (long lFlags)
Dll	*Use TCXlat*
Unix	Long TCMaxFrame (long lFlags)
XML	*Not supported*

Returns the maximum possible frame value for a time code type. See TCToFrame for flag definitions.

TCToFrame

ActiveX VB	vbx. TCToFrame (szTC As String, lFlags As Long) As Long
ActiveX C++	Long pcx-> TCToFrame (BSTR szTC, long lFlags)
Java	Long mci. TCToFrame (String szTC, long lFlags)
Dll	*Use TCXlat*
Unix	Long TCToFrame (char * szTC, long lFlags)
XML	*Not supported*

Convert a time code string to a frame count based on the flags.

Flags:

```
TC2_TCTYPE_MASK          0x000000FF
TC2_TCTYPE_FILM          0x00000001    // 24 fps
TC2_TCTYPE_NDF           0x00000002    // NTSC Non Drop Frame
TC2_TCTYPE_DF            0x00000004    // NTSC Drop Frame
TC2_TCTYPE_PAL           0x00000008    // PAL
TC2_TCTYPE_50            0x00000010    // PAL 720p (double rate)
TC2_TCTYPE_5994          0x00000020    // NTSC 59.94fps 720p
TC2_TCTYPE_60            0x00000040    // NTSC 60fps 720p
TC2_TCTYPE_NTSCFILM      0x00000080    // NTSC FILM 23.97

TC2_FTYPE_FIELD          0x10000000    // Field based (else frame)

// Basic sting tc representation types
TC2_STRTYPE_MASK         0x00000F00
TC2_STRTYPE_ASCII        0x00000100    // Std ascii string
TC2_STRTYPE_BCD          0x00000200    // RS-422 BCD or Packed
TC2_STRTYPE_HEX          0x00000400    // Hex packed DWORD
TC2_STRTYPE_GOP          0x00000800    // MPEG Gop TC
TC2_STRTYPE_INVERT       0x00001000    // Frames first

// Extended string handling
TC2_STREXT_MARKS         0x00010000    // Add : marks in string
TC2_STREXT_LEADING       0x00020000    // Include leading 0s
TC2_STREXT_TYPE          0x00040000    // Add ' N', ' D', ' P' or ' F' at end
TC2_STREXT_ALLCOLON      0x00080000    // No -.; just :
TC2_STREXT_FLAG          0x00100000    // Add DF Flag in BCD
TC2_STREXT_CF            0x00200000    // Add CF Flag in BCD
TC2_STREXT_MAX30         0x00400000    // Max 29 frames for output
TC2_STREXT_SHIFT7        0x40000000    // GOP Tc is shifted in DWORD
TC2_STREXT_SAVEBITS      0x80000000    // GOP Save unused bits
```

TCToString

ActiveX VB	vbx. TCToString (lTC As Long, lFlags As Long) As String
ActiveX C++	BSTR pcx-> TCToString (long lTC, long lFlags)
Java	String mci. TCToString (long lTC, long lFlags)
Dll	*Use TCXlat*
Unix	Char * TCToString (long lTC, long lFlags)
XML	*Not supported*

Convert a frame count to a time code string based on the flags. See TCToFrame for flag definitions.

VVWSpeedToPercentage

ActiveX VB	vbx. VVWSpeedToPercentage (lVVWSpeed As Long) As Double
ActiveX C++	Double pcx-> VVWSpeedToPercentage (long lVVWSpeed)
Java	Double mci. VVWSpeedToPercentage (long lVVWSpeed)
Dll	*Not implemented*
Unix	Double VVWSpeedToPercentage (long lVVWSpeed)
XML	*Not supported*

Convert a VVW speed (65520 based) to a percentage based speed (100.0).

PercentageToVVWSpeed

ActiveX VB	vbx. PercentageToVVWSpeed (double ddPercentageSpeed) As Long
ActiveX C++	Long pcx-> PercentageToVVWSpeed (double ddPercentageSpeed)
Java	Long mci. PercentageToVVWSpeed (double ddPercentageSpeed)
Dll	*Not implemented*
Unix	Long PercentageToVVWSpeed (double ddPercentageSpeed)
XML	*Not supported*

Convert a percentage speed (100.0) to a VVW speed (65520)

Java Data Structure Definitions

ClipInfo

Java	MediaCmdIF.ClipInfo clipData = new MediaCmdIF.ClipInfo(szClipName)

For use with the mci.GetClipInfo method. Must construct this data structure with a valid clip name.

Data members

String szClipName: clip name. This must be set to a valid clip name when a call is made to the mci.GetClipInfo method.
long lStart: the in point of the inquired clip.
long lEnd: the out point of the inquired clip.
long lVidEdit: number of video channels available for the requested clip.
long lAudEdit: number of audiochannels available for the inquired clip.
long lInfEdit: number of info channels available for the inquired clip.
String szFileName: file name of the requested clip.

VTREditLine

Java	MediaCmdIF.VTREditLine editInfo = new MediaCmdIF.VTREditLine();

For use with the mci.EDLGetEdit method. Values for the EDLGetEdit method are returned in the object's instance variables. The mci.EDLGetEdit method returns 0 for a successful retrieval and –1 on error. The VTREditLine instance variable will be set to null on error and will contain valid data on success.

Data members

long lRecordIn: time code on the VTR edit line.
long lPlayIn: the in point of the VTR edit line.
long lPlayOut: the out point of the VTR edit line.
long lVidEdit: number of video channels available for the VTR edit line.
long lAudEdit: number of audio channels available for the VTR edit line.
long lInfEdit: number of info channels available for the VTR edit line.
String szClipName: clip name of the current VTR edit line .
String szFileName: file name of the current VTR edit line.

GetValueMcmd

Java	MediaCmdIF.GetValueMcmd mCmdValues = new MediaCmdIF.GetValueMcmd(setValueType, pMin, pMax)

For use with the mci.ValueGet method. Must construct this data structure with a valid lValueType. If the min value of the get/set value is needed construct the GetValueMcmd instance with pMin different than –1. If the max value of the get/set value is needed construct the GetValueMcmd instance with pMax different than –1. Instantiating the GetValueMcmd with values of –1 for the pMin and/or pMax parameters will result in an invalid return for these instance variables.

Data members

long lValueType: the get/set value. Used by the mci.ValueGet method to determine which setting to retrieve.

long plMin: stores the min value for a get/set value if instantiated with a value different than –1.

long plMax: stores the max value for a get/set value if instantiated with a value different than –1.

AvailablePresets

Java	MediaCmdIF.AvailablePresets presets = new MediaCmdIF.AvaiablePresets()

For use with the mci.GetAvaolablePresets method. Values for the GetAvailablePresets method are returned in the object's instance variables.

Data members

long plVidEdit: supported video presets for a channel

long plAudEdit: supported audio presets for a channel

long plInfEdit: supported info presets for a channel

Appendix I – MediaCmd.h

```
/*****************************************************************
 * MediaCmd.h
 *
 * $HeadURL$:
 * $Author$:
 * $Revision$:
 * $Date$:
 *
 * Copyright (c) 1998-2010 Drastic Technologies Ltd.  All Rights Reserved.
 * 523 The Queensway, Suite 102 Toronto ON M8Y 1J7
 * 416 255 5636  fax 255 8780
 * engineering@drastictech.com      http://www.drastictech.com
 *****************************************************************/

//
// Common header describing the command interface between media control modules.
//

#ifndef _MEDIACMD_INCLUDED_H
#define _MEDIACMD_INCLUDED_H

#ifdef _WIN32
// Windows x64
#ifndef _QTCREATOR
#pragma warning(disable: 4996)          // deprecated security functions
#endif
#ifndef _CRT_SECURE_NO_WARNINGS
        #define _CRT_SECURE_NO_WARNINGS
#endif
#ifndef _QTCREATOR
#pragma warning( disable:4996 )
#endif
#endif

/* This is the 32 bit pointer version that compiles differently in 64 bit mode
 * last version was:  0x01010003
 */
// This allows a quick check for the head of the command
//! Major command versioning for upgrades to the command set. See
MEDIACMD::dwCmdID
#define MEDIACMD_VERSION_MAJOR_X32      0x0101UL
//! Minor command versioning for upgrades to the command set. See
MEDIACMD::dwCmdID
```

```
#define MEDIACMD_VERSION_MINOR_X32      0x0003UL
/* This is the 32/64 matched compatible version
 * current is:  0x02000000
 */
//! Major command versioning for upgrades to the command set. See
MEDIACMD::dwCmdID
#define MEDIACMD_VERSION_MAJOR  0x0200UL
//! Minor command versioning for upgrades to the command set. See
MEDIACMD::dwCmdID
#define MEDIACMD_VERSION_MINOR  0x0000UL
//! Mask for checking the command set version. See MEDIACMD::dwCmdID
#define MEDIACMD_VERSION_MASK    0xFFFFUL
//! Permanent magic number of command id. See MEDIACMD::dwCmdID
#define MEDIACMD_CHECK_VER                  0xFA250000UL
//! Mask for permanent magic number of command id. See MEDIACMD::dwCmdID
#define MEDIACMD_CHECK_MASK                 0xFFFF0000UL
//! Current version and magic number.  Place in MEDIACMD::dwCmdID
#define MEDIACMD_CURRENT            (MEDIACMD_VERSION_MAJOR |
MEDIACMD_VERSION_MINOR | MEDIACMD_CHECK_VER)

// Various speed limits and definitions
//! Forward play speed (normal) in VVW (65520) see MEDIACMD::lSpeed
#define SPD_FWD_PLAY                        65520L
//! Pause speed (0%) in VVW (0) see MEDIACMD::lSpeed
#define SPD_PAUSE                           0L
//! Reverse play speed (-100%) in VVW (-65520) see MEDIACMD::lSpeed
#define SPD_REV_PLAY                        (-SPD_FWD_PLAY)
//! Maximum possible play speed in VVW. See MEDIACMD::lSpeed
#define SPD_FWD_MAX                                 5896800
//! Minimum possible play speed in VVW. See MEDIACMD::lSpeed
#define SPD_REV_MAX                         (-SPD_FWD_MAX)
//! Max speed for bumping
#define SPD_FAST_BUMP                       114660L //1.75 times play speed
//!Min Sped for bumping
#defineSPD_SLOW_BUMP                        32760l  //.5 times play speed
//! Illegal speed, set MEDIACMD::lSpeed to this value if not used
#define SPD_ILLEGAL                         2147483647L
//! Illegal time code reference, set MEDIACMD::dwPosition, MEDIACMD::dwStart,
MEDIACMD::dwEnd to this if not used
#define TC_ILLEGAL                          0xFFFFFFFF
//! Illegal channel, or All Channels.  Set MEDIACMD::dwAudioChannels,
MEDIACMD::dwVideoChannels, MEDIACMD::dwInfoChannels to this if not used
#define CHAN_ILLEGAL                0xFFFFFFFF
/**
* Maximum clip ID length in the DEFAULT #MEDIACMD structure.  Larger versions may
be allocated and are legal.  Smaller versions should not have less than 10 unsigned chars
with zero padding for clip names (8 unsigned chars + NULL) and file name (NULL).
* Note that the array is actually allocated as CMD_MAX_CLIP_ID_LEN+2+2
* 1024+8+2+2 = 1036 unsigned chars (DWORD aligned)
```

```
* CLIP 8 unsigned chars
* NULL 1 unsigned char
* File 260 unsigned chars (_MAX_PATH - WINDOWS)
* File 1024 unsigned chars (_MAX_PATH - UNIX)
* NULL 1 unsigned char
* 2 Alignment padding
*/
#define CMD_MAX_CLIP_ID_LEN                    (1024+8+4)

// The possible commands and states of media
/**
* The legal commands for a #MEDIACMD structure.  Set MEDIACMD::ctCmd to one of these values and expect it to be set to one of these values on a valid return
*/
enum cmdType {
        /**
        * Stop - Stop all playback, and normally place all channels into passthrough
        * <HR>
        */
        ctStop,                                // Stop all action
        /**
        * Pause - Halt all channels.  Display current video frame and silent audio<BR>
        * Seek - With cmdFlags::cfUsePosition and MEDIACMD::dwPosition, go to that frame and Pause
        * <HR>
        */
        ctPause,                               // Pause, Seek
        /**
        * Play - Play all channels.  May be modified by (cmdFlags::cfUseSpeed + MEDIACMD::lSpeed) and <BR> ((cmdFlags::cfUsePosition or cmdFlags::cfUsePositionOffset) and MEDIACMD::dwPosition) or <BR> ((cmdFlags::cfUseStart or cmdFlags::cfUseStartOffset) and MEDIACMD::dwStart) or <BR> ((cmdFlags::cfUseEnd or cmdFlags::cfUseEndOffset) and MEDIACMD::dwEnd) as well as <BR> MEDIACMD::dwCmdAlt with certain #cmdFlags to play from-top, at speed or combinations of the above.
        * <HR>
        */
        ctPlay,                                // Play at specified speed (includes pause)
        /**
        * Record - Record one or a combination of video/audio/info to disk.  May be modified, as with #ctPlay by flags and structure members such as <BR> ((cmdFlags::cfUsePosition or cmdFlags::cfUsePositionOffset) and MEDIACMD::dwPosition) or <BR> ((cmdFlags::cfUseStart or cmdFlags::cfUseStartOffset) and MEDIACMD::dwStart) or <BR> ((cmdFlags::cfUseEnd or cmdFlags::cfUseEndOffset) and MEDIACMD::dwEnd) as well as <BR> (cmdFlags::cfUseClipID and MEDIACMD::arbID) or <BR> (cmdFlags::cfDeferred or cmdFlags::cfOverrideDeferred) or MEDIACMD::dwCmdAlt with certain #cmdFlags  to record from-top, at speed or combinations of the above.
        * <HR>
```

```
        */
        ctRecord,                       // Record at specified speed
        /**
        * Record Stop - Set the channel into a record ready state, normally passthrough
with the recording file pre-allocated, and possible pass start, end and name information.
See <BR> (cmdFlags::cfUseStart cmdFlags::cfUseStartOffset and MEDIACMD::dwStart),
<BR> (cmdFlags::cfUseEnd cmdFlags::cfUseEndOffset and MEDIACMD::dwEnd), <BR>
(cmdFlags::cfUseClipID and MEDIACMD::arbID) for record setups.
        * <HR>
        */
        ctRecStop,                      // Stop ready for recording
        /**
        * Eject - Stop the channel and unload removable media, if possible, else same as
stop
        * <HR>
        */
        ctEject,                    // Eject the current media
        /**
        * Transfer - Transfer media from one channel to another.  Normally used to
transfer internal media to or from an external tape device.
        * <HR>
        */
        ctTransfer,                     // Transfer source from one channel to another
        /**
        * Insert Clip or Timecode Area - Used in time code space (TCSpace.h) and Clip
Space (ClipSpace.h) to add new clips or areas.  Inserted media is defined by
(cmdFlags::cfUseStart - MEDIACMD::dwStart, cmdFlags::cfUseEnd - MEDIACMD::dwEnd)
for clip being added and (cmdFlags::cfUsePosition - MEDIACMD::dwPosition) for target.
Also (cmdFlags::cfUseClipID and MEDIACMD::arbID) may be used to specify a file name.
        * cmdFlags::cfUsePresets and MEDIACMD::dwVideoChannels,
MEDIACMD::dwAudioChannels, MEDIACMD::dwInfoChannels are also respected if set.
        * cmdFlags::cfRipple may also be used to insert over
        * <BR> NOTE - The ctTransfer command is ALWAYS sent to the target with the
SOURCE channel in the MEDIACMD::dwCmdAlt member and (cmdFlags::cfUseCmdAlt)
set UNLESS one of the devices is slow/high latency/sloppy (read VTR), in which case it
always receives the command so it can master the transfer and the (cmdFlags::cfInvert)
is used to set the direction.
        * <HR>
        */
        ctInsert,                       // Insert a new time code area
        /**
        * Blank a Timecode Area - Used in time code space (TCSpace.h) to set an area
to black and silent audio.
        * (cmdFlags::cfUseStart - MEDIACMD::dwStart, cmdFlags::cfUseEnd -
MEDIACMD::dwEnd) set the area to be blanked.
        * cmdFlags::cfUsePresets and MEDIACMD::dwVideoChannels,
MEDIACMD::dwAudioChannels, MEDIACMD::dwInfoChannels are also respected if set.
        * cmdFlags::cfRipple may also be used to remove blank area.  With this
command, no media is removed from storage.
```

```
        * <HR>
        */
        ctBlank,                    // Erase the specified TC area
        /**
        * Delete a clip (ClipSpace.h) or an area (TCSpace.h).
        * Deletes the media from storage and from the current space.
        * For ClipSpace, cmdFlags::cfUseClipID and MEDIACMD::arbID must be
specified, and if any sub clip or super clips exist, the ID will be removed but the media
will not be deleted.
        * For TCSpace, cmdFlags::cfUseStart and MEDIACMD::dwStart with
cmdFlags::cfUseEnd and MEDIACMD::dwEnd should be used to specify the time code
segment to be deleted.
        * cmdFlags::cfUsePresets and MEDIACMD::dwVideoChannels,
MEDIACMD::dwAudioChannels,
        * MEDIACMD::dwInfoChannels may also be used to delete specific channels.
        * If cmdFlags::cfRipple is set, then the TCSpace will close over the deleted
material, changing all timecode location beyond the deletion point by minus the size of
the deletion.
        * <HR>
        */
        ctDelete,                           // Slide segment within the specified TC area
        /**
        * Trim a clip or area - Currently not implemented.  Use cmdType::ctSetValue and
#cmdGetSetValue::gsClipInfo to trim a clip, or a combination of cmdType::ctInsert,
cmdType::ctDelete, cmdType::ctBlank to trim a tcspace area.
        * <HR>
        */
        ctTrim,                             // Trim the segment within the specified TC area
        /**
        * Channel select - select active channels, preview passthrough channels (to
preview and edit) recording channels (to create a split edit a la CMX)
        * Requires cmdFlags::cfUsePresets and MEDIACMD::dwVideoChannels,
MEDIACMD::dwAudioChannels, MEDIACMD::dwInfoChannels
        * <HR>
        */
        ctChanSelect,               // Passthrough requested channels

        /**
        * Get the current state of the controlled channel(s) - Fills the user supplied
#MEDIACMD structure with the current state.  Look for cmdType::ctError,
cmdType::ctStop, cmdType::ctEject, cmdType::ctPause, cmdType::ctPlay,
        * cmdType::ctRecStop, cmdType::ctRecord for basic state.  For valid fields,
check
        * <ul>
        * <li>cmdFlags::cfDeferred : we have a deferred clip
        * <li>cmdFlags::cfTimeMs : MEDIACMD::dwCmdAlt has millisecond performance
counter info
        * <li>cmdFlags::cfUseSpeed : MEDIACMD::lSpeed has the valid current speed
```

```
        * <li>cmdFlags::cfUsePresets : MEDIACMD::dwVideoChannels,
MEDIACMD::dwAudioChannels, MEDIACMD::dwInfoChannels contain preset information
        * <li>cmdFlags::cfUsePosition : MEDIACMD::dwPosition contains current position
        * <li>cmdFlags::cfUseStart : MEDIACMD::dwStart has starting frame position
        * <li>cmdFlags::cfUseEnd : MEDIACMD::dwEnd has end frame position (+1 the
Out is never included)
        * <li>cmdFlags::cfUseClipID : MEDIACMD::arbID has current clip name (8 char
for Louth and Odetics)
        * <li>cmdFlags::cfFields : MEDIACMD::dwPosition, MEDIACMD::dwStart and
MEDIACMD::dwEnd are in fields if they are valid
        * <li>cmdFlags::cfNoReturn : return is invalid.
        * </ul>
        * <HR>
        */
        ctGetState,                     // Returns TC and transport state information
        /**
        * Set the current state - Used for control type channels such as Serial 422 control
(vvwCtl.h) and network control (vvwNet.h).  Tells the controller or user what our current
state is.  The state should be reported honestly, as it is the receiver's responsibility to
transition states in an appropriate way for its controller.  For actual channels (vvwInt.h,
vvwExt.h, vvwNet.h-as controller, vvwDS2.h, etc), the state should be set by using one
of the transport commands (cmdType::ctPlay etc) above.
        * <HR>
        */
        ctSetState,                     // Sends a new state per GetState
        /**
        * Get a non transport setting - Used for one time setups on channel.
        * Includes audio levels, video proc amps, audio/video input, compression type
and level and many others
        * See:  #cmdGetSetValue for possible commands
        * <HR>
        */
        ctGetValue,                     // Get value of video, audio of internal variable
        /**
        * Set a non transport setting - Used for one time setups on channel.
        * Includes audio levels, video proc amps, audio/video input, compression type
and level and many others
        * See:  #cmdGetSetValue for possible commands
        * <HR>
        */
        ctSetValue,                     // Set value of video, audio of internal variable
        /**
        * Check support for a non transport setting - Used for one time setups on
channel.
        * Includes audio levels, video proc amps, audio/video input, compression type
and level and many others
        * See:  #cmdGetSetValue for possible commands
        * <HR>
        */
```

```
	ctValueSupported,	// Returns true is the specified Get/Set value is supported
	/**
	* Indicates that an error in the channel has occurred.  Return only. See
MEDIACMD::dwCmdAlt for error code and MEDIACMD::arbID for message if any.
	* These members will be valid if cmdFlags::cfUseCmdAlt and
cmdFlags::cfUseClipID are set
	* <HR>
	*/
	ctError,		// An error has occurred
	/**
	* Terminate Close A Channel - Only used by remote devices that cannot close
the channel directly such as vvwNet.h.  Channel may not actually close when this is
called, but the communications pipe will be closed and wait for another connection.
	* <HR>
	*/
	ctTerminate,		// Terminate the current command and move to the next
in the queue, if any
	/**
	* Abort the current operation - Use to abort operations that would normally
ignore extraneous commands such as non-linear playback sequences, records or if the
channel just seems to be stuck.  Makes a good panic button.
	* <HR>
	*/
	ctAbort,		// Abort any queued commands
	/**
	* Edit - this is an internal state set by an edit record (e.g. a record that is not
using all channels, so some channels are playing).  The two cases we are going to
support with AJA will be 'record video + play all audio' and 'play video + record some
audio'.  In the second case audio channels that a VTR would normally play (not armed)
will not be played, and will not be recorded.
	*/
	ctEdit,				// Edit record some channels, play others
	/**
	* Switch file source (later live source?) without changing mode or speed (for
replay)
	*/
	ctSwitch,			// Switch files seamlessly
};

/**
* Flags that modify #cmdType in the #MEDIACMD structure.  Mostly used to specify
which fields in the structure are valid.
*/
enum cmdFlags {
		// Normally, commands occur when the delay time is reached
	/**
	* Delay this command until the end of the previous one.  This is the method for
playing back clips non-linearly.  Send one clip to play, then send each clip after it with
this flag set and they will play seamlessly back to back.<BR>
```

```
        * In the case of ctInsert, the deferred indicates that the clip to be inserted will be translated from its current location to the current record directory and then added to the bin/tcspace.
        * <HR>
        */
        cfDeferred = 1,                         // 0x00000001 This is a delayed command (either at end of prev cmd, or absolute time)
        /**
        * Delay the command, as in #cfDeferred, but kill any other waiting commands and use this command as soon as the current command completes.
        * <HR>
        */
        cfOverrideDeferred = 1 << 30,// 0x40000000 Override all previous deferred commands
#define cfOverideDeferred       cfOverrideDeferred
        /**
        * Time is in milliseconds.  Applies only to the MEDIACMD::dwCmdAlt member. The millisecond reference is derived from the performance counter (or on extremely old machines timeGetTime()) via vsyncGetCurMs() which is implemented in DSync.DLL for user and kernel modes. The default timing without this flag set is in video frames.
        * <HR>
        */
        cfTimeMs = 1 << 1,                      // 0x00000002 Use Millisecond time for delayed time, not fields
        /**
        * Time is set for event occurrence.  This means the command will occur when the time specified is reached.  If this flag is not set and #cfTimeMs is set, then the time indicates the time the command was received and may be used for a deterministic offset. May be in frames (default) or milliseconds #cfTimeMs, requires #cfUseCmdAlt.
        * <HR>
        */
        cfTimeTarget = 1 << 2,                  // 0x00000004 Delayed time is offset from current time code
        /**
        * Time reference is the system clock (time of day) not the performance clock. This is used to sync network or serial based communication where there is no relationship between performance clocks.  For proper operation, the two devices must be genlocked to the same video source, which VVW will interpolate with the correct system clock to keep everything together.  Note:  This is only as accurate as the genlock readers and LTC or Network time transport connected to BOTH machines.  In general, a pair of VVWs are accurate to 1 field, which is ample for editing and broadcast insertion
        * <HR>
        */
        cfTimeHouseClock = 1 << 3,    // 0x00000008 Delayed time is based on absolute (real) time
        /**
        * Means the MEDIACMD::lSpeed member is valid.
        * <HR>
        */
```

```
        cfUseSpeed = 1 << 4,            // 0x00000010 Set the new speed
        /**
        * Means the MEDIACMD::dwVideoChannels, MEDIACMD::dwAudioChannels and
MEDIACMD::dwInfoChannels members are valid.
        * <HR>
        */
        cfUsePresets = 1 << 5,          // 0x00000020 Use video and audio edit presets
        /**
        * Means the MEDIACMD::dwPosition member is valid.
        * <HR>
        */
        cfUsePosition = 1 << 6,                 // 0x00000040 Use the position setting
        /**
        * Means the MEDIACMD::dwPosition member is valid and should be used as a
long (signed) against the current channel position counter.
        * <HR>
        */
        cfUsePositionOffset = 1 << 7,//0x00000080 Position is an offset
        /**
        * Means the MEDIACMD::dwStart member is valid.
        * <HR>
        */
        cfUseStart = 1 << 8,            // 0x00000100 Start a new timecode
        /**
        * Means the MEDIACMD::dwStart member is valid and should be used as a long
(signed) against the current channel position counter.
        * <HR>
        */
        cfUseStartOffset = 1 << 9,      // 0x00000200 Start is an offset from current TC
        /**
        * Means the MEDIACMD::dwEnd member is valid.
        * <HR>
        */
        cfUseEnd = 1 << 10,                     // 0x00000400 End command as
specified
        /**
        * Means the MEDIACMD::dwEnd member is valid and should be used as a long
(signed) against the current channel position counter.
        * <HR>
        */
        cfUseEndOffset =  1 << 11,      // 0x00000800 End is and offset from current TC

        /**
        * Causes the command to act on all IDs in the system.  Used for clipspace to
delete all IDs quickly.
        * <HR>
        */
        cfUseAllIDs = 1 << 12,          // 0x00001000 Use all clip IDs (usually erase
them)
```

```
        /**
        * Means the MEDIACMD::arbID member is valid.
        * <HR>
        */
        cfUseClipID = 1 << 13,          // 0x00002000 Use new clip ID, otherwise use
last or none
        /**
        * Copy the media to the current record folder when inserting
        * <HR>
        */
        cfCopy = 1 << 14,       // 0x00004000
        /**
        * Means the command should not be used on any clip or clip spaces
        * <HR>
        */
        cfNoClipFiles = cfCopy, // Deprecated
        /**
        * Convert the media to the current record folder when inserting
        * <HR>
        */
        cfConvert = 1 << 15,    // 0x00008000
        /**
        * Means the command should not be used on any clip within or the TCSpace
itself
        * <HR>
        */
        cfNoTCSpaces = cfConvert,               // Deprecated
        /**
        * Means the MEDIACMD::dwCmdAlt is valid
        * <HR>
        */
        cfUseCmdAlt = 1 << 16,                  // 0x00010000 Use the dwCmdAlt
        /**
        * Sent by shuttle/jog/var controllers for drivers that require a special play state
that takes too much time to get into.  If this flag is true, the command is a shuttle and
true play does not need to be used
        * <HR>
        */
        cfIsShuttle = 1 << 17,          // 0x00020000 Use speed in play for shuttle
        /**
        * If set then elements that are not illegal are current at the reception of the
command
        * <BR>If #cfUsePosition and #cfUsePositionOffset are NOT set and
#MEDIACMD::dwPosition is not #TC_ILLEGAL, then it is the current position when the
command was received
        * <BR>If #cfUseStart and #cfUseStartOffset are NOT set and
#MEDIACMD::dwStart is not #TC_ILLEGAL, then it is the current start location when the
command was received
```

```
        * <BR>If #cfUseEnd and #cfUseEndOffset are NOT set and
#MEDIACMD::dwEnd is not #TC_ILLEGAL, then it is the current end time when the
command was received
        * <BR>If #cfUseSpeed is set are NOT set and #MEDIACMD::lSpeed is not
#SPD_ILLEGAL, then it is the current speed when the command was received
        * <BR>If cfUsePresets is NOT set and #MEDIACMD::dwAudioChannels,
#MEDIACMD::dwVideoChannels and #MEDIACMD::dwInfoChannels are not 0xFFFFFFFF,
then they are the current presets when the command was received
        * <BR>If #cfUseClipID is NOT set and #MEDIACMD::arbID[0] is not equal to
NULL (""), then it is the current clip ID when the command was received.  If
#MEDIACMD::arbID[9] is not equal to NULL ("") then it is the current file name when the
command was received.
        * <HR>
        */
        cfUsingCurrent = 1 << 18,       // 0x000400000 any elem not flagged is current
        /**
        * If set then MEDIACMD::dwPosition, MEDIACMD::dwStart and
MEDIACMD::dwEnd are absolute (0 based) frame counts, else they are the current type
(CTL/CLIP(frame count) or LTC/VITC(time code offset)).
        * <HR>
        */
        cfUseFrameCount = 1 << 19,   // 0x000800000 Position, start and end are fields,
not frames
        /**
        * If set then MEDIACMD::dwPosition, MEDIACMD::dwStart and
MEDIACMD::dwEnd should be interpreted as fields, not frames, if they are valid
        * <HR>
        */
        cfFields = 1 << 20,                     // 0x00100000 Position, start and end
are fields, not frames
        /**
        * Close up any holes created by this command.  Most importantly
cmdType::ctDelete,
        * cmdType::ctBlank, cmdType::ctInsert and cmdType::ctTrim.
        * <HR>
        */
        cfRipple = 1 << 21,                     // 0x00200000 Ripple for insert or delete
        /**
        * Command should be looped.  Mostly used for loop playback where a start and
end are specified.  The play will begin at the start, proceed to the end, and once
reached, loop back to the start again.
        * <HR>
        */
        cfLoop = 1 << 22,                       // 0x00400000 Loop the clip or in out
        /**
        * INTERNAL - Allows one channel to setup a DSync trigger with another.  Use
cmdType::ctTransfer instead as this is very inefficient for non local command transports.
        * <HR>
        */
```

```
        cfTrigger = 1 << 23,            // 0x00800000 Trigger using dsync class
        /**
        * This command is part of a preview.  Either it notes a channel change
(passthrough to emulate an edit) or that the playback does not have to be consistent and
frame accurate.  Also returned if the channel can only produce preview quality playback
(as in VGA playback of HDTV media without hardware assist).
        * <HR>
        */
        cfPreview = 1 << 24,            // 0x01000000 Preview set (EE, non rt play)
        /**
        * This tells the DDR that the command originated from a remote machine before
being accepted from vvwNet. This is the only way to tell if we have full system access. If
this flag is set, Windows commands (HANDLES) will be ignored at the avHal Level. This is
for all commands not originating from localhost
        *
        *
        * <HR>
        */
        cfRemoteCommand = 1 << 25,
        /**
        * When returned in a status, it means the second field in time (the later field) is
the current one being displayed.  When sent, it indicates which field is to be displayed, if
only one field is going to be displayed, or which field to start the edit on (edit start is
NOT supported in the 3.0 version of VVW).
        * <HR>
        */
        cfSecondField = 1 << 27,                // 0x08000000 Is/Use second field
temporally
        /**

**********************************************************************
         * When used in pause command it will advance to the next

**********************************************************************
         */
        cfUseNextField = cfSecondField,
        /**
        * For cmdType::ctTransfer, invert the source and target.  Use to allow an
external device (such as a VTR) to always master the transfer procedure.  Because of the
high latency and poor ballistics of VTRs, the internal transfer slaves to it regardless of
whether it is the source or target of the transfer.
        * <HR>
        */
        cfInvert = 1 << 28,                     // 0x10000000 Invert a transfer
        /**
        * Means do not act on this command, but return #GS_NOT_SUPPORTED in
dwPosition if you cannot handle it.  Used to determine basic capabilities of the channel.
For instance, if itís an MPEG-2 playback channel, it can't record but if it has a
```

```
passthrough, it may be able to stop.  Using cfTest with cmdType::ctRecord,
cmdType::ctStop will tell the caller this so the interface may be adjusted accordingly.
        * <BR> Caution:  This flag has not been tested with all transport types.  Avoid
for now.
        * <HR>
        */
        cfTest = 1 << 29,                       // 0x20000000 See if the command
exists
        // NOTE:  1 << 30 in use by cfOverrideDeferred
        /**
        * Instructs the channel that no return is required.  The channel then has the
option of remembering the command and acting on it within a reasonable time.  This
means the caller does not know if the command completed successfully at return time,
but the status should be monitored anyways to figure that out.  Especially when long
time functions like a VTR seek will return that the command was successfully initiated,
but not wait for the completion of the seek, regardless of this flag.
        * <HR>
        */
        cfNoReturn = 1UL << 31UL                // 0x80000000 No return mediacmd is
required
};

// Video channels
//! Video channel bit array for MEDIACMD::dwVideoChannels
//! @{
enum cmdVidChan {
        vidChan0 = 1, vidChan1 = 1 << 1, vidChan2 = 1 << 2, vidChan3 = 1 << 3,
        vidChan4 = 1 << 4, vidChan5 = 1 << 5, vidChan6 = 1 << 6, vidChan7 = 1 <<
7,
        vidChan8 = 1 << 8, vidChan9 = 1 << 9, vidChan10 = 1 << 10, vidChan11 = 1
<< 11,
        vidChan12 = 1 << 12, vidChan13 = 1 << 13, vidChan14 = 1 << 14, vidChan15
= 1 << 15,
        vidChan16 = 1 << 16, vidChan17 = 1 << 17, vidChan18 = 1 << 18, vidChan19
= 1 << 19,
        vidChan20 = 1 << 20, vidChan21 = 1 << 21, vidChan22 = 1 << 22, vidChan23
= 1 << 23,
        vidChan24 = 1 << 24, vidChan25 = 1 << 25, vidChan26 = 1 << 26, vidChan27
= 1 << 27,
        vidChan28 = 1 << 28, vidChan29 = 1 << 29, vidChan30 = 1 << 30, vidChan31
= 1UL << 31UL,
#definevidChanAll 0xFFFFFFFFUL
};
//! @}

//! Audio channel bit array for MEDIACMD::dwAudioChannels
//! @{
enum cmdAudChan {
        audChan0 = 1, audChan1 = 1 << 1, audChan2 = 1 << 2, audChan3 = 1 << 3,
```

```
        audChan4 = 1 << 4, audChan5 = 1 << 5, audChan6 = 1 << 6, audChan7 = 1
<< 7,
        audChan8 = 1 << 8, audChan9 = 1 << 9, audChan10 = 1 << 10, audChan11 =
1 << 11,
        audChan12 = 1 << 12, audChan13 = 1 << 13, audChan14 = 1 << 14,
audChan15 = 1 << 15,
        audChan16 = 1 << 16, audChan17 = 1 << 17, audChan18 = 1 << 18,
audChan19 = 1 << 19,
        audChan20 = 1 << 20, audChan21 = 1 << 21, audChan22 = 1 << 22,
audChan23 = 1 << 23,
        audChan24 = 1 << 24, audChan25 = 1 << 25, audChan26 = 1 << 26,
audChan27 = 1 << 27,
        audChan28 = 1 << 28, audChan29 = 1 << 29, audChan30 = 1 << 30,
audChan31 = 1UL << 31UL,
#defineaudChanAll 0xFFFFFFFFUL
};
//! @}

//! Info channel bit array for MEDIACMD::dwInfoChannels
//! @}
enum cmdinf {
        //! LTC time code user bit channel
        infLtc = 1,
        //! VITC time code user bit channel
        infVitc = 1 << 1,
        //! Incoming source control time code
        infSrcCtl = 1 << 2,
        //! Incoming source LTC time code user bits
        infSrcLtc = 1 << 3,
        //! Incoming source VITC time code user bits
        infSrcVitc = 1 << 4,
        //! Record time of day
        infRecTime = 1 << 5,
        //! Record Data
        infRecDate = 1 << 6,
        //! Close caption information
        infCC = 1 << 7,
        //! Authorization information
        infAuth = 1 << 8,
        //! Copyright information
        infCopyright = 1 << 9,
        //! Ownership information
        infOwner = 1 << 10,
        //! Source media name
        infSourceName = 1 << 11,
        //! Source proxy name (if any)
        infProxyName = 1 << 12,
        //! Unused inf13 - inf21
        inf13 = 1 << 13, inf14 = 1 << 14, inf15 = 1 << 15,
```

```
        inf16 = 1 << 16, inf17 = 1 << 17, inf18 = 1 << 18, inf19 = 1 << 19,
        inf20 = 1 << 20, inf21 = 1 << 21, infVB0 = 1 << 22, infVB1 = 1 << 23,
        infVB2 = 1 << 24, infVB3 = 1 << 25, infVB4 = 1 << 26, infVB5 = 1 << 27,
        infVB6 = 1 << 28, infVB7 = 1 << 29, infVB8 = 1 << 30, infVB9 = 1UL << 31UL,
#define infChanAll 0xFFFFFFFFUL
};

//! @{
/**
* Enum sent in MEDIACMD::dwCmdAlt for the commands cmdType::ctGetValue,
* cmdType::ctSetValue, cmdType::ctValueSupported.  <BR>
* ctGetValue will return information in the mediacmd per this document <BR>
* ctSetValue will change the state of the channel using the members of mediacmd per
this document <BR>
* ctValueSupport will return GS_NOT_SUPPORTED in #MEDIACMD::dwPosition if it is
NOT supported.  If it is supported, #MEDIACMD::dwPosition will be set to some other
value <BR>
* <HR>
* NOTE:  <BR>
* \li 'nada' is Spanish for ìnothingî and is used here to indicate that the command is not
supported.
* \li  Time Code - There are three main time code types, each with their own user bit
information.  The 0 based absolute time code is referred to by 'Tc' and 'Ub'.  The LTC
(longitudinal time code or SMPTE time code often sent via audio) is referred to by 'LtcTc'
and 'LtcUb'.  The VITC (vertical interval time code, usually encoded in the vertical blank
area of the video signal) is referred to by 'VitcTc' and VitcUb'.  Not all devices will support
all types or the user bits values for some types.  Use value supported to determine
support
* <HR>
*/
enum cmdGetSetValue {
        /**
        * Current internal time - control or clip absolute zero based time code (0..total
frames exclusive)
        * \li cmdType::ctSetValue
        * <BR> nada
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition
        * <HR>
        */
        gsTc = 1,                                       // Current internal TC (dwPosition)
        /**
        * Current internal user bits
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  nada
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition
        * <HR>
```

```
        */
        gsUb,                                   // Current user bits (dwPosition)
        /**
        * Current LTC time
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - to set VITC generator for next gen (record) if
in preset mod
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition
        * <HR>
        */
        gsLtcTc,                                // Current LTC TC (dwPosition)
        /**
        * Current LTC user bits
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - to set LTC generator
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition
        * <HR>
        */
        gsLtcUb,                                // Current LTC user bits (dwPosition)
        /**
        * Current VITC time
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - to set LTC generator
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition
        * <HR>
        */
        gsVitcTc,                               // Current VITC TC (dwPosition)
        /**
        * Current VITC user bits
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - to set VITC generator
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition
        * <HR>
        */
        gsVitcUb,                               // Current VITC user bits (dwPosition)
        /**
        * Current time code source
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - #GS_TCSOURCE_LTC,
#GS_TCSOURCE_VITC, #GS_TCSOURCE_CTL or #GS_TCSOURCE_CLIP
```

```
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - #GS_TCSOURCE_LTC,
#GS_TCSOURCE_VITC, #GS_TCSOURCE_CTL or #GS_TCSOURCE_CLIP
        * <BR>  MEDIACMD::dwStart - supported types using bit array of above
        * <HR>
        */
        gsTcSource,                             // Default Source (dwPosition, supported
dwStart)
        /**
        * Current time code type
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - #TC2_TCTYPE_FILM, #TC2_TCTYPE_NDF,
#TC2_TCTYPE_DF, #TC2_TCTYPE_PAL, #TC2_TCTYPE_50, #TC2_TCTYPE_5994,
#TC2_TCTYPE_60, #TC2_TCTYPE_NTSCFILM, #TC2_TCTYPE_2398, #TC2_TCTYPE_100
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - #TC2_TCTYPE_FILM, #TC2_TCTYPE_NDF,
#TC2_TCTYPE_DF, #TC2_TCTYPE_PAL, #TC2_TCTYPE_50, #TC2_TCTYPE_5994,
#TC2_TCTYPE_60, #TC2_TCTYPE_NTSCFILM, #TC2_TCTYPE_2398, #TC2_TCTYPE_100
        * <BR>  MEDIACMD::dwStart - supported types using bit array of above
        * <HR>
        */
        gsTcType,                               // DF, NDF, PAL or FILM
(dwPosition,supported dwStart)
        /**
        * Lowest possible time code frame
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - New minimum value
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current minimum value
        * <BR>  MEDIACMD::dwStart - Absolute minimum possible value (usually 0)
        * <HR>
        */
        gsStart,                            // Lowest possible TC (current = dwPosition, min
= dwStart)
        /**
        * Highest possible time code frame plus 1 (out is never included)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - New maximum value + 1
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current maximum value + 1
        * <BR>  MEDIACMD::dwStart - Absolute maximum possible value (usually clip
end + 1)
        * <HR>
        */
        gsEnd,                                  // Highest possible TC (current =
dwPosition, max = dwStart)
```

```
/**
* Current mark in time set by any caller
* <BR>
* \li cmdType::ctSetValue
* <BR>  MEDIACMD::dwPosition - New in time
* \li cmdType::ctGetValue
* <BR>  MEDIACMD::dwPosition - Current in time
* <HR>
*/
gsIn,                                   // Current in point (dwPosition)
/**
* Previous (currently non active) mark in time set by RS-422 protocol
* <BR>
* \li cmdType::ctSetValue
* <BR>  - not supported (internal)
* \li cmdType::ctGetValue
* <BR>  MEDIACMD::dwPosition - last known in time
* <HR>
*/
gsLastIn,                               // Last in point (dwPosition)
/**
* Current mark out time set by any caller
* <BR>
* \li cmdType::ctSetValue
* <BR>  MEDIACMD::dwPosition - New out time
* \li cmdType::ctGetValue
* <BR>  MEDIACMD::dwPosition - Current out time
* <HR>
*/
gsOut,                                  // Current out point (dwPosition)
/**
* Previous (currently non active) mark out time set by RS-422 protocol
* <BR>
* \li cmdType::ctSetValue
* <BR>  - not supported (internal)
* \li cmdType::ctGetValue
* <BR>  MEDIACMD::dwPosition - last known out time
* <HR>
*/
gsLastOut,                              // Last out point (dwPosition)
/**
* Number of frames from Edit On command to start of Record (usually 4~7)
* <BR>
* \li cmdType::ctSetValue
* <BR>  MEDIACMD::dwPosition - New number of frames
* \li cmdType::ctGetValue
* <BR>  MEDIACMD::dwPosition - Current number of frames
* <HR>
*/
```

```
        gsEditOn,                               // Time to start an edit
        /**
        * Number of frames from Edit Off command to end of Record (usually 4~7)
should match #gsEditOn in most cases
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - New number of frames
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Current number of frames
        * <HR>
        */
        gsEditOff,                              // Time to end an edit
        /**
        * Number of frames to preroll before in point for an edit
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - New number of frames
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Current number of frames
        * <HR>
        */
        gsPreroll,                              // Edit pre roll time
        /**
        * Number of frames to postroll after an out point for an edit
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - New number of frames
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Current number of frames
        * <HR>
        */
        gsPostroll,                             // Edit post roll time

        /**
        * Switch from normal mode to auto mode.  For Sony VTR emulation it sets up
Pioneer dual head emulation.  For Louth and Odetics, enables preview play look ahead
for seamless clip playback
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - #GS_TRUE or #GS_FALSE
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Current auto mode state as above
        * <HR>
        */
        gsAutoMode,                             // Setup for NL Playback
        /**
        * Number of frames from receiving Play command to actual Play
        * <BR>
        * \li cmdType::ctSetValue
```

```
        * <BR>  MEDIACMD::dwPosition - New number of frames
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Current number of frames
        * <HR>
        */
        gsPlayDelay,                    // Time from pause to play
        /**
        * LTC time code preset (generator preset)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - to set generator for the next record.  Will be
used not in regen mode.
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - returns the current generator preset.
        * <HR>
        */
        gsLtcTcPreset,
        /**
        * LTC user bit preset (generator preset)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - to set generator for the next record.  Will be
used not in regen mode.
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - returns the current generator
        * preset.
        * <HR>
        */
        gsLtcUbPreset,
        /**
        * VITC time code preset (generator preset)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - to set generator for the next record.  Will be
used not in regen mode.
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - returns the current generator
        * preset.
        * <HR>
        */
        gsVitcTcPreset,
        /**
        * VITC time code preset (generator preset)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - to set generator for the next record.  Will be
used not in regen mode.
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - returns the current generator preset.
```

```
        * <HR>
        */
        gsVitcUbPreset,
        /**
        * Returns the block of data for a frame
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  Not really used at this point
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::arbID - The data block
        * <BR>  MEDIACMD::dwEnd - The size of the data block (0..n)
        * <BR>  MEDIACMD::dwPosition - The expected type of data
        * <HR>
        */
        gsFrameData,
        /**
        * Current Key Code
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwStart - Key Code prefix (4 unsigned chars)
        * <BR>  MEDIACMD::dwPosition - Key Code (4 unsigned chars)
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwStart - Key Code prefix (4 unsigned chars)
        * <BR>  MEDIACMD::dwPosition - Key Code (4 unsigned chars)
        * <HR>
        */
        gsKeyCode,                              // Key Code
        /**
        * Current Ink Code
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwStart - Ink Code prefix (4 unsigned chars)
        * <BR>  MEDIACMD::dwPosition - Ink Code (3 unsigned chars)
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwStart - Ink Code prefix (4 unsigned chars)
        * <BR>  MEDIACMD::dwPosition - Ink Code (3 unsigned chars)
        * <HR>
        */
        gsInkCode,                              // Ink Code
        /**
        * Current 215 Code Code
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwStart - Audio Phase [msb], Audio Modulus, pull down,
sequence [lsb]
        * <BR>  MEDIACMD::dwPosition - Absolute Frame
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwStart - Audio Phase [msb], Audio Modulus, pull down,
sequence [lsb]
```

```
        * <BR>  MEDIACMD::dwPosition - Absolute Frame
        * <HR>
        */
        gs215Code,                              // Other 215 Codes
        /**
        * Set the heads and tails to be used for the next record.  Automatically set to
zero after record/add
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - Heads and Tails size in frames
        * \li cmdType::ctGetValue - not supported
        * <BR>
        * <HR>
        */
        gsHeadsAndTails,                // Number of frames being added for heads and
tails
        /**
        * Set timecode sources limitations
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - #GS_TCSRC_DISABLE_EXTERNAL,
#GS_TCSRC_FORCE_VTR_TC, #GS_TCSRC_USE_TIMEOFDAY
        * \li cmdType::ctGetValue - not supported
        * <BR>
        * <HR>
        */
        gsTimecodeSources,              //


        gsGetNextClip = 90,             // Get clip name and info (was first/next - send
NULL arbID for first)
        /**
        * Obsolete - use #gsGetNextClip
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  - not supported in new drivers
        * \li cmdType::ctGetValue
        * <BR>  - not supported in new drivers
        * <HR>
        */
        gsFirstClip,                    // First clip name (arbID - name, dwStart, dwEnd
if avail)
        /**
        * Obsolete - use #gsGetNextClip
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  - not supported in new drivers
        * \li cmdType::ctGetValue
        * <BR>  - not supported in new drivers
```

```
        * <HR>
        */
        gsNextClip,                             // Next clip name, (arbID - name,
dwStart, dwEnd if avail)
        /**
        * Return the next state info when working through a time code space time line to
retrieve all the edits in order.  The state uses MEDIACMD::dwPosition,
MEDIACMD::dwVideoChannels, MEDIACMD::dwAudioChannels,
MEDIACMD::dwInfoChannels  to maintain the state (Please note that MEDIACMD::arbID
is reserved and must be maintained between calls).  The dwPosition describes the
current position in the timeline and the channel bits are set for channels already
returned.
        *  See gsTCSGetTLNextClip for more info
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  - not supported
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Current time line position
        * <BR>MEDIACMD::dwVideoChannels - Video channels used so far
        * <BR>MEDIACMD::dwAudioChannels - Audio channels used so far
        * <BR>MEDIACMD::dwInfoChannels - Info channels used so far
        * <HR>
        */
        gsTCSGetTLClipState,
        /**
        * CALL                                  Pos             Start   End             V
                A               I               arbID
        * gsTCSGetTLClipInfo    0               x               x               0
        0               0               x               - Restart at 0
        *       Rtn                                     0               0
        300             1               2               0               file1   - 10 sec
VA2 from file1
        * gsTCSGetTLNextState   0                                                       0
        0               0                               - First state 0
        *       Rtn                                     0
                1               2               0                                       - First
clip channels
        *       ( Copy prev gsTCSGetTLNextState into gsTCSGetTLClipInfo before
sending )
        * gsTCSGetTLClipInfo    0                                                       1
        2               0                               - Last get state
        *       Rtn                                     0               0
        150             0               1               0               file2   - 5 sec
A1 from file2
        *       ( Use last gsTCSGetTLNextState for this call )
        * gsTCSGetTLNextState   0                                                       1
        2               0                               - Use last state to get next
```

```
        *       Rtn                                     0
                1               3               0                               -
Channels used so far
        *       ( Copy prev gsTCSGetTLNextState into gsTCSGetTLClipInfo before
sending )
        * gsTCSGetTLClipInfo   0                                               1
        3               0                               - Last get state
        *       Rtn                                     150             150
        210             0               1               0               file3   - 2 sec
A1 from file3
        *       ( Use last gsTCSGetTLNextState for this call )
        * gsTCSGetTLNextState 0                                               1
        3               0                               - Use last state to get next
        *       Rtn                                     150
                0               1               0                               -
Channels used so far
        * Take the #MEDIACMD struct returned from gsTCSGetTLClipState and find the
next active clip.  For the first clip in time line, send all zeroes.  Other then the first call, all
calls should include the position/channel bits from the previous gsTCSGetTLNextState call
(other than the first call) and gsTCSGetTLNextState should be called immediately before
gsTCSGetTLClipInfo.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  - not supported
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::arbID - Clip ID
        * <BR>  for MCMD2 -out- MEDIACMD::arbID - Next 8 character ID and unc file
path separated by NULL or 8 NULLs if clip list complete
        * <BR>  MEDIACMD::cfFlags - Set cfUsePosition|cfUseStart|cfUseEnd to search
next clip, set cfUsePosition & MEDIACMD::dwPosition for info at specified position
        * <BR>  MEDIACMD::dwPosition - Reference time code for time line
        * <BR>  MEDIACMD::dwStart - First frame of clip
        * <BR>  MEDIACMD::dwEnd - Last frame of clip
        * <BR>MEDIACMD::dwVideoChannels - Channels this clip exists in for the
dwStart/dwEnd range
        * <BR>MEDIACMD::dwAudioChannels - Channels this clip exists in for the
dwStart/dwEnd range
        * <BR>MEDIACMD::dwInfoChannels - Channels this clip exists in for the
dwStart/dwEnd range
        * <HR>
        */
        gsTCSGetTLClipInfo,
        /**
        * Get or change the information on a clip (currently for clip space only)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  - not supported
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::arbID - Last returned clip ID or 8 NULLs for first clip
```

```
        * <BR>  for MCMD2 -out- MEDIACMD::arbID - Next 8 character ID and unc file
path separated by NULL or 8 NULLs if clip list complete
        * <BR>  MEDIACMD::dwPosition - Starting timecode if known, else First frame of
clip
        * <BR>  MEDIACMD::dwStart - First frame of clip
        * <BR>  MEDIACMD::dwEnd - Last frame of clip
        * <BR>MEDIACMD::dwVideoChannels - Channels this clip exists in for the
dwStart/dwEnd range
        * <BR>MEDIACMD::dwAudioChannels - Channels this clip exists in for the
dwStart/dwEnd range
        * <BR>MEDIACMD::dwInfoChannels - Channels this clip exists in for the
dwStart/dwEnd range
        * <HR>
        */
        gsClipInfo,                             // Same as above for named clip or
current (arbID - name, dwStart, dwEnd if avail)
        /**
        * Create a virtual copy of a clip from a current clip.  Must change at least name
to succeed.  To affect the source clip, use #gsClipInfo
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  Requires MEDIACMD::cfFlags set to affect stored clip info for each
member
        * <BR>  MEDIACMD::arbID - Source ClipID [8 unsigned chars], NULL, New
ClipID [8 unsigned chars] - min size 17 unsigned chars.
        * <BR>  MEDIACMD::dwStart - First frame of new clip (referenced from source
clip)
        * <BR>  MEDIACMD::dwEnd - Last frame of new clip (referenced from source
clip)
        * <BR>MEDIACMD::dwVideoChannels - Channels this clip exists in for the
dwStart/dwEnd range
        * <BR>MEDIACMD::dwAudioChannels - Channels this clip exists in for the
dwStart/dwEnd range
        * <BR>MEDIACMD::dwInfoChannels - Channels this clip exists in for the
dwStart/dwEnd range
        * \li cmdType::ctGetValue
        * <BR>- no supported
        * <HR>
        */
        gsClipCopy,                             // Copy current clip to specified name


        /**
        * Returns the available audio channels (read only)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>- not supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Available channels
```

```
        * <HR>
        */
        gsAudChan = 100,                // Available audio channels (dwPosition)
        /**
        * Returns the available video channels (read only)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>- not supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Available channels
        * <HR>
        */
        gsVidChan,                              // Available video channels (dwPosition)
        /**
        * Returns the available information channels (read only)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>- not supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Available channels
        * <HR>
        */
        gsInfChan,                              // Available info channels (dwPosition)
        /**
        * Return or set the selected audio channels
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New channel selection
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Currently selected channels
        * <BR>MEDIACMD::dwStart - Available channels for selection
        * <HR>
        */
        gsAudSelect,                    // Selected audio channels (dwPosition,
supported dwStart)
        /**
        * Return or set the selected video channels
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New channel selection
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Currently selected channels
        * <BR>MEDIACMD::dwStart - Available channels for selection
        * <HR>
        */
        gsVidSelect,                    // Selected video channels (dwPosition,
supported dwStart)
        /**
        * Return or set the selected information channels
```

```
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New channel selection
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Currently selected channels
        * <BR>MEDIACMD::dwStart - Available channels for selection
        * <HR>
        */
        gsInfSelect,                    // Selected info channels (dwPosition, supported
dwStart)
        /**
        * Return or set the audio channels for the next edit
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New channel edit selection
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Currently selected edit channels
        * <BR>MEDIACMD::dwStart - Available channels for edit
        * <HR>
        */
        gsAudEdit,                      // Edit ready audio channels (dwPosition,
supported dwStart)
        /**
        * Return or set the video channels for the next edit
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New channel edit selection
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Currently selected edit channels
        * <BR>MEDIACMD::dwStart - Available channels for edit
        * <HR>
        */
        gsVidEdit,                      // Edit ready video channels (dwPosition,
supported dwStart)
        /**
        * Return or set the information channels for the next edit
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New channel edit selection
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Currently selected edit channels
        * <BR>MEDIACMD::dwStart - Available channels for edit
        * <HR>
        */
        gsInfEdit,                      // Edit ready info channels (dwPosition,
supported dwStart)
        /**
        * Return or set the information channels for the next edit
        * <BR>
```

```
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New channel edit selection
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Currently selected edit channels
        * <BR>MEDIACMD::dwStart - Available channels for edit
        * <HR>
        */
        gsEditMode,                                 // Edit to use assemble or insert

        // MetaData set/retrieve
        /**
        * Access one metadata element for the current media. <BR>
        * See the enum #vvwInfoMetaTypes in vwwTypes.h
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwEnd - GS_TRUE if the element exists,
GS_NOT_SUPPORTED if not
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwEnd - GS_TRUE if the element exists,
GS_NOT_SUPPORTED if not
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * <HR>
        */
        gsMetaData = 150,
        /**
        * Access one metadata element for the directory of the current media
(./Default.xml). <BR>
        * See the enum #vvwInfoMetaTypes in vwwTypes.h
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * \li cmdType::ctGetValue
```

```
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * <HR>
        */
        gsMetaDataDirectory,
        /**
        * Access one metadata element for the drive/volume of the current media
(/Default.xml). <BR>
        * See the enum #vvwInfoMetaTypes in vwwTypes.h
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * <HR>
        */
        gsMetaDataVolume,
        /**
        * Access one metadata element for the default metadata of the current user
        * (HKEY_CURRENT_USER windows, /home/user/default.xml unix). <BR>
        * See the enum #vvwInfoMetaTypes in vwwTypes.h
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
```

```
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * <HR>
        */
        gsMetaDataCurrentUser,
        /**
        * Access one metadata element for the default metadata of the current user
        * (HKEY_LOCAL_MACHINE windows, /var/metadata/default.xml unix). <BR>
        * See the enum #vvwInfoMetaTypes in vwwTypes.h
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * <HR>
        */
        gsMetaDataLocalMachine,
        /**
        * Access one meta data element for the default metadata for the facility
        * (Requires group or facility media proxy and database). <BR>
        * See the enum #vvwInfoMetaTypes in vwwTypes.h
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - ID = #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::dwStart - value for #vvwInfoMetaTypes::vvwiTimeCode ..
#vvwInfoMetaTypes::vvwiGamma1000
        * <BR> MEDIACMD::arbID - value for #vvwInfoMetaTypes::vvwiFileName ..
#vvwInfoMetaTypes::vvwiFrameAttribute
        * <HR>
        */
        gsMetaDataGlobal,
```

```
        /**
        * Write or read the current metadata structure from the XML on the disk
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - to 0 = media file, 1 = directory, 2 = volume, 3
= current user, 4 = current machine, 5 = master server
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - from 0 = media file, 1 = directory, 2 =
volume, 3 = current user, 4 = current machine, 5 = master server
        * <HR>
        */
        gsMetaDataReadWrite,

#define METABASE_TYPE_UNKNOWN           0
#define METABASE_TYPE_CHAR                      1
#define METABASE_TYPE_INT               2

        /**
        * Open a metadata database file
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - DataBase File Name (IN)
        * <BR> MEDIACMD::dwPosition - Success/Failure (OUT)
        * NOTE: return Success/Failure (-1 if it doesn't exist)
        */
        gsMetaBaseOpen,
        /**
        * Create a new metadata database file
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - DataBase File Name (IN)
        * <BR> MEDIACMD::dwPosition - Success/Failure (OUT)
        */
        gsMetaBaseCreate,
        /**
        * \li cmdType::ctSetValue
        * Close previously opened database
        */
        gsMetaBaseClose,
        /**
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - Count number of entries in database (OUT)
        */
        gsMetaBaseFileCount,
        /**
        * Get each file item in database
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - Index (IN)
        * <BR> MEDIACMD::arbID - File Name (OUT)
        * <BR> MEDIACMD::dwPosition - Success/Index (OUT)
        */
```

```
        gsMetaBaseFileName,
        /**
        * Removing a file from the database
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - File Name (IN)
        * <BR> MEDIACMD::dwPosition - return -1 if failed (OUT)
        */
        gsMetaBaseFileRemove,
        /**
        * Add a new table entry for a file
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - FileName (IN)
        * <BR> MEDIACMD::dwPosition - Success/Index (-1 if exist) (OUT)
        */
        gsMetaBaseFileAdd,
        /**
        * Get metadata item and it's value for specified file item
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::arbID - FileName + MetaDataTag (IN)
        * <BR> MEDIACMD::dwPosition - Index of meta item, or -1 if !exist (OUT)
        */
        gsMetaBaseTagIndex,
        /**
        * Get metadata item and it's value for specified file item
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::arbID - File Name (Table in DataBase) (IN)
        * <BR> MEDIACMD::dwPosition - Index of meta item (IN)
        * <BR> MEDIACMD::arbID - Name/Value (Name is first starting at [0]) (OUT)
        * <BR> MEDIACMD::dwStart - Where in arbID the value starts (OUT)
        * <BR> MEDIACMD::dwPosition - Success/Index (OUT)
        */
        gsMetaBaseGetTag,
        /**
        * Set/Insert metadata item and it's value for specified file
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - Name/Tag/Value (Name is first starting at [0]) (IN)
        *                                       - Tag starts at MEDIACMD::dwStart,
Value starts at MEDIACMD::dwEnd
        */
        gsMetaBaseSetTag,
        /**
        * Insert all default metadata tags, (no values)
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - Name (IN)
        */
        gsMetaBaseDefaultTags,
        /**
        * Reset table (filename)
        * \li cmdType::ctSetValue
```

```
        * <BR> MEDIACMD::arbID - Name (IN)
        */
        gsMetaBaseFileRename,
        /**
        * Set replay mark 15 seconds after tc value sent
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID -
        */
        gsMetaBaseReplayMark,
        /**
        * Get the current database name
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::arbID -
        */
        gsMetaBaseGetName,
        /**
        * Get the current database name
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::arbID -
        */
        gsMetaBaseGetTable,
        /**
        * Reset and start the record session
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - Record Start (IN)
        * <BR> MEDIACMD::dwChannel - Record Channel (IN)
        * <BR> MEDIACMD::arbID - ClipName (IN)
        */
        gsMetaBaseResetAndRecord,

        // Audio settings use dwAudioChannels (except LTC)
        /**
        * Audio input select
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New audio input
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
        * #GS_AUDSELECT_BALANCED_10      #GS_AUDSELECT_BALANCED_4
#GS_AUDSELECT_SPDIF #GS_AUDSELECT_AES_EBU
        * #GS_AUDSELECT_EMBEDDED
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current input
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
        * #GS_AUDSELECT_BALANCED_10      #GS_AUDSELECT_BALANCED_4
#GS_AUDSELECT_SPDIF #GS_AUDSELECT_AES_EBU
        * #GS_AUDSELECT_EMBEDDED
        * <BR>MEDIACMD::dwStart - Bit array of available inputs, see above or
#GS_AUDSELECT_NONE
```

```
        * <BR>MEDIACMD::dwAudioChannels - Channels requested
        * <HR>
        */
        gsAudInSelect = 200,    // Audio Input Select (dwPosition, available = dwStart)
        /**
        * Audio output select (in general all outputs are active)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New audio output
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
        * #GS_AUDSELECT_BALANCED_10       #GS_AUDSELECT_BALANCED_4
#GS_AUDSELECT_SPDIF #GS_AUDSELECT_AES_EBU
        * #GS_AUDSELECT_EMBEDDED
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current output
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
        * #GS_AUDSELECT_BALANCED_10       #GS_AUDSELECT_BALANCED_4
#GS_AUDSELECT_SPDIF #GS_AUDSELECT_AES_EBU
        * #GS_AUDSELECT_EMBEDDED
        * <BR>MEDIACMD::dwStart - Bit array of available outputs, see above or
#GS_AUDSELECT_NONE
        * <BR>MEDIACMD::dwAudioChannels - Channels requested
        * <HR>
        */
        gsAudOutSelect,                         // Audio Output Select (dwPosition,
available = dwStart)

        /**
        * Audio input level (gain)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Level (0-65535)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Level (0-65535)
        * <BR>MEDIACMD::dwStart - Minimum level (usually 0)
        * <BR>MEDIACMD::dwEnd - Maximum level (usually 65535)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * <HR>
        */
        gsAudInputLevel,                // Input level setting (16 bit) (dwPosition, min =
dwStart, max = dwEnd)
        /**
        * Audio output level (master)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Level (0-65535)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
```

```
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Level (0-65535)
        * <BR>MEDIACMD::dwStart - Minimum level (usually 0)
        * <BR>MEDIACMD::dwEnd - Maximum level (usually 65535)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * <HR>
        */
        gsAudOutputLevel,               // Output level setting (16 bit) (dwPosition, min
= dwStart, max = dwEnd)
        /**
        * Audio advanced level (advanced cue head master) - Not Supported
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Level (0-65535)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Level (0-65535)
        * <BR>MEDIACMD::dwStart - Minimum level (usually 0)
        * <BR>MEDIACMD::dwEnd - Maximum level (usually 65535)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * <HR>
        */
        gsAudAdvanceLevel,              // ??? (Not Supported) (dwPosition, min =
dwStart, max = dwEnd)
        /**
        * Audio output phase
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Phase offset (default = 0) (0-65520 = degrees
* 182)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Phase offset (0-65520 = degrees * 182)
        * <BR>MEDIACMD::dwStart - Minimum phase available (usually 0)
        * <BR>MEDIACMD::dwEnd - Maximum phase available (usually 65520 = 360 *
182)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * <HR>
        */
        gsAudOutPhase,                          // In samples (dwPosition, min =
dwStart, max = dwEnd)
        /**
        * Audio advance phase (advanced cue head master) - Not Supported
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Phase offset (default = 0) (0-65520 = degrees
* 182)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * \li cmdType::ctGetValue
```

```
        * <BR>MEDIACMD::dwPosition - Phase offset (0-65520 = degrees * 182)
        * <BR>MEDIACMD::dwStart - Minimum phase available (usually 0)
        * <BR>MEDIACMD::dwEnd - Maximum phase available (usually 65520 = 360 *
182)
        * <BR>MEDIACMD::dwAudioChannels - Channels affected
        * <HR>
        */
        gsAudOutAdvancePhase,          // ??? (Not Supported) in samples (dwPosition,
min = dwStart, max = dwEnd)
        /**
        * Audio cross-fade time (clip effect overlap) - Not Supported
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Length of cross-fade in milliseconds
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Length of cross-fade in milliseconds
        * <BR>MEDIACMD::dwStart - Minimum cross-fade length (usually 0 = cut)
        * <BR>MEDIACMD::dwEnd - Maximum cross-fade length (depends on device)
        * <HR>
        */
        gsAudCrossFadeTime,            // Audio cross fade duration (dwPosition, min =
dwStart, max = dwEnd)
        /**
        * Enable LTC on an audio channel
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - #GS_ENABLE or #GS_DISABLE
        * <BR>MEDIACMD::dwAudioChannels - Bit for channel to use for LTC
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Is LTC enabled
        * <HR>
        */
        gsAudLtcEnable,                        // LTC enabled (dwposition)
        /**
        * Set audio channel to use for LTC input if enabled.  Currently will set LTC output
to same channel on all VVW drivers.
        *
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwAudioChannels - Bit for channel to use for LTC
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwAudioChannels - Bit channel is using for LTC
        * <HR>
        */
        gsAudInLtcChannel,             // LTC channel, -1 if disabled (dwPosition,
available = dwStart)
        /**
        * Set audio channel to use for LTC output if enabled.  Currently will set
        * LTC input to same channel on all VVW drivers.
```

```
        *
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwAudioChannels - Bit for channel to use for LTC
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwAudioChannels - Bit channel is using for LTC
        * <HR>
        */
        gsAudOutLtcChannel,             // 210 LTC channel, -1 if disabled (dwPosition,
available = dwStart)
        /**
        * Enable DTMF on an audio channel
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - #GS_ENABLE or #GS_DISABLE
        * <BR>MEDIACMD::dwAudioChannels - Bit for channel to use for DTMF
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Is DTMF enabled
        * <HR>
        */
        gsAudDtmfEnable,                // DTMF enabled (dwposition)
        /**
        * Set audio channel to use for DTMF input if enabled.  Currently will set DTMF
output to same channel on all VVW drivers.
        *
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwAudioChannels - Bit for channel to use for DTMF
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwAudioChannels - Bit channel is using for DTMF
        * <HR>
        */
        gsAudInDtmfChannel,             // DTMF channel, -1 if disabled (dwPosition,
available = dwStart)
        /**
        * Set audio channel to use for DTMF output if enabled.  Currently will set DTMF
input to same channel on all VVW drivers.
        *
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwAudioChannels - Bit for channel to use for DTMF
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwAudioChannels - Bit channel is using for DTMF
        * <HR>
        */
        gsAudOutDtmfChannel, // DTMF channel, -1 if disabled (dwPosition, available =
dwStart)
        /**
```

```
        * Return the last known RMS and peak value of the audio output.  Max 2
channels returned per call.  2 channels should always be requested
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>- Not Supported
        * \li cmdType::ctGetValue
        * <BR>-in- MEDIACMD::dwAudioChannels - Requested channels to check
        * <BR>MEDIACMD::dwStart - HIunsigned short=RMS channel +1, LOunsigned
short=RMS channel +0 (range 0-65535)
        * <BR>MEDIACMD::dwEnd - HIunsigned short=Peak channel +1, LOunsigned
short=Peak channel +0 (range 0-65535)
        * <BR>MEDIACMD::dwPosition - duplicates #MEDIACMD::dwStart
        * <HR>
        */
        gsAudWavePeakRMS,               // Current play or in peak|rms 0-
(dwStart:Hiunsigned short|Lounsigned short), 1-(dwEnd:Hiunsigned short|Lounsigned
short)
        /**
        * Get / Set the current bit rate for recording audio per call
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>- Not Supported
        * \li cmdType::ctGetValue
        * <BR>-in- MEDIACMD::dwAudioChannels - Requested channels to check
        */
        gsAudInputBitRate,                          // 215 Sets the bit rate for audio records
(Argus)
        /**
        * Set the audio input sample rate
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New sample rate (typically 48000 or 96000)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current sample rate (use
cmdType::ctValueSupported to get list)
        * <BR>MEDIACMD::dwStart - Lowest supported sample rate
        * <BR>MEDIACMD::dwEnd - Highest supported sample rate
        * <HR>
        */
        gsAudInputSampleRate,
        /**
        * Audio input mode
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - See Sony defines
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - See Sony defines
        * <HR>
        */
```

```
gsAudInputMode,
/**
* Audio input head room
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - Set required headroom
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition - Current headroom
* <BR>MEDIACMD::dwStart - Min headroom
* <BR>MEDIACMD::dwEnd - Max Headroom ;)
* <HR>
*/
gsAudInputHeadRoom,
/**
* Audio input original - see Sony def
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition -
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition -
* <HR>
*/
gsAudInputOriginal,
/**
* Enable and disable audio input error protection
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - GS_TRUE protection enabled, else GS_FALSE
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition - GS_TRUE protection enabled, else GS_FALSE
* <HR>
*/
gsAudInputErrorProtect,          //220
/**
* Does the audio input bit stream contain a copyright flag
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - GS_TRUE copyright flag is set, else GS_FALSE
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition - GS_TRUE copyright flag is set, else GS_FALSE
* <HR>
*/
gsAudInputCopyright,
/**
* Audio input is in slave mode
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - GS_TRUE if in slave mode, else GS_FALSE
* \li cmdType::ctGetValue
```

```
        * <BR>MEDIACMD::dwPosition - GS_TRUE if in slave mode, else GS_FALSE
        * <HR>
        */
        gsAudInputSlave,
        /**
        * Audio bass setting, hardware dependent
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 0..65536, 32768 being nominal and -1 being
default
        * <BR>MEDIACMD::dwAudioChannels - Bit(s) in use
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 0..65536, 32768 being nominal and -1 being
not supported
        * <BR>MEDIACMD::dwAudioChannels - Bit(s) in use
        * <HR>
        */
        gsAudInputBass,
        /**
        * Audio treble setting, hardware dependent
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 0..65536, 32768 being nominal and -1 being
default
        * <BR>MEDIACMD::dwAudioChannels - Bit(s) in use
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 0..65536, 32768 being nominal and -1 being
not supported
        * <BR>MEDIACMD::dwAudioChannels - Bit(s) in use
        * <HR>
        */
        gsAudInputTreble,
        /**
        * What audio channels are available, selected and valid
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Available
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Audio channel bits with valid inputs
        * <BR>MEDIACMD::dwStart - Audio channel bits on primary audio selection
        * <BR>MEDIACMD::dwEnd - Audio channel bits being monitored
        * <BR>MEDIACMD::dwAudioChannels - Audio channel bits available
        * <HR>
        */
        gsAudInputStatus,       // 225
        /**
        * Changes the audio mapping on the input
        * <BR>
        * \li cmdType::ctSetValue
```

```
        * <BR>Not Available
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Bit array of the mapping
        * <BR>MEDIACMD::dwAudioChannels - Audio channel to map
        * <HR>
        */
        gsAudioMappingInput,
        /**
        * Changes the audio mapping on the output
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Available
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Bit array of the mapping
        * <BR>MEDIACMD::dwAudioChannels - Audio channel to map
        * <HR>
        */
        gsAudioMappingOutput,
        /**
        * Selects the pair of channels to monitored (usually RCA analog)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Numeric pair 1/2=0, 3/4=1, 5/6=2, 7/8=3, etc
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Numeric pair 1/2=0, 3/4=1, 5/6=2, 7/8=3, etc
        * <BR>MEDIACMD::dwStart - Available pairs (bitwise)
        * <BR>  MEDIAMCD::dwEnd - Highest possible value (like position)
        * <HR>
        */
        gsAudMonitorSelect,
        /**
        * Selects the channels that are encoded (need raw capture/playback) for Dolby
(or doubley)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Bit pair 1/2=0x03, 3/4=0x0C, 5/6=0x30,
7/8=0xC0, etc
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Bit pair 1/2=0x03, 3/4=0x0C, 5/6=0x30,
7/8=0xC0, etc
        * <BR>MEDIACMD::dwStart - Available pairs
        * <HR>
        */
        gsAudChannelsEncoded,
        /**
        * Enable or disable audio output at non play speed (scrub)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 1-enable audio scrub, 0-disable
```

```
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 1-enable audio scrub, 0-disable
        * <BR>MEDIACMD::dwStart - 1 is available
        * <HR>
        */
        gsAudAudioScrub,        // 230

        /**
        * Freeze the video output
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Freeze type #GS_VIDFREEZE_NOT_FROZEN,
#GS_VIDFREEZE_FIELD0, #GS_VIDFREEZE_FIELD1, #GS_VIDFREEZE_FRAME
        * <BR>MEDIACMD::dwVideoChannels - Bit(s) for the channel(s) to freeze
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwAudioChannels - Bit(s) for the channel(s) currently frozen
        * <HR>
        */
        gsVidFreeze = 300,              // Freeze video 0-un, 1 field, 2 field, 3 both
(dwPosition)
        /**
        * Set DDR into pre read (read before write) mode - requires 2 or more channels
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Channel to use as output
        * <BR>MEDIACMD::dwStart - #GS_ENABLE or #GS_DISABLE
        * <BR>MEDIACMD::dwVideoChannels - Channels to record
        * <BR>MEDIACMD::dwAudioChannels - Channels to record
        * <BR>MEDIACMD::dwInfoChannels - Channels to record
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwStart - #GS_ENABLE or #GS_DISABLE - if not enabled,
the rest does not matter
        * <BR>MEDIACMD::dwPosition - Channel in use as output
        * <BR>MEDIACMD::dwVideoChannels - Channels recording
        * <BR>MEDIACMD::dwAudioChannels - Channels recording
        * <BR>MEDIACMD::dwInfoChannels - Channels recording
        * <HR>
        */
        gsVidPreReadMode,               // Setup for preread mode (dwPosition =
channel, dwStart = used channels, dwEnd = available channels);
        /**
        * First field recorded in edit
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Edit start field (#GS_FIELD2 for second, else
first)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Edit start field (#GS_FIELD2 for second, else
first)
```

```
        * <HR>
        */
        gsVidEditField,                 // Starting field for an edit (2=2nd, else 1st)
(dwPosition = channel, dwStart = available channels)
        /**
        * Record frames or fields
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_FIELD record single field, else record
frames (default - frames (both fields))
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_FIELD recording single field, else
recording frames (default - frames (both fields))
        * <HR>
        */
        gsVidRecFrame,                  // Record frame or field (dwPosition = channel,
dwStart = available channels)
        /**
        * Play frames or fields
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_FIELD play single field, else play frames
(default - frames (both fields))
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_FIELD recording fields, else recording
frames (default - frames (both fields))
        * <HR>
        */
        gsVidPlayFrame,                     // Play frame or field (dwPosition =
channel, dwStart = available channels)
        /**
        * Disable video edit to edit passthrough
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_TRUE Always in playback mode, else if
#GS_FALSE then will passthrough video
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_TRUE Always in playback mode, else if
#GS_FALSE then will passthrough video
        * <HR>
        */
        gsVidNoEE,                          // No E to E mode allowed (dwPosition =
channel, dwStart = available channels)
        /**
        * Enable superimposed tc/state/menu output in video
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_TRUE Superimpose, else normal video
```

```
        * <BR>MEDIACMD::dwStart - SuperImpose Type 0 = VTR Style 1= Film Full 2=
Film basic
        * <BR>MEDIACMD::dwEnd -  SuperImpose on VGA only
        * <BR>MEDIACMD::dwVideoChannels -  Height to start Imposing
        * <BR>MEDIACMD::dwAudioChannels -  Width to start Imposing
        * <BR>  MEDIACMD::lSpeed - Color of watermark (NOT SUPERIMPOSE TEXT
YET)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_TRUE Superimpose, #GS_FALSE Normal
Video, #GS_NOT_SUPPORTED cannot superimpose
        * <BR>MEDIACMD::dwStart - SuperImpose Type 0 = VTR Style 1= Film Full 2=
Film basic
        * <BR>MEDIACMD::dwEnd -  SuperImpose on VGA only
        * <BR>MEDIACMD::dwVideoChannels -  Height to start Imposing
        * <BR>MEDIACMD::dwAudioChannels -  Width to start Imposing
        * <HR>
        */
        gsVidSuperimpose,               // Super tc/state data on monitor output
(dwPosition = channel, dwStart = available channels)

        /**
        * Select the output type of the analog SD (Composite, SMPTE, RGB)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_VIDSELECT_COMPONENT_YUV
#GS_VIDSELECT_COMPONENT_YUV_M2 #GS_VIDSELECT_COMPONENT_YUV_SMPTE
#GS_VIDSELECT_COMPONENT_RGB #GS_VIDSELECT_COMPOSITE
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_VIDSELECT_COMPONENT_YUV
#GS_VIDSELECT_COMPONENT_YUV_M2 #GS_VIDSELECT_COMPONENT_YUV_SMPTE
#GS_VIDSELECT_COMPONENT_RGB #GS_VIDSELECT_COMPOSITE
        * <HR>
        */
        gsVidAnalogMonitorSDType,
        /**
        * Select the output type of the analog HD (RGB, SMPTE, xVidRGB)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_VIDSELECT_COMPONENT_YUV
#GS_VIDSELECT_COMPONENT_YUV_M2 #GS_VIDSELECT_COMPONENT_YUV_SMPTE
#GS_VIDSELECT_COMPONENT_RGB #GS_VIDSELECT_XVID_RGB
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_VIDSELECT_COMPONENT_YUV
#GS_VIDSELECT_COMPONENT_YUV_M2 #GS_VIDSELECT_COMPONENT_YUV_SMPTE
#GS_VIDSELECT_COMPONENT_RGB #GS_VIDSELECT_XVID_RGB
        * <HR>
        */
        gsVidAnalogMonitorHDType,
        /**
```

```
        * Set the type of up down convert to do
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_ANALOGMONITORMETHOD_DIRECT,
#GS_ANALOGMONITORMETHOD_SD,
        *   #GS_ANALOGMONITORMETHOD_HD720,
#GS_ANALOGMONITORMETHOD_HD1080, #GS_ANALOGMONITORMETHOD_FLIP720,
        *   #GS_ANALOGMONITORMETHOD_FLIP1080
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_ANALOGMONITORMETHOD_DIRECT,
#GS_ANALOGMONITORMETHOD_SD,
        *   #GS_ANALOGMONITORMETHOD_HD720,
#GS_ANALOGMONITORMETHOD_HD1080, #GS_ANALOGMONITORMETHOD_FLIP720,
        *   #GS_ANALOGMONITORMETHOD_FLIP1080 or GS_NOT_SUPPORTED
        * <HR>
        */
        gsVidAnalogMonitorMethod,
        /**
        * Select the method for upconverting to HD
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_UPCONVERT_ANAMORPHIC,
#GS_UPCONVERT_PILLARBOX,
        *    #GS_UPCONVERT_ZOOM14x9, #GS_UPCONVERT_LETTERBOX,
#GS_UPCONVERT_ZOOMWIDE
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_UPCONVERT_ANAMORPHIC,
#GS_UPCONVERT_PILLARBOX,
        *    #GS_UPCONVERT_ZOOM14x9, #GS_UPCONVERT_LETTERBOX,
#GS_UPCONVERT_ZOOMWIDE
        * <HR>
        */
        gsVidAnalogMonitorUpMode,
        /**
        * Select the method for downconverting to SD
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_DOWNCONVERT_LETTERBOX,
#GS_DOWNCONVERT_CROP, #GS_DOWNCONVERT_ANAMORPHIC
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_DOWNCONVERT_LETTERBOX,
#GS_DOWNCONVERT_CROP, #GS_DOWNCONVERT_ANAMORPHIC
        * <HR>
        */
        gsVidAnalogMonitorDownMode,

        /**
        * Set/Get the current pan scan pos and zoom
        * <BR>
```

```
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - when to do it (0xFFFFFFFF == immediate)
* <BR>MEDIACMD::dwStart - X (upper bit 0x80000000 is FLIP)
* <BR>MEDIACMD::dwEnd - Y (upper bit 0x80000000 is FLIP)
* <BR>MEDIACMD::dwSpeed - Z
* <BR>MEDIACMD::dwAudioChannels -  X Aspect
* <BR>MEDIACMD::dwVideoChannels -  Y Aspect
* <BR>MEDIACMD::dwInfoChannels - Rotate
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition - when to do it
* <BR>MEDIACMD::dwStart - X (upper bit 0x80000000 is FLIP)
* <BR>MEDIACMD::dwEnd - Y (upper bit 0x80000000 is FLIP)
* <BR>MEDIACMD::dwSpeed - Z
* <BR>MEDIACMD::dwAudioChannels -  X Aspect
* <BR>MEDIACMD::dwVideoChannels -  Y Aspect
* <BR>MEDIACMD::dwInfoChannels - Rotate
* <HR>
*/
gsVidPanScanZoom,
/**
* Slow motion mode - 'use field duplication'
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - 1 enabled, 0 disabled
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition - 1 enabled, 0 disabled
* <BR>  MEDIACMD::dwStart - 1 if available
* <HR>
*/
gsVidSlowMotionMode,
/**
* Varicam record/playback mode
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - GS_FRAMEDROPMODE_VARICAM_MASK_FPS
* <BR>MEDIACMD::dwPosition - GS_FRAMEDROPMODE_VARICAM_MASK_FPS
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition - GS_FRAMEDROPMODE_VARICAM_MASK_FPS
* <BR>  MEDIACMD::dwStart - 1 if available
* <HR>
*/
gsVidVariCamMode,
/**
* Add custom superimpose elements
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - 0 = clear all custom elements,<br>
*                            1 = Text element
*                                                              2 = Line
```

```
        *                                                       3 = Box (line)
        *                                                       4 = Rectangle (filled)
        *                                                       5 = Circle (line)
        *                                                       6 = Circle (filled)
        *                                                       7 = Oval
        *                                                       8 = Oval (filled)
        * <BR>MEDIACMD::dwStart - X position
        * <BR>MEDIACMD::dwEnd -  Y position
        * <BR>  MEDIACMD::lSpeed - Modifier (size for text)
        * <BR>MEDIACMD::dwVideoChannels - Width (not for text)
        * <BR>MEDIACMD::dwAudioChannels - Height (not for text)
        * <BR>MEDIACMD::dwInfoChannels - Color
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Number of custom elements, 0 = no custom
superimpose
        * <BR>MEDIACMD::dwStart - Available width of frame
        * <BR>MEDIACMD::dwEnd -  Available height of frame
        * <BR>MEDIACMD::dwVideoChannels -  Width of text element
        * <BR>MEDIACMD::dwAudioChannels -  Height of text element
        * <HR>
        */
        gsVidCustomSuperimpose,                 // Super tc/state data on monitor output
(dwPosition = channel, dwStart = available channels)

        // Video settings use dwVideoChannels
        /**
        * Select video input
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Video input to use
#GS_VIDSELECT_COMPOSITE, #GS_VIDSELECT_COMPOSITE_2,
#GS_VIDSELECT_SVIDEO,
        * #GS_VIDSELECT_COMPONENT_YUV,
#GS_VIDSELECT_COMPONENT_YUV_M2, #GS_VIDSELECT_COMPONENT_YUV_SMPTE,
#GS_VIDSELECT_COMPONENT_RGB,
        * #GS_VIDSELECT_D1_SERIAL, #GS_VIDSELECT_D1_PARALLEL,
#GS_VIDSELECT_SDTI, #GS_VIDSELECT_NONE
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Video input to use
#GS_VIDSELECT_COMPOSITE, #GS_VIDSELECT_COMPOSITE_2,
#GS_VIDSELECT_SVIDEO,
        * #GS_VIDSELECT_COMPONENT_YUV,
#GS_VIDSELECT_COMPONENT_YUV_M2, #GS_VIDSELECT_COMPONENT_YUV_SMPTE,
#GS_VIDSELECT_COMPONENT_RGB,
        * #GS_VIDSELECT_D1_SERIAL, #GS_VIDSELECT_D1_PARALLEL,
#GS_VIDSELECT_SDTI, #GS_VIDSELECT_NONE
        * <BR>MEDIACMD::dwStart - Supported video inputs (bit array using defines
from dwPosition)
        * <HR>
```

```
        */
        gsVidInSelect = 400,    // Select video input source (dwPosition, supported =
dwStart)
        /**
        * Select video input genlock type
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Video input lock type use
#GS_VIDLOCKTYPE_VTR or #GS_VIDLOCKTYPE_BROADCAST
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Video input lock type use
#GS_VIDLOCKTYPE_VTR or #GS_VIDLOCKTYPE_BROADCAST
        * <BR>MEDIACMD::dwStart - Supported video inputs (bit array using defines
from dwPosition)
        * <HR>
        */
        gsVidInLockType,                // Input channel lock type (1-Broadcast, 0-VTR)
        /**
        * Input TBC - Setup (~brightness) Normal range: 0-65535 (0x0000-0xffff)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC Setup
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC Setup
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidInSetup,                   // Input channel Setup (16 bit unsigned)
(dwPosition, min=dwStart, max=dwEnd)
        /**
        * Input TBC - Video (~contrast) Normal range: 0-65535 (0x0000-0xffff)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC Video
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC Video
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidInVideo,                   // Input channel Video (16 bit unsigned)
(dwPosition, min=dwStart, max=dwEnd)
        /**
        * Input TBC - Hue (~color angle) degrees * 182.  Normal range: 0-65520
(0x0000-0xfff0)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC Hue
```

```
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC Hue
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65520)
        * <HR>
        */
        gsVidInHue,                             // Input channel Hue (16 bit unsigned)
(dwPosition, min=dwStart, max=dwEnd)
        /**
        * Input TBC - Chroma (~saturation) Normal range: 0-65535 (0x0000-0xffff)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC Chroma
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC Chroma
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidInChroma,                  // Input channel Chroma (16 bit unsigned)
(dwPosition, min=dwStart, max=dwEnd)
        /**
        * Input TBC - U Chroma or Cb or Y'CrCb Normal range: 0-65535 (0x0000-0xffff)
        * Normally only affects the component video or D1 Serial inputs.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC U (Cb) Chroma
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC U (Cb) Chroma
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidInUChroma,                         // Input channel U Component Chroma
(16 bit unsigned) (dwPosition, min=dwStart, max=dwEnd)
        /**
        * Input TBC - V Chroma or Cr or Y'CrCb Normal range: 0-65535 (0x0000-0xffff)
        * Normally only affects the component video or D1 Serial inputs.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC V (Cr) Chroma
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Video input TBC V (Cr) Chroma
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidInVChroma,                         // Input channel V Component Chroma
(16 bit unsigned) (dwPosition, min=dwStart, max=dwEnd)
```

```
        /**
        * Remove color from input signal (black and white luminance data only)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - If 1, signal will have no chroma, if 0, normal
signal
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - If 1, signal will have no chroma, if 0, normal
signal
        * <HR>
        */
        gsVidInColorKiller,                 // Kill color on input (dwPosition DWORD)
        /**
        * Automatic gain control
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - If 1, signal adjust gain automatically, if 0, will
us cmdGetSetValue::gsVidInSetup and cmdGetSetValue::gsVidInVideo
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - If 1, signal adjust gain automatically, if 0, will
us cmdGetSetValue::gsVidInSetup and cmdGetSetValue::gsVidInVideo
        * <HR>
        */
        gsVidInAGC,                                 // Input channel automatic gain control
(dwPosition DWORD)
        /**
        * Maximum input bandwidth setting
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDBAND_STANDARD,
#GS_VIDBAND_MEDIUM, #GS_VIDBAND_HIGH, #GS_VIDBAND_NOTCH
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDBAND_STANDARD,
#GS_VIDBAND_MEDIUM, #GS_VIDBAND_HIGH, #GS_VIDBAND_NOTCH
        * <BR>MEDIACMD::dwStart - Bit array of allowable values as defined for
dwPosition above
        * <HR>
        */
        gsVidInBandwidth,                   // Signal bandwidth (dwPosition, supported =
dwStart)
        /**
        * Black type (NTSC only)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDBLACK_SETUP,
#GS_VIDBLACK_CRYSTAL, #GS_VIDBLACK_SUPER
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDBLACK_SETUP,
#GS_VIDBLACK_CRYSTAL, #GS_VIDBLACK_SUPER
```

```
        * <BR>MEDIACMD::dwStart - Bit array of allowable values as defined for
dwPosition above
        * <HR>
        */
        gsVidInBlack,                   // Black level (dwPosition, supported = dwStart)
        /**
        * White type (NTSC only)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDWHITE_CLAMP,
#GS_VIDWHITE_SCALE, #GS_VIDWHITE_FREE
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDWHITE_CLAMP,
#GS_VIDWHITE_SCALE, #GS_VIDWHITE_FREE
        * <BR>MEDIACMD::dwStart - Bit array of allowable values as defined for
dwPosition above
        * <HR>
        */
        gsVidInWhite,                   // Max white (dwPosition, supported = dwStart)
        /**
        * Input digital signal coring.  Removal of low order bits to remove DAC aliasing
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Remove bottom 0, 1 or 2 bits of digitized signal
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Remove bottom 0, 1 or 2 bits of digitized signal
        * <BR>MEDIACMD::dwStart - Bit array of allowable values as defined for
dwPosition above (0 always supported)
        * <HR>
        */
        gsVidInCoring,                  // Input channel coring 0, 1 or 2 bits (dwPosition,
supported = dwStart)
        /**
        * Remove (smooth) 100% signal spikes
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 0 leave signal intact, 1 smooth
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 0 leave signal intact, 1 smooth
        * <BR>MEDIACMD::dwStart - Bit array of allowable values as defined for
dwPosition above (0 always supported)
        * <HR>
        */
        gsVidInPeaking,                 // (dwPosition, supported = dwStart)
        /**
        * Set video transition sharpness
        * <BR>
        * \li cmdType::ctSetValue
```

```
        * <BR>MEDIACMD::dwPosition - Depends on cmdType::ctGetValue-
MEDIACMD::dwStart->MEDIACMD::dwEnd (typically 0-7, 0-100, 0-65535)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Video digitizing sharpness
        * <BR>MEDIACMD::dwStart - Lowest possible sharpness
        * <BR>MEDIACMD::dwEnd - Highest possible sharpness
        * <HR>
        */
        gsVidInSharpness,                   // (dwPosition, min=dwStart, max=dwEnd)
        /**
        * Set video gamma curve
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Depends on cmdType::ctGetValue-
MEDIACMD::dwStart->MEDIACMD::dwEnd (typically -32768->+32768)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Gamma curve weighting or offset
        * <BR>MEDIACMD::dwStart - Lowest possible sharpness
        * <BR>MEDIACMD::dwEnd - Highest possible sharpness
        * <HR>
        */
        gsVidInGamma,                       // (dwPosition, min=dwStart, max=dwEnd)
        /**
        * Video input signal format.  May be incorrect depending on some hardware
setups.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>- Not supported, please use #gsSignalFormat to set channel format to
match input
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_SIGFORM_NTSC #GS_SIGFORM_PAL
        * #GS_SIGFORM_CCIR_NTSC #GS_SIGFORM_CCIR_PAL
        * #GS_SIGFORM_1035i_30_260M #GS_SIGFORM_1035i_30X_260M
        * #GS_SIGFORM_1080i_30 #GS_SIGFORM_1080i_30X #GS_SIGFORM_1080i_25
#GS_SIGFORM_1080i_24 #GS_SIGFORM_1080i_24X
        * #GS_SIGFORM_1080_30 #GS_SIGFORM_1080_30X #GS_SIGFORM_1080_25
#GS_SIGFORM_1080_24 #GS_SIGFORM_1080_24X
        * #GS_SIGFORM_720_60 #GS_SIGFORM_720_60X
#GS_SIGFORM_NOT_PRESENT
        * <HR>
        */
        gsVidInSignalFormat,    // Input signal format (-1 for not present or bad setup)
        /**
        * Set video transition sharpness
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Depends on cmdType::ctGetValue-
MEDIACMD::dwStart->MEDIACMD::dwEnd (typically 0-100)
        * \li cmdType::ctGetValue
```

```
        * <BR>MEDIACMD::dwPosition - quality level
        * <BR>MEDIACMD::dwStart - Lowest possible quality from codec
        * <BR>MEDIACMD::dwEnd - Highest possible quality from codec
        * <HR>
        */
        gsVidInQuality,
        /**
        * Main TBC - Setup (~brightness) Normal range: 0-65535 (0x0000-0xffff)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - TBC Setup
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - TBC Setup
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidSetup = 500,                // Video 'Setup' (16 bit signed) (dwPosition,
min=dwStart, max=dwEnd)
        /**
        * Main TBC - Video (~contrast) Normal range: 0-65535 (0x0000-0xffff)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - TBC Video
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - TBC Video
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidVideo,                              // Video 'Video' (16 bit signed)
(dwPosition, min=dwStart, max=dwEnd)
        /**
        * Main TBC - Hue (~color angle) degrees * 182.  Normal range: 0-65520
(0x0000-0xfff0)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - TBC Hue
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Hue
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65520)
        * <HR>
        */
        gsVidHue,                                // Video 'Hue' (16 bit signed)
(dwPosition, min=dwStart, max=dwEnd)
        /**
        * Main TBC - Chroma (~saturation) Normal range: 0-65535 (0x0000-0xffff)
        * <BR>
```

```
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - TBC Chroma
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - TBC Chroma
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidChroma,                    // Video 'Chroma' (16 bit signed) (dwPosition,
min=dwStart, max=dwEnd)
        /**
        * Main TBC - U Chroma or Cb or Y'CrCb Normal range: 0-65535 (0x0000-0xffff)
        * Normally only affects the component video or D1 Serial paths.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - TBC U (Cb) Chroma
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - TBC U (Cb) Chroma
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidUChroma,                   // Input channel U Component Chroma (16 bit
signed) (dwPosition, min=dwStart, max=dwEnd)
        /**
        * Main TBC - V Chroma or Cr or Y'CrCb Normal range: 0-65535 (0x0000-0xffff)
        * Normally only affects the component video or D1 Serial paths.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - TBC V (Cr) Chroma
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - TBC V (Cr) Chroma
        * <BR>MEDIACMD::dwStart - Lowest possible value (usually 0)
        * <BR>MEDIACMD::dwEnd - Highest possible value (usually 65535)
        * <HR>
        */
        gsVidVChroma,                   // Input channel V Component Chroma (16 bit
signed) (dwPosition, min=dwStart, max=dwEnd)
        /**
        * Maximum channel bandwidth setting
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDBAND_STANDARD,
#GS_VIDBAND_MEDIUM, #GS_VIDBAND_HIGH, #GS_VIDBAND_NOTCH
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDBAND_STANDARD,
#GS_VIDBAND_MEDIUM, #GS_VIDBAND_HIGH, #GS_VIDBAND_NOTCH
        * <BR>MEDIACMD::dwStart - Bit array of allowable values as defined for
dwPosition above
```

```
        * <HR>
        */
        gsVidBandwidth,                         // Signal bandwidth SEE
gsVidInBandwidth (dwPosition, supported=dwStart)
        /**
        * Black type (NTSC only)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDBLACK_SETUP,
#GS_VIDBLACK_CRYSTAL, #GS_VIDBLACK_SUPER
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Uses #GS_VIDBLACK_SETUP,
#GS_VIDBLACK_CRYSTAL, #GS_VIDBLACK_SUPER
        * <BR>MEDIACMD::dwStart - Bit array of allowable values as defined for
dwPosition above
        * <HR>
        */
        gsVidBlackSetup,                // Super black, Crystal Black, NTSC Setup SEE
gsVidInBlack (dwPosition, supported=dwStart)
        /**
        * Remove color from signal path (black and white luminance data only)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - If 1, signal will have no chroma, if 0, normal
signal
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - If 1, signal will have no chroma, if 0, normal
signal
        * <HR>
        */
        gsVidColor,                             // Color signal or black and white
(dwPosition)

        /**
        * Select video output
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Video output to use
#GS_VIDSELECT_COMPOSITE, #GS_VIDSELECT_COMPOSITE_2,
#GS_VIDSELECT_SVIDEO,
        * #GS_VIDSELECT_COMPONENT_YUV,
#GS_VIDSELECT_COMPONENT_YUV_M2, #GS_VIDSELECT_COMPONENT_YUV_SMPTE,
#GS_VIDSELECT_COMPONENT_RGB,
        * #GS_VIDSELECT_D1_SERIAL, #GS_VIDSELECT_D1_PARALLEL,
#GS_VIDSELECT_SDTI, #GS_VIDSELECT_NONE
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current video output
#GS_VIDSELECT_COMPOSITE, #GS_VIDSELECT_COMPOSITE_2,
#GS_VIDSELECT_SVIDEO,
```

```
        * #GS_VIDSELECT_COMPONENT_YUV,
#GS_VIDSELECT_COMPONENT_YUV_M2, #GS_VIDSELECT_COMPONENT_YUV_SMPTE,
#GS_VIDSELECT_COMPONENT_RGB,
        * 7#GS_VIDSELECT_D1_SERIAL, #GS_VIDSELECT_D1_PARALLEL,
#GS_VIDSELECT_SDTI, #GS_VIDSELECT_NONE
        * <BR>MEDIACMD::dwStart - Supported video inputs (bit array using defines
from dwPosition)
        * <HR>
        */
        gsVidOutSelect = 600,  // Select main out (See gsVidInSelect)
        /**
        * Enable genlock (video black timing signal)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 1 using external ref genlock, 0 free running on
internal clock
        * (see gsGetSetCmdValue::gsVidOutGenlockSource)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 1 using external ref genlock, 0 free running on
internal clock
        * (see gsGetSetCmdValue::gsVidOutGenlockSource)
        * <BR>MEDIACMD::dwStart - If 1, external genlock supported
        * <HR>
        */
        gsVidOutGenlock,                    // Genlock enable (dwPosition,
dwStart=supported sources)
        /**
        * Select genlock (video black timing signal) source
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Genlock source to use #GS_LOCKSRC_NONE,
#GS_LOCKSRC_EXTIN,
        * #GS_LOCKSRC_INPUT, #GS_LOCKSRC_CVBS (composite video),
#GS_LOCKSRC_SVIDEO (svhs),
        * #GS_LOCKSRC_IN_Y (y of component in), #GS_LOCKSRC_SDI (D1 Digital In)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Genlock source to use #GS_LOCKSRC_NONE,
#GS_LOCKSRC_EXTIN,
        * #GS_LOCKSRC_INPUT, #GS_LOCKSRC_CVBS (composite video),
#GS_LOCKSRC_SVIDEO (svhs),
        * #GS_LOCKSRC_IN_Y (y of component in), #GS_LOCKSRC_SDI (D1 Digital In)
        * <BR>MEDIACMD::dwStart - Supported genlock inputs (bit array using defines
from dwPosition)
        * <HR>
        */
        gsVidOutGenlockSource,
        /**
        * Select genlock type (quality)
        * <BR>
```

```
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Genlock lock type use #GS_VIDLOCKTYPE_VTR
or #GS_VIDLOCKTYPE_BROADCAST
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Genlock lock type use #GS_VIDLOCKTYPE_VTR
or #GS_VIDLOCKTYPE_BROADCAST
        * <BR>MEDIACMD::dwStart - Supported video inputs (bit array using defines
from dwPosition)
        * <HR>
        */
        gsVidOutLockType,               // Genlock lock type (0-Broadcast, 1-VTR)
(dwPosition )
        /**
        * Horizontal output phase
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Depends on cmdType::ctGetValue-
MEDIACMD::dwStart->MEDIACMD::dwEnd (typically 0->65535 or -32768->32768)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Horizontal phase setting
        * <BR>MEDIACMD::dwStart - Lowest possible horizontal phase
        * <BR>MEDIACMD::dwEnd - Highest possible horizontal phase
        * <HR>
        */
        gsVidOutHPhase,                         // Horizontal Phase (16 bit signed)
(dwPosition, min=dwStart, max=dwEnd)
        /**
        * Video genlock subcarrier phase timing
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Depends on cmdType::ctGetValue-
MEDIACMD::dwStart->MEDIACMD::dwEnd (typically 0->65520 == degrees * 182)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Sub carrier phase setting
        * <BR>MEDIACMD::dwStart - Lowest possible sub carrier phase
        * <BR>MEDIACMD::dwEnd - Highest possible sub carrier phase
        * <HR>
        */
        gsVidOutSubCarrier,             // Sub Carrier Phase (16 bit signed) (dwPosition,
min=dwStart, max=dwEnd)
        /**
        * Digital output signal coring.  Removal of low order bits to remove DAC aliasing
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Remove bottom 0, 1 or 2 bits of digitized signal
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Remove bottom 0, 1 or 2 bits of digitized signal
        * <BR>MEDIACMD::dwStart - Bit array of allowable values as defined for
dwPosition above (0 always supported)
```

```
        * <HR>
        */
        gsVidOutCoring,                         // Core 0, 1 or 2 bits from signal (0, 1, 2)
(dwPosition, supported=dwStart)
        /**
        * Remove (smooth) 100% signal spikes on output
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 0 leave signal intact, 1 smooth
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 0 leave signal intact, 1 smooth
        * <BR>MEDIACMD::dwStart - Bit array of allowable values as defined for
dwPosition above (0 always supported)
        * <HR>
        */
        gsVidOutPeaking,                // Peaking (dwPosition, supported=dwStart)
        /**
        * Generic advanced adjustment 1 (hardware dependent)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Depends on cmdType::ctGetValue-
MEDIACMD::dwStart->MEDIACMD::dwEnd (typically 0->65535 or -32768->32768)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Adjust 1 setting
        * <BR>MEDIACMD::dwStart - Lowest possible adjust 1 setting
        * <BR>MEDIACMD::dwEnd - Highest possible adjust 1 setting
        * <HR>
        */
        gsVidOutAdjust1,                //
        /**
        * Generic advanced adjustment 2 (hardware dependent)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Depends on cmdType::ctGetValue-
MEDIACMD::dwStart->MEDIACMD::dwEnd (typically 0->65535 or -32768->32768)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Adjust 2 setting
        * <BR>MEDIACMD::dwStart - Lowest possible adjust 2 setting
        * <BR>MEDIACMD::dwEnd - Highest possible adjust 2 setting
        * <HR>
        */
        gsVidOutAdjust2,                //
        /**
        * Genlock output delay (not currently used)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Depends on cmdType::ctGetValue-
MEDIACMD::dwStart->MEDIACMD::dwEnd (typically 0->65535 or -32768->32768)
        * \li cmdType::ctGetValue
```

```
        * <BR>MEDIACMD::dwPosition - Genlock timing delay
        * <BR>MEDIACMD::dwStart - Lowest possible delay
        * <BR>MEDIACMD::dwEnd - Highest possible delay
        * <HR>
        */
        gsVidOutGenlockDelay, //
        /**
        * Video output genlock input signal format.  May be incorrect depending on some
hardware setups.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>- Not supported, please use #gsSignalFormat to set channel format to
match input
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_SIGFORM_NTSC #GS_SIGFORM_PAL
        * #GS_SIGFORM_CCIR_NTSC #GS_SIGFORM_CCIR_PAL
        * #GS_SIGFORM_1035i_30_260M #GS_SIGFORM_1035i_30X_260M
        * #GS_SIGFORM_1080i_30 #GS_SIGFORM_1080i_30X #GS_SIGFORM_1080i_25
#GS_SIGFORM_1080i_24 #GS_SIGFORM_1080i_24X
        * #GS_SIGFORM_1080_30 #GS_SIGFORM_1080_30X #GS_SIGFORM_1080_25
#GS_SIGFORM_1080_24 #GS_SIGFORM_1080_24X
        * #GS_SIGFORM_720_60 #GS_SIGFORM_720_60X
#GS_SIGFORM_NOT_PRESENT
        * <HR>
        */
        gsVidOutLockSignalFormat,       // Genlock input signal format (-1 for not present
or bad setup)
        /**
        * When video input is in DualLink, the out switches to dual link to.  If this is set,
then
        * the output will stay in single link and convert the dual link 4:4:4 to 4:2:2
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_TRUE, #GS_FALSE
        * match input
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_TRUE, #GS_FALSE
        * <HR>
        */
        gsVidOutDisableDualLink,        // Disable dual link out when in dual link in
        /**
        * Set the output of the Kona to show a wipe or dissolve against the current
frame
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 0=dissolve, 1=Wipe
        * <BR>MEDIACMD::dwStart - Wipe Type,
0=horiz,1=vert,2=upperright,3=upperleft
```

```
        * <BR>MEDIACMD::dwEnd - Wipe amount, 0..65535 (0..100%) where it is a
percentage of stored frame (e.g. 0=showinput,65535=showframe)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 0=dissolve, 1=Wipe
        * <HR>
        */
        gsVidOutReferenceWipeMix,               // Wipe or Mix between the input and the
frame on disk
        /**
        *       Disable genlocking of board to allow out->in connection for capture of
output
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 1 to disable, 0 default
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 1 to disable, 0 default
        * <HR>
        */
        gsDisableGenlockForInfiniteLoop,


        // Compression and internal signal parameters
        /**
        *       Size of picture   Y (Vertical)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Vertical size of video frame
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Vertical size of video frame
        * <HR>
        */
        gsCompChVerticalRes = 700,
        //! Alias for #gsCompChVerticalRes
        gsMpegVerticalRes = gsCompChVerticalRes,
        /**
        *       Size of picture   X (Horizontal)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Horizontal size of video frame
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Horizontal size of video frame
        * <HR>
        */
        gsCompChHorizontalRes,
        //! Alias for #gsCompChHorizontalRes
        gsMpegHorizontalRes = gsCompChHorizontalRes,
        /**
        *       Chroma type 4:0:0, 4:2:0, 4:2:2, 4:4:4
        * <BR>
```

```
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - One of #GS_MPEG_CHROMA_FORMAT_420,
        * #GS_MPEG_CHROMA_FORMAT_422, #GS_MPEG_CHROMA_FORMAT_444
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - One of #GS_MPEG_CHROMA_FORMAT_420,
        * #GS_MPEG_CHROMA_FORMAT_422, #GS_MPEG_CHROMA_FORMAT_444
        * <HR>
        */
        gsCompChChromaFormat,
        //! Alias for #gsCompChChromaFormat
        gsMpegChromaFormat = gsCompChChromaFormat,
        /**
        *       DC Precision (mostly MPEG) 8..12
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - #GS_MPEG_DC_PRECISION_8,
#GS_MPEG_DC_PRECISION_9,
        * #GS_MPEG_DC_PRECISION_10, #GS_MPEG_DC_PRECISION_11
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - #GS_MPEG_DC_PRECISION_8,
#GS_MPEG_DC_PRECISION_9,
        * #GS_MPEG_DC_PRECISION_10, #GS_MPEG_DC_PRECISION_11
        * <HR>
        */
        gsCompChDCPrecision,
        //! Alias for #gsCompChDCPrecision
        gsMpegDCPrecision = gsCompChDCPrecision,
        /**
        *       Video signal aspect ratio
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - #GS_ASPECT_RATIO_SQUARE,
#GS_ASPECT_RATIO_4x3,
        * #GS_ASPECT_RATIO_16x9, #GS_ASPECT_RATIO_2_21x1
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - #GS_ASPECT_RATIO_SQUARE,
#GS_ASPECT_RATIO_4x3,
        * #GS_ASPECT_RATIO_16x9, #GS_ASPECT_RATIO_2_21x1
        * <HR>
        */
        gsCompChAspectRatio,
        //! Alias for #gsCompChAspectRatio
        gsMpegAspectRatio = gsCompChAspectRatio,
        /**
        *       MPEG file stream standard
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - #GS_MPEG_STANDARD_SYSTEM,
#GS_MPEG_STANDARD_PROGRAM,
```

```
        * #GS_MPEG_STANDARD_TRANSPORT, #GS_MPEG_STANDARD_ELEMENTARY
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - #GS_MPEG_STANDARD_SYSTEM,
#GS_MPEG_STANDARD_PROGRAM,
        * #GS_MPEG_STANDARD_TRANSPORT, #GS_MPEG_STANDARD_ELEMENTARY
        * <HR>
        */
        gsCompChStandard,
        //! Alias for #gsCompChStandard
        gsMpegStandard = gsCompChStandard,
        /**
        *   Video/Audio Language Code
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - #GS_MPEG_LANGUAGE_ENGLISH, etc
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - #GS_MPEG_LANGUAGE_ENGLISH, etc
        * <HR>
        */
        gsCompChLanguageCode,
        //! Alias for #gsCompChLanguageCode
        gsMpegLanguageCode = gsCompChLanguageCode,
        /**
        *       Closed Captioning Format
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - #GS_MPEG_CC_FORMAT_CCUBE,
#GS_MPEG_CC_FORMAT_ATSC,
        * #GS_MPEG_CC_FORMAT_CCUBE_REORDER,
#GS_MPEG_CC_FORMAT_ATSC_REORDER
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - #GS_MPEG_CC_FORMAT_CCUBE,
#GS_MPEG_CC_FORMAT_ATSC,
        * #GS_MPEG_CC_FORMAT_CCUBE_REORDER,
#GS_MPEG_CC_FORMAT_ATSC_REORDER
        * <HR>
        */
        gsCompChCCFormat,
        //! Alias for #gsCompChCCFormat
        gsMpegCCFormat = gsCompChCCFormat,
        /**
        *       MPEG Concealment Vector
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - Not sure - Argus Encoder
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - Not sure - Argus Encoder
        * <HR>
        */
```

```
gsCompChConcealmentVector,
//! Alias for #gsCompChConcealmentVector
gsMpegConcealmentVector = gsCompChConcealmentVector,
/**
*       Set encoding to closed GOP or open GOP
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - 0 = open GOP, 1 = closed GOP
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition -  0 = open GOP, 1 = closed GOP
* <HR>
*/
gsCompChClosedGop,
//! Alias for #gsCompChClosedGop
gsMpegClosedGop = gsCompChClosedGop,
/**
*       Set the next GOP start time code value
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - Time code in frames (used def tctype)
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - Time code in frames (used def tctype)
* <HR>
*/
gsCompChAdjustGopTC,
//! Alias for #gsCompChAdjustGopTC
gsMpegAdjustGopTC = gsCompChAdjustGopTC,
/**
*       Set MPEG encoder to use alternate co-efficient tables
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - 1/0
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - 1/0
* <HR>
*/
gsCompChAltCoEffTable,
//! Alias for #gsCompChAltCoEffTable
gsMpegAltCoEffTable = gsCompChAltCoEffTable,
/**
*       Set encoder to use non linear quantization
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - 1/0
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - 1/0
* <HR>
*/
gsCompChNonLinearQuant,
```

```
//! Alias for #gsCompChNonLinearQuant
gsMpegNonLinearQuant = gsCompChNonLinearQuant,
/**
*       Set the multiplexer (overall) bit rate
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - Bits per second
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - Bits per second
* <HR>
*/
gsCompChMuxRate,
//! Alias for #gsCompChMuxRate
gsMpegMuxRate = gsCompChMuxRate,
/**
*       Audio packet size
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - Size of an audio packet in unsigned chars
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - Size of an audio packet in unsigned chars
* <HR>
*/
gsCompChAudPacketSize,
//! Alias for #gsCompChAudPacketSize
gsMpegAudPacketSize = gsCompChAudPacketSize,
/**
*       Video packet size
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - Size of a video packet in unsigned chars
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - Size of a video packet in unsigned chars
* <HR>
*/
gsCompChVidPacketSize,
//! Alias for #gsCompChVidPacketSize
gsMpegVidPacketSize = gsCompChVidPacketSize,
/**
*       Stream ID for AUDIO 0xc0 (0x1c0)
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - Audio Stream ID
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - Audio Stream ID
* <HR>
*/
gsCompChAudioStreamID,
//! Alias for #gsCompChVideoStreamID
```

```
gsMpegAudioStreamID = gsCompChAudioStreamID,
/**
*       Stream ID for VIDEO 0xe0 (0x1e0)
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - Video Stream ID
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - Video Stream ID
* <HR>
*/
gsCompChVideoStreamID,
//! Alias for #gsCompChVideoStreamID
gsMpegVideoStreamID = gsCompChVideoStreamID,
/**
*       Program ID of the video stream within a transport container
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - Video program id (PID)
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - Video program id (PID)
* <HR>
*/
gsCompChAudioStreamPID,
//! Alias for #gsCompChAudioStreamPID
gsMpegAudioStreamPID = gsCompChAudioStreamPID,
/**
*       Program ID of the audio stream within a transport container
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - Audio program id (PID)
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - Audio program id (PID)
* <HR>
*/
gsCompChVideoStreamPID,
//! Alias for #gsCompChVideoStreamPID
gsMpegVideoStreamPID = gsCompChVideoStreamPID,
/**
*       Allow settings to be changed.  Used to determine if settings
* can be changed (on the fly, without restart),
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - 0 disable changes, 1 allow changes
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - 0 disable changes, 1 allow changes
* <HR>
*/
gsCompChAllowSettings,
//! Alias for #gsCompChAllowSettings
```

```
gsMpegAllowSettings = gsCompChAllowSettings,
/**
* Fourcc code for compression.  Set one video or audio channel
* set to return fourcc code in dwPosition
* <BR>
* \li cmdType::ctSetValue - See fccDef.h for know fourcc
* <BR>
* \li cmdType::ctGetValue - See fccDef.h for know fourcc
* <HR>
*/
gsCompChFourCC,
/**
* bit count for compression 8 / 10 / 24 / 32
* set to return bit count in dwPosition
* <BR>
* \li cmdType::ctSetValue - See fccDef.h for know fourcc
* <BR>
* \li cmdType::ctGetValue - See fccDef.h for know fourcc
* <HR>
*/
gsCompChBitCount,
/**
* Size of each image in unsigned chars
* set to return fourcc code in dwPosition
* <BR>
* \li cmdType::ctSetValue - See fccDef.h for know fourcc
* <BR>
* \li cmdType::ctGetValue - See fccDef.h for know fourcc
* <HR>
*/
gsCompChSizeImage,
/**
* rate of frame
* set to return fourcc code in dwPosition
* <BR>
* \li cmdType::ctSetValue - See fccDef.h for know fourcc
* <BR>
* \li cmdType::ctGetValue - See fccDef.h for know fourcc
* <HR>
*/
gsCompChRate,
/**
* Scale for frame rate
* set to return fourcc code in dwPosition
* <BR>
* \li cmdType::ctSetValue - See fccDef.h for know fourcc
* <BR>
* \li cmdType::ctGetValue - See fccDef.h for know fourcc
* <HR>
```

```
        */
        gsCompChScale,
        /**
        * unsigned chars per video line
        * set to return fourcc code in dwPosition
        * <BR>
        * \li cmdType::ctSetValue - See fccDef.h for know fourcc
        * <BR>
        * \li cmdType::ctGetValue - See fccDef.h for know fourcc
        * <HR>
        */
        gsCompChPitch,
        /**
        * Encoding compression format i.e. avi mov dpx
        * set to return fourcc code in dwPosition
        * <BR>
        * \li cmdType::ctSetValue - See defines for file types
        * <BR> MEDIACMD::dwPosition - #VIDEOWRITETYPE_AVI,
#VIDEOWRITETYPE_MOV, #VIDEOWRITETYPE_WMV, #VIDEOWRITETYPE_GEN,
#VIDEOWRITETYPE_JS,
        *                       #VIDEOWRITETYPE_DVS, #VIDEOWRITETYPE_IHSS,
#VIDEOWRITETYPE_HDR, #VIDEOWRITETYPE_YUV, #VIDEOWRITETYPE_RAW,
        *                       #VIDEOWRITETYPE_TGA, #VIDEOWRITETYPE_BMP,
#VIDEOWRITETYPE_TIFF, #VIDEOWRITETYPE_DPX, #VIDEOWRITETYPE_AVCI_MXF
        *                       #VIDEOWRITETYPE_MPG, #VIDEOWRITETYPE_4224,
#VIDEOWRITETYPE_SONY_MXF, #VIDEOWRITETYPE_P2_MXF,
#VIDEOWRITETYPE_AVID_MXF
        *                       #VIDEOWRITETYPE_ARRI,
#VIDEOWRITETYPE_OP1a_MXF, #VIDEOWRITETYPE_DCP_MXF,
        *
#VIDEOWRITETYPE_TS, #VIDEOWRITETYPE_MP4, #VIDEOWRITETYPE_FLASH
        * <BR>
        * \li cmdType::ctGetValue - See defines for file types
        * <HR>
        * <BR> MEDIACMD::dwPosition - #VIDEOWRITETYPE_AVI,
#VIDEOWRITETYPE_MOV, #VIDEOWRITETYPE_WMV, #VIDEOWRITETYPE_GEN,
#VIDEOWRITETYPE_JS,
        *                       #VIDEOWRITETYPE_DVS, #VIDEOWRITETYPE_IHSS,
#VIDEOWRITETYPE_HDR, #VIDEOWRITETYPE_YUV, #VIDEOWRITETYPE_RAW,
        *                       #VIDEOWRITETYPE_TGA, #VIDEOWRITETYPE_BMP,
#VIDEOWRITETYPE_TIFF, #VIDEOWRITETYPE_DPX, #VIDEOWRITETYPE_AVCI_MXF
        *                       #VIDEOWRITETYPE_MPG, #VIDEOWRITETYPE_4224,
#VIDEOWRITETYPE_SONY_MXF, #VIDEOWRITETYPE_P2_MXF,
#VIDEOWRITETYPE_AVID_MXF
        *                       #VIDEOWRITETYPE_ARRI,
#VIDEOWRITETYPE_OP1a_MXF, #VIDEOWRITETYPE_DCP_MXF,
        *
#VIDEOWRITETYPE_TS, #VIDEOWRITETYPE_MP4, #VIDEOWRITETYPE_FLASH
        * <BR> MEDIACMD::dwStart - Bit array of available file formats
```

```
        */
        gsVideoEncodeFormat,
        /**
        * Encoding compression format i.e. WAV, AIFF
        * set to return fourcc code in dwPosition
        * <BR>
        * \li cmdType::ctSetValue - See defines for file types
        * <BR> MEDIACMD::dwPosition - The file type and the internal format<br>
        * #AUDIOWRITETYPE_INTERNAL, #AUDIOWRITETYPE_WAVE,
#AUDIOWRITETYPE_AIFF
        * #AUDIOWRITETYPE_MONO, #AUDIOWRITETYPE_STEREO,
#AUDIOWRITETYPE_MULTI
        * <BR>
        * \li cmdType::ctGetValue - See fccDef.h for know fourcc
        * <BR> MEDIACMD::dwPosition - The file type and the internal format<br>
        * #AUDIOWRITETYPE_INTERNAL, #AUDIOWRITETYPE_WAVE,
#AUDIOWRITETYPE_AIFF
        * #AUDIOWRITETYPE_MONO, #AUDIOWRITETYPE_STEREO,
#AUDIOWRITETYPE_MULTI
        * <BR> MEDIACMD::dwStart - Bit array of available file formats
        * <HR>
        */
        gsAudioEncodeFormat,
        /**
        * last compression change ms for updating the clip bin
        * set to return fourcc code in dwPosition
        * <BR>
        * \li cmdType::ctSetValue - See fccDef.h for know fourcc
        * <BR>
        * \li cmdType::ctGetValue - See fccDef.h for know fourcc
        * <HR>
        */
        gsCompChannelChangeMs,
        /**
        * single link, dual link or alpha
        * set to return fourcc code in dwPosition
        * <BR>
        * \li cmdType::ctSetValue - See fccDef.h for know fourcc
        * <BR>
        * \li cmdType::ctGetValue - See fccDef.h for know fourcc
        * <HR>
        */
        gsAlphaChromaSource,
        /**
        * RGBA BGRA ycbcr
        * set to return fourcc code in dwPosition
        * <BR>
        * \li cmdType::ctSetValue - See fccDef.h for know fourcc
        * <BR>
```

```
        * \li cmdType::ctGetValue - See fccDef.h for know fourcc
        * <HR>
        */
        gsCompressionType,
        /**
        * <BR>
        * \li cmdType::ctSetValue -
        * <BR>
        * \li cmdType::ctGetValue -
        * <HR>
        */
        gsVideoStandard,
        /**
        * Reset the channel to the new setup
        * <BR>
        * \li cmdType::ctSetValue -
        * <BR>
        * \li cmdType::ctGetValue -
        * <HR>
        */
        gsResetChannel,
        /**
        * <BR>
        * \li cmdType::ctSetValue -
        * <BR>
        * \li cmdType::ctGetValue -
        * <HR>
        */
        gsEnableHDSDFormat,
        /**
        * Enable capture of vertical blank?
        * <BR>
        * \li cmdType::ctSetValue -
        * <BR> MEDIACMD::dwPosition - 1 (bit_0) capture vertical blank, 2 (bit_1) save
vertical blank
        * <BR>
        * \li cmdType::ctGetValue -
        * <BR> MEDIACMD::dwPosition - 1 (bit_0) capture vertical blank, 2 (bit_1) save
vertical blank
        * <HR>
        */
        gsVBlankEnable,
        /**
        * Enable/Disable LUTs
        * <BR>
        * \li cmdType::ctSetValue -
        * <BR> MEDIACMD::dwPosition - 1 (bit_0) play enable, 2 (bit_1) linear / log if
not set, 4 (bit_2) record enable
        * <BR>
```

```
            * \li cmdType::ctGetValue -
            * <BR> MEDIACMD::dwPosition - 1 (bit_0) play enable, 2 (bit_1) linear / log if
not set, 4 (bit_2) record enable
            * <HR>
            */
            gsLUTEnable,
            /**
            *
            * <BR>
            * \li cmdType::ctSetValue -
            * <BR>
            * \li cmdType::ctGetValue -
            * <HR>
            */
            gsAudioFileType,
            /**
            *
            * <BR>
            * \li cmdType::ctSetValue -
            * <BR>
            * \li cmdType::ctGetValue -
            * <HR>
            */
            gsAudioBitSize,
            /**
            *
            * <BR>
            * \li cmdType::ctSetValue -
            * <BR>
            * \li cmdType::ctGetValue -
            * <HR>
            */
            gsAudioFrequency,
            /**
            * Enabled/disable/ mediafile overlapped writes
            * <BR>
            * \li cmdType::ctGetValue
            * <BR>MEDIACMD::dwPosition  - Enabled 0,1
            * \li cmdType::ctSetValue
            * <BR>MEDIACMD::dwPosition  - Enabled 0,1
            * <HR>
            */
            gsEnableOverlappedWrites,
            /**
            * Match video output to current clip settings
            * <BR>
            * \li cmdType::ctGetValue
            * <BR>MEDIACMD::dwPosition  - Enabled 0,1
            * \li cmdType::ctSetValue
```

```
        * <BR>MEDIACMD::dwPosition  - Enabled 0,1
        * <HR>
        */
        gsMatchOutputToClip,
        /**
        * Allow each channel to be configure separately for file type, bit depth, etc.
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Enabled 0,1
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Enabled 0,1
        * <HR>
        */
        gsAllowIndependentChanConfig,
#define gsAllowIndependantChanConfig          gsAllowIndependentChanConfig
        /**
        * Channel Compression format
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition -
        * #GS_SIGFORM_CCIR_NTSC #GS_SIGFORM_CCIR_PAL
        * #GS_SIGFORM_1035i_30_260M #GS_SIGFORM_1035i_30X_260M
        * #GS_SIGFORM_1080i_30 #GS_SIGFORM_1080i_30X #GS_SIGFORM_1080i_25
#GS_SIGFORM_1080i_24 #GS_SIGFORM_1080i_24X
        * #GS_SIGFORM_1080_30 #GS_SIGFORM_1080_30X #GS_SIGFORM_1080_25
#GS_SIGFORM_1080_24 #GS_SIGFORM_1080_24X
        * #GS_SIGFORM_720_60 #GS_SIGFORM_720_60X #GS_SIGFORM_CUSTOM
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_SIGFORM_NTSC #GS_SIGFORM_PAL
        * #GS_SIGFORM_CCIR_NTSC #GS_SIGFORM_CCIR_PAL
        * #GS_SIGFORM_1035i_30_260M #GS_SIGFORM_1035i_30X_260M
        * #GS_SIGFORM_1080i_30 #GS_SIGFORM_1080i_30X #GS_SIGFORM_1080i_25
#GS_SIGFORM_1080i_24 #GS_SIGFORM_1080i_24X
        * #GS_SIGFORM_1080_30 #GS_SIGFORM_1080_30X #GS_SIGFORM_1080_25
#GS_SIGFORM_1080_24 #GS_SIGFORM_1080_24X
        * #GS_SIGFORM_720_60 #GS_SIGFORM_720_60X #GS_SIGFORM_CUSTOM
        * <BR>MEDIACMD::dwStart - Bit array of supported types
        * <HR>
        */

        gsSignalFormat = 900,  // NTSC CCIR HD 16x9 etc (dwPosition,
supported=dwStart)

        /**
        * Channel Compression format/type
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition -
```

```
        * <BR>#GS_COMPTYPE_SOFTWARE    Software passed codec on main
processor
        * <BR>#GS_COMPTYPE_MJPEG Motion JPEG hardware codec (LSI, Zoran, C-
Cube, etc)
        * <BR>#GS_COMPTYPE_WAVELET Wavelet hardware codec
        * <BR>#GS_COMPTYPE_MPEG1 MPEG 1 hardware compatible codec
        * <BR>#GS_COMPTYPE_MPEG2 MPEG 2 hardware compatible codec
        * <BR>#GS_COMPTYPE_MPEG2I Editable MPEG 2 I Frame Only compatible
codec
        * <BR>#GS_COMPTYPE_MPEG2IBP      MPEG 2 long GOP hardware compatible
codec
        * <BR>#GS_COMPTYPE_DV25 Hardware DV25, DVCPRO, DVCPRO25
        * <BR>#GS_COMPTYPE_DV50 Hardware DV50, DVCPRO50
        * <BR>#GS_COMPTYPE_DVSD Hardware Standard DV Bluebook, DVPRO, DVSD
        * <BR>#GS_COMPTYPE_DV100 High Def DV codec
        * <BR>#GS_COMPTYPE_HDPAN Panasonic HD to SDI codec
        * <BR>#GS_COMPTYPE_HDSONY Sony HD to SDI codec
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition -
        * <BR>#GS_COMPTYPE_SOFTWARE    Software passed codec on main
processor
        * <BR>#GS_COMPTYPE_MJPEG Motion JPEG hardware codec (LSI, Zoran, C-
Cube, etc)
        * <BR>#GS_COMPTYPE_WAVELET Wavelet hardware codec
        * <BR>#GS_COMPTYPE_MPEG1 MPEG 1 hardware compatible codec
        * <BR>#GS_COMPTYPE_MPEG2 MPEG 2 hardware compatible codec
        * <BR>#GS_COMPTYPE_MPEG2I Editable MPEG 2 I Frame Only compatible
codec
        * <BR>#GS_COMPTYPE_MPEG2IBP      MPEG 2 long GOP hardware compatible
codec
        * <BR>#GS_COMPTYPE_DV25 Hardware DV25, DVCPRO, DVCPRO25
        * <BR>#GS_COMPTYPE_DV50 Hardware DV50, DVCPRO50
        * <BR>#GS_COMPTYPE_DVSD Hardware Standard DV Bluebook, DVPRO, DVSD
        * <BR>#GS_COMPTYPE_DV100 High Def DV codec
        * <BR>#GS_COMPTYPE_HDPAN Panasonic HD to SDI codec
        * <BR>#GS_COMPTYPE_HDSONY Sony HD to SDI codec
        * <BR>MEDIACMD::dwStart - Bit array of supported types
        * <HR>
        */
        gsCompType,                             // MJPG, MPEG2, Uncompressed
(dwPosition, supported=dwStart)

        /**
        * Compression setting by total throughput
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Size of compressed stream in kilounsigned
chars per second
        * \li cmdType::ctGetValue
```

```
        * <BR>MEDIACMD::dwPosition - Size of compressed stream in kilounsigned
chars per second
        * <BR>MEDIACMD::dwStart - Smallest size possible
        * <BR>MEDIACMD::dwEnd - Largest size possible
        * <HR>
        */
        gsCompRateSize,                         // Total data throughput - frame size
(dwPosition, min=dwStart, max=dwEnd)
        /**
        * Compression setting by compression ratio
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Ratio * 100 (e.g. 2:1 = 200)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Ratio * 100 (e.g. 2:1 = 200)
        * <BR>MEDIACMD::dwStart - Smallest available ration * 100
        * <BR>MEDIACMD::dwEnd - Largest available ration * 100
        * <HR>
        */
        gsCompRateRatio,                // Total data rate - ratio * 100 (e.g. 2:1 = 200)
(dwPosition, min=dwStart, max=dwEnd)
        /**
        * Compression setting by compression percentage of original size
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Percentage * 100 (e.g. 50% compression =
5000)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Percentage * 100 (e.g. 50% compression =
5000)
        * <BR>MEDIACMD::dwStart - Smallest available percentage (usually 0)
        * <BR>MEDIACMD::dwEnd - Largest available percentage (usually 10000)
        * <HR>
        */
        gsCompRatePercent,              // Total data rate - percentage * 100 (0-10000)
of maximum (dwPosition, min=dwStart, max=dwEnd)
        /**
        * Number of frames per 'group of pictures'.  For MPEG compression as well as
        * defining keyframe interval for Cinepac, Indeo, MPEG-4, etc.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Number of frames between keyframes of MPEG
'GOP' frame length
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Number of frames between keyframes of MPEG
'GOP' frame length
        * <BR>MEDIACMD::dwStart - Minimum possible size of group of pictures
(usually 0)
```

```
        * <BR>MEDIACMD::dwEnd - Largest possible size of group of pictures (up to
10000 for MPEG 4)
        * <HR>
        */
        gsCompGOPSize,                          // Number of elements in GOP
        /**
        * Number of I Frame elements per GOP
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Kilounsigned chars available on drive
        * <HR>
        */
        gsCompIFactor,                  // Number of elements in GOP
        /**
        * Number of B Frame elements per GOP
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Kilounsigned chars available on drive
        * <HR>
        */
        gsCompBFactor,                          // Number of elements in GOP
        /**
        * Number of P Frame elements per GOP
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Kilounsigned chars available on drive
        * <HR>
        */
        gsCompPFactor,                          // Number of elements in GOP
        /**
        * Reference period to determine amount and order of P and B frames
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Kilounsigned chars available on drive
        * <HR>
        */
        gsCompRefPeriod,                        // Number of elements in GOP
        /**
        * Total storage available on current recording drive in kilounsigned chars
        * <BR>
        * \li cmdType::ctSetValue
```

```
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Kilounsigned chars available on drive
        * <HR>
        */
        gsTotalStorageAvail,     // Total available storage (dwPosition)
        /**
        * Total storage free on current recording drive in kilounsigned chars
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Kilounsigned chars free on drive
        * <HR>
        */
        gsTotalStorageFree,              // Total free storage (total free space/cur data
rate) (dwPosition)
        /**
        * Total recording time available on current recording drive at current
compression level
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Number of frames available to record to
        * <HR>
        */
        gsTotalTimeAvail,                // Total available time (dwPosition)
        /**
        * Total recording time free on current recording drive at current compression
level
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Number of frames free to record to
        * <HR>
        */
        gsTotalTimeFree,                 // Total free time (total free space/cur data rate)
(dwPosition)
        /**
        * VTR emulation ID type
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Any unsigned short VTR ID - See Control key in
registry docs and LocalConfig.exe
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Any unsigned short VTR ID - See Control key in
registry docs and LocalConfig.exe
```

```
        * <BR>  MEDIACMD::arbID - String description of VTR (short (8 char), then
long)
        * <HR>
        */
        gsVtrType,                                  // Emulation type (dwPosition)
        /**
        * Bayer signal wrapped in HD-SDI (for Arri, Weisscam, LMP, etc)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - HDSDI bayer transfer type:
#GS_HDSDIBAYER_DUALBIT, #GS_HDSDIBAYER_DUALLINKBIT,
#GS_HDSDIBAYER_ARRI_D21, #GS_HDSDIBAYER_ARRI_ALEXA,
        * #GS_HDSDIBAYER_WIESS_ONEFRAME, #GS_HDSDIBAYER_WIESS_2K1536,
#GS_HDSDIBAYER_WIESS_TWOFRAME, #GS_HDSDIBAYER_WIESS_QUADFRAME,
        * #GS_HDSDIBAYER_WIESS_TWO2K1536
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - HDSDI bayer transfer type:
#GS_HDSDIBAYER_DUALBIT, #GS_HDSDIBAYER_DUALLINKBIT,
#GS_HDSDIBAYER_ARRI_D21, #GS_HDSDIBAYER_ARRI_ALEXA,
        * #GS_HDSDIBAYER_WIESS_ONEFRAME, #GS_HDSDIBAYER_WIESS_2K1536,
#GS_HDSDIBAYER_WIESS_TWOFRAME, #GS_HDSDIBAYER_WIESS_QUADFRAME,
        * #GS_HDSDIBAYER_WIESS_TWO2K1536
        * <HR>
        */
        gsHDSDIBayerType,               // Type of signal if using HDSDI bayer

        /**
        * Front panel/GUI Interface Local Mode
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - If 1 then local control available, else remote
only
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - If 1 then local control available, else remote
only
        * <HR>
        */
        gsLocal = 1000,                             // Local enable/disable (dwPosition)
        /**
        * Supported read/write file types
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Currently #GS_SUPFILE_AVI,
#GS_SUPFILE_ODML,
        * #GS_SUPFILE_QT, #GS_SUPFILE_OMFI, #GS_SUPFILE_FIX,
#GS_SUPFILE_AUDONLY,
        * #GS_SUPFILE_STILLS, #GS_SUPFILE_UNK, #GS_SUPFILE_ANY
        * \li cmdType::ctGetValue
```

```
        * <BR>MEDIACMD::dwPosition - Currently #GS_SUPFILE_AVI,
#GS_SUPFILE_ODML,
        * #GS_SUPFILE_QT, #GS_SUPFILE_OMFI, #GS_SUPFILE_FIX,
#GS_SUPFILE_AUDONLY,
        * #GS_SUPFILE_STILLS, #GS_SUPFILE_UNK, #GS_SUPFILE_ANY
        * <BR>MEDIACMD::dwStart - Bit array of supported types per dwPosition above.
        * <HR>
        */
        gsSupportedFileTypes,  // In order of favorites (dwPosition)
        /**
        * File types for this channel to ignore
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Currently #GS_SUPFILE_AVI,
#GS_SUPFILE_ODML,
        * #GS_SUPFILE_QT, #GS_SUPFILE_OMFI, #GS_SUPFILE_FIX,
#GS_SUPFILE_AUDONLY,
        * #GS_SUPFILE_STILLS, #GS_SUPFILE_UNK, #GS_SUPFILE_ANY
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Currently #GS_SUPFILE_AVI,
#GS_SUPFILE_ODML,
        * #GS_SUPFILE_QT, #GS_SUPFILE_OMFI, #GS_SUPFILE_FIX,
#GS_SUPFILE_AUDONLY,
        * #GS_SUPFILE_STILLS, #GS_SUPFILE_UNK, #GS_SUPFILE_ANY
        * <HR>
        */
        gsIgnoreFileTypes,              // Per above     (dwPosition)
        /**
        * Disable recording on this channel or this channel does not support recording.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 1 to disable recording, or 0 to enable
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 1 to disable recording, or 0 to enable (play only
channels always return 1)
        * <HR>
        */
        gsRecInhibit,                   // Inhibit recording (dwPosition)
        /**
        * Select recording drive
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Bit representing drive where 0=C:, 1=D: etc
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Bit representing drive where 0=C:, 1=D: etc
        * <BR>MEDIACMD::dwStart - Bit array of available drives
        * <HR>
        */
```

```
        gsRecDrive,                             // Record drives (dwPosition,
available=dwStart)
        /**
        * Change the default record filename
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::arbID -  New next record filename
        * <BR>  MEDIACMD::cfFlags - must be set to cfUseClipID
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - gsTrue/gsFalse
        * <BR>MEDIACMD::arbID -  Next record filename
        * <BR>  MEDIACMD::cfFlags - must be set to cfUseClipID
        * <HR>
        */
        gsRecFileName,                          // Record drives (dwPosition,
available=dwStart)
        /**
        * Recording rate by throughput in kilobytes per second
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Target size of recorded stream in kilounsigned
chars per second
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Target size of recorded stream in kilounsigned
chars per second
        * <BR>MEDIACMD::dwStart - Smallest size possible
        * <BR>MEDIACMD::dwEnd - Largest size possible
        * <HR>
        */
        gsRecRate,                              // Default recording rate (dwPosition,
default=dwStart)
        /**
        * Default video/stream record file type
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Uses mftXXX enum from MediaReactorTypes.h
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Uses mftXXX enum from MediaReactorTypes.h
        * <HR>
        */
        gsRecFileFormat,                // Type of file to record SEE suppfiletypes
(dwPosition, avail=dwStart)
        /**
        * Default audio record file type
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Uses mftXXX enum from MediaReactorTypes.h
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Uses mftXXX enum from MediaReactorTypes.h
```

```
        * <HR>
        */
        gsRecAudFileFormat,             // Type of audio file, or -1 if embedded
(dwPosition, avail=dwStart)
        /**
        * Disable file deletion on this channel
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 1 to disable delete command, or 0 to enable
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 1 to disable delete command, or 0 to enable
        * <HR>
        */
        gsDelInhibit,                   // Inhibit deletion (dwPosition)
        /**
        * Allows/Inhibits clips being Deleted from Bin or TC Space
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - TRUE/FALSE
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - TRUE/FALSE
        * <HR>
        */
        // 1010 (dec)
        gsInsInhibit,                   // Inhibit insertion of clips (dwPosition)
        /**
        * Allows/Inhibits clips being added to Bin or TC Space
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - TRUE/FALSE
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - TRUE/FALSE
        * <BR>
        */
        gsConvertFileFormat,    // Type of file to convert to (dwPosition, avail = dwStart)
        /**
        * Default audio conversion file type
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Uses mftXXX enum from MediaReactorTypes.h
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Uses mftXXX enum from MediaReactorTypes.h
        * <HR>
        */
        gsConvertAudFileFormat,         // Type of audio file, or -1 if embedded
(dwPosition, avail = dwStart)
        /**
        * Default length, in frames, for a still graphics file being added as a clip
        * <BR>
```

```
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Duration in frames
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Duration in frames
        * <HR>
        */
        gsDefStillLen,                  // Default still length (dwPosition)
        /**
        * Current reference system time for house VITC or house LTC if available, if
        * not then from system clock interpolated with performance counter.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Current time code position
        * <BR>MEDIACMD::dwStart - Current milliseconds position
        * <BR>MEDIACMD::dwEnd - Current date
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current time code position
        * <BR>MEDIACMD::dwStart - Current milliseconds position
        * <BR>MEDIACMD::dwEnd - Current date
        * <HR>
        */
        gsSysTime,                      // System Reference time (dwPostion,
dwStart, dwEnd)
        /**
        * Current reference system time for house VITC or house LTC if available, if
        * not then from system clock interpolated with performance counter.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Current dysnc milliseconds
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current dysnc milliseconds
        * <HR>
        */
        gsDSyncMs,                      // DSync MS counter for current channel
        /**
        * Current hardware port used by channel.  Mostly for COMx: port
        * selection of CTL and EXT channels.
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - New com port
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current com port
        * <BR>MEDIACMD::dwStart - Available com ports as bit array
        * <HR>
        */
        gsHwPort,                       // Currently only for ext drv (dwPostion
= current, dwStart = available)
        /**
        * Playback only output or allow edit to edit
```

```
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_PBEE_AUTO (playback or e to e),
#GS_PBEE_PB (playback only)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_PBEE_AUTO (playback or e to e),
#GS_PBEE_PB (playback only), #GS_PBEE_DEFAULT (device default read only)
        * <BR>MEDIACMD::dwStart - Bit array of available commands per dwPosition
settings above
        * <HR>
        */
        gsPBEE,                                         // dwPosition
        /**
        * Video reference for servo select
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_SERVOREF_AUTO (ext is avail, else int),
#GS_SERVOREF_EXT (always external)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_SERVOREF_AUTO (ext is avail, else int),
#GS_SERVOREF_EXT (always external), #GS_SERVOREF_DEFAULT (device default read
only)
        * <BR>MEDIACMD::dwStart - Bit array of available commands per dwPosition
settings above
        * <HR>
        */
        gsServoRefSelect,               // dwPosition
        /**
        * Head select
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_HEADSEL_RECPLAY,
#GS_HEADSEL_PLAY
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_HEADSEL_RECPLAY,
#GS_HEADSEL_PLAY, GS_HEADSEL_DEFAULT (device default read only)
        * <BR>MEDIACMD::dwStart - Bit array of available commands per dwPosition
settings above
        * <HR>
        */
        gsHeadSelect,                   // dwPosition
        /**
        * Colour frame select
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_CLRFRM_2FLD, #GS_CLRFRM_4FLD,
#GS_CLRFRM_8FLD
        * \li cmdType::ctGetValue
```

```
        * <BR>MEDIACMD::dwPosition - #GS_CLRFRM_2FLD, #GS_CLRFRM_4FLD,
#GS_CLRFRM_8FLD, GS_CLRFRM_DEFAULT
        * <BR>MEDIACMD::dwStart - Bit array of available commands per dwPosition
settings above
        * <HR>
        */
        // 1020 (dec)
        gsColorFrame,                   // dwPosition
        /**
        * Video reference disable
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - GS_VIDREF_DISABLE, GS_VIDREF_ENABLE
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - GS_VIDREF_DISABLE, GS_VIDREF_ENABLE
        * <BR>MEDIACMD::dwStart - Bit array of available commands per dwPosition
settings above
        * <HR>
        */
        gsVidRefDisable,                // dwPosition
        /**
        * Get Play count delay for the VTR interp
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition stores the value 7 for default
        * <HR>
        */
        gsPlayCountDelay,               //dwPoisition     1022
        /**
        * Use fake edit mode for MPEG bumping.  Basically, all non
        * play speed commands will be emulated, and once a play (lock)
        * is reached the card will be synced to that time and play.
        * Dangerous if sync does not happen quickly...
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition 1 turns on fake edit, 0 turns off
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition 1 fake edit on, 0 fake edit of
        * <HR>
        */
        gsEmulateEditBumping,
        /**
        * Special command alt value for position requests, goes
        * to key frame nearest to requested frame
        * <BR>
        * \li cmdType::ctPause, cmdType::ctPlay
        * <BR>MEDIACMD::dwPosition Target Position
```

```
* <HR>
*/
cmdaltNearestKeyFrame,
/**
* Special command alt value for position requests, goes
* to key frame after the requested frame
* <BR>
* \li cmdType::ctPause, cmdType::ctPlay
* <BR>MEDIACMD::dwPosition Target Position
* <HR>
*/
cmdaltNextKeyFrame,
/**
* Special command alt value for position requests, goes
* to key frame before the requested frame
* <BR>
* \li cmdType::ctPause, cmdType::ctPlay
* <BR>MEDIACMD::dwPosition Target Position
* <HR>
*/
cmdaltPrevKeyFrame,
/**
* Special command alt value for position requests, goes
* to first frame of actual (non-black) video after the requested frame
* <BR>
* \li cmdType::ctPause, cmdType::ctPlay
* <BR>MEDIACMD::dwPosition Target Position
* <HR>
*/
cmdaltStartOfMessage,
/**
* Special command is video input valid
* GS_TRUE / GS_FALSE
* <BR>
* \li cmdType::ctGetValue, cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition  -True / False is input Valid
* <BR>  MEDIACMD::dwStart     -signal format of the input
* <HR>
*/
gsVidInputValid,
/**
* Special command is genlock input valid
* GS_TRUE / GS_FALSE
* <BR>
* \li cmdType::ctGetValue, cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition  -True / False is genlock Valid
* <BR>  MEDIACMD::dwStart     -signal format of the genlock signal
* <HR>
*/
```

```
        gsVidGenlockValid,
        /**
        * Special command to set/get edit mode for slow MPEG boards
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - #GS_SERIALEDITMODE_NONE,
#GS_SERIALEDITMODE_IGNORE, #GS_SERIALEDITMODE_FAKE
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - #GS_SERIALEDITMODE_NONE,
#GS_SERIALEDITMODE_IGNORE, #GS_SERIALEDITMODE_FAKE
        * <HR>
        */
        // 1030 (dec)
        gsSerialEditMode,
        /**
        * Enable/Disable serial types for ext or ctl channels
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - #GS_SERIALPROTOCOLS_SONY422,
#GS_SERIALPROTOCOLS_ODETICS, #GS_SERIALPROTOCOLS_VDCP
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - #GS_SERIALPROTOCOLS_SONY422,
#GS_SERIALPROTOCOLS_ODETICS, #GS_SERIALPROTOCOLS_VDCP
        * <BR>MEDIACMD::dwStart  - Bit array of supported protocols
        * <HR>
        */
        gsSerialProtocols,
        /**
        * Enable/Disable pause being called on first stop (ee) command.  A stop in stop
will stop (ee)
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - 1 enabled
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - 1 enabled
        * <HR>
        */
        gsPauseBeforeStop,
        /**
        * Pause delay in frames.  Wait this many frames after pause received before
pausing
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - # frames, default 0 (fast as possible)
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - # frames, default 0 (fast as possible)
        * <HR>
        */
        gsPauseDelay,
```

```
        /**
        * Enable/disable/setup front panel
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - enable (1), disable (0)
        * <BR>MEDIACMD::dwStart  - com port connected to panel (0 for true usb/vga)
        * <BR>MEDIACMD::dwEnd  - Panel type, set startup
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - enable (1), disable (0)
        * <BR>MEDIACMD::dwStart  - com port connected to panel (0 for true usb/vga)
        * <HR>
        */
        gsFrontPanel,
        /**
        * Enable/disable/setup front panel
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Comport Number i.e. 2,3,4
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Comport Number i.e. 2,3,4
        * <HR>
        */
        gsFrontPanelComPort,
        /**
        * Sent from the ctl module when a VDCP controller sets a new disk
        * preroll value.  It is in frames, but is not nec.  When commands
        * that use disk preroll are received, the correct offset is calculated
        * by the ctl and sent, so this value is for info only.  Do on use it.
* Version (4) 206 or greater
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>Not Supported
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - disk preroll frame send from ctl
        * <HR>
        */
        gsVDCPPreroll,  // 1036 dec
        /*
        * This will have the add/remove/setup channel
        */
        /**
        * Add a new channel (int, ext, ctl, net client)
        */
        gsChannelAdd = 2000,
        /**
        * Delete a channel (int, ext, ctl, net client)
        */
        gsChannelDel,
        /**
```

```
* Enable/Disable channel
*/
gsChannelEnable,
/**
* If channel supports, a network address
*/
gsChannelAddress,
/**
* If channel supports, a network port
*/
gsChannelPort,
/**
* If channel supports, a com port
*/
gsChannelComPort,
/**
* Target of channels command for ctl and network
*/
gsChannelTarget,
/**
* Path to channel support files (e.g. HTML files for the HTTP server)
*/
gsChannelPath,
/**
* Type of channel (read only?
*/
gsChannelType,
/**
* UserName
*/
gsChannelUserName,
/**
* PassWord
*/
gsChannelPassWord,

/**
* Set/Get user data.  Could be string, start, end and/or position
* kept in a mediacmd
* the cmdalt returned will be the time is was set or -1
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - Any user data
* <BR>MEDIACMD::dwStart - Any user data
* <BR>MEDIACMD::dwEnd - Any user data
* <BR>MEDIACMD::arbID - Any user data
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwCmdAlt - Ms time the user data was set or -1 for not set
* <BR>MEDIACMD::dwPosition - Any user data
```

```
* <BR>MEDIACMD::dwStart - Any user data
* <BR>MEDIACMD::dwEnd - Any user data
* <BR>MEDIACMD::arbID - Any user data
* <HR>
*/
gsUserData0 = 3000,
//! See #cmdGetSetValue::gsUserData0
gsUserData1,
//! See #cmdGetSetValue::gsUserData0
gsUserData2,
//! See #cmdGetSetValue::gsUserData0
gsUserData3,
//! See #cmdGetSetValue::gsUserData0
gsUserData4,
//! See #cmdGetSetValue::gsUserData0
gsUserData5,
//! See #cmdGetSetValue::gsUserData0
gsUserData6,
//! See #cmdGetSetValue::gsUserData0
gsUserData7,
//! See #cmdGetSetValue::gsUserData0
gsUserData8,
//! See #cmdGetSetValue::gsUserData0
gsUserData9,


/**
* Enable/Disable/Flush error log
* <BR>
* \li cmdType::ctSetValue
* <BR>MEDIACMD::dwPosition - 1 Enable, 0 Disable, -1 Flush
* <BR>MEDIACMD::dwStart - Start messages
* <BR>MEDIACMD::dwEnd - Maximum messages
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition - 1 Enable, 0 Disable
* <BR>MEDIACMD::dwStart - Total messages
* <BR>MEDIACMD::dwEnd - Maximum messages
* <HR>
*/
gsErrorLog = 10000,
/**
* Get/Set Error Log Name
* \li cmdType::ctSetValue
* <BR>MEDIACMD::arbID - New Log Name
* \li cmdType::ctGetValue
* <BR>MEDIACMD::arbID - Current Log Name
* <HR>
*/
gsErrorLogName,
```

```
        /**
        * Gets the starting ms value in message units
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - First message ms (ever)
        * <HR>
        */
        gsErrorLogStartMs,
        /**
        * Gets current ms time per log entries (for relative and date absolute)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Current ms
        * <HR>
        */
        gsErrorLogCurrentMs,
        /**
        * Gets the last change value, modified with each new entry.
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Last change value
        * <BR>MEDIACMD::dwStart - First available message number
        */
        gsErrorLogLastChange,
        /**
        * Gets/Set an error message to/from the log
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Send -1 for first num, send last returned
number to get next, Return num
        * <BR>MEDIACMD::lSpeed - Send format type
        * <BR>MEDIACMD::dwStart - Error Code
        * <BR>MEDIACMD::dwCmdAlt - Time of message ms
        * <BR>MEDIACMD::cfFlags - cfPreview causes RAW return
        * <BR>MEDIACMD::dwStart - First available message number
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwStart - Error Code
        * <BR>MEDIACMD::arbID - Raw message
        */
        gsErrorLogMessage,

        /**
        * Buffer levels for playback/record (real time)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> Not supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Buffers stored in board
        * <BR>MEDIACMD::dwStart - Buffers in queue to board
        * <BR>MEDIACMD::dwEnd - Buffers in queue from disk
        * <BR>. MEDIACMD:;dwInfoChannels - total system buffers available in board +
memory / system frame size
        * <BR>MEDIACMD::dwVideoChannels - Selected channel for buffer request
```

```
        * <BR>MEDIACMD::dwAudioChannels - Selected channel for buffer request
        * <HR>
        */
        gsSysBufferLevel = 20000,
        /**
        * Memory usage
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> Not supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Memory we are using
        * <BR>MEDIACMD::dwStart - Memory used in system
        * <BR>MEDIACMD::dwEnd - Total memory in system
        * <HR>
        */
        gsSysMemoryUsage,
        /**
        * CPU usage
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> Not supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - CPU we are using
        * <BR>MEDIACMD::dwStart - CPU used in system
        * <BR>MEDIACMD::dwEnd - Portion of CPU in kernel mode
        * <HR>
        */
        gsSysCPUUsage,
        /**
        * Dropped frames
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> Not supported
        * \li cmdType::ctGetValue
    * <BR>    MEDIACMD::dwPosition - Last drop number (includes off speed play that
should drop)
    * <BR>    MEDIACMD::lSpeed - millisecond time of last drop
    * <BR>    MEDIACMD::dwStart - Total playback dropped (since first run)
    * <BR>    MEDIACMD::dwEnd - Total record dropped (since first run)
        * <HR>
        */
        gsDroppedFrames,

        /**
        * Sets/Returns AutoProxy mode
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Proxy mode 0=disabled,
        * \li cmdType::ctSetValue
```

```
* <BR>MEDIACMD::dwPosition - Proxy mode 0=disabled,
*/
gsProxyMode = 50000,
/**
* Returns the status of any proxy generation
* <BR>
* \li MEDIACMD::ctCmd -
* \li cmdType::ctPlay - creating a proxy from file on disk
* \li cmdType::ctRecord - creating a proxy from a recording file
* \li MEDIACMD::dwCmdAlt - Last change in list ms
* \li MEDIACMD::dwPosition - Current encode frame
* \li MEDIACMD::dwEnd - current length of file
* \li MEDIACMD::dwStart - current processor percentage
* \li MEDIACMD::arbID - current encode path + name
*/
gsProxyStatus,
/**
* Return the next proxy source file name.  If the previous name
* is set to NULL then return the first clip in the list.
* <BR>
* \li cmdType::ctSetValue
* <BR>  - not supported
* \li cmdType::ctGetValue
* <BR>  -in-  MEDIACMD::arbID - Last returned file name or NULL
* <BR>  -out- MEDIACMD::arbID - Next file name in proxy list
* <BR>  MEDIACMD::dwPosition - Not used yet
* <BR>  MEDIACMD::dwStart - Not used yet
* <BR>  MEDIACMD::dwEnd - Not used yet
* <HR>
*/
gsGetNextProxy,
/**
* Add a new proxy file to the list.  Also see #gsAddProxyAndOutputName
* <BR>
* \li MEDIACMD::arbID - File name to proxy
* \li MEDIACMD::dwStart - Start frame of proxy
* \li MEDIACMD::dwEnd - End frame of proxy
*/
gsAddProxy,
/**
* Set a proxy file to next in list
* <BR>
* \li MEDIACMD::arbID - File to promote
*/
gsPromoteProxy,
/**
* Remove a proxy file from the list
* <BR>
* \li MEDIACMD::arbID - File to remove
```

```
*/
gsRemoveProxy,          // 50005
/**
* Get Set the max CPU percentage
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition - 0 No CPU limit, 1..100 max CPU usage
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - Current CPU usage
* <BR> MEDIACMD::dwStart - Current MAX CPU usage
* <BR> MEDIACMD::dwEnd - Default CPU Max Usage
*/
gsProxyCPUUsage,
/**
* Get Set a transfer into the archives
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID - file / clip to transfer
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - progress
* <BR> MEDIACMD::dwStart - # in queue
*/
gsTransferToArchive,
/**
* Get Set a transfer from the archives
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID - file / clip to transfer
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - progress
* <BR> MEDIACMD::dwStart - # in queue
*/
gsTransferFromArchive,
/**
* Get Archive list (NULL string = first)
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID - file / clip transfered
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - # of Clips in archive total
* <BR> MEDIACMD::dwCmdAlt - ms Since last Update of List (getLastChangeMs)
*/
gsGetNextArchiveClip,
/**
* Get Archive list (NULL string = first)
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID - file / clip transfering
* \li cmdType::ctGetValue
```

```
        * <BR> MEDIACMD::dwPosition - # of Clips transfering to archive
        * <BR> MEDIACMD::dwCmdAlt - ms Since last Update of List (getLastChangeMs)
        */
        gsGetNextTransferToArchiveClip,
        /**
        * Get Archive list (NULL string = first)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - file / clip transfering
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - # of Clips transfering from archive
        * <BR> MEDIACMD::dwCmdAlt - ms Since last Update of List (getLastChangeMs)
        */
        gsGetNextTransferFromArchiveClip,
        /**
        * Add a new proxy file with output name and i/o.  Extension of #gsAddProxy
        * <BR>
        * \li MEDIACMD::arbID[0] - "File name to proxy"\0"Ouptut name to use"\0
        * \li MEDIACMD::dwStart - Start frame of proxy
        * \li MEDIACMD::dwEnd - End frame of proxy
        */
        gsAddProxyAndOutputName,

        /**
        * Get VVW version number
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::arbID - Zero terminated ansi string with version number
        * <HR>
        */
        gsVVWVersion = 60000,          // VVW Version (dwPosition, arbID = string)
        /**
        * Get MediaReactor version number
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::arbID - Zero terminated ansi string with version number
        * <HR>
        */
        gsMEVersion,                   // Media Reactor Version  (dwPosition, arbID =
string)
        /**
        * Get VVW type description
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
```

```
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::arbID - Zero terminated ansi string with VVW machine type
        * <HR>
        */
        gsVVWType,                              // Name of VVW model (dwPosition,
arbID = string)
        /**
        * Get VVW channel type
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::arbID - Zero terminated ansi string with channel type
        * <HR>
        */
        gsVVWChannelType,               // Description of channel (dwPosition, arbID =
string)
        /**
        * Get/Set VVW channel name
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 0 Get Current, 1 Get Default
        * <BR>MEDIACMD::arbID - Zero terminated ansi string with desired channel
name
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::arbID - Zero terminated ansi string with current channel
name
        * <HR>
        */
        gsVVWChannelName,               // Name of channel (dwPosition, arbID = string)
        /**
        * Get VVW License status
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not supported
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - License type (-1=invalid, 0=perm, 1=days,
2=runs, 3=users)
        * <BR>MEDIACMD::dwStart - 0 for perm, # for days/runs/users
        * <BR>  MEDIACMD::dwEnd - License flags (app dependent)
        * <BR>  MEDIACMD::lSpeed - License level (app dependent)
        * <BR>MEDIACMD::arbID - Zero terminated ansi string about current license
        * <HR>
        */
        gsVVWLicense,                   // Status of license


        /**
        * Setup on screen monitor (VGA Monitor)
```

```
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - 1 to enable, 0 to disable
        * <BR>MEDIACMD::dwEnd - Left corner
        * <BR>MEDIACMD::dwStart - Top corner
        * <BR>MEDIACMD::dwCmdAlt - Size (0 = Default, 1 = Full, 2 = Half, 3 =
Quarter)
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - 1 to enable, 0 to disable
        * <BR>MEDIACMD::dwEnd - Left corner
        * <BR>MEDIACMD::dwStart - Top corner
        * <BR>MEDIACMD::dwCmdAlt - Size (0 = Default, 1 = Full, 2 = Half, 3 =
Quarter)
        * <HR>
        */
        gsMonitor = 64000,              // Set/Get info on VGA or Secondary NTSC
monitor
        /**
        * Set handles to windows
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Handle to target window or -1
        * <BR>MEDIACMD::dwEnd - Handle to owner application window
        * <BR>MEDIACMD::dwStart - Window flags
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Handle to target window
        * <BR>MEDIACMD::dwEnd - Handle to owner application window or -1
        * <BR>MEDIACMD::dwStart - Window flags
        * <HR>
        */
        gsMonitorHwnds,
        //! Alias for #gsMonitorHwnds for older apps
        gsHwnds = gsMonitorHwnds,                       // Get above info
        /**
        * Turns VGA display on / off without killing the window
        * Can use this later to set refresh rates - aspect ratios or what not
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - GS_TRUE, GS_FALSE
        * <BR>MEDIACMD::dwEnd -
        * <BR>MEDIACMD::dwStart -
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - GS_TRUE, GS_FALSE
        * <BR>MEDIACMD::dwEnd -
        * <BR>MEDIACMD::dwStart
        * <HR>
        */
        gsMonitorDisplay,
        /**
```

```
        *   Get a capture of the current output (input passthrough or
        * current clip output).  Use #cmdType::ctGetValue to get a preview.
        *<BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Type of capture that is going to be used
        * <BR>MEDIACMD::arbID - Optional, depends on dwPosition
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Preview type #GS_MONITORGRAB_NONE,
        * #GS_MONITORGRAB_TYPE_BMP, #GS_MONITORGRAB_TYPE_JPG,
#GS_MONITORGRAB_SIZE_FULL,
        * #GS_MONITORGRAB_SIZE_HALF, #GS_MONITORGRAB_SIZE_QUARTER,
#GS_MONITORGRAB_TO_MEMORY,
        * #GS_MONITORGRAB_TO_UNC_PATH, #GS_MONITORGRAB_TO_HTTP,
#GS_MONITORGRAB_TO_NETWORK
        * <BR>MEDIACMD::arbID - Image data, if returned not saved
        * <BR>
        * To create a grab type, combine on type (bmp, jpg) with a size
        * (full, half, quarter) and a target (memory, path, http, network). <BR>
        * Depending on target, the arbID member will be filled in as follows:
        * \li GS_MONITORGRAB_TO_MEMORY - arbID not used
        * \li GS_MONITORGRAB_TO_UNC_PATH - arbID unified naming conventions
(UNC) path
        * \li GS_MONITORGRAB_TO_HTTP - arbID contains the name
        * \li GS_MONITORGRAB_TO_NETWORK - Not implemented
        * <HR>
        */
        gsMonitorGrab,
        /**
        * Get/Set a pointer to the Utility Monitor DTDraw class
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Pointer in DWORD to DTDraw class (always
RGB32)
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Pointer in DWORD to DTDraw class (always
RGB32)
        * <BR>
        * <HR>
        */
        gsUtilityMonitorDraw,
        /**
        * Get/Set the layout of the utility monitor
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Main layout of screen
        *         // Normal quad screen<BR>
        *         #define AVUM_CONFIG_QUAD_SPLIT                    0<BR>
        *         // One full screen on left, 3 1/4 on right<BR>
```

```
        *       #define AVUM_CONFIG_ONELEFT_THREERIGHT      1<BR>
        *       // One full screen on top, 3 1/2 on bottom<BR>
        *       #define AVUM_CONFIG_ONETOP_THREEBOTTOM      2<BR>
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Main layout of screen
        * <BR>
        * <HR>
        */
        gsUtilityMonitorDrawSetup,
        /**
        * Get/Set a waveform, vectorscope, etc on DTDraw, VGA output or Main output
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - Pointer to the DTDraw structure to draw on
        * <BR>  MEDIACMD::dwStart - Enable or disable  #GS_ENABLE #GS_DISABLE
        * <BR>  MEDIACMD::dwEnd - Target #GS_WAVEVECTOR_TARGET_DTDRAW,
#GS_WAVEVECTOR_TARGET_VGA, #GS_WAVEVECTOR_TARGET_OUTPUT
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Returns the DTDraw structure it is drawing on
        * <BR>  MEDIACMD::dwStart - Is enabled or disabled  #GS_ENABLE
#GS_DISABLE
        * <BR>  MEDIACMD::dwEnd - Target #GS_WAVEVECTOR_TARGET_DTDRAW,
#GS_WAVEVECTOR_TARGET_VGA, #GS_WAVEVECTOR_TARGET_OUTPUT
        * <HR>
        */
        gsWaveVectorSetup,
        /**
        * Get/Set type of waveform vector
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - Waveform vector type(s)
#GS_WAVEVECTOR_PICTURE, #GS_WAVEVECTOR_VECTORSCOPE,
#GS_WAVEVECTOR_WAVEFORM
        * <BR>  MEDIACMD::dwStart - Channels to enable
#GS_WAVEVECTOR_CHANNEL_R, #GS_WAVEVECTOR_CHANNEL_G,
#GS_WAVEVECTOR_CHANNEL_B, #GS_WAVEVECTOR_CHANNEL_A,
#GS_WAVEVECTOR_CHANNEL_Y. #GS_WAVEVECTOR_CHANNEL_CR,
#GS_WAVEVECTOR_CHANNEL_CB
        * <BR>  MEDIACMD::dwEnd -
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Waveform vector type(s)
#GS_WAVEVECTOR_PICTURE, #GS_WAVEVECTOR_VECTORSCOPE,
#GS_WAVEVECTOR_WAVEFORM
        * <BR>  MEDIACMD::dwStart - Channels to enable
#GS_WAVEVECTOR_CHANNEL_R, #GS_WAVEVECTOR_CHANNEL_G,
#GS_WAVEVECTOR_CHANNEL_B, #GS_WAVEVECTOR_CHANNEL_A,
#GS_WAVEVECTOR_CHANNEL_Y. #GS_WAVEVECTOR_CHANNEL_CR,
#GS_WAVEVECTOR_CHANNEL_CB
```

```
        * <BR>  MEDIACMD::dwEnd
        * <HR>
        */
        gsWaveVectorType,
        /**
        * Get/Set area to use as source
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition -
        * <BR>  MEDIACMD::dwStart - Start line 0..(height-1)
        * <BR>  MEDIACMD::dwEnd - End line 1..height
        * <BR>  MEDIACMD::dwVideoChannels - Start pixel 0..(width-1) - not supported
yet
        * <BR>  MEDIACMD::dwAudioChannels - End pixel 1..width - not supported yet
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition -
        * <BR>  MEDIACMD::dwStart - Start line 0..(height-1)
        * <BR>  MEDIACMD::dwEnd - End line 1..height
        * <BR>  MEDIACMD::dwVideoChannels - Start pixel 0..(width-1) - not supported
yet
        * <BR>  MEDIACMD::dwAudioChannels - End pixel 1..width - not supported yet
        * <HR>
        */
        gsWaveVectorArea,
        /**
        * Get last update time in milliseconds
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Time of last update in milliseconds
        */
        gsWaveVectorLastChangeMs,

        /** Not to be used.  See VVWXMLNextDirEntry and VVWXMLFileInfo
        */
        gsDirGetList = 64250,
        /** Not to be used.  See VVWXMLNextDirEntry and VVWXMLFileInfo
        */
        gsDirGetInfo,
        /** Not to be used.  See VVWXMLNextDirEntry and VVWXMLFileInfo
        */
        gsDirGetFileInfo,
        /** Not to be used.  See VVWXMLNextDirEntry and VVWXMLFileInfo
        */
        gsDirGetFileGrab,
        /**
        * Allow VGA display to function
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Enabled 0,1
```

```
        * <BR>MEDIACMD::dwStart      - Fullscreen Enabled 0,1
        * <BR>MEDIACMD::dwEnd                - Reduced Frame Rate Setting (1= 1/1
2 = 1/2 3 = 1/3 or 4 = 1/4)
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Enabled 0,1
        * <BR>MEDIACMD::dwEnd                - Reduced Frame Rate Setting (1= 1/1
2 = 1/2 3 = 1/3 or 4 = 1/4)
        * <HR>
        */
        gsVgaDisplayEnable,
        /**
        * Allow VGA display to function
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Allow DirectX
        * <BR>MEDIACMD::dwStart      - Bit Array for YUV, RGB and Overlay allow
#RGB_OVERLAY #RGB_DIRECT #YUV_OVERLAY #YUV_DIRECT
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Allow DirectX
        * <BR>MEDIACMD::dwStart      - Bit Array for YUV, RGB and Overlay allow
#RGB_OVERLAY #RGB_DIRECT #YUV_OVERLAY #YUV_DIRECT
        * <HR>
        */
        gsVgaDirectXConfig,
        /**
        * Setup 3D VGA Output
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - 3D Display Type #GS_3DVGA_LEFTEYE
#GS_3DVGA_RIGHTEYE #GS_3DVGA_ANAGLYPH_REDCYAN
#GS_3DVGA_ANAGLYPH_REDBLUE #GS_3DVGA_ANAGLYPH_AMBERBLUE
#GS_3DVGA_ANAGLYPH_GREENMAGENTA #GS_3DVGA_INTERLACED
#GS_3DVGA_ONIONSKIN #GS_3DVGA_DIFFERENCE #GS_3DVGA_OVERUNDER
#GS_3DVGA_SIDEBYSIDE #GS_3DVGA_SPLIT #GS_3DVGA_MIRROR
#GS_3DVGA_BUTTERFLY #GS_3DVGA_AMINUSB_THRESHOLD #GS_3DVGA_DISSOLVE
#GS_3DVGA_WIPE
        * <BR> #GS_3DVGA_FLAG_INVERT #GS_3DVGA_FLAG_FLIPVERT
#GS_3DVGA_FLAG_FLIPHORIZ #GS_3DVGA_LUMA_DIFF
        * <BR>MEDIACMD::dwStart      - Bit Array of available 3D display types
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - 3D Display Type #GS_3DVGA_LEFTEYE
#GS_3DVGA_RIGHTEYE #GS_3DVGA_ANAGLYPH_REDCYAN
#GS_3DVGA_ANAGLYPH_REDBLUE #GS_3DVGA_ANAGLYPH_AMBERBLUE
#GS_3DVGA_ANAGLYPH_GREENMAGENTA #GS_3DVGA_INTERLACED
#GS_3DVGA_ONIONSKIN #GS_3DVGA_DIFFERENCE #GS_3DVGA_OVERUNDER
#GS_3DVGA_SIDEBYSIDE #GS_3DVGA_SPLIT #GS_3DVGA_MIRROR
#GS_3DVGA_BUTTERFLY #GS_3DVGA_AMINUSB_THRESHOLD #GS_3DVGA_DISSOLVE
#GS_3DVGA_WIPE
```

```
        * <BR> #GS_3DVGA_FLAG_INVERT #GS_3DVGA_FLAG_FLIPVERT
#GS_3DVGA_FLAG_FLIPHORIZ #GS_3DVGA_LUMA_DIFF
        * <HR>
        */
        gsVga3DConfig,
        /**
        * Setup 3D VGA type for #GS_3DVGA_WIPE (use SMPTE)
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Wipe type
        * <BR>MEDIACMD::dwStart      - Wipe range low (0)
        * <BR>MEDIACMD::dwEnd                - Wip range high (65536)
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Wipe type
        * <HR>
        */
        gsVga3DWipeType,
        /**
        * Setup 3D VGA mix value
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Mix value (16 bit)
        * <BR>MEDIACMD::dwStart      - Mix range low (0)
        * <BR>MEDIACMD::dwEnd                - Mix range high (65536)
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Mixs value (16 bit)
        * <HR>
        */
        gsVga3DMix,
        /**
        * Setup 3D VGA Threshold
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Threshold value (16 bit)
        * <BR>MEDIACMD::dwStart      - Threshold range low (0)
        * <BR>MEDIACMD::dwEnd        - Threshold range high (65536)
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Threshold value (16 bit)
        * <HR>
        */
        gsVga3DThreshold,
        /**
        * Setup 3D VGA horizontal split
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Split position in pixels
        * <BR>MEDIACMD::dwStart      - Left (0)
        * <BR>MEDIACMD::dwEnd                - Right (width)
        * \li cmdType::ctSetValue
```

```
        * <BR>MEDIACMD::dwPosition  - Split position in pixels
        * <HR>
        */
        gsVga3DSplitHorizontal,
        /**
        * Setup 3D VGA vertical split
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Split position in lines
        * <BR>MEDIACMD::dwStart      - Left (0)
        * <BR>MEDIACMD::dwEnd                - Right (height)
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Split position in lines
        * <HR>
        */
        gsVga3DSplitVertical,
        /**
        * Overlay a grid on the display, either percentage or sizes
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Percentage size (if #GS_NOT_SUPPORTED /
-1, then use start / end)
        * <BR>MEDIACMD::dwStart      - Horizontal pixels to next line in grid
        * <BR>MEDIACMD::dwEnd                - Veritical pixels to next line in grid
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Percentage size
        * <BR>MEDIACMD::dwStart      - Horizontal lines
        * <BR>MEDIACMD::dwEnd                - Veritical pixels to next line in grid
        * <HR>
        */
        gsVga3DGridSize,
        /**
        * Overlay a grid on the display
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Set type to (0=off,1=percent,2=pixel)
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Current Type
        * <HR>
        */
        gsVga3DGridType,
        /**
        * Allow Fullscreen VGA on secondary monitor
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition  - Set type to (0=off,1=on)
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Current Value
        * <BR>  MEDIACMD::dwStart     - number of monitors
```

```
        * <HR>
        */
        gsVgaFullscreenEnable,
        /**
        * Check if channels exist
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwVideoChannels - Possible Video Channels
        * <BR>MEDIACMD::dwAudioChannels - Possible Audio Channels
        * <BR>MEDIACMD::dwInfoChannels - Possible Info Channels
        * <HR>
        */
        gsChannelsExist = 65536,// Do the channels exist (dwPosition - used
dwvid/aud/inf channels)
        /**
        * Get/Set clip mode state (else time code mode)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - #GS_CLIPMODE_CLIPSPACE,
#GS_CLIPMODE_TCSPACE, #GS_CLIPMODE_SINGLE, #GS_CLIPMODE_FILM current
types
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_CLIPMODE_CLIPSPACE,
#GS_CLIPMODE_TCSPACE, #GS_CLIPMODE_SINGLE, #GS_CLIPMODE_FILM current
types
        * <HR>
        */
        gsClipMode,                                 // Are we in clip or timecode space mode
(dwPosition)
        /**
        * Get/Set record offset for VVW3x00 replay mode
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Time code offset or 0 to reset
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Time code offset or 0 to reset
        * <HR>
        */
        gsRecOffset,                        // Mostly for multi DigiSuite records
        /**
        * Get channel capabilities
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Bitwise array #GS_CHANCAP_PLAY,
#GS_CHANCAP_REVPLAY,
```

```
        * #GS_CHANCAP_PAUSE, #GS_CHANCAP_JOG, #GS_CHANCAP_SHUTTLE,
#GS_CHANCAP_SEEK,
        * #GS_CHANCAP_PREVIEW, #GS_CHANCAP_STOP, #GS_CHANCAP_ETOE,
#GS_CHANCAP_RECORD,
        * #GS_CHANCAP_EDIT, #GS_CHANCAP_RECSTOP,
#GS_CHANCAP_SELECTPRESET,
        * #GS_CHANCAP_EJECT, #GS_CHANCAP_LOOP, #GS_CHANCAP_VGAPREVIEW,
#GS_CHANCAP_AUDPREVIEW,
        * #GS_CHANCAP_FILE, #GS_CHANCAP_NET, #GS_CHANCAP_CLIPSPACE,
        * #GS_CHANCAP_TCSPACE, #GS_CHANCAP_ALL
        * <BR>
        * <HR>
        */
        gsChanCapabilities,                         // Return capabilities of VVW channel
        /**
        * Get last change millisecond time from clip space, tc space or file
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>Not Supported
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - last change in ms aligned with Dsync
        * <HR>
        */
        gsLastChangeMs,                                 // Are we in clip or timecode
space mode (dwPosition)
        /**
        * Get the state of the GPI ins, reset them with set.  Only GPIs that are
        * included in the 'Mask' will be effected.  If you want to set the GPI 2,
        * you need to set dwPosition to 0x00000002 and the dwVideoChannels (mask) to
        * 0x00000002.  Setting the dwPosition to 0x00000000 and the dwVideoChannels
to
        * 0x00000002 will turn off the GPI.  If the dwVideoChannels is 0, then
        * nothing will change.
        * <BR>
        * \li cmdType::ctSetValue - reset in events to nothing
        * <BR>MEDIACMD::dwPosition      - GPI (0-31  / 1-32)
        * <BR>MEDIACMD::dwStart        - GPI (32-63 / 33-64)
        * <BR>MEDIACMD::dwEnd          -     GPI (64-95 / 65-96)
        * <BR>MEDIACMD::dwVideoChannels - Mask for 0-31
        * <BR>MEDIACMD::dwAudioChannels - Mask for 32-63
        * <BR>MEDIACMD::dwInfoChannels -  Mask for 64-95
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition      - GPI (0-31  / 1-32)
        * <BR>MEDIACMD::dwStart        - GPI (32-63 / 33-64)
        * <BR>MEDIACMD::dwEnd          -     GPI (64-95 / 65-96)
        * <BR>MEDIACMD::dwVideoChannels - Mask for 0-31
        * <BR>MEDIACMD::dwAudioChannels - Mask for 32-63
        * <BR>MEDIACMD::dwInfoChannels -  Mask for 64-95
        * <BR>MEDIACMD::lSpeed - last change in ms aligned with Dsync
```

```
            * <BR>A 1 in the GPI bitwise array mean it triggered, 0 means it has not.
            * <HR>
            */
            gsGPIIn,                                        // Get/Set GPI inputs
            /**
            * Get the state of the GPI outs,  Only GPIs that are
            * included in the 'Mask' will be effected.  If you want to set the GPI 2,
            * you need to set dwPosition to 0x00000002 and the dwVideoChannels (mask) to
            * 0x00000002.  Setting the dwPosition to 0x00000000 and the dwVideoChannels
to
            * 0x00000002 will turn off the GPI.  If the dwVideoChannels is 0, then
            * nothing will change.
            * <BR>
            * \li cmdType::ctSetValue - set the GPIs up or down or pulse
            * <BR>MEDIACMD::dwPosition      - GPI (0-31  / 1-32)
            * <BR>MEDIACMD::dwStart        - GPI (32-63 / 33-64)
            * <BR>MEDIACMD::dwEnd          -     GPI (64-95 / 65-96)
            * <BR>MEDIACMD::dwVideoChannels - Mask for 0-31
            * <BR>MEDIACMD::dwAudioChannels - Mask for 32-63
            * <BR>MEDIACMD::dwInfoChannels -  Mask for 64-95
            * \li cmdType::ctGetValue - get the current GPI output state.
            * <BR>MEDIACMD::dwPosition      - GPI (0-31  / 1-32)
            * <BR>MEDIACMD::dwStart        - GPI (32-63 / 33-64)
            * <BR>MEDIACMD::dwEnd          -     GPI (64-95 / 65-96)
            * <BR>MEDIACMD::dwVideoChannels - Mask for 0-31
            * <BR>MEDIACMD::dwAudioChannels - Mask for 32-63
            * <BR>MEDIACMD::dwInfoChannels -  Mask for 64-95
            * <BR>MEDIACMD::lSpeed - Last Change Ms
            * <BR>A 1 in the GPI bitwise array means on or triggered, 0
            * is down or off.
            * <HR>
            */
            gsGPIOut,                                       // Get/Set GPI outputs
            /**
            * Get the current millisecond counter on the machine (NOT aligned to anything).
            * NOTE:  This is handled directly for network channels on server side
            * <BR>
            * \li cmdType::ctSetValue
            * <BR>Not Supported
            * \li cmdType::ctGetValue
            * <BR>MEDIACMD::dwPosition - the current millisecond counter
            * <HR>
            */
            gsCurrentMs,                                    // Current ms (dwPosition)
            /**
            * Get/Set whether we are adding on minute of black to start and end of each clip
            * <BR>
            * \li cmdType::ctSetValue
            * <BR>MEDIACMD::dwPosition - #GS_TRUE, #GS_FALSE
```

```
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - #GS_TRUE, #GS_FALSE
        * <HR>
        */
        gsClipModePreroll,                              // Are we adding black around
the clips

        /**
        * Get number of backups in the current media space
        * Set the number to back up from
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - Which backup to make active
        * <BR>  MEDIACMD::dwStart    - Clip mode to use (-1 = Current)
#GS_CLIPMODE_CLIPSPACE, #GS_CLIPMODE_TCSPACE
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Number of current backups available
        * <BR>  MEDIACMD::dwStart    - Clip mode to use (-1 = Current)
#GS_CLIPMODE_CLIPSPACE, #GS_CLIPMODE_TCSPACE
        * <HR>
        */
        gsClipModeBackup,                               // Use / Query number of backup
files for media space

        /*** Save the current parameters to the registry
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>No Parameters
        * \li cmdType::ctGetValue
        * <BR>Not Used
        * <HR>
        */
        gsSaveCurrent = 100000,         // Save the current params
        /**
        * Load a new clip space (or new if file does not exist)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - File name of new or existing clip space
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::arbID - returns current file name of clip space
        * <HR>
        */
        gsLoadClipSpace,         // Load a new clip space
        /**
        * Load a new tc space (or new if file does not exist)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - File name of new or existing tc space
        * \li cmdType::ctGetValue
```

```
        * <BR> MEDIACMD::arbID - returns current file name of tc space
        * <HR>
        */
        gsLoadTCSpace,          // Load a new ::VTR_TC space
        /**
        * Load a new film (or new if file does not exist)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - File name of new or existing film
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::arbID - returns current file name of film
        * <HR>
        */
        gsLoadFilmSpace,        // Load a new ::Film space
        /**
        * Load a new edit (or new if file does not exist)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - File name of new or existing edit clip
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::arbID - returns current file name of edit clip
        * <HR>
        */
        gsLoadEditSpace,        // Load a new ::Edit space

        /**
        * Log in user (must have rights on the local machine)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - User name zero terminated followed by
passunsigned short zero terminated.
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::arbID - Returns user name zero terminated.
        * <HR>
        */
        gsUserLogIn = 900000,
        /**
        * Last change in user status
        * <BR>
        * \li cmdType::ctSetValue
        * Not Supported
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - Last change in users (status or number)
        * <HR>
        */
        gsUserLastChangeMs,
        /**
        * Return a list of currently logged in users
        * <BR>
```

```
            * \li cmdType::ctSetValue
            * Not Supported
            * \li cmdType::ctGetValue
            * <BR>MEDIACMD::dwPosition - 0..max user
            * <BR>  MEDIACMD::arbID - User name zero terminated, location info zero
terminated
            * <BR>MEDIACMD::dwStart - User rights
            * NULL when all users have been returned
            * <HR>
            */
            gsUserList,
            /**
            * Allow a user access to the unit - ONLY AVAILABLE ON LOCAL MACHINE
            * <BR>
            * \li cmdType::ctSetValue
            * <BR> MEDIACMD::arbID - User name zero terminated, passunsigned short
zero terminated
            * <BR> MEDIACMD::dwStart - User Rights
            * \li cmdType::ctGetValue
            * Not Supported
            * <HR>
            */
            gsUserAdd,
            /**
            * Remove a user's access to the unit - ONLY AVAILABLE ON LOCAL MACHINE
            * <BR>
            * \li cmdType::ctSetValue
            * <BR> MEDIACMD::arbID - User name zero terminated.
            * \li cmdType::ctGetValue
            * Not Supported
            * <HR>
            */
            gsUserDel,
            /**
            * Change a users rights - ONLY AVAILABLE ON LOCAL MACHINE
            * <BR>
            * \li cmdType::ctSetValue
            * <BR> MEDIACMD::arbID - User name zero terminated.
            * <BR> MEDIACMD::dwPosition - the user rights
            * \li cmdType::ctGetValue
            * <BR> MEDIACMD::arbID - User name zero terminated (current if NULL)
            * <BR> MEDIACMD::dwPosition - the user rights
            * <HR>
            */
            gsUserRights,
            /**
            * Change current users passunsigned short (must be logged in)
            * <BR>
            * \li cmdType::ctSetValue
```

```
        * <BR> MEDIACMD::arbID - new passunsigned short zero terminated
        * new passunsigned short zero terminated.
        * \li cmdType::ctGetValue
        * Not Supported
        * <HR>
        */
        gsUserPasswd,


        /**
        * Create 1 small jpg image (picon) for a file frame
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                - frame number
        * <BR> MEDIACMD::arbID                     - file and directory
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                - (GS_TRUE | GS_FALSE) is set
picon
        * <BR> MEDIACMD::arbID                     - file and directory
        */
        gsPiconFrame = 1000000,
        /**
        * Get a full resolution jpg of a frame
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                - frame number
        * <BR> MEDIACMD::arbID                     - file and directory
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                - (GS_TRUE | GS_FALSE) is set
jpeg
        * <BR> MEDIACMD::arbID                     - file and directory
        */
        gsJpegFrame,
        /**
        * Default image directory
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID                     - directory to use (NULL = Use
Network Directory)
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::arbID                     - directory using for images
        */
        gsImageDirectory,
        /**
        * Get a frame's info
        * <BR>
```

```
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                - IN = frame number, OUT =
Width
        * <BR> MEDIACMD::dwStart            - Height
        * <BR> MEDIACMD::dwEnd        - Bits
        * <BR> MEDIACMD::arbID                     - Encoding
        * <BR> return frame number
        */
        gsFrameInfo,
        /**
        * Get a raw frame in frame's default format
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                - frame number
        * <BR> MEDIACMD::dwStart            - Format (original, RGBA)
        * <BR> MEDIACMD::dwEnd        - Frame Size
        * <BR> MEDIACMD::arbID                     - Image Data
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                - frame number
        * <BR> MEDIACMD::dwStart            - Format (original, RGBA)
        * <BR> MEDIACMD::dwEnd        - Frame Size
        * <BR> MEDIACMD::arbID                     - Image Data
        */
        gsRawFrame,
        /**
        * Create a black empty file for ::edit
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                - userbits
        * <BR> MEDIACMD::dwStart            - starting time code
        * <BR> MEDIACMD::dwEnd        - duration
        * <BR> MEDIACMD::arbID                     - Filename
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                - current frame writing
(0..dwEnd), -1 if no file writing
        * <BR> MEDIACMD::dwStart            - starting time code
        * <BR> MEDIACMD::dwEnd        - duration
        * <BR> MEDIACMD::arbID                     - Filename, null if no file writing
        */
        gsPreallocateEditFile,
        /**
        * Create a file from the clip ::edit
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                - userbits
        * <BR> MEDIACMD::dwStart            - starting time code
        * <BR> MEDIACMD::dwEnd        - duration
```

```
        * <BR> MEDIACMD::arbID                  - Filename
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition              - current frame writing
(0..dwEnd), -1 if no file writing
        * <BR> MEDIACMD::dwStart          - starting time code
        * <BR> MEDIACMD::dwEnd       - duration
        * <BR> MEDIACMD::arbID                  - Filename, null if no file writing
        */
        gsCreateEditFile,
        /**
        * Get a preview frame from AvHAL (usually via the network)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition              - Frame size
        * <BR> MEDIACMD::arbID                  - Image Data
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition     - Frame Size
        * <BR> MEDIACMD::dwStart          - If 1, return pointer, else return data
(note: return data not implemented)
        * <BR> MEDIACMD::arbID                  - Image Data
        */
        gsPreviewFrame,

        /**
        * Service
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>
        * <HR>
        */
        gsVVWService  = 1100000,


        /**
        * For clip copy and translation lists: Set removes an item, Get returns the list like
a cliplist
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>
        * <HR>
        */
        gsInsertQueue,
        /**
```

```
        * For clip copy queue manipulation
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>
        * <HR>
        */
        gsXlatQueue,

        /**
        * Set the rate and scale in an XML file for later opening
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>
        * \li cmdType::ctGetValue
        * <BR>
        * <HR>
        */
        gsXMLRateScale,

        /**
        * Set the rate and scale in an XML file for later opening
        * <BR>
        * \li cmdType::ctSetValue

        * <BR>
        * \li cmdType::ctGetValue
        * <BR>
        * <HR>
        */
        gsXMLFileProperties,
        /**
        * Set the clip file EDL or settings for XML export
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition          - XML Value below
        * <BR> MEDIACMD::dwStart        - DWORD setting
        * <BR> MEDIACMD::arbID          - Filename or string value
        * \li cmdType::ctGetValue
        * <BR>
        * <HR>
        */
        gsDTProjectToXml,
#define GS_XML_FILENAME                                 1       //Set the
XML File Name
#define GS_XML_CLIPFILE                                 2       //Set the
project clipfilename
```

```
#define GS_XML_EDL                                          3       //Set the
EDL name
#define GS_XML_STRING                                       4       //set a
string in the project file
#define GS_XML_DWORD                                        5       //add an
DWORD to the project file
#define GS_XML_SAVE                                         6       //save
the already open project
#define GS_XML_CHECK_OPEN                           7       //see if this
project has conflicting EDLs or clipfiles
#define GS_XML_OPEN_DELETE_FILES            8       //Open the project while
deleting same named EDLs & clip logs
#define GS_XML_OPEN_IGNORE_FILES            9       //Open the project while
leaving same named EDLs & clip logs
#define GS_ERROR_FILE_EXISTS                        -2      //Returned when
a conflict occurs (GS_XML_CHECK_OPEN)

        /**
        * Set the calling application to allow for app specific behaviors of the DDR
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                  - #GS_APP_NONE,
#GS_APP_QUICKCLIP, #GS_APP_QUICKCLIPXO, #GS_APP_VTRID,
#GS_APP_MEDIANXS, #GS_APP_DTREPLAYLIVE, #GS_APP_DTOUCH
        * <BR>
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                  - #GS_APP_NONE,
#GS_APP_QUICKCLIP, #GS_APP_QUICKCLIPXO, #GS_APP_VTRID,
#GS_APP_MEDIANXS, #GS_APP_DTREPLAYLIVE, #GS_APP_DTOUCH
        * <BR>
        * <HR>
        */
        gsApplicationID,        // 1100006
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_NONE                             0x00000000
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_QUICKCLIP            0x00000001
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_QUICKCLIPXO          0x00000002
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_VTRID                            0x00000004
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_MEDIANXS                         0x00000008
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_DTREPLAYLIVE                     0x00000010
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_DTOUCH                           0x00000020

        //
```

```
// Commands for timeclock overlay

/**
* Set the Video Inlay enabled = 1 or diabled = 0
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition                 - Enable Inlay
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition                 - Inlay Enabled
* <HR>
*/
gsInlay,
/**
* Set the Video Inlay File
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID              - Inlay Source File
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::arbID              - Inlay Source File
* <HR>
*/
gsInlayFile,
/**
* Set the Inlay Source Rect
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition                 - Inlay Source X
* <BR> MEDIACMD::dwStart            - Inlay Source Y
* <BR> MEDIACMD::dwEnd                      - Inlay Source Height
* <BR> MEDIACMD::lSpeed                     - Inlay Source Width
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition                 - Inlay Source X
* <BR> MEDIACMD::dwStart            - Inlay Source Y
* <BR> MEDIACMD::dwEnd                      - Inlay Source Height
* <BR> MEDIACMD::lSpeed                     - Inlay Source Width
* <HR>
*/
gsInlaySourceArea,
/**
* Set the Inlay Destination Rect
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition                 - Inlay Destination X
* <BR> MEDIACMD::dwStart            - Inlay Destination Y
* <BR> MEDIACMD::dwEnd                      - Inlay Destination Height
* <BR> MEDIACMD::lSpeed                     - Inlay Destination Width
```

```
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                 - Inlay Destination X
        * <BR> MEDIACMD::dwStart            - Inlay Destination Y
        * <BR> MEDIACMD::dwEnd                      - Inlay Destination Height
        * <BR> MEDIACMD::lSpeed                     - Inlay Destination Width
        * <HR>
        */
        gsInlayDestinationArea,
        /**
        * Set the frame offset of the Inlay Source vs Destination
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                 - Inlay frame offset
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                 - Inlay frame offset
        * <HR>
        */
        gsInlayOffset,
        /**
        * Set the frame of the Inlay Source as DF / NDF
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                 - TC_TYPE_DF / TC_TYPE_NDF
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                 - Inlay tc source (TC_TYPE_DF /
TC_TYPE_NDF)
        * <HR>
        */
        gsInlayTcType,
        /**
        * Exports a section of a file
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwStart = In point;
        * <BR> MEDIACMD::dwEnd = Out point;
        * <BR> MEDIACMD::dwPosition = Camera Number;
        * <BR> MEDIACMD::lSpeed = Mark Number;
        * <BR> MEDIACMD::arbID[0] = Source File Name;
        *
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition                 - Progress x / 1000
        * <HR>
        */
        gsExportClip,
        //
        // Special system commands at the end
```

```
        /**
        * Close the DDR and currently running application
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                 - Must be 0x01010101
        * <BR> MEDIACMD::dwStart            - Must be 0xA5A5A5A5
        * <BR> MEDIACMD::dwEnd                      - Must be 0x5F5F5F5F
        * <BR>
        * \li cmdType::ctGetValue Not supported
        * <BR>
        */
        gsShutdownApplication = 2147418112,                                     //
0x7FFF0000
#define GS_SHUTDOWNAPPLICATION_POSITION 0x01010101
#define GS_SHUTDOWNAPPLICATION_START 0xA5A5A5A5
#define GS_SHUTDOWNAPPLICATION_END 0x5F5F5F5F
        /**
        * Close the DDR and currently running application
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                 - Must be 0x11111111
        * <BR> MEDIACMD::dwStart            - Must be 0x5F5F5F5F
        * <BR> MEDIACMD::dwEnd                      - Must be 0xA5A5A5A5
        * <BR>
        * \li cmdType::ctGetValue Not supported
        * <BR>
        */
        gsShutdownSystem,
#define GS_SHUTDOWNSYSTEM_POSITION 0x11111111
#define GS_SHUTDOWNSYSTEM_START 0x5F5F5F5F
#define GS_SHUTDOWNSYSTEM_END 0xA5A5A5A5
        /**
        * Clean off the root, or the whold record derive
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition                 - Must be
GS_CLEANRECORDWIPE_ROOTDIR || GS_CLEANRECORDWIPE_WHOLEDRIVE
        * <BR> MEDIACMD::dwStart            - Must be 0x5F5F5F5F
        * <BR> MEDIACMD::dwEnd                      - Must be 0xA5A5A5A5
        * <BR>
        * \li cmdType::ctGetValue Not supported
        * <BR>
        */
#define GS_CLEANRECORDWIPE_ROOTDIR 0x0
#define GS_CLEANRECORDWIPE_WHOLEDRIVE 0x1
#define GS_CLEANRECORDWIPE_START 0x5F5F5F5F
#define GS_CLEANRECORDWIPE_END 0xA5A5A5A5
        gsCleanRecordWipeDrive,
```

```
        //
        // Internal Properties.  Do not add to Java/VB/HTTP/etc
        //

        /** Get the optimal offset for a video frame to allow a header to be added
        */
        gsInternalGetImageOffset = 0xFFFFFFFDUL
};      // cmdGetSetValue

//! Known file types
//@{
#define GS_CLIPMODE_ILLEGAL                     0xFFFFFFFF      //
filetypeIllegal
#define GS_CLIPMODE_CLIPSPACE           0                       //
filetypeClipSpace
#define GS_CLIPMODE_TCSPACE                     1
        // filetypeTCSpace
#define GS_CLIPMODE_SINGLE                      2
        // filetypeSingle
#define GS_CLIPMODE_FILM                3                       //
filetypeFilm
//@}


// Standard values for Get/Set/Supported commands
//! For cmdGetSetValue::gsTcSource - Using LTC
#define GS_TCSOURCE_LTC             1
//! For cmdGetSetValue::gsTcSource - Using VITC
#define GS_TCSOURCE_VITC    2
//! For cmdGetSetValue::gsTcSource - Using CTL
#define GS_TCSOURCE_CTL             4
//! For cmdGetSetValue::gsTcSource - Using absolute clip
#define GS_TCSOURCE_CLIP    7

//! No data, unknown data for cmdGetSetValue::gsFrameData,
cmdType::ctSetValue/cmdType::ctGetValue
#define GS_FRAMEDATA_UNKNOWN    0x00000
//! ASCII data (all printable) for cmdGetSetValue::gsFrameData,
cmdType::ctSetValue/cmdType::ctGetValue
#define GS_FRAMEDATA_ASCII          0x00001
//! Binary (hex) data for cmdGetSetValue::gsFrameData,
cmdType::ctSetValue/cmdType::ctGetValue
#define GS_FRAMEDATA_HEX            0x00002
//! Telecine RP-215 / DPX Data for cmdGetSetValue::gsFrameData,
cmdType::ctSetValue/cmdType::ctGetValue
#define GS_FRAMEDATA_TELECINE     0x10001
//! Close caption/teletext for cmdGetSetValue::gsFrameData,
cmdType::ctSetValue/cmdType::ctGetValue
```

```
#define GS_FRAMEDATA_CC_TTEXT     0x10002
//! Navy telemetry data for cmdGetSetValue::gsFrameData,
cmdType::ctSetValue/cmdType::ctGetValue
#define GS_FRAMEDATA_NAVY          0x10003

//! Set cmdGetSetValue::gsEditMode for an insert edit
#define GS_INSERT_EDIT             0x01
//! Set cmdGetSetValue::gsEditMode for an assemble edit
#define GS_ASSEMBLE_EDIT    0x02

//! Audio in/out unbalanced (RCA connector) high impedance at -10db
(cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_UNBALANCED_10          0x001
//! Audio in/out unbalanced (RCA connector) high impedance at -4db
(cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_UNBALANCED_4           0x002
//! Audio in/out balanced (XLR connector) 600ohm impedance at -10db
(cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_BALANCED_10            0x010
//! Audio in/out balanced (XLR connector) 600ohm impedance at +4db
(cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_BALANCED_4             0x020
//! Audio in/out digital single wire (cmdGetSetValue::gsAudInSelect
cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_SPDIF                  0x100
//! Audio in/out digital balanced with clock (cmdGetSetValue::gsAudInSelect
cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_AES_EBU                0x200
//! Audio in/out embedded in SDI or HD-SDI video signal
(cmdGetSetValue::gsAudInSelect cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_EMBEDDED               0x400
//! Audio in/out digital balanced with clock (cmdGetSetValue::gsAudInSelect
cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_AES_EBU_PRO            0x800
//! Use audio embedded in the HDMI signal
#define GS_AUDSELECT_HDMI                   0x1000
//! No audio in/out available, or cannot be configured (cmdGetSetValue::gsAudInSelect
cmdGetSetValue::gsAudOutSelect)
#define GS_AUDSELECT_NONE                   0
//! No Audio Selected leave silent
#define GS_AUDSELECT_SILENT                         0x040

//Defines for various audio bit rates.
#define GS_AUD_BIT_RATE_32000               0x0001
#define GS_AUD_BIT_RATE_41100               0x0002
#define GS_AUD_BIT_RATE_48000               0x0004
#define GS_AUD_BIT_RATE_56000               0x0008
#define GS_AUD_BIT_RATE_64000               0x0010
#define GS_AUD_BIT_RATE_80000               0x0020
```

```
#define GS_AUD_BIT_RATE_96000                   0x0040
#define GS_AUD_BIT_RATE_112000                  0x0080
#define GS_AUD_BIT_RATE_128000                  0x0100
#define GS_AUD_BIT_RATE_160000                  0x0200
#define GS_AUD_BIT_RATE_192000                  0x0400
#define GS_AUD_BIT_RATE_224000                  0x0800
#define GS_AUD_BIT_RATE_256000                  0x1000
#define GS_AUD_BIT_RATE_320000                  0x2000
#define GS_AUD_BIT_RATE_384000                  0x4000

#define GS_AUD_STEREO                                   0x001
#define GS_AUD_JOINT_STEREO                             0x002
#define GS_AUD_DUAL                                     0x004
#define GS_AUD_SINGLE                                   0x008
#define GS_AUD_MULTIPLE                                 0x010
#define GS_AUD_HEADROOM_18                              0x01
#define GS_AUD_HEADROOM_20                              0x02

//! Freeze - no freeze (cmdGetSetValue::gsVidFreeze)
#define GS_VIDFREEZE_NOT_FROZEN                 0
//! Freeze - first (0) field (cmdGetSetValue::gsVidFreeze)
#define GS_VIDFREEZE_FIELD0                             1
//! Freeze - second (1) field (cmdGetSetValue::gsVidFreeze)
#define GS_VIDFREEZE_FIELD1                             2
//! Freeze - both fields (cmdGetSetValue::gsVidFreeze)
#define GS_VIDFREEZE_FRAME                              3

//! Standard NTSC or PAL composite video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPOSITE                          0x001
//! SVHS/S-Video four wire NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_SVIDEO                                     0x002
//! Secondary NTSC or PAL video (often monitor selection)
(cmdGetSetValue::gsVidInSelect cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPOSITE_2                0x004
//! third NTSC or PAL video (often monitor selection) (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPOSITE_3                0x008
//! BetaCam level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV                      0x010
//! Panasonic M2 level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV_M2           0x020
//! SMPTE standard level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV_SMPTE        0x040
```

```
//! RGB at video standard rate (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_RGB                  0x080
//! D1 Serial Digital or HDSDI video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_D1_SERIAL                      0x100
//! D1 Serial Parallel video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_D1_PARALLEL                0x200
//! SDTI/SDI including high speed transfer video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_SDTI                           0x400
///Extra for Digital Rapids--order is screwed up but as long as it works I guess
//! Secondary NTSC or PAL video (often monitor selection)
(cmdGetSetValue::gsVidInSelect cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPOSITE_4                0x800
//! Secondary NTSC or PAL video (often monitor selection)
(cmdGetSetValue::gsVidInSelect cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_SVIDEO_2                       0x1000
//! Secondary NTSC or PAL video (often monitor selection)
(cmdGetSetValue::gsVidInSelect cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV_2            0x2000
//! Secondary NTSC or PAL video (often monitor selection)
(cmdGetSetValue::gsVidInSelect cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_D1_SERIAL_2                0x4000
//! Standard NTSC or PAL composite video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPOSITE_JAPAN            0x8000
//! SVHS/S-Video four wire NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_SVIDEO_JAPAN                   0x10000
//! BetaCam level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV_JAPAN    0x20000
//! SMPTE standard level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV_SMPTE_JAPAN      0x40000
//! xVGA compatible analog RGB
#define GS_VIDSELECT_XVID_RGB                               0x80000
//! HDMI - Auto YCbCr/RGB
#define GS_VIDSELECT_HDMI                                   0x100000
//! HDMI - RGB I/O
#define GS_VIDSELECT_HDMI_RGB                               0x200000
//! HDMI - YCBCR I/O
#define GS_VIDSELECT_HDMI_YCBCR                             0x400000
//! DVI Protocol
#define GS_VIDSELECT_DVI                                    0x800000
//! 2 HDSDI YCbCr signals at once
#define GS_VIDSELECT_3G_DUAL_RATE                           0x1000000
```

```
//! Dual link 4:4:4 over 1 cable
#define GS_VIDSELECT_3G_DUAL_LINK                                       0x2000000
//! No video available or no configurable settings (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_NONE                                       0

//! VTR (unruly hsync) lock for cmdGetSetValue::gsVidInLockType
cmdGetSetValue::gsVidOutLockType cmdGetSetValue::gsVidOutLockType
#define GS_VIDLOCKTYPE_VTR                                      1
//! Perfect lock for cmdGetSetValue::gsVidInLockType
cmdGetSetValue::gsVidOutLockType cmdGetSetValue::gsVidOutLockType
#define GS_VIDLOCKTYPE_BROADCAST                        2

//! Allow normal bandwidth (gsGetSetValue::gsVidInBandwidth
gsGetSetValue::gsVidBandwidth)
#define GS_VIDBAND_STANDARD                     0x01
//! Allow medium bandwidth (gsGetSetValue::gsVidInBandwidth
gsGetSetValue::gsVidBandwidth)
#define GS_VIDBAND_MEDIUM           0x02
//! Allow high bandwidth (gsGetSetValue::gsVidInBandwidth
gsGetSetValue::gsVidBandwidth)
#define GS_VIDBAND_HIGH                         0x04
//! Impose notch filter on bandwidth (gsGetSetValue::gsVidInBandwidth
gsGetSetValue::gsVidBandwidth)
#define GS_VIDBAND_NOTCH            0x08

//! Black at normal level (7.5 IRE NTSC, 0 IRE PAL) gsGetSetValue::gsVidBlackSetup
gsGetSetValue::gsVidInBlack
#define GS_VIDBLACK_SETUP           0x01    // 7.5 ire NTSC, 0 ire pal
//! Crystal black level (0 IRE NTSC, 0 IRE PAL) gsGetSetValue::gsVidBlackSetup
gsGetSetValue::gsVidInBlack
#define GS_VIDBLACK_CRYSTAL                     0x02    // 0 ire NTSC, 0 ire pal
//! Super black level (0 > IRE NTSC/PAL) gsGetSetValue::gsVidBlackSetup
gsGetSetValue::gsVidInBlack
#define GS_VIDBLACK_SUPER           0x04    // < 0 ire NTSC/pal

//! Whites are clamped or 100 IRE (gsGetSetValue::gsVidInWhite)
#define GS_VIDWHITE_CLAMP           0x01    // 100 IRE white max
//! Whites are scaled automatically from black level to 100 IRE
(gsGetSetValue::gsVidInWhite)
#define GS_VIDWHITE_SCALE           0x02    // Scale like AGC
//! Whites are allowed to be greater then 100 IRE (gsGetSetValue::gsVidInWhite)
#define GS_VIDWHITE_FREE            0x04    // Any white

//! No external genlock source (free running on internal clock)
(gsGetSetValue::gsVidOutGenlockSource)
#define GS_LOCKSRC_NONE             0x0001 // No genlock (free run master)
//! External ref in is genlock source (gsGetSetValue::gsVidOutGenlockSource)
#define GS_LOCKSRC_EXTIN    0x0002 // Lock to external in
```

```
//! Current input is genlock source (gsGetSetValue::gsVidOutGenlockSource)
#define GS_LOCKSRC_INPUT    0x0004 // Lock to current input
//! Composite (CVBS) input is genlock source (gsGetSetValue::gsVidOutGenlockSource)
#define GS_LOCKSRC_CVBS             0x0008 // Lock to composite input
//! S-Video (SVHS) input is genlock source (gsGetSetValue::gsVidOutGenlockSource)
#define GS_LOCKSRC_SVIDEO 0x0010 // Lock to S-Video In
//! Component Y input is genlock source (gsGetSetValue::gsVidOutGenlockSource)
#define GS_LOCKSRC_IN_Y             0x0020 // Lock to Y of BetaCam input
//! SDI serial digital input is genlock source (gsGetSetValue::gsVidOutGenlockSource)
#define GS_LOCKSRC_SDI              0x0040 // Lock to Digital Module Ref In.
//DigiSuiteLite
//! HDMI genlock
#define GS_LOCKSRC_HDMI             0x0080 // Lock to HDMI (usually input)

//! Keep analog monitor in line with digital (HD=HD, SD=SD)
#define GS_ANALOGMONITORMETHOD_DIRECT       0x0001
//! Convert everything to the nearest SD type
#define GS_ANALOGMONITORMETHOD_SD           0x0002
//! Convert everything to the nearest 720 HD type
#define GS_ANALOGMONITORMETHOD_HD7200x0004
//! Convert everything to the nearest 1080 HD type
#define GS_ANALOGMONITORMETHOD_HD1080       0x0008
//! SD->HD720 and HD->SD
#define GS_ANALOGMONITORMETHOD_FLIP720      0x0010
//! SD->HD720 and HD->SD
#define GS_ANALOGMONITORMETHOD_FLIP1080     0x0020
//! HD / SD -> HDSL
#define GS_ANALOGMONITORMETHOD_HSDL         0x0040

//! Upconvert to whole screen
#define GS_UPCONVERT_ANAMORPHIC                 0x0001
//! Upconvert with bars
#define GS_UPCONVERT_PILLARBOX              0x0002
//! Upconvert with some zoom
#define GS_UPCONVERT_ZOOM14x9               0x0004
//! Upconvert to letter box
#define GS_UPCONVERT_LETTERBOX              0x0008
//! Upconvert to wide zoom
#define GS_UPCONVERT_ZOOMWIDE               0x0010

//! Down convert with top/bottom black bars
#define GS_DOWNCONVERT_LETTERBOX            0x0001
//! Down convert and crop image
#define GS_DOWNCONVERT_CROP                     0x0002
//! Down convert to whole screen
#define GS_DOWNCONVERT_ANAMORPHIC           0x0004
//! Down convert to 14x9
#define GS_DOWNCONVERT_14x9                     0x0008
```

```
//! Standard MPEG resolution 120
#define GS_MPEG_RESOLUTION_120                  0x001
//! Standard MPEG resolution 240
#define GS_MPEG_RESOLUTION_240                  0x002
//! Standard MPEG resolution 288
#define GS_MPEG_RESOLUTION_288                  0x004
//! Standard MPEG resolution 352
#define GS_MPEG_RESOLUTION_352                  0x008
//! Standard MPEG resolution 480
#define GS_MPEG_RESOLUTION_480                  0x010
//! Standard MPEG resolution 512
#define GS_MPEG_RESOLUTION_512                  0x020
//! Standard MPEG resolution 522
#define GS_MPEG_RESOLUTION_544                  0x040
//! Standard MPEG resolution 576
#define GS_MPEG_RESOLUTION_576                  0x080
//! Standard MPEG resolution 608
#define GS_MPEG_RESOLUTION_608                  0x100
//! Standard MPEG resolution 704
#define GS_MPEG_RESOLUTION_704                  0x200
//! Standard MPEG resolution 720
#define GS_MPEG_RESOLUTION_720                  0x400

//! Chroma format 4:2:0
#define GS_CHROMA_FORMAT_420                    0x1
//! Chroma format 4:2:2
#define GS_CHROMA_FORMAT_422                    0x2
//! Chroma format 4:4:4
#define GS_CHROMA_FORMAT_444                    0x4
//! Chroma format 4:1:1
#define GS_CHROMA_FORMAT_411                    0x8

//! MPEG chroma format 4:2:0 see #GS_CHROMA_FORMAT_420
#define GS_MPEG_CHROMA_FORMAT_420               GS_CHROMA_FORMAT_420
//! MPEG chroma format 4:2:2 see #GS_CHROMA_FORMAT_422
#define GS_MPEG_CHROMA_FORMAT_422               GS_CHROMA_FORMAT_422
//! MPEG chroma format 4:4:4 see #GS_CHROMA_FORMAT_444
#define GS_MPEG_CHROMA_FORMAT_444               GS_CHROMA_FORMAT_444

//! MPEG DCT Precision 8 bits
#define GS_MPEG_DC_PRECISION_8                  0x1
//! MPEG DCT Precision 9 bits
#define GS_MPEG_DC_PRECISION_9                  0x2
//! MPEG DCT Precision 10 bits
#define GS_MPEG_DC_PRECISION_10                 0x4
//! MPEG DCT Precision 11 bits
#define GS_MPEG_DC_PRECISION_11                 0x8

//! Aspect ratio square
```

```
#define GS_ASPECT_RATIO_SQUARE          0x1
//! Aspect ratio 4:3
#define GS_ASPECT_RATIO_4x3                     0x2
//! Aspect ratio 16:9
#define GS_ASPECT_RATIO_16x9            0x4
//! Aspect ratio 2.21:1
#define GS_ASPECT_RATIO_2_21x1          0x8
//! MPEG aspect ratio square see #GS_ASPECT_RATIO_SQUARE
#define GS_MPEG_ASPECT_RATIO_SQUARE             GS_ASPECT_RATIO_SQUARE
//! MPEG aspect ratio square see #GS_ASPECT_RATIO_4x3
#define GS_MPEG_ASPECT_RATIO_4x3        GS_ASPECT_RATIO_4x3
//! MPEG aspect ratio square see #GS_ASPECT_RATIO_16x9
#define GS_MPEG_ASPECT_RATIO_16x9               GS_ASPECT_RATIO_16x9
//! MPEG aspect ratio square see #GS_ASPECT_RATIO_2_21x1
#define GS_MPEG_ASPECT_RATIO_2_21x1             GS_ASPECT_RATIO_2_21x1

#define GS_MPEG_STANDARD_SYSTEM                         0x1
#define GS_MPEG_STANDARD_PROGRAM                0x2
#define GS_MPEG_STANDARD_TRANSPORT              0x4
#define GS_MPEG_STANDARD_ELEMENTARY             0x8
#define GS_MPEG_STANDARD_ELEMENTRY
        GS_MPEG_STANDARD_ELEMENTARY

#define GS_MPEG_LANGUAGE_ENGLISH                0x0001
#define GS_MPEG_LANGUAGE_SPANISH                0x0002
#define GS_MPEG_LANGUAGE_FRENCH                         0x0004
#define GS_MPEG_LANGUAGE_GERMAN                         0x0008
#define GS_MPEG_LANGUAGE_JAPANESE               0x0010
#define GS_MPEG_LANGUAGE_DUTCH                  0x0020
#define GS_MPEG_LANGUAGE_DANISH                 0x0040
#define GS_MPEG_LANGUAGE_FINNISH                0x0080
#define GS_MPEG_LANGUAGE_ITALIAN                0x0100
#define GS_MPEG_LANGUAGE_GREEK                  0x0200
#define GS_MPEG_LANGUAGE_PORTUGUESE             0x0400
#define GS_MPEG_LANGUAGE_SWEDISH                0x0800
#define GS_MPEG_LANGUAGE_RUSSIAN                0x1000
#define GS_MPEG_LANGUAGE_CHINESE                0x2000

#define GS_MPEG_CC_FORMAT_CCUBE                         0x1
#define GS_MPEG_CC_FORMAT_ATSC                  0x2
#define GS_MPEG_CC_FORMAT_CCUBE_REORDER 0x4
#define GS_MPEG_CC_FORMAT_ATSC_REORDER          0x8

#define GS_MPEG_ONE_FRAMES                              0x0001
#define GS_MPEG_TWO_FRAMES                              0x0002
#define GS_MPEG_THREE_FRAMES                    0x0004
#define GS_MPEG_FOUR_FRAMES                             0x0008
#define GS_MPEG_FIVE_FRAMES                             0x0010
#define GS_MPEG_SIX_FRAMES                              0x0020
```

```
#define GS_MPEG_SEVEN_FRAMES                    0x0040
#define GS_MPEG_EIGHT_FRAMES                    0x0080
#define GS_MPEG_NINE_FRAMES                             0x0100
#define GS_MPEG_TEN_FRAMES                              0x0200
#define GS_MPEG_ELEVEN_FRAMES                   0x0400
#define GS_MPEG_TWELVE_FRAMES                   0x0800
#define GS_MPEG_THIRTEEN_FRAMES                 0x1000
#define GS_MPEG_FOURTEEN_FRAMES                         0x2000
#define GS_MPEG_FIFTEEN_FRAMES                  0x4000
#define GS_MPEG_SIXTEEN_FRAMES                  0x8000

//! Video for windows avi (audio video interleave)
#define VIDEOWRITETYPE_AVI              0x00000001 //standard uncompressed AVI
//! QuickTime movie (apple)
#define VIDEOWRITETYPE_MOV                      0x00000002 //Uncompressed
QuickTime
//! Windows Media Video (Microsoft)
#define VIDEOWRITETYPE_WMV                      0x00000004 //uncompressed windows
media
//! SoftImage/Avid uncompressed GEN
#define VIDEOWRITETYPE_GEN                      0x00000008 //Avid uncompressed
Gen File
//! Jaleo uncompressed format
// removed #define VIDEOWRITETYPE_JS                0x00000010      //Uncompressed
Jaleo
//! MXF - Sony HDCam 4:2:0/4:2:2 MPEG
#define VIDEOWRITETYPE_SONY_HD_MXF      0x00000010      //mxf 4:2:0/4:2:2 MPEG
//! DVS file format
// removed #define VIDEOWRITETYPE_DVS               0x00000020      //
//! Iridas 8 bit RGB format
// removed #define VIDEOWRITETYPE_IHSS              0x00000040      //
//! HDR+Raw descriptor for raw streams
#define VIDEOWRITETYPE_HDR                      0x00000080 //Header YUV
//! Stills - 8/10 bit YCbCr .yuv or .v210
#define VIDEOWRITETYPE_YUV                      0x00000100 //YUV still file format
//! Stills - Raw 24/32 bit RGB/RGBA
#define VIDEOWRITETYPE_RAW                      0x00000200 //uncompressed raw still
file format
//! Stills - Targa 24/32 bit RGB
#define VIDEOWRITETYPE_TGA                      0x00000400 //Targa still file format
//! Stills - no longer supported
#define VIDEOWRITETYPE_BMP                      0x00000800 //birmap still file format
//! Stills - Tiff 24/32 bit RGB/RGBA
#define VIDEOWRITETYPE_TIFF                     0x00001000 //TIFF still file format
//! MXF - Pansonic AVCi - Different P2 plugin
#define VIDEOWRITETYPE_AVCI_MXF         0x00002000      //mxf DV25/50/100, AVCi
//! Stills - DPX (SMPTE/Kodak) 10 bit RGB
#define VIDEOWRITETYPE_DPX                      0x00004000 //dpx still file format
```

```
//! MPEG program or transport stream - Note:  VVW send YCbCr 8 unc to/from board, compression done in MediaFile/PlugIn
#define VIDEOWRITETYPE_MPG                  0x00008000 //mpeg program/system
//! Stills - 4224 individual frames of 8 or 10 bit YCbCr+A
#define VIDEOWRITETYPE_4224                 0x00010000 //4:2:2:4 YCbCr 8/10 bit
//! MXF - Sony XDCam SD
#define VIDEOWRITETYPE_SONY_MXF         0x00020000      //mxf D10/HDV
//! MXF - Pansonic P2 DV25/50/100, AVCi
#define VIDEOWRITETYPE_P2_MXF           0x00040000      //mxf DV25/50/100, AVCi
//! MXF - Avid DNxHD, Uncompressed, JPEG
#define VIDEOWRITETYPE_AVID_MXF         0x00080000      //mxf DNxHD, Uncomp, JPEG
//! ARI - Raw Arri frame format
#define VIDEOWRITETYPE_ARRI                 0x00100000 //Bayer pattern raw
//! Jp2 - Jpeg2000 Still frames
#define VIDEOWRITETYPE_JP2K                 0x00200000 //Jpeg 2000
//! MXF - Omneon AVCi, DVxx, MPEG
#define VIDEOWRITETYPE_OP1a_MXF         0x00400000      //mxf Omneon AVCi, DVxx, MPEG, YCbCr
//! MXF - DCP XYZ or RGB JPEG-2000
#define VIDEOWRITETYPE_DCP_MXF          0x00800000      //mxf XYZ or RGB JPEG-2000
//! MPEG transport stream - Note:  VVW send YCbCr 8 unc to/from board, compression done in MediaFile/PlugIn
#define VIDEOWRITETYPE_TS               0x01000000 //mpeg transport (mpeg-2/mpeg-4/h264)
//! MPEG program or transport stream - Note:  VVW send YCbCr 8 unc to/from board, compression done in MediaFile/PlugIn
#define VIDEOWRITETYPE_MP4                  0x02000000 //mpeg program/system
//! Flash video (264+mp3) - Note:  VVW send YCbCr 8 unc to/from board, compression done in MediaFile/PlugIn
#define VIDEOWRITETYPE_FLASH        0x04000000 //flash video (264)
//! DNG - Cinema DNG format
#define VIDEOWRITETYPE_DNG                  0x08000000 //Bayer pattern in various forms
//
//0x10000000
//0x20000000
//0x40000000
//0x80000000

//! Audio in stereo channels
#define AUDIOWRITETYPE_STEREO           0x00000001
//! Audio in mono channels
#define AUDIOWRITETYPE_MONO                     0x00000002
//! Audio in multichannel
#define AUDIOWRITETYPE_MULTI            0x00000004
//! Audio write type internal
#define AUDIOWRITETYPE_INTERNAL         0x00000000
```

```
//! Audio write type aiff
#define AUDIOWRITETYPE_AIFF                         0x00000010
//! Audio write type wave
#define AUDIOWRITETYPE_WAVE                         0x00000020
// Sets
#define AUDIOWRITETYPE_WAVE_INTERNAL
        (AUDIOWRITETYPE_WAVE|AUDIOWRITETYPE_INTERNAL)
#define AUDIOWRITETYPE_WAVE_STEREO
        (AUDIOWRITETYPE_WAVE|AUDIOWRITETYPE_STEREO)
#define AUDIOWRITETYPE_WAVE_MONO
        (AUDIOWRITETYPE_WAVE|AUDIOWRITETYPE_MONO)
#define AUDIOWRITETYPE_WAVE_MULTI
        (AUDIOWRITETYPE_WAVE|AUDIOWRITETYPE_MULTI)
#define AUDIOWRITETYPE_AIFF_INTERNAL
        (AUDIOWRITETYPE_AIFF|AUDIOWRITETYPE_INTERNAL)
#define AUDIOWRITETYPE_AIFF_STEREO
        (AUDIOWRITETYPE_AIFF|AUDIOWRITETYPE_STEREO)

//! Turn off monitor
#define GS_MONITORGRAB_NONE                         0x0000
//! Type mask (jpg, bmp)
#define GS_MONITORGRAB_TYPE_MASK        0x00F0
//! Use BMP format for image
#define GS_MONITORGRAB_TYPE_BMP                     0x0000
//! Use JPEG format for image
#define GS_MONITORGRAB_TYPE_JPG         0x0010
//! Size mask (full, half, quarter)
#define GS_MONITORGRAB_SIZE_MASK        0x000F
//! Full size image captured
#define GS_MONITORGRAB_SIZE_FULL        0x0001
//! Half size image captured
#define GS_MONITORGRAB_SIZE_HALF        0x0002
//! Quarter size image captured
#define GS_MONITORGRAB_SIZE_QUARTER     0x0004
//! Target/To mask
#define GS_MONITORGRAB_TARGET_MASK      0x0F00
//! Use the arbID area
#define GS_MONITORGRAB_TO_MEMORY        0x0100
//! Save image to a UNC path
#define GS_MONITORGRAB_TO_UNC_PATH      0x0200
//! Save image to web server (use name sent in arbID)
#define GS_MONITORGRAB_TO_HTTP          0x0400
//! Save image through 'to be announced' network transport
#define GS_MONITORGRAB_TO_NETWORK       0x0800

//@{
#define GS_USERRIGHTS_NONE                  0x0000
#define GS_USERRIGHTS_READ                  0x0001
#define GS_USERRIGHTS_MODIFY        0x0002
```

```
#define GS_USERRIGHTS_WRITE             0x0004
#define GS_USERRIGHTS_SETUP             0x0008
#define GS_USERRIGHTS_PLAY              0x0010
#define GS_USERRIGHTS_RECORD    0x0020
#define GS_USERRIGHTS_ADD       0x0100
#define GS_USERRIGHTS_DELETE    0x0200
#define GS_USERRIGHTS_FULL      0x7FFF
#define GS_USERRIGHTS_ADMIN             0x8000
//@}

//
// Masks and shifts
//
//! Frame rate mask (portion of return for frame rate)
#define GS_SIGFORMMASK_FRAMERATE        0x000001ff
//! Shift frame rate to 0
#define GS_SIGFORMSHIFT_FRAMERATE       0
//! Horizontal / 8 mask (portion of return for frame rate)
#define GS_SIGFORMMASK_HORIZONTAL       0x000ffe00
//! Horizontal / 8 shift to 0
#define GS_SIGFORMSHIFT_HORIZONTAL      9
//! Vertical / 8 mask (portion of return for frame rate)
#define GS_SIGFORMMASK_VERTICAL         0x0ff00000
//! Vertical / 8 shift to 0
#define GS_SIGFORMSHIFT_VERTICAL 20
//! Frame type mask (portion of return for frame rate)
#define GS_SIGFORMMASK_FRAMETYPE        0xF0000000UL
//! Frame type shift to 0
#define GS_SIGFORMSHIFT_FRAMETYPE       28

//
// Basic frame rate types
//
#define GS_SIGFORMFRAMERATE_5           5
#define GS_SIGFORMFRAMERATE_6           6
#define GS_SIGFORMFRAMERATE_7_5         7
#define GS_SIGFORMFRAMERATE_10          10
#define GS_SIGFORMFRAMERATE_14_98       14
#define GS_SIGFORMFRAMERATE_15          15
#define GS_SIGFORMFRAMERATE_23_98       23
#define GS_SIGFORMFRAMERATE_24          24
#define GS_SIGFORMFRAMERATE_25          25
#define GS_SIGFORMFRAMERATE_29_97       29
#define GS_SIGFORMFRAMERATE_30          30
#define GS_SIGFORMFRAMERATE_47_95       47
#define GS_SIGFORMFRAMERATE_48          48
#define GS_SIGFORMFRAMERATE_50          50
#define GS_SIGFORMFRAMERATE_59_94       59
#define GS_SIGFORMFRAMERATE_60          60
```

```
#define GS_SIGFORMFRAMERATE_71_93       71
#define GS_SIGFORMFRAMERATE_72          72
#define GS_SIGFORMFRAMERATE_100         100             // 0x64
#define GS_SIGFORMFRAMERATE_119_88      119             // 0x77
#define GS_SIGFORMFRAMERATE_CUSTOM      0x100

//
// Sizing elements
//
#defineGS_SIGFORMSIZE_240               0x01
#defineGS_SIGFORMSIZE_243               0x02
#defineGS_SIGFORMSIZE_288               0x03    // Gap 4
#define GS_SIGFORMSIZE_320              0x08
#define GS_SIGFORMSIZE_352              0x09
#define GS_SIGFORMSIZE_360              0x0a    // Gap 5
#define GS_SIGFORMSIZE_480              0x10
#define GS_SIGFORMSIZE_483              0x11
#define GS_SIGFORMSIZE_486              0x12    // Gap 1
#define GS_SIGFORMSIZE_496              0x14    // Gap 2
#define GS_SIGFORMSIZE_504              0x16
#define GS_SIGFORMSIZE_512              0x17    // Gap 2
#define GS_SIGFORMSIZE_576              0x1a
#define GS_SIGFORMSIZE_600              0x1b
#define GS_SIGFORMSIZE_608              0x1c    // Gap 3
#define GS_SIGFORMSIZE_640              0x20
#define GS_SIGFORMSIZE_720              0x21
#define GS_SIGFORMSIZE_768              0x22
#define GS_SIGFORMSIZE_800              0x23
#define GS_SIGFORMSIZE_864              0x24
#define GS_SIGFORMSIZE_988              0x25    // Gap 3
#define GS_SIGFORMSIZE_857              0x26
#define GS_SIGFORMSIZE_960              0x28
#define GS_SIGFORMSIZE_968              0x29
#define GS_SIGFORMSIZE_778              0x2A
#define GS_SIGFORMSIZE_872              0x2B    // Gap 4
#define GS_SIGFORMSIZE_1024                     0x30
#define GS_SIGFORMSIZE_1035                     0x31
#define GS_SIGFORMSIZE_1044                     0x32
#define GS_SIGFORMSIZE_1052                     0x33
#define GS_SIGFORMSIZE_1050                     0x34    // Gap 4
#define GS_SIGFORMSIZE_1080                     0x38
#define GS_SIGFORMSIZE_1088                     0x39
#define GS_SIGFORMSIZE_1096                     0x3a    // Gap 3
#define GS_SIGFORMSIZE_1102                     0x3E    // Gap 1
#define GS_SIGFORMSIZE_1152                     0x40
#define GS_SIGFORMSIZE_1200                     0x41    // Gap 1
#define GS_SIGFORMSIZE_1234                     0x43    // Gap 6
#define GS_SIGFORMSIZE_1280                     0x48
#define GS_SIGFORMSIZE_1332                     0x49    // Gap 1
```

```
#define GS_SIGFORMSIZE_1400                     0x4B
#define GS_SIGFORMSIZE_1440                     0x4C    // Gap 3
#define GS_SIGFORMSIZE_1536                     0x50
#define GS_SIGFORMSIZE_1556                     0x51
#define GS_SIGFORMSIZE_1588                     0x52    // Gap 4
#define GS_SIGFORMSIZE_1828                     0x56
#define GS_SIGFORMSIZE_1714                     0x57
#define GS_SIGFORMSIZE_1600                     0x58
#define GS_SIGFORMSIZE_1920                     0x59
#define GS_SIGFORMSIZE_1782                     0x5A    // Gap 6
#define GS_SIGFORMSIZE_2048                     0x60    // Gap 3
#define GS_SIGFORMSIZE_2160                     0x64    // Gap 4
#define GS_SIGFORMSIZE_2650                     0x68    // Gap 4
#define GS_SIGFORMSIZE_2880                     0x6A    // Gap 1
#define GS_SIGFORMSIZE_3112                     0x6b    // Gap 3
#define GS_SIGFORMSIZE_4096                     0x80

//
// Basic Frame Sizes
//
#define GS_SIGFORMSIZE_640x480              ((GS_SIGFORMSIZE_640 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_480 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_640x576              ((GS_SIGFORMSIZE_640 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_576 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_720x480              ((GS_SIGFORMSIZE_720 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_480 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_720x483              ((GS_SIGFORMSIZE_720 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_483 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_720x486              ((GS_SIGFORMSIZE_720 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_486 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_720x512              ((GS_SIGFORMSIZE_720 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_512 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_720x576              ((GS_SIGFORMSIZE_720 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_576 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_720x608              ((GS_SIGFORMSIZE_720 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_608 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_720x504              ((GS_SIGFORMSIZE_720 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_504 <<
GS_SIGFORMSHIFT_VERTICAL))
```

```
#define GS_SIGFORMSIZE_800x600          ((GS_SIGFORMSIZE_800 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_600 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_960x486          ((GS_SIGFORMSIZE_960 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_486 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_960x576          ((GS_SIGFORMSIZE_960 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_576 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_960x504          ((GS_SIGFORMSIZE_960 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_504 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1024x768         ((GS_SIGFORMSIZE_1024 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_768 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1024x1024 ((GS_SIGFORMSIZE_1024 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1024 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1152x864         ((GS_SIGFORMSIZE_1152 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_864 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1280x1024 ((GS_SIGFORMSIZE_1280 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1024 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1400x1050 ((GS_SIGFORMSIZE_1400 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1050 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1600x1200 ((GS_SIGFORMSIZE_1600 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1200 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1280x720         ((GS_SIGFORMSIZE_1280 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_720 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1828x778    ((GS_SIGFORMSIZE_1828 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_778 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1828x988    ((GS_SIGFORMSIZE_1828 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_988 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1828x1102   ((GS_SIGFORMSIZE_1828 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1102 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1828x1332   ((GS_SIGFORMSIZE_1828 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1332 << GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1920x1035 ((GS_SIGFORMSIZE_1920 << GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1035 << GS_SIGFORMSHIFT_VERTICAL))
```

```
#define GS_SIGFORMSIZE_1920x1080 ((GS_SIGFORMSIZE_1920 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1080 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_1920x1088 ((GS_SIGFORMSIZE_1920 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1088 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_2560x1080 ((GS_SIGFORMSIZE_2650 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1080 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_2048x857     ((GS_SIGFORMSIZE_2048 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_857 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_2048x872     ((GS_SIGFORMSIZE_2048 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_872 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_2048x1102    ((GS_SIGFORMSIZE_2048 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1102 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_2048x1234    ((GS_SIGFORMSIZE_2048 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1234 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_2048x1080 ((GS_SIGFORMSIZE_2048 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1080 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_2048x1536 ((GS_SIGFORMSIZE_2048 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1536 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_2048x1556 ((GS_SIGFORMSIZE_2048 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1556 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_4096x1714 ((GS_SIGFORMSIZE_4096 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1714 <<
GS_SIGFORMSHIFT_VERTICAL))
#define GS_SIGFORMSIZE_4096x3112 ((GS_SIGFORMSIZE_4096 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_3112 <<
GS_SIGFORMSHIFT_VERTICAL))
// Arri D21
#define GS_SIGFORMSIZE_2880x2160 ((GS_SIGFORMSIZE_2880 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_2160 <<
GS_SIGFORMSHIFT_VERTICAL))
// Arri Alexa
#define GS_SIGFORMSIZE_2880x1782 ((GS_SIGFORMSIZE_2880 <<
GS_SIGFORMSHIFT_HORIZONTAL) | (GS_SIGFORMSIZE_1782 <<
GS_SIGFORMSHIFT_VERTICAL))

//
// Basic frame types
//
#define GS_SIGFORMTYPE_UNKNOWN                    (0)
```

```
#define GS_SIGFORMTYPE_INTERLACED                (1 <<
GS_SIGFORMSHIFT_FRAMETYPE)
#define GS_SIGFORMTYPE_PROGRESSIVE               (2 <<
GS_SIGFORMSHIFT_FRAMETYPE)
#define GS_SIGFORMTYPE_SEGMENTEDFRAME            (4 <<
GS_SIGFORMSHIFT_FRAMETYPE)

//! Signal format NTSC square pixel (320x240 or 640x480) @ 29.97 or 30 fps
gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_NTSC                          (GS_SIGFORMSIZE_640x480 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_29_97)
//! Signal format PAL square pixel (320x288 or 640x576) @ 25 fps
gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_PAL                           (GS_SIGFORMSIZE_640x576 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_25)
//! Signal format NTSC square pixel (360/352x243/240 or 720/704x486/480) @ 29.97 or
30 fps gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_CCIR_NTSC                (GS_SIGFORMSIZE_720x486 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_29_97)
//! Signal format NTSC square pixel (360/352x243/240 or 720/704x486/480) @ 29.97 or
30 fps gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_CCIR_NTSC_P483                (GS_SIGFORMSIZE_720x483 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_29_97)
//! Signal format PAL square pixel (360/352x288 or 720/704x576) @ 25 fps
gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_CCIR_PAL                      (GS_SIGFORMSIZE_720x576 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_25)
//! Signal format NTSC at 30 hz Progressive
#define GS_SIGFORM_CCIR_PNTSC_30(GS_SIGFORMSIZE_720x486 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_30)
//! Signal format PAL at 25 hz Progressive
#define GS_SIGFORM_CCIR_PPAL_25            (GS_SIGFORMSIZE_720x576 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
//! Signal format NTSC 23.98
#define GS_SIGFORM_CCIR_NTSC2398(GS_SIGFORMSIZE_720x486 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_23_98)
//! Signal format compressed HD 960x504 29.97
#define GS_SIGFORM_HD360                   (GS_SIGFORMSIZE_960x504 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_29_97)
//! Signal format NTSC High Res (960x486)
#define GS_SIGFORM_ALT_NTSC                      (GS_SIGFORMSIZE_960x486 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_29_97)
//! Signal format PAL High Res (960x576)
#define GS_SIGFORM_ALT_PAL                       (GS_SIGFORMSIZE_960x576 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_25)

//! 2200x1125 raster, 1920x1035 production aperture (1888x1017 clean) @ 30 fps
gsGetSetValue::gsSignalFormat
```

```
#define GS_SIGFORM_1035i_30_260M (GS_SIGFORMSIZE_1920x1035 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_30)
//! 2200x1125 raster, 1920x1035 production aperture (1888x1017 clean) @ 29.97 fp
gsGetSetValue::gsSignalFormats
#define GS_SIGFORM_1035i_30X_260M          (GS_SIGFORMSIZE_1920x1035 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_29_97)
//! 1920x1080i (274M-1997 Table1 System 4) @ 29.97 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080i_30                (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_30)        /*        (274M-
1997 Table1 System 4) */
#define GS_SIGFORM_1080sf_30               (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_30)          /*
        (274M-1997 Table1 System 4)   */
//! 1920x1080i (274M-1997 Table1 System 4) @ 30 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080i_30X               (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_29_97)    /*        (274M-
1997 Table1 System 5) */
#define GS_SIGFORM_1080sf_30X              (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_29_97)      /*
        (274M-1997 Table1 System 5)   */
//! 1920x1080i (274M-1997 Table1 System 4) @ 25 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080i_25                (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_25)        /*        (274M-
1997 Table1 System 6) */
#define GS_SIGFORM_1080sf_25               (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_25)          /*
        (274M-1997 Table1 System 6)   */
//! 1920x1080sf (274M-1997 Table1 System 4) @ 24 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080i_24                (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_24)
#define GS_SIGFORM_1080sf_24               (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_24)
//! 1920x1080sf (274M-1997 Table1 System 4) @ 23.98 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080i_24X               (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_23_98)
#define GS_SIGFORM_1080sf_24X              (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_23_98)
//! 1920x1080P (274M-1997 Table1 System 4) @ 30 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080_30                 (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_30)       /*        (274M-
1997 Table1 System 7) */
//! 1920x1080P (274M-1997 Table1 System 4) @ 29.97 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080_30X                (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_29_97)  /*        (274M-
1997 Table1 System 8) */
//! 1920x1080P (274M-1997 Table1 System 4) @ 25 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080_25                 (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)       /*        (274M-
1997 Table1 System 9) */
```

```
//! 1920x1080P (274M-1997 Table1 System 4) @ 24 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080_24                  (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)      /*       (274M-
1997 Table1 System 10)          */
//! 1920x1080P (274M-1997 Table1 System 4) @ 23.98 gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_1080_24X                         (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_23_98)  /*       (274M-
1997 Table1 System 11)          */


//! 1920x1080P 60 (Dual P30)
#define GS_SIGFORM_1080_60                  (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_60)
//! 1920x1080P 59.94 (Dual P29.97)
#define GS_SIGFORM_1080_60X                         (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_59_94)
//! 1920x1080P 50 (Dual 25)
#define GS_SIGFORM_1080_50                  (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_50)
//! 1920x1080P 48 (Dual 24)
#define GS_SIGFORM_1080_48                  (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_48)
//! 1920x1080P 47.95 (Dual 23.98)
#define GS_SIGFORM_1080_48X                         (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_47_95)


//! 1650x750 raster, 1280x720 production aperture (1248x702 clean): @ 60
gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_720_60                   (GS_SIGFORMSIZE_1280x720 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_60)      /*       (296M-
1996 Table1 System 1) */
//! 1650x750 raster, 1280x720 production aperture (1248x702 clean): @ 59.97
gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_720_60X                          (GS_SIGFORMSIZE_1280x720 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_59_94)  /*       (296M-
1996 Table1 System 2) */
//! 50 Hz DVS, IRT
#define GS_SIGFORM_720_50                   (GS_SIGFORMSIZE_1280x720 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_50)      /*       (296M-
1996 Table1 System 1) */

//! Half frame rate 720/60
#define GS_SIGFORM_720_30                   (GS_SIGFORMSIZE_1280x720 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_30)
//! Half frame rate 720/59.94
#define GS_SIGFORM_720_30X                          (GS_SIGFORMSIZE_1280x720 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_29_97)
//! Half 50 Hz DVS, IRT
```

```
#define GS_SIGFORM_720_25                    (GS_SIGFORMSIZE_1280x720 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)      /*       (296M-
1996 Table1 System 1) */
//! 720x1280 true 24 (Varicam)
#define GS_SIGFORM_720_24                    (GS_SIGFORMSIZE_1280x720 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)      /*       (296M-
1996 Table1 System 1) */

/** VGA res
*/
//! gsGetSetValue::gsSignalFormat Vesa 640x480@72
#define GS_SIGFORM_VESA_640_72               (GS_SIGFORMSIZE_640x480 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_72)
//! gsGetSetValue::gsSignalFormat Vesa 800x600@71.9
#define GS_SIGFORM_VESA_800_71X              (GS_SIGFORMSIZE_800x600 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_71_93)
//! gsGetSetValue::gsSignalFormat Vesa 800x600@72
#define GS_SIGFORM_VESA_800_72               (GS_SIGFORMSIZE_800x600 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_72)
//! gsGetSetValue::gsSignalFormat Vesa 1024x768@71.9
#define GS_SIGFORM_VESA_1024_71X(GS_SIGFORMSIZE_1024x768 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_71_93)
//! gsGetSetValue::gsSignalFormat Vesa 1024x766@72
#define GS_SIGFORM_VESA_1024_72              (GS_SIGFORMSIZE_1024x768 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_72)
//! gsGetSetValue::gsSignalFormat Vesa 1280x1024@24
#define GS_SIGFORM_VESA_1280_24              (GS_SIGFORMSIZE_1280x1024 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Vesa 1280x1024@30
#define GS_SIGFORM_VESA_1280i_30  (GS_SIGFORMSIZE_1280x1024 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_30)
//! gsGetSetValue::gsSignalFormat Vesa 1280x1024@71.9
#define GS_SIGFORM_VESA_1280_71X(GS_SIGFORMSIZE_1280x1024 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_71_93)
//! gsGetSetValue::gsSignalFormat Vesa 1280x1024@72
#define GS_SIGFORM_VESA_1280_72              (GS_SIGFORMSIZE_1280x1024 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_72)
//! gsGetSetValue::gsSignalFormat Vesa 1600x1200i@30
#define GS_SIGFORM_VESA_1600i_30  (GS_SIGFORMSIZE_1600x1200 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_30)

/** Presentation res
*/
//! gsGetSetValue::gsSignalFormat Presentation
#define GS_SIGFORM_DVI_1400_1050_24      (GS_SIGFORMSIZE_1400x1050 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Presentation
#define GS_SIGFORM_DVI_1400_1050_25      (GS_SIGFORMSIZE_1400x1050 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
```

```
//! gsGetSetValue::gsSignalFormat Presentation
#define GS_SIGFORM_DCIN_2048_25            (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
#define GS_SIGFORM_DCIN_2048sf_25                  (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_25)

/** Digital cinema 2048x1080
*/
//! gsGetSetValue::gsSignalFormat Digital Cinema
#define GS_SIGFORM_DCIN_2048sf_24X         (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_23_98)
//! gsGetSetValue::gsSignalFormat Digital Cinema
#define GS_SIGFORM_DCIN_2048sf_24          (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Digital Cinema
#define GS_SIGFORM_DCIN_2048_24X(GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_23_98)
//! gsGetSetValue::gsSignalFormat Digital Cinema
#define GS_SIGFORM_DCIN_2048_24            (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)

//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_1828_778_24  (GS_SIGFORMSIZE_1828x778 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_1828_778_25  (GS_SIGFORMSIZE_1828x778 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_1828_988_24  (GS_SIGFORMSIZE_1828x988 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_1828_988_25  (GS_SIGFORMSIZE_1828x988 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_1828_1102_24 (GS_SIGFORMSIZE_1828x1102 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_1828_1102_25 (GS_SIGFORMSIZE_1828x1102 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_1828_1332_24 (GS_SIGFORMSIZE_1828x1332 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_1828_1332_25 (GS_SIGFORMSIZE_1828x1332 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)

//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_2048_857_24  (GS_SIGFORMSIZE_2048x857 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
```

```
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_2048_857_25  (GS_SIGFORMSIZE_2048x857 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)

//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_2048_872_24  (GS_SIGFORMSIZE_2048x872 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_2048_872_25  (GS_SIGFORMSIZE_2048x872 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_2048_1102_24 (GS_SIGFORMSIZE_2048x1102 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_2048_1102_25 (GS_SIGFORMSIZE_2048x1102 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_2048_1234_24 (GS_SIGFORMSIZE_2048x1234 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film
#define GS_SIGFORM_FILM_2048_1234_25 (GS_SIGFORMSIZE_2048x1234 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)

/** Film transfer
*/
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048_15X (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_14_98)
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048_14          GS_SIGFORM_FILM_2048_15X  //
Deprecated, do not use
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048_15          (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_15)
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048sf_15X       (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_14_98)
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048sf_15        (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_15)
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048_24X (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_23_98)
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048_24          (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048sf_24X       (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_23_98)
```

```
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048sf_24(GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048_48          (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_48)

//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536_25          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536sf_25          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_25)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_25
        (GS_SIGFORMSIZE_2048x1556 | GS_SIGFORMTYPE_PROGRESSIVE |
GS_SIGFORMFRAMERATE_25)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048sf_25          (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_25)

//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536_15X          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_14_98)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536_15          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_15)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536sf_15X          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_14_98)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536sf_15          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_15)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536_24X          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_23_98)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536_24          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536sf_24X          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_23_98)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536sf_24          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
#define GS_SIGFORM_FILM_2048_1536_48X          (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_47_95)
//! gsGetSetValue::gsSignalFormat Film 2K(1536)
```

```
#define GS_SIGFORM_FILM_2048_1536_48            (GS_SIGFORMSIZE_2048x1536 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_48)

//! gsGetSetValue::gsSignalFormat Film 4K Half
#define GS_SIGFORM_FILM_4096_1714_24            (GS_SIGFORMSIZE_4096x1714 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film 4K Half
#define GS_SIGFORM_FILM_4096_1714_24X           (GS_SIGFORMSIZE_4096x1714 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_23_98)

//! gsGetSetValue::gsSignalFormat Film 4K
#define GS_SIGFORM_FILM_4096_3112sf_5           (GS_SIGFORMSIZE_4096x3112 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_5)
//! gsGetSetValue::gsSignalFormat Film 4K
#define GS_SIGFORM_FILM_4096_3112_24            (GS_SIGFORMSIZE_4096x3112 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Film 4K
#define GS_SIGFORM_FILM_4096_3112_24X           (GS_SIGFORMSIZE_4096x3112 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_23_98)

#define GS_SIGFORM_ARRI_D21
        (GS_SIGFORMSIZE_2880x2160 | GS_SIGFORMTYPE_PROGRESSIVE)
#define GS_SIGFORM_ARRI_ALEXA
        (GS_SIGFORMSIZE_2880x1782 | GS_SIGFORMTYPE_PROGRESSIVE)

/** Dump from June 24 2010, signal format values

        var GS_SIGFORM_NTSC = 285229085; //0x1100401D
        var GS_SIGFORM_PAL = 295714841; //0x11A04019
        var GS_SIGFORM_CCIR_NTSC = 287326749; //0x1120421D
        var GS_SIGFORM_CCIR_NTSC_P483 = 554713629; //0x2110421D
        var GS_SIGFORM_CCIR_PAL = 295715353; //0x11A04219
        var GS_SIGFORM_CCIR_PNTSC_30 = 555762206; //0x2120421E
        var GS_SIGFORM_CCIR_PPAL_25 = 564150809; //0x21A04219
        var GS_SIGFORM_CCIR_NTSC2398 = 287326743; //0x11204217
        var GS_SIGFORM_HD360 = 291524637; //0x1160501D
        var GS_SIGFORM_ALT_NTSC = 287330333; //0x1120501D
        var GS_SIGFORM_ALT_PAL = 295718937; //0x11A05019
        var GS_SIGFORM_1035i_30_260M = 319861278; //0x1310B21E
        var GS_SIGFORM_1035i_30X_260M = 319861277; //0x1310B21D
        var GS_SIGFORM_1080i_30 = 327201310; //0x1380B21E
        var GS_SIGFORM_1080sf_30 = 1132507678; //0x4380B21E
        var GS_SIGFORM_1080i_30X = 327201309; //0x1380B21D
        var GS_SIGFORM_1080sf_30X = 1132507677; //0x4380B21D
        var GS_SIGFORM_1080i_25 = 327201305; //0x1380B219
        var GS_SIGFORM_1080sf_25 = 1132507673; //0x4380B219
        var GS_SIGFORM_1080i_24 = 327201304; //0x1380B218
        var GS_SIGFORM_1080sf_24 = 1132507672; //0x4380B218
        var GS_SIGFORM_1080i_24X = 327201303; //0x1380B217
```

```
var GS_SIGFORM_1080sf_24X = 1132507671; //0x4380B217
var GS_SIGFORM_1080_30 = 595636766; //0x2380B21E
var GS_SIGFORM_1080_30X = 595636765; //0x2380B21D
var GS_SIGFORM_1080_25 = 595636761; //0x2380B219
var GS_SIGFORM_1080_24 = 595636760; //0x2380B218
var GS_SIGFORM_1080_24X = 595636759; //0x2380B217
var GS_SIGFORM_1080_60 = 595636796; //0x2380B23C
var GS_SIGFORM_1080_60X = 595636795; //0x2380B23B
var GS_SIGFORM_1080_50 = 595636786; //0x2380B232
var GS_SIGFORM_1080_48 = ;
var GS_SIGFORM_1080_48X = ;
var GS_SIGFORM_720_60 = 571510844; //0x2210903C
var GS_SIGFORM_720_60X = 571510843; //0x2210903B
var GS_SIGFORM_720_50 = 571510834; //0x22109032
var GS_SIGFORM_720_30 = 571510814; //0x2210901E
var GS_SIGFORM_720_30X = 571510813; //0x2210901D
var GS_SIGFORM_720_25 = 571510809; //0x22109019
var GS_SIGFORM_720_24 = 571510808; //0x22109018
var GS_SIGFORM_VESA_640_72 = 553664584; //0x21004048
var GS_SIGFORM_VESA_800_71X = 565200455; //0x21B04647
var GS_SIGFORM_VESA_800_72 = 565200456; //0x21B04648
var GS_SIGFORM_VESA_1024_71X = 572547143; //0x22206047
var GS_SIGFORM_VESA_1024_72 = 572547144; //0x22206048
var GS_SIGFORM_VESA_1280_24 = 587239448; //0x23009018
var GS_SIGFORM_VESA_1280i_30 = 318803998; //0x1300901E
var GS_SIGFORM_VESA_1280_71X = 587239495; //0x23009047
var GS_SIGFORM_VESA_1280_72 = 587239496; //0x23009048
var GS_SIGFORM_VESA_1600i_30 = 336637982; //0x1410B01E
var GS_SIGFORM_DVI_1400_1050_24 = 591435288; //0x23409618
var GS_SIGFORM_DVI_1400_1050_25 = 591435289; //0x23409619
var GS_SIGFORM_DCIN_2048_25 = 595640345; //0x2380C019
var GS_SIGFORM_DCIN_2048sf_25 = 1132511257; //0x4380C019
var GS_SIGFORM_DCIN_2048sf_24X = 1132511255; //0x4380C017
var GS_SIGFORM_DCIN_2048sf_24 = 1132511256; //0x4380C018
var GS_SIGFORM_DCIN_2048_24X = 595640343; //0x2380C017
var GS_SIGFORM_DCIN_2048_24 = 595640344; //0x2380C018
var GS_SIGFORM_FILM_1828_778_24 = 580955160; //0x22A0AC18
var GS_SIGFORM_FILM_1828_778_25 = 580955161; //0x22A0AC19
var GS_SIGFORM_FILM_1828_988_24 = 575712280; //0x2250AC18
var GS_SIGFORM_FILM_1828_988_25 = 575712281; //0x2250AC19
var GS_SIGFORM_FILM_1828_1102_24 = 601926680; //0x23E0AC18
var GS_SIGFORM_FILM_1828_1102_25 = 601926681; //0x23E0AC19
var GS_SIGFORM_FILM_1828_1332_24 = 613461016; //0x2490AC18
var GS_SIGFORM_FILM_1828_1332_25 = 613461017; //0x2490AC19
var GS_SIGFORM_FILM_2048_857_24 = 576765976; //0x2260C018
var GS_SIGFORM_FILM_2048_857_25 = 576765977; //0x2260C019
var GS_SIGFORM_FILM_2048_872_24 = 582008856; //0x22B0C018
var GS_SIGFORM_FILM_2048_872_25 = 582008857; //0x22B0C019
var GS_SIGFORM_FILM_2048_1102_24 = 601931800; //0x23E0C018
```

```
        var GS_SIGFORM_FILM_2048_1102_25 = 601931801; //0x23E0C019
        var GS_SIGFORM_FILM_2048_1234_24 = 607174680; //0x2430C018
        var GS_SIGFORM_FILM_2048_1234_25 = 607174681; //0x2430C019
        var GS_SIGFORM_FILM_2048_15X = 621854734; //0x2510C00E
        var GS_SIGFORM_FILM_2048_14 = 621854734; //0x2510C00E
        var GS_SIGFORM_FILM_2048_15 = 621854735; //0x2510C00F
        var GS_SIGFORM_FILM_2048sf_15X = 1158725646; //0x4510C00E
        var GS_SIGFORM_FILM_2048sf_15 = 1158725647; //0x4510C00F
        var GS_SIGFORM_FILM_2048_24X = 621854743; //0x2510C017
        var GS_SIGFORM_FILM_2048_24 = 621854744; //0x2510C018
        var GS_SIGFORM_FILM_2048sf_24X = 1158725655; //0x4510C017
        var GS_SIGFORM_FILM_2048sf_24 = 1158725656; //0x4510C018
        var GS_SIGFORM_FILM_2048_48 = 621854768; //0x2510C030
        var GS_SIGFORM_FILM_2048_1536_25 = 620806169; //0x2500C019
        var GS_SIGFORM_FILM_2048_1536sf_25 = 1157677081; //0x4500C019
        var GS_SIGFORM_FILM_2048_25 = 621854745; //0x2510C019
        var GS_SIGFORM_FILM_2048sf_25 = 1158725657; //0x4510C019
        var GS_SIGFORM_FILM_2048_1536_15X = 620806158; //0x2500C00E
        var GS_SIGFORM_FILM_2048_1536_15 = 620806159; //0x2500C00F
        var GS_SIGFORM_FILM_2048_1536sf_15X = 1157677070; //0x4500C00E
        var GS_SIGFORM_FILM_2048_1536sf_15 = 1157677071; //0x4500C00F
        var GS_SIGFORM_FILM_2048_1536_24X = 620806167; //0x2500C017
        var GS_SIGFORM_FILM_2048_1536_24 = 620806168; //0x2500C018
        var GS_SIGFORM_FILM_2048_1536sf_24X = 1157677079; //0x4500C017
        var GS_SIGFORM_FILM_2048_1536sf_24 = 1157677080; //0x4500C018
        var GS_SIGFORM_FILM_2048_1536_48X = 620806191; //0x2500C02F
        var GS_SIGFORM_FILM_2048_1536_48 = 620806192; //0x2500C030
        var GS_SIGFORM_FILM_4096_1714_24 = 628162584; //0x25710018
        var GS_SIGFORM_FILM_4096_1714_24X = 628162583; //0x25710017
        var GS_SIGFORM_FILM_4096_3112sf_5 = 1186004997; //0x46B10005
        var GS_SIGFORM_FILM_4096_3112_24 = 649134104; //0x26B10018
        var GS_SIGFORM_FILM_4096_3112_24X = 649134103; //0x26B10017
*/
//! All non video rate types (e.g.. 15fps, 10fps, 37fps) gsGetSetValue::gsSignalFormat
#define GS_SIGFORM_CUSTOM                       0xF0000000UL
//! For input and genlock status returns
#define GS_SIGFORM_NOT_PRESENT          0

/**
* Supported formats
*/
//! NTSC (CCIR or sqp) 720x480/486/508/512
#define GS_SIGFORM_SUPPORTS_NTSC        0x00000001
//! PAL (CCIR or sqp) 720x576/608
#define GS_SIGFORM_SUPPORTS_PAL         0x00000002
//! 960 width SD, not used
#define GS_SIGFORM_SUPPORTS_HR          0x00000004
//! 360 compressed, not used
#define GS_SIGFORM_SUPPORTS_360         0x00000008
```

```
//! 720p Rasters (59/60 and sometimes 50)
#define GS_SIGFORM_SUPPORTS_720          0x00000010
//! 1035x1080 Production rasters
#define GS_SIGFORM_SUPPORTS_1035         0x00000020
//! 1080/1088/1092/1112x1920 HD rasters
#define GS_SIGFORM_SUPPORTS_1080         0x00000040
//! 1088 HD rasters
#define GS_SIGFORM_SUPPORTS_EXTRA8       0x00000080          // 728, 1044,
1088
//! Film 2K 1536 lines
#define GS_SIGFORM_SUPPORTS_1536         0x00000100          // 2048x1536
//! Film 2K 1556 lines
#define GS_SIGFORM_SUPPORTS_1556         0x00000200          // 2048x1556
//! Digital Cinema 2048x1080
#define GS_SIGFORM_SUPPORTS_DCIN         0x00000400          // 2048x1080
//! Presentation 1440x1050
#define GS_SIGFORM_SUPPORTS_1400         0x00000800          // 1440x1050
//! Vesa 640x480
#define GS_SIGFORM_SUPPORTS_V480         0x00010000          // Vesa 640x480
//! Vesa 800x600
#define GS_SIGFORM_SUPPORTS_V600         0x00020000          // Vesa 800x600
//! Vesa 1024x768
#define GS_SIGFORM_SUPPORTS_V768         0x00040000          // Vesa
1024x768
//! Vesa 1280x1024
#define GS_SIGFORM_SUPPORTS_V1024        0x00080000          // Vesa
1280x1024
//! Vesa 1600x1200
#define GS_SIGFORM_SUPPORTS_V1200        0x00100000          // Vesa
1600x1200
//! Vesa 1600
#define GS_SIGFORM_SUPPORTS_V1600        0x00200000
//! Modifier times 2
#define GS_SIGFORM_SUPPORTS_X2           0x20000000
//! Modifier times 3
#define GS_SIGFORM_SUPPORTS_X3           0x40000000
//! Modifier times 4
#define GS_SIGFORM_SUPPORTS_X4           0x80000000
//! Supports 1080p 50/59/60
#define GS_SIGFORM_SUPPORTS_1080_X2      (GS_SIGFORM_SUPPORTS_1080|
GS_SIGFORM_SUPPORTS_X2)


//! Software passed codec on main processor (gsGetSetValue::gsCompType)
#define GS_COMPTYPE_SOFTWARE             0x00000001
//! Motion JPEG hardware codec (LSI, Zoran, C-Cube, etc)
(gsGetSetValue::gsCompType)
//#define GS_COMPTYPE_MJPEG                        0x00000002
//! Bayer
```

```
#define GS_COMPTYPE_BAYER                   0x00000002
//! MPEG-4 h.264
#define GS_COMPTYPE_H264                    0x00000004
//! Wavelet hardware codec (gsGetSetValue::gsCompType)
////#define GS_COMPTYPE_WAVELET             0x00000008
#define GS_COMPTYPE_JPEG2000                0x00000008
//! MPEG 1 hardware compatible codec (gsGetSetValue::gsCompType)
//#define GS_COMPTYPE_MPEG1                         0x00000010
//!
#define GS_COMPTYPE_CINEFORM_3D             0x00000010
//! MPEG 2 hardware compatible codec (gsGetSetValue::gsCompType)
/////#define GS_COMPTYPE_MPEG2               0x00000020
//! Uncompressed BGR 24 Bit
#define GS_COMPTYPE_BGR                           0x00000020      // was MPEG2
//! Editable MPEG 2 I Frame Only compatible codec (gsGetSetValue::gsCompType)
//#define GS_COMPTYPE_MPEG2I                0x00000040
#define GS_COMPTYPE_HDCAM                         0x00000040
//! MPEG 2 long GOP or IFrame hardware compatible codec
(gsGetSetValue::gsCompType)
//#define GS_COMPTYPE_MPEG2IBP    0x00000080
#define GS_COMPTYPE_MPEG                    0x00000080      // BitDepth 16=4:2:2,
12=4:2:0
//! Hardware DV25, DVCPRO. DVCPRO25  (gsGetSetValue::gsCompType)
#define GS_COMPTYPE_DV25                    0x00000100              // DV25,
DVCPRO. DVCPRO25
//! Hardware DV50, DVCPRO50 (gsGetSetValue::gsCompType)
#define GS_COMPTYPE_DV50                    0x00000200              // DV50,
DVCPRO50
//! Hardware Standard DV Bluebook, DVPRO, DVSD  (gsGetSetValue::gsCompType)
#define GS_COMPTYPE_DVSD                    0x00000400              // Standard DV
Bluebook, DVPRO, DVSD
//! High Def DV codec (gsGetSetValue::gsCompType)
#define GS_COMPTYPE_DV100                   0x00000800
//! 8Bit Y'CrCb uncompressed video (gsGetSetValue::gsCompType)
// No longer used #define GS_COMPTYPE_UN8BIT               0x00001000 use
GS_COMPTYPE_YCRCB_422
//!
#define GS_COMPTYPE_CINEFORM                0x00001000
//! 10Bit Y'CrCb uncompressed video (gsGetSetValue::gsCompType)
#define GS_COMPTYPE_YCRCB_V210              0x00002000      // was
GS_COMPTYPE_UN10BIT
//! Uncompressed RGB 24 Bit
#define GS_COMPTYPE_RGB                           0x00004000
//! Avid DNxHD
#define GS_COMPTYPE_DNxHD                         0x00008000
// WAS !!! #define GS_COMPTYPE_HDPAN                0x00010000
//! Pansonic AVCi (gsGetSetValue::gsCompType)
#define GS_COMPTYPE_AVCi                    0x00010000
// WAS !!! #define GS_COMPTYPE_HDSONY               0x00020000
```

```
//! Apple ProRes (gsGetSetValue::gsCompType)
#define GS_COMPTYPE_PRORES                      0x00020000
//!Inverted 32 bit TGA
#define GS_COMPTYPE_BGRA_INVERT         0x00040000
//! DPX 10 bit YCbCr
#define GS_COMPTYPE_DPX_YCBCR10         0x00080000
//! Uncompressed RGB (DVS)
#define GS_COMPTYPE_ARGB                0x00100000
//! Uncompressed RGBA (DVS)
#define GS_COMPTYPE_RGBA                0x00200000
//! Uncompressed A BGR - TIFF
#define GS_COMPTYPE_ABGR                0x00400000
//! Uncompressed BGR A - BMP/TGA
#define GS_COMPTYPE_BGRA                0x00800000
//! Uncompressed Y'CrCb 4:2:2 (DVS, VG)
#define GS_COMPTYPE_YCRCB_422           0x01000000
//! Uncompressed Y'CrCb 4:2:2A (DVS, Dual VG)
#define GS_COMPTYPE_YCRCB_422A          0x02000000
//! Uncompressed Y'CrCb 4:4:4 (DVS, Dual VG)
#define GS_COMPTYPE_YCRCB_444           0x04000000
//! Uncompressed Y'CrCb 4:4:4A (DVS, Dual VG)
#define GS_COMPTYPE_YCRCB_444A          0x08000000
//! Uncompressed Y'CrCb 4:4:4A (DVS, Dual VG) or 3D 8, 10, 30 or 32 bit
#define GS_COMPTYPE_STEREO                      0x10000000
//! Uncompressed Y'CrCb 4:2:0
#define GS_COMPTYPE_YCRCB_420           0x20000000
//! DPX 10 bit rgb
#define GS_COMPTYPE_DPX_RGB10           0x40000000
//! Use as generic alternative for use through AVCodec
#define GS_COMPTYPE_ALT                         0x80000000

//! Flag bit for dual rate capture
#define GS_HDSDIBAYER_DUALBIT                   0x10000000
//! Flag bit for dual pipe capture
#define GS_HDSDIBAYER_DUALLINKBIT               0x20000000
//! Raw bayer HD-SDI:  Arri D21 T-Link
#define GS_HDSDIBAYER_ARRI_D21                  0x00000001
//! Raw bayer HD-SDI:  Arri Alexa
#define GS_HDSDIBAYER_ARRI_ALEXA        0x00000002
//! Raw bayer HD-SDI:  Weisscam 1:1 - up to 30 in 30
#define GS_HDSDIBAYER_WIESS_ONEFRAME  0x00000100
//! Raw bayer HD-SDI:  Weisscam Film2K at 25p
#define GS_HDSDIBAYER_WIESS_2K1536              0x00000400
//! Raw bayer HD-SDI:  Weisscam 720p at 100p, 1080p up to 60
#define GS_HDSDIBAYER_WIESS_TWOFRAME  (GS_HDSDIBAYER_WIESS_ONEFRAME |
GS_HDSDIBAYER_DUALBIT)
//! Raw bayer HD-SDI:  Weisscam 720p at 200p, 1080p up to 120
#define GS_HDSDIBAYER_WIESS_QUADFRAME(GS_HDSDIBAYER_WIESS_ONEFRAME |
GS_HDSDIBAYER_DUALBIT | GS_HDSDIBAYER_DUALLINKBIT)
```

```
//! Raw dual 2K film
#define GS_HDSDIBAYER_WIESS_TWO2K1536 (GS_HDSDIBAYER_WIESS_2K1536 |
GS_HDSDIBAYER_DUALBIT)

/** @{ */
/** Standard Windows AVI container (gsGetSetValue::gsSupportedFileTypes
gsGetSetValue::gsIgnoreFileTypes)
* gsGetSetValue::gsRecFileFormat gsGetSetValue::gsRecAudFileFormat
gsGetSetValue::gsConvertFileFormat gsGetSetValue::gsConvertAudFileFormat */
#define GS_SUPFILE_AVI                  0x00000001
/** OpenDML AVI container (gsGetSetValue::gsSupportedFileTypes
gsGetSetValue::gsIgnoreFileTypes)
* gsGetSetValue::gsRecFileFormat gsGetSetValue::gsRecAudFileFormat
gsGetSetValue::gsConvertFileFormat gsGetSetValue::gsConvertAudFileFormat */
#define GS_SUPFILE_ODML                 0x00000002
/** QuickTime Mov/MooV container (gsGetSetValue::gsSupportedFileTypes
gsGetSetValue::gsIgnoreFileTypes)
* gsGetSetValue::gsRecFileFormat gsGetSetValue::gsRecAudFileFormat
gsGetSetValue::gsConvertFileFormat gsGetSetValue::gsConvertAudFileFormat */
#define GS_SUPFILE_QT                   0x00000004
/** Avid Open Media Format container (gsGetSetValue::gsSupportedFileTypes
gsGetSetValue::gsIgnoreFileTypes)
* gsGetSetValue::gsRecFileFormat gsGetSetValue::gsRecAudFileFormat
gsGetSetValue::gsConvertFileFormat gsGetSetValue::gsConvertAudFileFormat */
#define GS_SUPFILE_OMFI                 0x00000008
/** Drastic Fixed Frame container (gsGetSetValue::gsSupportedFileTypes
gsGetSetValue::gsIgnoreFileTypes)
* gsGetSetValue::gsRecFileFormat gsGetSetValue::gsRecAudFileFormat
gsGetSetValue::gsConvertFileFormat gsGetSetValue::gsConvertAudFileFormat */
#define GS_SUPFILE_FIX                  0x00000100
/** Audio only or separate audio formats (gsGetSetValue::gsSupportedFileTypes
gsGetSetValue::gsIgnoreFileTypes)
* gsGetSetValue::gsRecFileFormat gsGetSetValue::gsRecAudFileFormat
gsGetSetValue::gsConvertFileFormat gsGetSetValue::gsConvertAudFileFormat */
#define GS_SUPFILE_AUDONLY              0x00010000
/** Series of still file formats (gsGetSetValue::gsSupportedFileTypes
gsGetSetValue::gsIgnoreFileTypes)
* gsGetSetValue::gsRecFileFormat gsGetSetValue::gsRecAudFileFormat
gsGetSetValue::gsConvertFileFormat gsGetSetValue::gsConvertAudFileFormat */
#define GS_SUPFILE_STILLS       0x00100000
/** Other unspecified formats (gsGetSetValue::gsSupportedFileTypes
gsGetSetValue::gsIgnoreFileTypes)
* gsGetSetValue::gsRecFileFormat gsGetSetValue::gsRecAudFileFormat
gsGetSetValue::gsConvertFileFormat gsGetSetValue::gsConvertAudFileFormat */
#define GS_SUPFILE_UNK                  0x40000000
/** Any supported MediaReactor format (gsGetSetValue::gsSupportedFileTypes
gsGetSetValue::gsIgnoreFileTypes)
* gsGetSetValue::gsRecFileFormat gsGetSetValue::gsRecAudFileFormat
gsGetSetValue::gsConvertFileFormat gsGetSetValue::gsConvertAudFileFormat */
```

```
#define GS_SUPFILE_ANY                  0x80000000
/** @} */

//! Allow playback or edit to edit output as necessary (gsGetSetValue::gsPBEE)
#define GS_PBEE_AUTO                    0x00000000
//! Allow playback only output - no passthrough (gsGetSetValue::gsPBEE)
#define GS_PBEE_PB                      0x00000001
//! Allow passthrough only output - no playback (gsGetSetValue::gsPBEE)
#define GS_PBEE_EE                      0x00000002
//! Device dependent output (gsGetSetValue::gsPBEE)
#define GS_PBEE_DEFAULT                 0x000000FF

//! Video servo reference auto (gsGetSetValue::gsServoRefSelect)
#define GS_SERVOREF_AUTO        0x00000000
//! Video servo reference external only (gsGetSetValue::gsServoRefSelect)
#define GS_SERVOREF_EXT                 0x00000001
//! Video servo reference device default (gsGetSetValue::gsServoRefSelect)
#define GS_SERVOREF_DEFAULT             0x000000FF

//! Use record/play head (gsGetSetValue::gsHeadSelect)
#define GS_HEADSEL_RECPLAY              0x00000000
//! Use play head (gsGetSetValue::gsHeadSelect)
#define GS_HEADSEL_PLAY                 0x00000001
//! (gsGetSetValue::gsHeadSelect)
#define GS_HEADSEL_DEFAULT              0x000000FF

//! Edit color frame 2 field (gsGetSetValue::gsColorFrame)
#define GS_CLRFRM_2FLD                  0x00000000
//! Edit color frame 4 field (gsGetSetValue::gsColorFrame)
#define GS_CLRFRM_4FLD                  0x00000001
//! Edit color frame 8 field (gsGetSetValue::gsColorFrame)
#define GS_CLRFRM_8FLD                  0x00000002
//! Edit color frame device default (gsGetSetValue::gsColorFrame)
#define GS_CLRFRM_DEFAULT       0x000000FF

//! Disable video reference (gsGetSetValue::gsVidRefDisable)
#define GS_VIDREF_DISABLE       0x00000000
//! Enable video reference (gsGetSetValue::gsVidRefDisable)
#define GS_VIDREF_ENABLE        0x00000001

//! Channel can play (video or audio or both) (gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_PLAY                 0x00000001
//! Channel can reverse play (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_REVPLAY              0x00000002
//! Channel can pause and display frame (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_PAUSE        0x00000004
```

```
//! Channel can jog below play speed (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_JOG                  0x00000008
//! Channel can shuttle above play speed (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_SHUTTLE              0x00000010
//! Channel can seek to any point (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_SEEK                 0x00000020
//! Channel can preview from in to out (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_PREVIEW              0x00000040
//! Channel has a stop mode (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_STOP                 0x00001000
//! Channel can pass through video (in stop) (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_ETOE                 0x00002000
//! Channel can record (video or audio or both) (gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_RECORD               0x00004000
//! Channel can edit from in to out (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_EDIT                 0x00008000
//! Channel can set clip name and prep record (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_RECSTOP              0x00010000
//! Channel can select recording channels (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_SELECTPRESET  0x00020000
//! Channel can eject the media (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_EJECT            0x00040000
//! Channel can play in a loop (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_LOOP                 0x00100000
//! Channel can display a VGA preview (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_VGAPREVIEW    0x00200000
//! Channel can preview audio on a multi media card (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_AUDPREVIEW    0x00200000
//! Channel can play from a file (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_FILE                 0x01000000
//! Channel can play from a network feed (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_NET                  0x02000000
//! Channel can act like a clip space (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
```

```
#define GS_CHANCAP_CLIPSPACE        0x10000000
//! Channel can act like a VTR time code space (video or audio or both)
(gsGetSetValue::gsChanCapabilities)
#define GS_CHANCAP_TCSPACE                  0x20000000
//! Channel can be configured as MPEG -- opens a whole bunch of
//! settings on the options (specifically for Argus board right now).
#define GS_CHANCAP_MPEG_ENC                 0x40000000
//! Do not use this bit - indicates error
#define GS_CHANCAP_ERROR            0x80000000
//! Channel can do anything except MPEG_ENC (by default this should not be)
#define GS_CHANCAP_ALL                      0x3FFFFFFF

//! Stop if frames dropped in playback
#define GS_PRODUCTION_MODE_PLAY                     0x01
//! Stop if frames dropped in record
#define GS_PRODUCTION_MODE_RECORD       0x02

//! Normal editing mode, no special speed compensation
cmdGetSetValue::gsSerialEditMode
#define GS_SERIALEDITMODE_NONE              0
//! Ignore all off speed play commands (CBS TimeLogic Mode)
cmdGetSetValue::gsSerialEditMode
#define GS_SERIALEDITMODE_IGNORE            1
//! Pause at each speed change, call time play when real play comes (CTV mode)
cmdGetSetValue::gsSerialEditMode
#define GS_SERIALEDITMODE_FAKE              2

//! Enable Sony VTR 422 (gsSerialProtocols)
#define GS_SERIALPROTOCOLS_SONY422        1
//! Enable Odetics extensions (gsSerialProtocols)
#define GS_SERIALPROTOCOLS_ODETICS        2
//! Enable VDCP Louth (gsSerialProtocols)
#define GS_SERIALPROTOCOLS_VDCP             4

//! cmdGetSetValue::gsProxyMode Do not automatically proxy anything
#define GS_PROXYMODE_NOTHING                0x000000
//! cmdGetSetValue::gsProxyMode Proxy any file that is opened (for read/write/check)
#define GS_PROXYMODE_EVERYTHING                     0x000002
//! cmdGetSetValue::gsProxyMode Proxy files while they are recording (with supported
source types)
#define GS_PROXYMODE_RECORD                         0x000001
//! cmdGetSetValue::gsProxyMode Proxy files once they have finished recording
#define GS_PROXYMODE_AFTERRECORD        0x000010
//! cmdGetSetValue::gsProxyMode       Abort all active proxies
#define GS_PROXYMODE_ABORTALL               0x0ffff0

// WaveForm, Vectorscope control

//! cmdGetSetValue::gsWaveVectorSetup standard picture
```

```
#define GS_WAVEVECTOR_PICTURE                   0x00000001
//! cmdGetSetValue::gsWaveVectorSetup standard vectorscope
#define GS_WAVEVECTOR_VECTORSCOPE               0x00000002
//! cmdGetSetValue::gsWaveVectorSetup standard waveform
#define GS_WAVEVECTOR_WAVEFORM                      0x00000004
//! cmdGetSetValue::gsWaveVectorSetup parade RGB waveform
#define GS_WAVEVECTOR_WAVEFORM_RGB              0x00000008
//! cmdGetSetValue::gsWaveVectorSetup parade Y CB CR waveform
#define GS_WAVEVECTOR_WAVEFORM_YCBCR0x00000010
//! cmdGetSetValue::gsWaveVectorSetup histogram
#define GS_WAVEVECTOR_HISTOGRAM                     0x00000020
//! cmdGetSetValue::gsWaveVectorSetup parade histogram
#define GS_WAVEVECTOR_HISTOGRAM_SEP             0x00000040
//! cmdGetSetValue::gsWaveVectorSetup illegal colors
#define GS_WAVEVECTOR_DATA                          0x00000080
//! cmdGetSetValue::gsWaveVectorSetup illegal colors
#define GS_WAVEVECTOR_ILLEGAL                   0x00000100
//! Video types mask
#define GS_WAVEVECTOR_MASK                          0x000001FF
//! Audio wave form
#define GS_WAVEAUDIO_WAVE                           0x00000200
//! Audio meters
#define GS_WAVEVAUDIO_METERS                    0x00000400
//! Audio Lissajous X-Y
#define GS_WAVEAUDIO_LISSAJOUSXY                0x00000800
//! Surround monitor
#define GS_WAVEAUDIO_SURROUND                   0x00001000
#define GS_WAVEAUDIO_2                              0x00002000
#define GS_WAVEAUDIO_4                              0x00004000
#define GS_WAVEAUDIO_8                              0x00008000
//! Audio types mask
#define GS_WAVEAUDIO_MASK                           0x0000FE00
//
#define GS_WAVEVECTOR_FLAG_MASK                     0xFFFF0000
#define GS_WAVEVECTOR_ALT_GRATICULE             0x10000000
// Vectorscope graticule settings
#define GS_WAVEVECTOR_HIDE75VECTOR              0x01000000
#define GS_WAVEVECTOR_HIDE100VECTOR             0x02000000
#define GS_WAVEVECTOR_HIDEFLESHVECTOR0x04000000
#define GS_WAVEVECTOR_HIDEANGLES                0x08000000
// Waveform graticule settings
#define GS_WAVEVECTOR_USESMPTESCALE             0x00000000
#define GS_WAVEVECTOR_USEFULLSCALE              0x01000000
// Picture settings
#define GS_WAVEVECTOR_PICT_CLEAN                0x00000000
#define GS_WAVEVECTOR_PICT_SAFE                 0x01000000
#define GS_WAVEVECTOR_PICT_TITLE_SAFE   0x02000000

#define GS_WAVEVECTOR_CHANNEL_MASK     0x00FF0000
```

```
//! cmdGetSetValue::gsWaveVectorSetup dwStart color channel RED
#define GS_WAVEVECTOR_CHANNEL_R                 0x00010000
//! cmdGetSetValue::gsWaveVectorSetup dwStart color channel GREEN
#define GS_WAVEVECTOR_CHANNEL_G                 0x00020000
//! cmdGetSetValue::gsWaveVectorSetup dwStart color channel BLUE
#define GS_WAVEVECTOR_CHANNEL_B                 0x00040000
//! cmdGetSetValue::gsWaveVectorSetup dwStart color channel ALPHA
#define GS_WAVEVECTOR_CHANNEL_A                 0x00080000
//! cmdGetSetValue::gsWaveVectorSetup dwStart color channel Y (Luma)
#define GS_WAVEVECTOR_CHANNEL_Y                 0x00100000
//! cmdGetSetValue::gsWaveVectorSetup dwStart color channel CB
#define GS_WAVEVECTOR_CHANNEL_CB        0x00200000
//! cmdGetSetValue::gsWaveVectorSetup dwStart color channel CR
#define GS_WAVEVECTOR_CHANNEL_CR        0x00400000

//! cmdGetSetValue::gsTimecodeSources Don't use any external time code of any kind
#define GS_TCSRC_DISABLE_EXTERNAL       0x8000
//! cmdGetSetValue::gsTimecodeSources Use the RS-422 VTR time code
#define GS_TCSRC_FORCE_VTR_TC           0x0002
//! cmdGetSetValue::gsTimecodeSources Use the time of day as time code
#define GS_TCSRC_USE_TIMEOFDAY          0x0080

//! DirectX allow direct RGB plane (for cmdGetSetValue::gsVgaDirectXConfig)
#define GS_DXRGB_DIRECT                 0x001
//! DirectX allow overlay RGB plane (for cmdGetSetValue::gsVgaDirectXConfig)
#define GS_DXRGB_OVERLAY            0x002
//! DirectX allow overlay YUV plane (for cmdGetSetValue::gsVgaDirectXConfig)
#define GS_DXYUV_OVERLAY            0x004
//! DirectX allow direct YUV plane (for cmdGetSetValue::gsVgaDirectXConfig)
#define GS_DXYUV_DIRECT                 0x008

//! 3D VGA viewing mode MASK for all modes (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_TYPE_MASK                              0x00FFFFFF
//! 3D VGA viewing mode MASK for flags (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_FLAGS_MASK                             0xFF000000
//! 3D VGA view left eye only (top picture) (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_LEFTEYE                                0x00000001
//! 3D VGA view right eye only (bottom picture) (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_RIGHTEYE                               0x00000002
//! 3D VGA view comics red/blue (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_ANAGLYPH_REDBLUE               0x00000004
//! 3D VGA view 50s movie red/cyan (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_ANAGLYPH_REDCYAN               0x00000008
//! 3D VGA view 50s movie amber/blue  (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_ANAGLYPH_AMBERBLUE             0x00000010
//! 3D VGA view 50s movie green/magenta (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_ANAGLYPH_GREENMAGENTA      0x00000020
//! 3D VGA view interlaced (Zalman, real 3D, IMAX) (for
cmdGetSetValue::gsVga3DConfig)
```

```
#define GS_3DVGA_INTERLACED                                    0x00000040
//! 3D VGA view onion skin like 2D animation programs (for
cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_ONIONSKIN                                     0x00000080
//! 3D VGA view absolute difference (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_DIFFERENCE                                    0x00000100
//! 3D VGA view images on top and bottom (squeeze vert) (for
cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_OVERUNDER                                     0x00000200
//! 3D VGA view image next to each other (squeeze horiz) (for
cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_SIDEBYSIDE                                    0x00000400
//! 3D VGA view aribitrary split (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_SPLIT                                         0x00000800
//! 3D VGA view sqeeze and mirror (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_MIRROR                                        0x00001000
//! 3D VGA view invert right and split (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_BUTTERFLY                                     0x00002000
//! 3D VGA view difference above a certain threshold (for
cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_AMINUSB_THRESHOLD                 0x00004000
//! 3D VGA dissolve between (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_DISSOLVE                                      0x00008000
//! 3D VGA wipe (smpte +) (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_WIPE                                          0x00010000
//! 3D VGA luma invert diff (invert luma of second frame, then diff)
#define GS_3DVGA_LUMA_DIFF                                     0x00020000

//! 3D VGA 2D show component (0xFF000000 Alpha, 0x00FF0000 Red, 0x0000FF00
Green, 0x000000FF Blue, 0xFFFFFFFF Gray)
#define GS_3DVGA_2DSHOWCOMPONENT                   0x00800000

//! 3D VGA view flag to add a grid (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_FLAG_ADDGRID                      0x01000000
//! 3D VGA view flag to invert left/right (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_FLAG_INVERT                       0x80000000
//! 3D VGA view flag to flip one eye vertically (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_FLAG_FLIPVERT                     0x40000000
//! 3D VGA view flag to flip one eye vertically (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_FLAG_FLIPHORIZ                    0x20000000
//! 3D VGA view split flags make it a vertical split/mirror/butterfly
#define GS_3DVGA_FLAG_SPLITVERT                    0x10000000
//#define GS_3DVGA_FLAG_SPLITHORIZ                 0x00000000     // default
#define GS_3DVGA_FLAG_LENTICULAR          GS_3DVGA_FLAG_SPLITVERT

//! Command is not supported see cmdType::ctGetValue, cmdType::ctSetValue,
cmdType::ctValueSupported
#define GS_NOT_SUPPORTED 0xFFFFFFFF
/**
```

```
*  Parameter is bad see cmdType::ctGetValue, cmdType::ctSetValue,
cmdType::ctValueSupported and
* MEDIACMD::dwPosition, MEDIACMD::dwStart and MEDIACMD::dwEnd
*/
#define GS_BAD_PARAM          0xFFFFFFFE
/**
* False for boolean cmdType::ctGetValue, cmdType::ctSetValue
*/
#define GS_FALSE       0x00
/**
* True for boolean cmdType::ctGetValue, cmdType::ctSetValue
*/
#define GS_TRUE                  0x01
/**
* Disable a feature or command
*/
#define GS_DISABLE    0x00
/**
* Enable a feature or command
*/
#define GS_ENABLE     0x01
/**
* Default for cmdType::ctGetValue, cmdType::ctSetValue (usually in relation to VTR
setup)
*/
#define GS_DEFAULT   0xFF     /* Default set by user on device */
//! Use field cmdType::ctGetValue, cmdType::ctSetValue (for pause/freeze as opposed to
frame)
#define GS_FIELD       0x00     /* Frame or Field one (gs dependent) */
//! Use field 1 cmdType::ctGetValue, cmdType::ctSetValue (for record/playback starts
and edits)
#define GS_FIELD1      0x01     /* Field 1 - norm editing / single (1) freeze */
//! Use field 2 cmdType::ctGetValue, cmdType::ctSetValue (for record/playback starts
and edits)
#define GS_FIELD2      0x02     /* Field 2 - off editing / single (2) freeze */
//! Use frame cmdType::ctGetValue, cmdType::ctSetValue (for pause/freeze as opposed
to field)
#define GS_FRAME      0x03     /* Freeze frame */
//! Set value to unity (levels, tbc) or default (compression type, amount)
#define GS_UNITY        0xFFFFFFFF

//! AvHAL input set normal SDI or Analog single link
#define GS_ALPHACHROMA_SINGLE     0x01
//! AvHAL input set normal SDI plus a Y only SDI alpha plane
#define GS_ALPHACHROMA_ALPHA      0x02
//! Dual Link or HSDL input setup (2 HD-SDI 4:4:4 combined)
#define GS_ALPHACHROMA_DUAL                  0x04

//! Supports 8 bits per pixel component (normally YCbCr, for RGB see below)
```

```
#define GS_BITCOUNT_8                   0x01
//! Supports 10 bits per pixel component (normally YCbCr, for RGB see below)
#define GS_BITCOUNT_10                  0x02
//! Supports 3 (RGB) 8 bit components per pixel
#define GS_BITCOUNT_24                  0x04
//! Supports 3 (RGB) 10 bit components per pixel (e.g. standard DPX)
#define GS_BITCOUNT_30                  0x08
//! Supports 4 (RGBA) 8 bit components per pixel (e.g. standard TGA)
#define GS_BITCOUNT_32                  0x10
//! Supports YCbCr 4:2:0 (YUV) 8 bit components (e.g. i420, yv12)
#define GS_BITCOUNT_12                  0x20    /* 4:2:0 */

#define GS_FRAMEDROPMODE_NONE                       0x000000
#define GS_FRAMEDROPMODE_VARICAM_MASK_FPS   0x0000FF
#define GS_FRAMEDROPMODE_VARICAM_2398               0x000023
#define GS_FRAMEDROPMODE_VARICAM_24                 0x000024
#define GS_FRAMEDROPMODE_VARICAM_25                 0x000025
#define GS_FRAMEDROPMODE_VARICAM_2997               0x000029
#define GS_FRAMEDROPMODE_VARICAM_30                 0x000030
#define GS_FRAMEDROPMODE_VARICAM_50                 0x000050
#define GS_FRAMEDROPMODE_VARICAM_5994               0x000059
#define GS_FRAMEDROPMODE_VARICAM_60                 0x000060
#define GS_FRAMEDROPMODE_VARICAM_VARI               0x000001
#define GS_FRAMEDROPMODE_VARICAM_ILLEGAL    0x0000FF
#define GS_FRAMEDROPMODE_VARICAM_UB_INVERT  0x000100
#define GS_FRAMEDROPMODE_HALF                       0x010000
// This is used to send a command to the system from any driver
// Also used for returning current state
// e.g. RS-422, RS-232, Keyboard, User Interface, etc.
//Lets the last change know we need to reload the file
#define GS_FILE_HAS_CHANGED_REMOTELY        0xFFFFFFFD

#define GS_ONTRAK_NONE                  0x00000000  // all off
#define GS_ONTRAK_K0                    0x00000001  // start counter
#define GS_ONTRAK_K1                    0x00000002  // reset counter
#define GS_ONTRAK_K2                    0x00000004  // unused
#define GS_ONTRAK_K3                    0x00000008  // led on
#define GS_ONTRAK_ILLEGAL           0xFFFFFFFF
#define GS_ONTRAK_PA0                   0x00000001  // external pager
#define GS_ONTRAK_PA1                   0x00000002  // mark replay
#define GS_ONTRAK_PA2                   0x00000004  // unused
#define GS_ONTRAK_PA3                   0x00000008  // unused

#pragma pack(4)

#ifndef __midl
// This is used to make sure enum == 32 bits
#define COMPILE_ASSERT(x) extern int __dummy[(int)x]
//COMPILE_ASSERT(sizeof(cmdType) == sizeof(DWORD));
```

```
#endif

typedef struct /*tagCMD_QUEUE_ELEM*/ {
        // NOTE:  Must match D_LNODE
#ifndef __midl
        //! INTERNAL dlist.dll Link List Pointers - do not use
  void * pPrev; // Next lnode
        //! INTERNAL dlist.dll Link List Pointers - do not use
  void * pNext; // Prev lnode
        //! INTERNAL dlist.dll Link List Pointers - do not use
  void * pList;   // Parent or List owner
#elif defined(_WIN64)
  __int64 pPrev, pNext, pList;
#else
  DWORD pPrev, pNext, pList;
#endif
#ifndef __LP64__
        /** We need to make sure the MediaCMD structure is the same size
         * in 64 bit and 32 bit compiles.  The 3 x 64 bit pointers are
         * 24 bytes in total.  The 3 x 32 pointers are 12, so we have
         * 24 bytes added in the xxxPtr32Fills below.
         */
        DWORD dwPrevPtr32Fill, dwNextPtr32Fill, dwListPtr32Fill;
#endif
        /**
        * The command identifier is used to confirm that the command is properly
formatted and
        * of a version that the receiver understands.  This member should always be set
to #MEDIACMD_CURRENT
        */
        DWORD                   dwCmdID;
        /**
        * This member contains the entire size of the structure being sent.  Certain
commands may not
        * require all fields to be completely understood.  Most commands will send the
bulk of the
        * structure and remove only unused parts of arbID[], though this should not be
counted on.
        * It may be assumed that all commands will include the #pPrev, #pNext, #pList,
#dwCmdID,
        * #dwStructSize, #dwChannel, #ctCmd, and #cfFlags members with every
command so dwStructSize
        * must be at least 32 unsigned chars in length.
        */
        DWORD                   dwStructSize;
        /**
        * The target channel.  For direct connect channels (e.g.. in the same machine)
this member
        * is ignored.  For serial/network/piped channels, this allows a single transport to
```

```
        * access multiple channels.  It is also used for kernel<->user mode commands
for
        * DComCtrl.sys and vvwCtl.dll as well as vvwNet.dll.
        */
        DWORD                   dwChannel;              //
        /**
        * The ctCmd member contains the basic or overall command.  It is of the type
#cmdType.
        * It includes transport and setup commands that may be immediate or delayed
depending
        * on the rest of the structure.  Basic commands include:  cmdType::ctPlay,
        * cmdType::ctStop, cmdType::ctPause (or seek), cmdType::ctRecord,
cmdType::ctGetState,
        * cmdType::ctGetValue and cmdType::ctSetValue.
        * CAUTION:  This is an enum and we send this stucture as a binary between
various
        * systems.  Fortunately MSVC 2005/2008 and gcc 4.x both still make enums 32
bits, so
        * this works.  If we evey port to a 64 bit compiler that changes this, it will
        * break the binary compatibility.
        */
        enum cmdType ctCmd;                     // The command - enum added for ansi
'C' compiles
        /**
        * The cfFlags member contains flags that modify the operation of the other
structure
        * members.  These flags are of the type #cmdFlags.  The flags normally specify
which
        * other members are to be used for the command as well as modifiers for
delaying
        * or otherwise augmenting commands.  Basic flags include:
cmdFlags::cfDeferred,
        * cmdFlags::cfUseSpeed, cmdFlags::cfUsePosition, cmdFlags::cfUseStart,
cmdFlags::cfUseEnd,
        * cmdFlags::cfUsePositionOffset, cmdFlags::cfUseClipID.
        */
        DWORD                   cfFlags;        // Command flags
        /**
        * This member controls the speed of a command.  Normally it is used with
cmdType::ctPlay and
        * required cmdFlags::cfUseSpeed to be set in the #cfFlags member to be used.
The defines
        * #SPD_FWD_PLAY, #SPD_PAUSE, #SPD_REV_PLAY, #SPD_FWD_MAX,
#SPD_REV_MAX can be used, or any
        * other valid speed.  This table outlines the basic linear speed changes.
        \code
   SPD_REV_MAX     SPD_REV_PLAY    SPD_PAUSE       SPD_FWD_PLAY
SPD_FWD_MAX
   -5896800        -65520          0               65520           5896800
```

```
-90x          -1x           0            1x           90x
-9000%        -100%         0            100%         90000%
-90.0         -1.0          0            1.0          90.0

Some Normal Speeds (for reverse, set to minus)
10 times play   10.0    1000%   655200
5 times play    5.0     500%    327600
2 times Play    2.0     200%    131040
Play Normal     1.0     100%    65520
Two Third       0.66    66%     43680
Half Play       0.5     50%     32760
One Third       0.33    33%     21840

note:  Speed table is linear (log like serial control must be converted)
        \endcode
        */
        LONG            lSpeed;                 // '-'Reverse '0'pause '+'forward
        /**
        * VIDEO Bit array of channels where<BR>
        * 1 = first channel<BR>
        * 2 = second channel<BR>
        * 4 = third channel<BR>
        * 8 = fourth channel<BR>
        * Also, #cmdVidChan enum may be used
        */
        DWORD                   dwVideoChannels;// Video channels the cmd involves
        /**
        * AUDIO Bit array of channels where<BR>
        * 1 = first channel<BR>
        * 2 = second channel<BR>
        * 4 = third channel<BR>
        * 8 = fourth channel<BR>
        * Also, #cmdAudChan enum may be used
        */
        DWORD                   dwAudioChannels;// Audio channels the cmd involves
        /**
        * INFO Bit array if channels.  See #cmdinf for possible channels
        * including cmdinf::LTC, cmdinf::VITC, cmdinf::Copyright, etc.
        */
        DWORD                   dwInfoChannels; // Info channels the cmd involves
        /**
        * This member may have many meaning depending on the rest of the
        * MEDIACMD structure.  <BR>
        * if #ctCmd is cmdType::ctGetValue, cmdType::ctSetValue or
cmdType::ctValueSupported then it contains the #cmdGetSetValue to use
        * if #ctCmd is cmdType::ctTransfer it contains the source channel for the
transfer (the command is always set to the target)
        * if #cfFlags includes cmdFlags::cfUseCmdAlt and cmdFlags::cfTimeMs is
contains a millisecond version of the performance clock
```

```
        * if #ctCmd is cmdType::ctRecord then it may be used as a millisecond offset to
start of record
        * if #ctCmd is cmdType::ctPlay then it may be used as a millisecond offset to
start of playback
        */
        DWORD                   dwCmdAlt;               // Time delay, alternate channel
        /**
        * For most commands, this will be the current or target frame counter position
for the command.<BR>
        * Check of cmdFlags::cfUsePosition or cmdFlags::cfUsePositionOffset (becomes
long against current position) to make sure this member is valid.<BR>
        * For cmdType::ctGetValue, cmdType::ctSetValue or cmdType::ctValueSupported
this is primary set and return member.
        */
        DWORD                   dwPosition;             // Current or third edit point
position
        /**
        * For most commands, this will be the starting frame counter position for the
command.<BR>
        * Check of cmdFlags::cfUseStart or cmdFlags::cfUseStartOffset (becomes long
against current position) to make sure this member is valid.<BR>
        * For cmdType::ctGetValue, cmdType::ctSetValue or cmdType::ctValueSupported
this is secondary set and return member.
        */
        DWORD                   dwStart;                // Command start point in
frames,
        /**
        * For most commands, this will be the ending frame counter position for the
command.<BR>
        * Check of cmdFlags::cfUseEnd or cmdFlags::cfUseEndOffset (becomes long
against current position) to make sure this member is valid.<BR>
        * For cmdType::ctGetValue, cmdType::ctSetValue or cmdType::ctValueSupported
this is secondary set and return member.
        */
        DWORD                   dwEnd;                  // Command end point
(exclusive) in frames
        /**
        * Free form memory area mostly used for string and clip handling. Valid if the
        * #cfFlags member includes cmdFlags::cfUseClipID
        * Basic Clip Name:      first 8 unsigned charS alpha number, may include
spaces, terminated by 0<BR>
        * Two Clip Names:  'Basic Clip Name' followed immediately by another 'Basic Clip
Name'
        * Clip+File Name:  'Basic Clip Name' followed by a null terminated file/path
name.
        * String:          Null terminated ANSI string.
        */
        unsigned char           arbID[CMD_MAX_CLIP_ID_LEN+2+2];  // Odetics/Louth
Identifier
```

```
} MEDIACMD, *PMEDIACMD;

//! Old name, use MEDIACMD instead
#define CmdQueueElem MEDIACMD
//! Old name, use PMEDIACMD instead
#define pCmdQueueElem PMEDIACMD

#ifndef __LP64__
//! Declare a media cmd structure to all pause
#define DECLARE_MEDIACMD(__mCmd_) MEDIACMD __mCmd_ =      \
        {NULL, NULL, NULL,
        \
        0, 0, 0,
        \
        MEDIACMD_CURRENT, (DWORD)sizeof(MEDIACMD), CHAN_ILLEGAL,   \
        ctPause, 0UL, SPD_ILLEGAL,                                   \
        (DWORD)vidChanAll, (DWORD)audChanAll,                          \
        (DWORD)infChanAll, (DWORD)0,                                 \
        (DWORD)TC_ILLEGAL, (DWORD)TC_ILLEGAL, (DWORD)TC_ILLEGAL,   \
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,        \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \

                    \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,              \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,           \
        }};
#else
        // 64 bit
#define DECLARE_MEDIACMD(__mCmd_) MEDIACMD __mCmd_ =      \
        {NULL, NULL, NULL,
        \
        MEDIACMD_CURRENT, (DWORD)sizeof(MEDIACMD), CHAN_ILLEGAL,   \
        ctPause, 0UL, SPD_ILLEGAL,                                   \
        (DWORD)vidChanAll, (DWORD)audChanAll,                          \
        (DWORD)infChanAll, (DWORD)0,                                 \
        (DWORD)TC_ILLEGAL, (DWORD)TC_ILLEGAL, (DWORD)TC_ILLEGAL,   \
        {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,        \
```

```
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,                  \
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,             \
        }};
#endif
//! Initialize a media cmd structure to all illegal (no command)
#define INIT_MEDIACMD(__mCmd_)   {                                            \
        memset(&__mCmd_,0, sizeof(MEDIACMD));                          \
        __mCmd_.dwCmdID = MEDIACMD_CURRENT;                                   \
        __mCmd_.dwStructSize = sizeof(MEDIACMD);               \
        __mCmd_.dwChannel = CHAN_ILLEGAL;                                     \
        __mCmd_.lSpeed = SPD_ILLEGAL;                                         \
        __mCmd_.dwVideoChannels = (DWORD)vidChanAll;           \
        __mCmd_.dwAudioChannels = (DWORD)audChanAll;           \
        __mCmd_.dwInfoChannels = (DWORD)infChanAll;                    \
        __mCmd_.dwCmdAlt = 0;
        \
        __mCmd_.dwPosition = TC_ILLEGAL;                               \
        __mCmd_.dwStart = TC_ILLEGAL;                                         \
        __mCmd_.dwEnd = TC_ILLEGAL;        }

//! Initialize a media cmd pointer to all illegal (no command)
#define INIT_PMEDIACMD(__mCmd_) {
        \
        memset(__mCmd_, 0,sizeof(MEDIACMD));                                  \
        ((PMEDIACMD)__mCmd_)->dwCmdID = MEDIACMD_CURRENT;                     \
        ((PMEDIACMD)__mCmd_)->dwStructSize = sizeof(MEDIACMD);\
        ((PMEDIACMD)__mCmd_)->dwChannel = CHAN_ILLEGAL;                       \
        ((PMEDIACMD)__mCmd_)->lSpeed = SPD_ILLEGAL;                           \
        ((PMEDIACMD)__mCmd_)->dwVideoChannels = (DWORD)vidChanAll;            \
        ((PMEDIACMD)__mCmd_)->dwAudioChannels = (DWORD)audChanAll;            \
        ((PMEDIACMD)__mCmd_)->dwInfoChannels = (DWORD)infChanAll;             \
        ((PMEDIACMD)__mCmd_)->dwCmdAlt = CHAN_ILLEGAL;                        \
        ((PMEDIACMD)__mCmd_)->dwPosition = TC_ILLEGAL;                 \
        ((PMEDIACMD)__mCmd_)->dwStart = TC_ILLEGAL;                           \
        ((PMEDIACMD)__mCmd_)->dwEnd = TC_ILLEGAL;      }
```

```
#pragma pack()

//! SizeOf a command queue without the arbID
#define CMD_QUEUE_ELEMSIZE ((size_t)(&((pCmdQueueElem)(0))->arbID[0]))
//! SizeOf basic mediacmd structure without any clip id
#define SIZEOF_MEDIACMD_BASE CMD_QUEUE_ELEMSIZE
//! SizeOf mediacmd structure with a 8 unsigned char clip id and terminating 0
#define SIZEOF_MEDIACMD_CLIPID (CMD_QUEUE_ELEMSIZE + 9)
//! SizeOf a complete mediacmd structure
#define SIZEOF_MEDIACMD sizeof(MEDIACMD)

/**
* \page mediacmdfastinfo MEDIACMD Fast Info Page

  \section mediamcmdinto MEDIACMD Introduction

<BR>This document covers inter-device communication protocol used for control and
information exchange across a Drastic Media machine, Intranet (LAN) or Internet (WAN).
This document does not include information concerning low-level file structures, raw data
access, or hardware specific variations.
<BR>
<BR>Drastic uses an inter-process piping metaphor to allow the implementation of
media distribution and control over a small or large sized installation.  A media
distribution system normally includes the following components:
<BR>
\li Receiving and interpreting external time code based instructions (Sony/SMPTE/VDR
serial protocols)
\li Receiving and implementing external file or clip based instructions (Odetics/Alamar)
\li Configuration and local control (Graphical User Interface)
\li Storage management (Clip/File/Time Code/LTC/VITC/User)
\li Transport control (On time/Pre load/Pathing/QOS)
\li Hardware audio/video/compression control (Local/Effective Remote)
<BR>
<BR>Each of these components must be able to intercommunicate with one or more
other components to maintain control and status information within the processes or
systems.  To allow transportation of these command elements across a local or wide area
network, a protocol and transport method compatible with the running platforms and the
networks connecting them must be used.
<BR>
<BR>Once the protocol is in place, each of the drivers within a machine or network must
have an established command set to inter communicate.  The object of each component
of the system is to receive media commands, send media commands or control media
based on those commands.  This requirement is met by the drastic MediaCmd structure,
which is used for communication between all devices.
<BR>
<BR>Physical Transport
<BR>
```


The physical mechanisms for transporting commands throughout a system are encapsulated in the Drastic VVWNet library. The connection is in some ways similar to a Unix or NT, but does not actually use pipes. The VVWNet provides a simple interface for sending command packets between drivers within the same machine or different machines attached by a network. The VVWNet may use any underlying network protocol, but currently supports TCP/IP and IPX.

\section mediacmdlinks MEDIACMD QuickLinks
<CODE>

 The structure: MEDIACMD

 The commands: #cmdType

 The Flags: #cmdFlags

 Channels: #cmdVidChan, #cmdAudChan, #cmdinf

 GetSet Commands #cmdGetSetValue

 Basic cmdGetSetValue::gsTc

 Clip cmdGetSetValue::gsGetNextClip

 Channel cmdGetSetValue::gsAudChan

 Audio cmdGetSetValue::gsAudInSelect

 Video General cmdGetSetValue::gsVidFreeze

 Video Input cmdGetSetValue::gsVidInSelect

 Video TBC cmdGetSetValue::gsVidSetup

 Video Output cmdGetSetValue::gsVidOutSelect

 Signal Type cmdGetSetValue::gsSignalFormat

 Compression cmdGetSetValue::gsCompType

 Storage cmdGetSetValue::gsTotalStorageAvail

 Control cmdGetSetValue::gsLocal

 Info cmdGetSetValue::gsVVWVersion

 Internal cmdGetSetValue::gsSetHwnds

 Mode+Info cmdGetSetValue::gsChannelExist

 Returns #GS_NOT_SUPPORTED

 Declaration Local #DECLARE_MEDIACMD

 Init Local #INIT_MEDIACMD

 Init Pointer #INIT_PMEDIACMD

 Sizes #SIZEOF_MEDIACMD_BASE

</CODE>
\section mediacmdsamplecmds MEDIACMD Sample Commands

The following are sample commands as sent through media command. The samples are taken from the VVW series of products, and as such are guaranteed to work with them. Other OEMs and manufacturers supporting the MediaCmd interface may vary from this specification.

\b Simple \b Play
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 28
MEDIACMD::ctCmd = #ctPlay

\b Simple \b Pause

MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 28
MEDIACMD::ctCmd = #ctPause

\b Simple \b Stop
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 28
MEDIACMD::ctCmd = #ctStop

\b Play \b From-To
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctPlay
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd
MEDIACMD::dwStart = 1800 // 1 minute in frames NTSC NDF
MEDIACMD::dwEnd = 2100 // 1 minute 10 seconds NTSC NDF

\b Play \b DMC \b (Play \b at \b speed)
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 36
MEDIACMD::ctCmd = #ctPlay
MEDIACMD::cfFlags = #cfUseSpeed
MEDIACMD::lSpeed = 32760 // Half play speed forward

\b Record \b Edit
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctRecord
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd | #cfUsePresets
MEDIACMD::dwVideoChannels = 0x01 // Record V
MEDIACMD::dwAudioChannels = 0x03 // Record A1 & A2 (AA)
MEDIACMD::dwInfoChannels = 0x00
MEDIACMD::dwStart = 3000
MEDIACMD::dwEnd = 3200

\b Play \b Two \b Segments
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctPlay
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd
MEDIACMD::dwStart = 2200
MEDIACMD::dwEnd = 2500

MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctPlay
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd | #cfDeferred
MEDIACMD::dwStart = 5300
MEDIACMD::dwEnd = 5450

\b Play \b A \b Named \b Clip
Char szName = "C:\Adir\Afile.OMF"
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60 + strlen(szName)
MEDIACMD::ctCmd = #ctPlay
MEDIACMD::cfFlags = #cfUseClipID
strcpy(MEDIACMD::arbID , szName)

\b Seek \b To \b 1 \b Minute
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 48
MEDIACMD::ctCmd = #ctPause
MEDIACMD::cfFlags = #cfUsePosition
MEDIACMD::dwPosition = 1800 // Frames NTSC NDF

\b Step \b Reverse \b 1 \b Frame
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 48
MEDIACMD::ctCmd = #ctPause
MEDIACMD::cfFlags = #cfUsePositionOffset
MEDIACMD::dwPosition = (DWORD) -1

\b Record \b A \b 1 \b Minute \b Named \b Clip
Char szName = "C:\Adir\ArecordFile.MOV"
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60 + strlen(szName)
MEDIACMD::ctCmd = #ctRecord
MEDIACMD::cfFlags = #cfUseClipID | #cfUseEnd
MEDIACMD::dwEnd = 1800
strcpy(MEDIACMD::arbID , szName)

\b Record \b An \b Odetics \b or \b Louth \b Clip \b (ext)
(Note: The stop command is required for accuracy on some DDRs)
Char szName = "THISCLIP" // Max 8 char
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60 + strlen(szName)
MEDIACMD::ctCmd = #ctStop // Record ready
MEDIACMD::cfFlags = #cfUseClipID
strcpy(MEDIACMD::arbID , szName)

MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60 + strlen(szName)
MEDIACMD::ctCmd = #ctRecord
MEDIACMD::cfFlags = #cfUseClipID
strcpy(MEDIACMD::arbID , szName)

\b Eject \b The \b Current \b Media
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT

MEDIACMD::dwStructSize = 28
MEDIACMD::ctCmd = #ctEject

\b Setup \b The \b Video \b To \b Nominal
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctSetValue
MEDIACMD::dwCmdAlt = gsVidSetup
MEDIACMD::dwStart = 0

MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctSetValue
MEDIACMD::dwCmdAlt = gsVidVideo
MEDIACMD::dwStart = 0

MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctSetValue
MEDIACMD::dwCmdAlt = gsVidHue
MEDIACMD::dwStart = 0

MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctSetValue
MEDIACMD::dwCmdAlt = gsVidChroma
MEDIACMD::dwStart = 0

\b Check \b The \b Device \b Type
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctGetValue
MEDIACMD::dwCmdAlt = gsVtrType
MEDIACMD::dwStart = 0
(Returns: VTR/DDR/Media type in .dwStart)

\b Change \b The \b Default \b TC \b Type \b To \b VITC
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctSetValue
MEDIACMD::dwCmdAlt = gsVitcTc
MEDIACMD::dwStart = GS_DEFAULT

\b Transfer \b From \b External \b VTR \b To \b Internal \b Channel
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctTransfer
MEDIACMD::cfFlags = #cfUsePosition | #cfUseStart | #cfUseEnd | #cfUsePresets
MEDIACMD::dwVideoChannels = 0x01 // Capture Video

MEDIACMD::dwAudioChannels = 0x00 // No Audio
MEDIACMD::dwInfoChannels = 0x00
MEDIACMD::dwCmdAlt = hExternalChannel // The VTR channel
MEDIACMD::dwPosition = 2100 // Where to record to
MEDIACMD::dwStart = 10850 // Source In on VTR
MEDIACMD::dwEnd = 11000 // Source Out on VTR

\b Insert \b Media \b From \b File \b Over \b Current
Char szName = "C:\Adir\ArecordFile.MOV"
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60 + strlen(szName)
MEDIACMD::ctCmd = #ctInsert
MEDIACMD::cfFlags = #cfUsePosition | #cfUseStart | #cfUseEnd | #cfUseClipID
MEDIACMD::dwPosition = 2100 // Insert @ on target channel
MEDIACMD::dwStart = 0 // Start on source file
MEDIACMD::dwEnd = 180 // End on source file
strcpy(MEDIACMD::arbID , szName) // Name of source file

\b Delete \b A \b Section \b Of \b Media \b (No \b Hole)
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctDelete
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd | #cfRipple // Remove #cfRipple to leave hole
MEDIACMD::dwStart = 140500 // Start on target media
MEDIACMD::dwEnd = 150010 // End on target media

\b Trim \b A \b Clip
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 60
MEDIACMD::ctCmd = #ctTrim
MEDIACMD::cfFlags = #cfUsePosition | #cfUseStart | #cfUseEnd
MEDIACMD::dwPosition = 2100 // Clip @ position
MEDIACMD::dwStart = +32 // Clip 32 frames from start
MEDIACMD::dwEnd = (DWORD) -12 // Clip 12 frame from end

\b Terminate \b Session \b (Restart \b At \b Default)
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 28
MEDIACMD::ctCmd = #ctTerminate

\b Abort \b Current \b Command
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT
MEDIACMD::dwStructSize = 28
MEDIACMD::ctCmd = #ctAbort

* \section mediacmdsamplertns MEDIACMD Sample Returns

Sample Returns

MEDIACMD::dwCmdID == #MEDIACMD_CURRENT
MEDIACMD::dwStructSize == #SIZEOF_MEDIACMD
MEDIACMD::ctCmd == cmdType::ctPlay
MEDIACMD::cfFlags == 0
\b Normal \b Play \b (100% \b Play \b Speed)

MEDIACMD::dwCmdID == #MEDIACMD_CURRENT
MEDIACMD::dwStructSize == #SIZEOF_MEDIACMD
MEDIACMD::ctCmd == cmdType::ctRecord
MEDIACMD::cfFlags == 0
\b Normal \b Record \b (Crash \b Record)

*/

/*! \mainpage MediaCmd SDK
 *
 * \section intro_sec Introduction
 *
The Drastic MediaCmd SDK is the mechanism by which all the elements of Drastic's DDRs communicate with one and other. This includes:

* Controlling Drastic Titan and VVW Series DDR Servers, QuickClip locally, and QuickClip with network option remotely
* Controlling 9 pin serial VTRs and Servers via Sony, Odetics or VDCP protocol
* Receiving commands from 9 pin serial controller via Sony, Odetics or VDCP protocol
* Receiving commands from Drastic GUIs, servers and controllers
* Building HTML/Ajax status and control pages

MediaCmd is the communication method used within Drastic's DDR products. Any operation you see in a Drastic interface is available to your application through MediaCmd.

* \section overview_sec Overview

MediaCmd is a simple structure that supports a small, well defined set of commands for communicating transport, status and setup information between components in Drastic's DDR software. There are a number of fields in the structure, but the important fields are:

* ctCmd ñ the primary command of this packet (Play, Pause, Stop, Record, etc)
* lSpeed ñ the transport speed for any play commands (integer where 65520 = normal forward play)
* dwPosition ñ the frame position for any play, pause or record commands
* dwStart ñ the starting frame for any play or record commands (inclusive)
* dwEnd ñ the ending frame for any play or record commands (exclusive)
* arbID ñ clip name, file name or other string/binary data for the command

* cfFlags ñ denotes which fields above are valid and their meaning

With the standard initialization of the structure, you can quickly build commands in this structure by changing a few members and sending it. The primary motion commands are ctPlay, ctPause, ctStop, ctRecStop, ctRecord, ctEject and ctTransfer. To get the current state (position, speed, start and end, current clip), the command ctGetState will return a filled in MediaCmd. For setup and less common status (e.g. video input, audio rms level, genlock) there is ctGetValue and ctSetValue. This is documented in the Low Level Header Docs.

Hopefully, you will not have to deal with the MediaCmd structure directly. The SDK includes a series of simple commands that should provide 99% of what your application needs. These functions are simply wrappers that create and send MediaCmd structures. The source for all these functions is provided in the SDK under SRC/General/vvwIF.cpp in case you need to modify or create new commands. The commands have slightly different names depending on which interface you use, but have the same root name, such as: Play(), PlayFromTo(), Stop(), Pause(), Seek(), Record() and UpdateStatus(). Commands are also included for getting clip lists (GetNextClip()) and EDL elements from ::VTR_TC time code spaces (EDLResetToStart(), EDLGetEdit()). A selection of the most common settings are also included (SetVideoInput(), SetAudioInput(), SetVideoGenlock(), GetAudioPeakRMS(), etc). This interface is documented in the MediaCmd Documentation (previously called ìVVW Interface Specificationî).

* \section installation_sec Installation

To properly work with the MediaCmd SDK, you should have a copy of the QuickClip software installed on your development system. Even if your target application will only use a part of the QuickClip software, it should all be installed for the development phase. Before proceeding with the SDK you should familiarize your self with QuickClip's operation and toolset. All the elements available within QuickClip are the same elements available to your application through the SDK.
Once you have QuickClip installed, you should install the MediaCmd SDK. This will install the headers, libraries and source needed to control QuickClip from your application.

* \section accessmethod_sec Choosing An Access Method

The SDK access method you should use depends on what you you would like your application to do, what programming language you are using and how involved you would like to/need to get in the low level MediaCmd structures. No matter which method you choose, the MediaCmd structure packets are exactly the same. Here are the main access methods, with their pros and cons:

ActiveX
Type: Microsoft ActiveX/COM access method

Pros: Easy to program, 1:1 relationship with QuickClip/XO interface.
Cons: Uses same config as QuickClip/XO. Requires a local copy of QuickClip.
Setup: Register VVW.DLL using RegSvr32.exe in the QuickClip installation directory.
Issues: Difficult to use when communicating via TCP/IP within the same machine.
Can be overcome by using the default pipe communication system, but this requires changes for remote network control.

Direct Link
Type: Direct link to VVW.DLL
Pros: No ActiveX layer, code compatible with Linux, Irix, Mac OS-X.
Cons: Uses default config from QuickClip/XO, application must be run in QuickClip directory. Requires a local copy of QuickClip.
Setup: Link to vvw.lib, include vvw.h. Copy application into the QuickClip directory before running
Issues: Needs access to VVW.dll and all its support DLLs/D1Xs. Still needs to be setup by LocalConfig.exe or QuickClip/XO

Network DLL
Type: Direct line to vvwNet2.dll
Pros: Consistent interface between local/remote and various OSs. Does not require a local
copy of QuickClip.
Cons: Requires vvwNet2.dll and support dlls
Setup: Link to vvwNet2.lib, include vvwNet2.h. Copy dll set from SDK/bin directory with your application
Issues: Use the netOpenLocal function to avoid QuickClip configuration issues. Requires a few dDLLs to be added to your application installations. Does not run the client software automatically, so your application may need to start it, depending on what your application
is doing.

Network Direct
Type: Direct compile of network sources in your app or your DLL.
Pros: No extra dlls. Easy to customize and modify. Lots of commands already written.
Cons: You app needs to handle setup and may need to run QuickClip.exe/VVWServer.exe/QCRun.exe.
Setup: Copy source files from vvwNet2 into your project, modify and compile
Issues: Does not run the client software automatically, so your application may need to start it, depending on what your application is doing.

Manual
Type: Use the structures and defines to write your own communication and control layer.
Pros: This is required if you are using an unsupported development platform like PHP.
Cons: Everything has to be built and tested from the ground up.
Setup: None.
Issues: Unless you absolutely have to, this method is not recommended.

 * \section sdkstructure_sec SDK Structure

The location of the SDK directories will depend on the location you choose during the installation, but the directories within there will always be the same:

    * /BIN ñ Copies of the minimum dll set from a QuickClip installation.
    * /LIB ñ Libraries required to link the vvwNet2.dll, examples and your application
    * /INC ñ Header files required to compile vvwNet2.dll, examples and your application
    * /Src/vwwNet2 ñ The source to our vvwNet2.dll from QuickClip
    * /Src/General ñ Useful source files that do not compile into examples directly. The most important would be vvwIF.cpp that is the code behind the SDK functions described below.
    * /Sample ñ Broken down into sub directories based on access type
        o /ActiveX ñ Examples that use the ActiveX control
        o /Direct ñ Examples that link directly to DLLs
        o /Java ñ Java based examples
        o /HTTP ñ Ajax based examples (must use QuickClip HTTP server to run)

 * \section maindoclink_sec Main Documentation Links

*** PDF version of the MediaCmd Documentation
http://www.drastictech.com/manuals/VVW%20Interface%20Specification.pdf

*** Online version of the MediaCmd Documentation
http://www.drastictech.com/manuals/VVW%20Interface%20Specification.html

 * \section lowlevellink_sec Low Level Header Documentation Links

*** Windows CHM help file version of the MediaCmd headers
http://www.drastictech.com/manuals/MediaCmd.chm

*** Online version of the MediaCmd headers
http://www.drastictech.com/manuals/mediacmd/

 * \section htmlajaxlink_sec HTTP XML AJAX Documentation Links

*** Wiki area for HTTP XML MediaCmd
http://www.drasticpreview.org/wakka.php?wakka=DrasticHttpCommands&v=d9c
 *
 *
 */

/**
' Licensing Flags

```
*/
#endif //_MEDIACMD_INCLUDED_H
```

Appendix II – VVWIF.h

```
/
*************************************************************************
 * $Id$:
 *
 * $HeadURL$:
 * $Author$:
 * $Revision$:
 * $Date$:
 *
 * Copyright (c) 1998-2011 Drastic Technologies Ltd.  All Rights Reserved.
 * 523 The Queensway, Suite 102 Toronto ON M8V 1Y7
 * 416 255 5636  fax 255 8780
 * engineering@drastictech.com      http://www.drastictech.com
 *************************************************************************/

#ifdef _WIN32
        #include <direct.h>
        #ifndef _CRT_SECURE_NO_WARNINGS
                #define _CRT_SECURE_NO_WARNINGS
        #endif
#else
        #include <string.h>
        #include <sys/timeb.h>
        //#include "vvwtcpip.h"
        //#include "vvwnet2.h"
#endif

// Access functions exported from DLL
#ifdef __cplusplus
extern "C" {
#endif

/*
unsigned LONG __stdcall vvwOpenNetworkChannel(char * szAddress, unsigned LONG
dwPort, unsigned LONG dwChannel);
*/
/**
Get the current millisecond time.
*/
DWORD __stdcall vvwGetCurMs(LONG lChannel);

/**
* Convert a 0..3 channel to 0, 64, 65536
* Mostly internal
```

```
* Undocumented
*/
VVWHANDLE __stdcall vvwChannelToHandle(LONG lChannel);

/**
* Convert a 0, 64, 65536 to 0..3
* Mostly internal
* Undocumented
*/
LONG __stdcall vvwHandleToChannel(VVWHANDLE hVvw);

/**
Enable or disable channels based on the bit array supplied.  VVW can contain up to 256
channels per access point.  Channels 193-255 are disabled by default.  The remaining
channels may be enabled (if the corresponding bit is set to 1) or disabled (if the
corresponding bit is set to 0) with this call.  The first 64 channels (0 through 63) are
reserved for internal ddr channels.  Then next 64 channels (64 through 127) are reserved
for VTR or DDR devices controlled via serial, Odetics or Louth protocol.  The remaining
channels are for controlling other devices through the network.  Please note that a
network channel controls all the channels on the network server box, so disabling one
network connection may disable more than one channel.  Always call GetMaxChannels()
after setting the bits to make sure all the channels you expect exist actually exist.  This
should be the first call made to the activex component.
*/
LONG __stdcall vvwEnableChannels(LONG lInternal0_31,
                                 LONG lInternal32_63,
                                 LONG lExternal64_95,
                                 LONG lExternal96_127,
                                 LONG lNetwork128_159,
                                 LONG
lNetwork160_191);
/**
* Release memory allocated to channels
*/
LONG __stdcall vvwReleaseChannels(void);
/**
Returns the maximum number of channels available for control.  Channels start at 0 and
end at max channels – 1.  This return is one greater than the largest value available for
SetCurChannel(), GetCurChannel() and the lChannel parameter for the DLL interface.
*/
LONG __stdcall vvwGetMaxChannels(void);
/**
Returns the maximum number of internal channels available for control.  Channels start
at 0 and end at 62 – 1.  This return is one greater than the largest value available for
SetCurChannel(), GetCurChannel() and the lChannel parameter for the DLL interface.
*/
LONG __stdcall vvwGetMaxInternalChannels(void);
/**
```

```
Get the name of the current channel.  For unix and dll access, pass a null to get the
channel name size, then pass in a pointer that points to a memory size of at least that
many bytes (ANSI characters only).
*/
LONG __stdcall vvwGetChannelName(LONG lChannel, char * szChannelName);

/**
Returns the basic type of the channel (VTR, Internal, User, House)
        VVW_CHANTYPE_HOUSE              0x1             1
        VVW_CHANTYPE_INTERNAL           0x2             2
        VVW_CHANTYPE_VTR_DDR            0x4             4
        VVW_CHANTYPE_UNKNOWN            0xFFFFFFFF      -1
*/
LONG __stdcall vvwGetChannelType(LONG lChannel);

/**
Show the configuration dialog box for the current channel.  If the channel does not have
a configuration dialog, this function will return an error.  It is not available in Java or unix
as the dialog only shows up on the local machine, and cannot be seem through the
network.
*/
LONG __stdcall vvwShowConfigDialog(LONG lChannel, LONG hWnd);

//
// Transport
//

#define vvwWaitForState(__hChan_,__ctCmd_)
                        \
                        LONG __nDur_ = 1, __nWait_ = 0;
                                \
                        while(vvwGetState(__hChan_) != __ctCmd_ && __nWait_ <
500){   \
                                Sleep(__nDur_); __nWait_++;
                                \
                        }
                                                        \
/**
Play at normal speed.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwPlay(LONG lChannel);

/**
Play at a particular VVW speed.  VVW speeds use a base play speed of 65520.  This
means that play = 65520, reverse play = -65520, four times play = 262080, half play
speed = 32760.  Percentage play speeds may be converted to VVW speeds using the
PercentageToVVWSpeed() function. For Speed calculations please see GetSpeed() below.
        Returns 0 if successful, else an error code.
```

```
*/
LONG __stdcall vvwPlayAtSpeed(LONG lChannel, LONG lVVWSpeed);

/**
Play from a frame to another frame.  As with editing systems, the 'from' point is included
and will be displayed but the to point is NOT included and will not be displayed.  This
means that the last frame displayed will be lFrom – 1.  The deferred flag allows

PlayFromTos to be stacked so that they will play back to back.  The deferred flag in the
status return should be false before another deferred command is added.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwPlayFromTo(LONG lChannel, LONG lFrom, LONG lTo, int fDeferred);

/**
Clip Mode Only.  Load a clip into the channel and display the lStartFrame.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwLoadClip(LONG lChannel, char * sz8CharClipName, LONG
lStartFrame);

/**
Clip Mode Only.  Switch to a different clip without changing play/pause mode.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwSwitchClip(LONG lChannel, char * sz8CharClipName, LONG lPosition,
BOOL bUseFrameCount);

/**
Clip Mode Only.  Play the entire clip specified by clip name.  If the deferred flag is true,
clip playback will only occur once the currently playing clip has finished.  If there is no
currently playing clip, playback will occur immediately.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwPlayClip(LONG lChannel, char * sz8CharClipName, int fDeferred);

/**
Clip Mode Only.  Play the specified portion of the clip specified by clip name.  If the
deferred flag is true, clip playback will only occur once the currently playing clip has
finished.  If there is no clip currently playing, playback will occur immediately.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwPlayClipFromTo(LONG lChannel, char * sz8CharClipName, LONG
lFrom, LONG lTo, int fDeferred);

/**
Set the channel into its fastest possible forward motion state.
        Returns 0 if successful, else an error code.
*/
```

```
LONG __stdcall vvwFastForward(LONG lChannel);

/**
Set the channel into its fastest possible reverse motion state.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwFastRewind(LONG lChannel);

/**
Stop playback and display the current frame.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwPause(LONG lChannel);

/**
Seek to a particular frame and display it to the user.  This call will return before the seek
is complete.  Once the Position return in the status reaches the lFrame, the seek is
complete.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwSeek(LONG lChannel, LONG lFrame);

/**
Seek a certain number of frames from the current position.  Positive offsets imply
forward direction, negative offset imply reverse.
*/
LONG __stdcall vvwSeekRelative(LONG lChannel, LONG lFrameOffset);

/**
Stop the output of the controlled channel and display the input video (not supported on
all devices). On unsupported devices stop will be the same as a pause.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwStop(LONG lChannel);

/**
Start the channel recording. In clip mode a default clip name will be used with a duration
set to infinity. The record will stop on any transport command or at the point that the
disk is full.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwRecord(LONG lChannel);

/**
Record from a frame value to a frame value. As with editing systems, the 'from' point is
included and will be recorded but the to point is NOT included and will not be recorded.
This means that the last frame recorded will be lFrom – 1.
        Returns 0 if successful, else an error code.
*/
```

```
LONG __stdcall vvwRecordFromTo(LONG lChannel, LONG lFrom, LONG lTo);

/**
Clip Mode Only.  Set the clip name and length of time to record in frames.  The record
will not actually start until Record() is called.  If the lDuration is set to –1 the record will
continue until Stop() is called or the channel runs out of space.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwRecordStop(LONG lChannel, char * sz8CharClipName, LONG
lDuration);
/**
Clip Mode Only.  Set the clip name filename and length of time to record in frames.  The
record will not actually start until Record() is called.  If the lDuration is set to –1 the
record will continue until Stop() is called or the channel runs out of space.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwRecordStopFileName(LONG lChannel, char * sz8CharClipName, char
* sz256CharFileName, LONG lDuration);
/**
Set the channels to record. Using –1 values implies that the Preset should be set to all
available channels. Record Presets will remain set until the user changes them.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwSetRecordPresets(LONG lChannel, LONG lVidEdit, LONG lAudEdit,
LONG lInfEdit);

/**
Reset Record file and/or delete all files found in the record folder
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwCleanRecordWipeDrive(LONG lChannel, LONG lWipeFolder);

/**
Eject the current media if it is removable.  Normally only used with VTRs.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwEject(LONG lChannel);

/**
* MediaCmd direct access
*/
LONG __stdcall vvwMediaCmd(LONG lChannel, void * pMediaCmd);

//
// Special Commands (lChannel must be vtr type, lTargetChannel must be internal type)
//
/**
Transfer media from one channel to another.  Only supported by VTR channels.
Currently only implemented for VTR to internal channels or internal channels to VTR
```

```
channels.  To record media from a VTR, the fToTape should be false, to record media
onto a VTR the fToTape should be true. The start and end point are from the playback
device. The edit will occur at the current timecode location on the recorder.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwTransfer(LONG lChannel, LONG lTargetChannel, LONG lPosition,
LONG lStart, LONG lEnd, LONG lVidEdit, LONG lAudEdit, LONG lInfEdit, char *
szClipName, int fToTape);

/**
Retrieve the current status from the controlled device.  The status is automatically
updated by the interface, but this call ensures that the status is current when you are
checking it.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwUpdateStatus(LONG lChannel);

/**
Returns the current state

ctStop          0               // Stop all action
ctPause         1               // Pause, Seek
ctPlay          2               // Play at specified speed (includes pause)
ctRecord        3               // Record at specified speed
ctRecStop       4               // Stop ready for recording
ctEject         5               // Eject the current media
ctError         17              // An error has occurred
ctAbort         19              // Abort any queued commands
*/
LONG __stdcall vvwGetState(LONG lChannel);

/**
        Returns the current flags

cfDeferred = 1,                         // 0x00000001 This is a delayed
cfOverrideDeferred = 1 << 30,// 0x40000000 Overide all previous deferred
commands
cfTimeMs = 1 << 1,                      // 0x00000002 Use Millisecond time for delayed
time, not fields
cfTimeTarget = 1 << 2,                  // 0x00000004 Delayed time is offset from
current time code
cfTimeHouseClock = 1 << 3,    // 0x00000008 Delayed time is based on absolute (real)
time
cfUseSpeed = 1 << 4,            // 0x00000010 Set the new speed
cfUsePresets = 1 << 5,          // 0x00000020 Use video and audio edit presets
cfUsePosition = 1 << 6,                 // 0x00000040 Use the position setting
cfUsePositionOffset = 1 << 7,  // 0x00000080 Position is an offset
cfUseStart = 1 << 8,            // 0x00000100 Start a new timecode
cfUseStartOffset = 1 << 9,      // 0x00000200 Start is an offset from current tc
```

```
cfUseEnd = 1 << 10,                     // 0x00000400 End command as specified
cfUseEndOffset =  1 << 11,      // 0x00000800 End is and offset from current tc
cfUseAllIDs = 1 << 12,          // 0x00001000 Use all clip IDs
cfUseClipID = 1 << 13,          // 0x00002000 Use new clip ID, otherwise use last or
none
cfNoClipFiles = 1 << 14,        // 0x00004000 Use new clip ID, otherwise use last or
none
cfNoTCSpaces = 1 << 15,                 // 0x00008000 Use new clip ID, otherwise use
last or none
cfUseCmdAlt = 1 << 16,                  // 0x00010000 Use the dwCmdAlt
cfIsShuttle = 1 << 17,          // 0x00020000 Use speed in play for shuttle
cfFields = 1 << 20,                     // 0x00100000 Position, start and end are fields,
not frames
cfRipple = 1 << 21,                     // 0x00200000 Ripple for insert or delete
cfLoop = 1 << 22,                       // 0x00400000 Loop the clip or in out
cfTrigger = 1 << 23,            // 0x00800000 Trigger using dsync class
cfPreview = 1 << 24,            // 0x01000000 Preview set (EE, non rt play)
cfInvert = 1 << 28,                     // 0x10000000 Invert a transfer
cfTest = 1 << 29,                       // 0x20000000 See if the command exists
cfNoReturn = 1 << 31,           // 0x80000000 No return mediacmd is required
*/
LONG __stdcall vvwGetFlags(LONG lChannel);

/**
Returns the current VVW speed if the cfUseSpeed flag is set, otherwise pause or full play
speed.  VVW speeds are based on 65520 as the play speed.  To translate to decimal
number where 1.0 represents play, use the following formula:
        D1Speed = ((double)VVWSpeed / 65520.0)
For percentages, where 100.0 represents play speed, use the following formula:
        Dpercent = (((double)VVWSpeed * 100.0) / 65520.0)
                = ((double)VVWSpeed / 655.2)
XML: See \<MediaCmd\> root element, \<Speed\> sub-element

Typical VVW speeds (note speeds are linear):
Pause           0%              0
Play            100%    65520
Half Play       50%             32760
Rev Play        -100%   -65520
Rev 2 x Play-200%       131040
10 x Play       1000%   655200
Max Play        90000%          5896800
Max Rev                 -90000%-5896800
*/
LONG __stdcall vvwGetSpeed(LONG lChannel);

/**
Returns the current position if the cfUsePosition flag is set, otherwise invalid.
*/
LONG __stdcall vvwGetPosition(LONG lChannel);
```

```
/**
Returns the millisecond time the last status occurred (time of the last vertical blank).
*/
LONG __stdcall vvwGetLastMs(LONG lChannel);

/**
Returns the current start or in point if the cfUseStart flag is set.
*/
LONG __stdcall vvwGetStart(LONG lChannel);

/**
Return the current end point or out point if cfUseEnd is set.
*/
LONG __stdcall vvwGetEnd(LONG lChannel);

/**
Only supported in clip Mode. Returns the current clip name, if any.  For dll access, the
memory must be at least 9 bytes LONG (8 character bytes + NULL) and is always ANSI.
*/
LONG __stdcall vvwGetClipName(LONG lChannel, char * sz8CharClipName);

/**
Returns the current file name, if any.  For dll access, the memory must be at lease 261
bytes LONG (260 bytes max path + NULL) and is always ANSI.
*/
LONG __stdcall vvwGetFileName(LONG lChannel, char * sz260CharFileName);

/**
Returns the current time code as a string (e.g. "00:01:00:00").  For dll access, the
memory must always be at least 15 bytes LONG (14 byte time code plus id + NULL) and
is always ANSI.
*/
LONG __stdcall vvwGetCurTC(LONG lChannel, char * sz14ByteTC);

/**
Returns the current state as a string (e.g. "Play"). For dll access, the memory must
always be at least 15 bytes LONG (14 byte state + NULL) and is always ANSI.
*/
LONG __stdcall vvwGetCurState(LONG lChannel, char * sz14ByteState);

//
// Media Operation
//

// Clip Mode
/**
Clip Mode Only.  Returns the next clip identifier.  To get the first clip, szLastClip should
be an empty string.  Once the last clip available has been returned, GetNextClip will
```

```
return an error or NULL for unix/dll access.  Please note:  For unix/dll access, the
sz8CharLastClipCurClip memory area is used for the new clip.  The previous clip name is
therefore lost and the memory is not allocated by the vvw.
        Returns 0 if successful, else an error code.
*/
char * __stdcall vvwGetNextClip(LONG lChannel, char * sz8CharLastClipCurClip);

/**
Retuns the basic information from szClip.  The information is located in lStart, lEnd,
lVidEdit, lAudEdit and szFileName as the in point, out point, number of video channels,
number of audio channels, and the file name respectively.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwGetClipInfo (LONG lChannel, char * sz8CharClipName, LONG * lStart,
LONG * lEnd, LONG * lVidEdit, LONG * lAudEdit, LONG * lInfEdit, char * szFileName);

/**
Sets the meta data for szClip.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwSetMetaData (LONG lChannel, char * sz8CharClipName, LONG
vvwiInfoRequest, LONG nValue, char * szValue);

/**
Retuns the meta data from szClip.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwGetMetaData (LONG lChannel, char * sz8CharClipName, char *
sz260CharFileName, LONG vvwiInfoRequest, char * szValue);

/**
Retuns the extended information from szClip.  The information is located in lStart, lEnd,
lVidEdit, lAudEdit and szFileName as time of creation, last modified date, the file size,
and the number of fragments in the file respectively.
*/
LONG __stdcall vvwGetNextClipEx (LONG lChannel, char * sz8CharClipName, LONG *
lCreation, LONG * lLastModification, LONG * lFileSize, LONG * lDiskFragments);

/**
Create a virtual copy of a clip, changing the in and out points if necessary.  To use the
whole clip, set lStart to 0 and the end to –1.
        Returns 0 if successful, else an error code.
*/
LONG __stdcall vvwCopyClip (LONG lChannel, char * szSourceClip, char * szDestClip,
LONG lStart, LONG lEnd);

// VTR Mode
/**
Reset the edl returns in VTR mode to the first element of the list.
```

```
*/
LONG __stdcall vvwEDLResetToStart(LONG lChannel);

/**
Returns an edit line from the VTR space of an internal channel. The function will continue
to return the next edit in the timecode space until the last edit is returned, after which an
error will be returned. To reset to the start of the Edl use EDLResetToStart.
        Returns 0 if successfule else an Error code.
*/
LONG __stdcall vvwEDLGetEdit (LONG lChannel, LONG * lRecordIn, LONG * lPlayIn,
LONG * lPlayOut, LONG * lVidEdit, LONG * lAudEdit, LONG * lInfEdit, char *
sz8CharClipName, char * sz260CharFileName, bool bClipInfo);



// Shared
/**
Returns the millisecond time of the last change in the transfer queue
*/
LONG __stdcall vvwGetLastChangeXferMs(LONG lChannel);
/**
Returns the millisecond time of the last change in the current mode (clip or vtr).
*/
LONG __stdcall vvwGetLastChangeMs(LONG lChannel, LONG lClipSpace, LONG *
lNumClips);

/**
*/
LONG __stdcall vvwInsert (LONG lChannel, char * szClipName, char * szFileName, LONG
lPosition, LONG lStart, LONG lEnd, LONG lVidEdit, LONG lAudEdit, LONG lInfEdit, int
fRipple);

/**
*/
LONG __stdcall vvwBlank (LONG lChannel, char * szClipName, LONG lStart, LONG lEnd,
LONG lVidEdit, LONG lAudEdit, LONG lInfEdit, int fRipple);

/**
*/
LONG __stdcall vvwDelete (LONG lChannel, char * szClipName, LONG lStart, LONG lEnd,
LONG lVidEdit, LONG lAudEdit, LONG lInfEdit, int fRipple);

/**
*/
LONG __stdcall vvwTrim (LONG lChannel, LONG lPosition, LONG lStartOffset, LONG
lEndOffset, LONG lVidEdit, LONG lAudEdit, LONG lInfEdit, int fRipple);

//
// Settings
```

```
//
/**
Returns the supported attributes of a get/set value  (gsClipMode, gsTcSource, etc) or –1
for not supported.
*/
LONG __stdcall vvwValueSupported(LONG lChannel, LONG lValueType);

/**
Returns the current setting for a get/set value.
*/
LONG __stdcall vvwValueGet(LONG lChannel, LONG lValueType, LONG * plMin, LONG *
plMax);

/**
Sets the get/set value to setting.
*/
LONG __stdcall vvwValueSet(LONG lChannel, LONG lValueType, LONG lSetting);

/**
Sets the get/set value to setting with extended parameters.  Please set unused
parameters to NULL.
*/
LONG __stdcall vvwValueSet2(LONG lChannel, LONG lValueType, LONG lSetting, LONG
lStart, LONG lEnd, LONG lVidChan, LONG lAudChan, LONG lInfChan);

/**
Calls ValueXXX with gsClipMode.  If equal to 1 then the channel is in clip mode, if 0 the
channel is in VTR mode.
*/
LONG __stdcall vvwGetClipMode(LONG lChannel);
/**
Calls ValueXXX with gsClipMode.  If equal to 1 then the channel is in clip mode, if 0 the
channel is in VTR mode.
*/
LONG __stdcall vvwSetClipMode(LONG lChannel, LONG lSetting);

/**
Calls ValueXXX with gsTcType (drop frame, non drop frame, pal).
#TC2_TCTYPE_FILM, #TC2_TCTYPE_NDF, #TC2_TCTYPE_DF, #TC2_TCTYPE_PAL,
#TC2_TCTYPE_50, #TC2_TCTYPE_5994, #TC2_TCTYPE_60, #TC2_TCTYPE_NTSCFILM,
#TC2_TCTYPE_2398, #TC2_TCTYPE_100
*/
LONG __stdcall vvwGetTCType(LONG lChannel);
/**
Calls ValueXXX with gsTcType (drop frame, non drop frame, pal).
#TC2_TCTYPE_FILM, #TC2_TCTYPE_NDF, #TC2_TCTYPE_DF, #TC2_TCTYPE_PAL,
#TC2_TCTYPE_50, #TC2_TCTYPE_5994, #TC2_TCTYPE_60, #TC2_TCTYPE_NTSCFILM,
#TC2_TCTYPE_2398, #TC2_TCTYPE_100
*/
```

```
LONG __stdcall vvwSetTCType(LONG lChannel, LONG lSetting);

/**
Calls ValueXXX with gsTcSource (VITC, LTC, Control, Clip).
#GS_TCSOURCE_LTC, #GS_TCSOURCE_VITC, #GS_TCSOURCE_CTL or
#GS_TCSOURCE_CLIP
*/
LONG __stdcall vvwGetTCSource(LONG lChannel);
/**
Calls ValueXXX with gsTcSource (VITC, LTC, Control, Clip).
#GS_TCSOURCE_LTC, #GS_TCSOURCE_VITC, #GS_TCSOURCE_CTL or
#GS_TCSOURCE_CLIP
*/
LONG __stdcall vvwSetTCSource(LONG lChannel, LONG lSetting);

/**
Calls ValueXXX with gsAutoMode.  Required for play lists, deferred commands and auto
edit commands on VTRs.
*/
LONG __stdcall vvwGetAutoMode(LONG lChannel);
/**
Calls ValueXXX with gsAutoMode.  Required for play lists, deferred commands and auto
edit commands on VTRs.
*/
LONG __stdcall vvwSetAutoMode(LONG lChannel, LONG lSetting);


/**
ADD FUNCTIONS lVidEdit, lAudEdit, lInfEdit
        Returns the curent audio, video and info presets for a channel.
*/
LONG __stdcall vvwGetCurrentPresets(LONG lChannel, LONG * plVidEdit, LONG *
plAudEdit, LONG * plInfEdit);
/**
ADD FUNCTIONS lVidEdit, lAudEdit, lInfEdit
        Returns the supported audio, video and info presets for a channel.
*/
LONG __stdcall vvwGetAvailablePresets(LONG lChannel, LONG * plVidEdit, LONG *
plAudEdit, LONG * plInfEdit);

/**
ADD FUNCTION lAudIn
        Get the current audio input.
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
#GS_AUDSELECT_BALANCED_10       #GS_AUDSELECT_BALANCED_4
#GS_AUDSELECT_SPDIF #GS_AUDSELECT_AES_EBU
#GS_AUDSELECT_EMBEDDED
*/
LONG __stdcall vvwGetAudioInput(LONG lChannel);
```

```
/**
ADD FUNCTION lAudIn
        Set the current audio input.
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
#GS_AUDSELECT_BALANCED_10       #GS_AUDSELECT_BALANCED_4
#GS_AUDSELECT_SPDIF #GS_AUDSELECT_AES_EBU
#GS_AUDSELECT_EMBEDDED
*/
LONG __stdcall vvwSetAudioInput(LONG lChannel, LONG lSetting);

/**
Get the current audio input level
*/
LONG __stdcall vvwGetAudioInputLevel(LONG lChannel);
/**
Get the current audio input level
*/
LONG __stdcall vvwSetAudioInputLevel(LONG lChannel, LONG lSetting);

/**
Get the current audio Output – See Get/SetAudioInput
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
#GS_AUDSELECT_BALANCED_10       #GS_AUDSELECT_BALANCED_4
#GS_AUDSELECT_SPDIF #GS_AUDSELECT_AES_EBU
#GS_AUDSELECT_EMBEDDED
*/
LONG __stdcall vvwGetAudioOutput(LONG lChannel);
/**
Set the current audio Output – See Get/SetAudioInput
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
#GS_AUDSELECT_BALANCED_10       #GS_AUDSELECT_BALANCED_4
#GS_AUDSELECT_SPDIF #GS_AUDSELECT_AES_EBU
#GS_AUDSELECT_EMBEDDED
*/
LONG __stdcall vvwSetAudioOutput(LONG lChannel, LONG lSetting);

/**
Get the current audio output level.
*/
LONG __stdcall vvwGetAudioOutputLevel(LONG lChannel);
/**
Get the current audio output level.
*/
LONG __stdcall vvwSetAudioOutputLevel(LONG lChannel, LONG lSetting);

/**
Returns the RMS and Peak audio levels of the input (stop/record) or output (play/pause).
*/
LONG __stdcall vvwGetAudioPeakRMS(LONG lChannel, LONG lAudEdit, LONG * plPeaks);
```

```
/**
Get the current video input.
#GS_VIDSELECT_COMPOSITE, #GS_VIDSELECT_COMPOSITE_2,
#GS_VIDSELECT_SVIDEO,
#GS_VIDSELECT_COMPONENT_YUV, #GS_VIDSELECT_COMPONENT_YUV_M2,
#GS_VIDSELECT_COMPONENT_YUV_SMPTE, #GS_VIDSELECT_COMPONENT_RGB,
#GS_VIDSELECT_D1_SERIAL, #GS_VIDSELECT_D1_PARALLEL, #GS_VIDSELECT_SDTI,
#GS_VIDSELECT_NONE
*/
LONG __stdcall vvwGetVideoInput(LONG lChannel);
/**
Set the current video input.
#GS_VIDSELECT_COMPOSITE, #GS_VIDSELECT_COMPOSITE_2,
#GS_VIDSELECT_SVIDEO,
#GS_VIDSELECT_COMPONENT_YUV, #GS_VIDSELECT_COMPONENT_YUV_M2,
#GS_VIDSELECT_COMPONENT_YUV_SMPTE, #GS_VIDSELECT_COMPONENT_RGB,
#GS_VIDSELECT_D1_SERIAL, #GS_VIDSELECT_D1_PARALLEL, #GS_VIDSELECT_SDTI,
#GS_VIDSELECT_NONE
*/
LONG __stdcall vvwSetVideoInput(LONG lChannel, LONG lSetting);

/**
Get the current video output. See Get/SetVideoInput for settings.
#GS_VIDSELECT_COMPOSITE, #GS_VIDSELECT_COMPOSITE_2,
#GS_VIDSELECT_SVIDEO,
#GS_VIDSELECT_COMPONENT_YUV, #GS_VIDSELECT_COMPONENT_YUV_M2,
#GS_VIDSELECT_COMPONENT_YUV_SMPTE, #GS_VIDSELECT_COMPONENT_RGB,
#GS_VIDSELECT_D1_SERIAL, #GS_VIDSELECT_D1_PARALLEL, #GS_VIDSELECT_SDTI,
#GS_VIDSELECT_NONE
*/
LONG __stdcall vvwGetVideoOutput(LONG lChannel);
/**
Set the current video output. See Get/SetVideoInput for settings.
#GS_VIDSELECT_COMPOSITE, #GS_VIDSELECT_COMPOSITE_2,
#GS_VIDSELECT_SVIDEO,
#GS_VIDSELECT_COMPONENT_YUV, #GS_VIDSELECT_COMPONENT_YUV_M2,
#GS_VIDSELECT_COMPONENT_YUV_SMPTE, #GS_VIDSELECT_COMPONENT_RGB,
#GS_VIDSELECT_D1_SERIAL, #GS_VIDSELECT_D1_PARALLEL, #GS_VIDSELECT_SDTI,
#GS_VIDSELECT_NONE
*/
LONG __stdcall vvwSetVideoOutput(LONG lChannel, LONG lSetting);


/**
Get the current video input's 'Setup' TBC setting.
*/
LONG __stdcall vvwGetVideoInputSetup(LONG lChannel);
/**
```

```
Set the current video input's 'Setup' TBC setting.
*/
LONG __stdcall vvwSetVideoInputSetup(LONG lChannel, LONG lSetting);
/**
Get the current video input's 'Video' TBC setting.
*/
LONG __stdcall vvwGetVideoInputVideo(LONG lChannel);
/**
Set the current video input's 'Video' TBC setting.
*/
LONG __stdcall vvwSetVideoInputVideo(LONG lChannel, LONG lSetting);
/**
Get the current video input's 'Hue' TBC setting.
*/
LONG __stdcall vvwGetVideoInputHue(LONG lChannel);
/**
Set the current video input's 'Hue' TBC setting.
*/
LONG __stdcall vvwSetVideoInputHue(LONG lChannel, LONG lSetting);
/**
Get the current video input's 'Chroma' TBC setting.
*/
LONG __stdcall vvwGetVideoInputChroma(LONG lChannel);
/**
Set the current video input's 'Chroma' TBC setting.
*/
LONG __stdcall vvwSetVideoInputChroma(LONG lChannel, LONG lSetting);

/**
Get the current global TBC's 'Setup' setting.
*/
LONG __stdcall vvwGetVideoTBCSetup(LONG lChannel);
/**
Set the current global TBC's 'Setup' setting.
*/
LONG __stdcall vvwSetVideoTBCSetup(LONG lChannel, LONG lSetting);
/**
Get the current global TBC's 'Video' setting.
*/
LONG __stdcall vvwGetVideoTBCVideo(LONG lChannel);
/**
Set the current global TBC's 'Video' setting.
*/
LONG __stdcall vvwSetVideoTBCVideo(LONG lChannel, LONG lSetting);
/**
Get the current global TBC's 'Hue' setting.
*/
LONG __stdcall vvwGetVideoTBCHue(LONG lChannel);
/**
```

```
Set the current global TBC's 'Hue' setting.
*/
LONG __stdcall vvwSetVideoTBCHue(LONG lChannel, LONG lSetting);
/**
Get the current global TBC's 'Chroma' setting.
*/
LONG __stdcall vvwGetVideoTBCChroma(LONG lChannel);
/**
Set the current global TBC's 'Chroma' setting.
*/
LONG __stdcall vvwSetVideoTBCChroma(LONG lChannel, LONG lSetting);

/**
Turn the house/reference lock on or off
*/
LONG __stdcall vvwGetVideoGenlock(LONG lChannel);
/**
Turn the house/reference lock on or off
*/
LONG __stdcall vvwSetVideoGenlock(LONG lChannel, LONG lSetting);

/**
Set the genlock source to input or external reference
*/
LONG __stdcall vvwGetVideoGenlockSource(LONG lChannel);
/**
Set the genlock source to input or external reference
*/
LONG __stdcall vvwSetVideoGenlockSource(LONG lChannel, LONG lSetting);

/**
Get the current compression rate
*/
LONG __stdcall vvwGetCompressionRate(LONG lChannel);
/**
Set the current compression rate
*/
LONG __stdcall vvwSetCompressionRate(LONG lChannel, LONG lSetting);

/**
Get the status of the super imposed time code overlay
*/
LONG __stdcall vvwGetSuperImpose(LONG lChannel);
/**
Set the status of the super imposed time code overlay
*/
LONG __stdcall vvwSetSuperImpose(LONG lChannel, LONG lSetting);
/**
Get the ms time the last error was added to the error log
```

```
*/
LONG __stdcall vvwGetErrorLogMs();
/**
Set the error log pointer to the message you want
*/
LONG __stdcall vvwSetErrorLog(LONG lSetting);
/**
Get the number of current errors
*/
LONG __stdcall vvwGetNumberOfErrors(LONG * plErrors);
/**
Get the length of the current error string
*/
LONG __stdcall vvwGetErrorLength(LONG * plLastError, LONG * plErrorLength);
/**
Get the current error.  Sets pointer to the next one automatically
*/
LONG __stdcall vvwGetError(LONG * plLastError, LONG * plSeverity, char * szError,
SHORT * psYear, SHORT * psMonth, SHORT * psDay, SHORT * psHour, SHORT *
psMinute, SHORT * psSecond, SHORT * psMillisecond);

/**
Returns the total number of frames of storage available at current compression rate if
the storage space was empty.
*/
LONG __stdcall vvwGetTotalTime(LONG lChannel);
/**
Returns the remaining number of frames of storage available at current compression
rate.
*/
LONG __stdcall vvwGetFreeTime(LONG lChannel);
/**
Returns the total storage connected in megabytes.
*/
LONG __stdcall vvwGetTotalStorage(LONG lChannel);
/**
Returns the amount of available storage for recording in megabytes.
*/
LONG __stdcall vvwGetFreeStorage(LONG lChannel);

/**
Get the available commands for a channel.
*/
LONG __stdcall vvwGetChannelCapabilities(LONG lChannel);

/**
Returns the version string of the VVW subsystem.
*/
char * __stdcall vvwGetVVWVersion();
```

```
/**
Returns the version string of the MediaReactor subsystem.
*/
char * __stdcall vvwGetMRVersion();

/**
Returns the type string of the VVW channel.
*/
char * __stdcall vvwGetVVWType(LONG lChannel);

/**
Get the name of a   file from the media file's name
*/
LONG __stdcall vvwGetPiconName(char * szFileName);

/**
Start playback at a specified MS
*/
LONG __stdcall vvwPlayAtMs(LONG lChannel, LONG lMs);

/**
Start Recording at a specified MS
*/
LONG __stdcall vvwRecordAtMs(LONG lChannel, LONG lMs, LONG lStart, LONG lEnd);

/***
Export to XML functions
*/
LONG __stdcall vvwXMLProjectSave(char * lpszProjectName);
/* Check for file conflicts*/
LONG __stdcall vvwXMLProjectCheckOpen(char * lpszProjectName);
/* Open the project*/
LONG __stdcall vvwXMLProjectOpen(char * lpszProjectName, int bDeleteConflictingFiles);
/*** Get number of backups on disk
*/
LONG __stdcall vvwGetNumberOfBackUps(LONG lChannel, LONG * pdwBackUps, LONG
dwClipMode);
/***
Revert to specified backup
*/
LONG __stdcall vvwSetBackUpNumber(LONG lChannel, LONG dwBackUp, LONG
dwClipMode);
/***
Check if the current channel settings match the specified file
return values:  -1 = problem finding values for source or system
                                    0 = file params and system settings do not match, if
bInstantMatch == true then return is from vvwChannel2Video
                                    1 = settings are the same or have been matched
```

```
*/
LONG __stdcall vvwChannel2File(LONG lChannel, char * lpszFileName, LONG *
plFileType, LONG * plCompression, LONG * plBitCount,
                                                    LONG * plWidth, LONG *
plHeight,
                                                    LONG * plRate, LONG *
plScale,
                                                    BOOL bInstantMatch);
/***
Set the channel settings to match VVWVIDEO of a desired file
return values:  -1 = problem with values for source or system
                                    0 = file params and system settings already match
                                    1 = success
*/
LONG __stdcall vvwChannel2Video(LONG lChannel, LONG * plFileType, LONG *
plCompression, LONG * plBitCount,
                                                          LONG * plWidth, LONG *
plHeight,
                                                    LONG * plRate, LONG *
plScale);

//
// Utility
//
/**
Free a string value returned by the channel.
*/
void __stdcall vvwFreeString(char * szString);

//
// MetaData DataBase Access
//
/**
*/

/**
// Get database name
*/
LONG __stdcall vvwMetaBaseGetName(char * szDataBase);

/**
// Open a database file
*/
LONG __stdcall vvwMetaBaseOpen(char * szDataBase);

/**
// Create a new database file
*/
LONG __stdcall vvwMetaBaseCreate(char * szDataBase);
```

```
/**
// Close previously opened database
*/
LONG __stdcall vvwMetaBaseClose();

/**
// Get number of metadata file items in database (table count)
//  return -1 if not open or table doesn't exist
*/
LONG __stdcall vvwMetaBaseFileCount(char * szFileName, char * szClipName, char *
szUniqueID, char * szMetaFilter, bool bUseAnd, bool bCountDBs);

/**
// Get each file item in database
//  return -1 if not open or table doesn't exist
*/
LONG __stdcall vvwMetaBaseFileName(LONG nIndex, char * szFileName, char *
szClipName, char * szUniqueID, char * szMetaFilter, bool bUseAnd, bool bCountDBs);

/**
// Remove a file item
//  return -1 if failed
*/
LONG __stdcall vvwMetaBaseFileRemove(char * szFileName, char * szClipName, char *
szUnique);

/**
// Change file item name
//  return -1 if failed
*/
LONG __stdcall vvwMetaBaseFileRename(char * szUniqueID, char * szNewFileName);

/**
// Add new file item
//  return -1 if already exist
*/
LONG __stdcall vvwMetaBaseFileAdd(char * szFileName, char * szClipName, LONG
lAddDefaults, LONG lIgnoreGUID);

/**
* Get the index of given tag name, or -1 if !exist
*/
LONG __stdcall vvwMetaBaseTagIndex(char * szFileName, char * szMetaDataTag);

/**
*
*/
LONG __stdcall vvwOutputReport(char * szFileName, long nExportType);
```

```
/**
*
*/
LONG __stdcall vvwMetaBaseGetAllTags(char * szFileName, char * szMetaDataValue);
/**
// Get metadata item and it's value for specified file item
//  return eof to determine if we reached the end
*/
LONG __stdcall vvwMetaBaseGetTag(char * szFileName, LONG nIndex, char *
szMetaDataTag, char * szMetaDataValue, LONG * plMetaDataValue, LONG *
plMetaDataType);

/**
// Set metadata item and it's value for specified file item
*/
LONG __stdcall vvwMetaBaseSetTag(char * szFileName, char * szMetaDataTag, char *
szMetaDataValue, LONG lMetaDataValue, LONG lMetaDataType);

LONG __stdcall vvwMetaBaseSavePicon(char * szDataBase, char * szPlay, char *
szPicon);

/**
// Set metadata item and it's value for specified file item
*/
LONG __stdcall vvwMetaBaseDefaultTags(char * szTable, char * szFileName);
LONG __stdcall vvwMetaBaseXOSTags(char * szTable);
LONG __stdcall vvwMetaBaseSMPTETags(char * szTable, char * szFileName);

/**
// Handle metabase commands
*/
LONG __stdcall vvwMetaBaseCmd(PMEDIACMD pCmd);
LONG __stdcall vvwMetaBaseDeletePicons();
LONG __stdcall vvwMetaBaseSetReplayMark(char * szFullTable, char * szMetaTag, char
* szMetaVal, LONG lPosition);
LONG __stdcall vvwMetaBaseGetReplayMark(char * szFullTable, char * szMetaTag, char
* szMetaVal, LONG * lPosition);

//! MetaData

// DSync GPI Control for OnTrack
DWORD __stdcall dtGpiSetMode(VVWHANDLE hDSync, DWORD dwInOut, DWORD
dwCmd, DWORD dwSetOn, BOOL bSet);
// debug tracking
DWORD __stdcall dtGpiSetModeD(VVWHANDLE hDSync, DWORD dwInOut, DWORD
dwCmd, DWORD dwSetOn, BOOL bSet, BOOL bTrace);

#ifdef __cplusplus
```

```
}
#endif

#if !defined(_VVW_TYPES_HAVE_ALREADY_BEEN_INCLUDED)

// Various speed limits and definitions
//! Forward play speed (normal) in VVW (65520) see MEDIACMD::lSpeed
#define SPD_FWD_PLAY                    65520L
//! Pause speed (0%) in VVW (0) see MEDIACMD::lSpeed
#define SPD_PAUSE                       0L
//! Reverse play speed (-100%) in VVW (-65520) see MEDIACMD::lSpeed
#define SPD_REV_PLAY                    (-SPD_FWD_PLAY)
//! Maximum possible play speed in VVW see MEDIACMD::lSpeed
#define SPD_FWD_MAX                         5896800
//! Minimum possible play speed in VVW see MEDIACMD::lSpeed
#define SPD_REV_MAX                     (-SPD_FWD_MAX)
//! Illegal speed, set MEDIACMD::lSpeed to this value if not used
#define SPD_ILLEGAL                     2147483647L

#endif
```

Appendix III – VvwTypes.h

```
/
*****************************************************************************
 * $Id$:
 *
 * $HeadURL$:
 * $Author$:
 * $Revision$:
 * $Date$:
 *
 * Copyright © 1998-2007 Drastic Technologies Ltd.  All Rights Reserved.
 * 523 The Queensway, Suite 102 Toronto ON M8V 1Y7
 * 416 255 5636  fax 255 8780
 * engineering@drastictech.com      http://www.drastictech.com
 *****************************************************************************/

#if !defined(_VVW_TYPES_HAVE_ALREADY_BEEN_INCLUDED)
#define _VVW_TYPES_HAVE_ALREADY_BEEN_INCLUDED

#ifdef _WIN32
// Windows x64
#ifndef _QTCREATOR
#pragma warning(disable: 4996)         // deprecated security functions
#endif

#ifndef _CRT_SECURE_NO_WARNINGS
        #define _CRT_SECURE_NO_WARNINGS
#endif
        #include <stdlib.h>
#endif

#include "mediacmd.h"

/**
* \page vvwtypesfastinfo VVWTypes Fast Info Page
*
* \section vvwtypeintro Introduction
*
* VVWTypes.h are internal types used to pass information between
VVW/QuickClip/MediaReactor
* modules when a #MEDIACMD is not possible or appropriate.  These are provided as a
part
* of the mediacmd sdk to fill in any holes or references not directly available in
* mediacmd.h.
```

```
*
*/

//
//! String return sizes (mediabase/mediafile plugin)
//@{
//! Maximum length for the supported file extensions string (fmt: *.xxx;*.yyy;*.ttt)
#define MB_FILE_EXT_LEN                 200
//! Maximum length for the base or short description (used in drop down)
#define MB_FILE_SHORT_LEN  64
//! Maximum length for the long description
#define MB_FILE_DESC_LEN    255
//! Maximum length for the short codec name
#define MB_CODEC_SHORT_LEN          64
//! Maximum length for the full (long) codec description
#define MB_CODEC_DESC_LEN 255
//@}

//
/**
* The universal preview flag - Top bit of a flag DWORD.  This bit
* should not be used for other purposes as the whole system should
* know if we are in preview.  Main places it is used:
*         intPreviewOpen(dwFlags)
*         avOpen(dwFlags)
*         mfOpen(dwOpenFlags)
*         dtcodecOpen(fccFlags)  - I think
*/
#define DTVVW_PREVIEW       0x80000000

//
// Some useful defines
//
//! No frame exists - is the same as TC_ILLEGAL - careful, only valid when tc cannot go
negative, else it means -1
#define VVW_INVALID_FRAME 0x80000001      // 2147483648 dec // old value
0xFFFFFFFF

//! Absolute maximum timecode in a time code space (99:59:59:29 NDF) - Actual max is
usually 23:59:59:x9 depending on type (see tc2Maximum())
#define VVW_ABS_MAX_FRAME 10692000                  // A325A0 hex

//! The size of a clip name string (8 char plus NULL term) (1 for safety) - This is for
odetics/louth compatibility
#define _MAX_CLIP_NAME_SIZE         9

//@{
/**
* Channel definition for VVW:
```

```
*/
/**
* Special sync channel
*/
#define _VVW_DSYNC_CHANNEL                          65536
/**
* First ltc channel
*/
#define _VVW_DSYNC_LTC_CHANNEL            _VVW_DSYNC_CHANNEL + 1
/**
* First real channel
*/
#define _VVW_CONTROL_START_LOCAL 0
/**
* This is the maximum number of channels that may exist on
* a machine locally.  They are numbered from 0..63 inclusive
* to the controlling software
*/
#define _VVW_CONTROL_MAX_LOCAL 64U
/**
* This is the maximum number of local serial control channels,
* LTC, VITC or any other info/control channels
* that may exist on a single machine.  The are numbered 64..127
*/
#define _VVW_CONTROL_START_EXTERNAL _VVW_CONTROL_MAX_LOCAL
//! Maximum number of external channels (63..127)
#define _VVW_CONTROL_MAX_EXTERNAL 64U
/**
* This is the maximum number of of network control channels.
* Each of the other channels exports a network server and may
* be connected to any of these channels as the user wishes
* They are numbered 128..255
*/
#define _VVW_CONTROL_START_NETWORK (_VVW_CONTROL_MAX_EXTERNAL +
_VVW_CONTROL_MAX_EXTERNAL)
//! Maximum number of network channels (128..255)
#define _VVW_CONTROL_MAX_NETWORK 128U
//! This is the maximum number of control inputs available
#define _VVW_CONTROL_MAX_CONTROL _VVW_CONTROL_MAX_LOCAL
//! Abs Max Channels on any system
#define _VVW_ABS_MAX_CHANNELS    256
//@}


// To match windows structures, we need to pack these things
#ifndef RC_INVOKED
#pragma pack(1)
#endif
```

```
/**
* Those stupid idiots at Microsoft forgot to change this structure
* when making the VfW headers for Win32.  The result is that the
* AVIStreamHeader is correct in 16-bit Windows, but the 32-bit
* headers define RECT as longs instead of shorts.  This creates
* invalid AVI files!!!!
*/
typedef struct {
        //! left side of rect - see microsoft RECT struct
        short    left;
        //! top of rect - see microsoft RECT struct
        short    top;
        //! right side of rect - see microsoft RECT struct
        short    right;
        //! bottom of rect - see microsoft RECT struct
        short    bottom;
} RECT16;

#ifndef RC_INVOKED
#pragma pack()
#endif

#ifndef RC_INVOKED
#pragma pack(4)
#endif

////////////////////////////////////////////////////////////////////
/**
* A Frame Info Packet - Part of the DFrame structure maintained
* with each memory frame passed through the MR or VVW system
* DFrame maintains the memory and internal timing info for the
* system, whereas frame_info maintains the external meta data
* from the original source.
*/
typedef struct /*tagFRAME_INFO*/ {
        //! Current time code, Control (CTL), our internal time code
        DWORD           dwFrame;
        //! Current LTC time code from original tape or internal generation
        DWORD           dwLtcFrame;
        //! Current LTC user Bits from original tape or internal generation
        DWORD           dwLtcUb;
        //! Current VITC time code from original tape or internal generation
        DWORD           dwVitcFrame;
        //! Current VITC user bits from original tape or internal generation
        DWORD           dwVitcUb;
        //! Version 4:  Key code value (all used)
        BYTE    arbKeyCode[8];
        //! Version 4:  Ink code value (6-8 = count, upper 1-5 = code)
        BYTE    arbInkCode[8];
```

```
        //! Data is EIA-608B SD closed caption data
#define FRAMEINFO_DATA_EIA608               1
        //! Data is EIA-708 HD closed caption data
#define FRAMEINFO_DATA_EIA708               2
        //! Data is RP-215 KLV data
#define FRAMEINFO_DATA_RP215_KLV            4
        /** Version 4:  Flags of what is in the data space
        */
        DWORD           dwDataFlags;
        /** Version 4:  Size used in the data space
        */
        DWORD           dwDataSize;
        //! Size of the arbData area
#define FRAMEINFO_MAX_DATA_SIZE  256
        /** Version 4:  extra space for EIA/CIE-708 captioning.  The old SD
        * 608 could only manage 4 characters per frame (2 per field).  The HD
        * 708 supports up to 40 characters per second.
        */
        BYTE    arbData[FRAMEINFO_MAX_DATA_SIZE];
        /** Pre V4!: Current VITC aux for extra vitc time code (3 line) or other info,
        * or if DFRAME_TYPE_FI_PTR_DATA is dwType of PDFRAME is set, then this
        * is a pointer to the extended data area.  This are should exist at
        * the end of the DFRAME structure.
        * NOTE:  When the DFRAME_TYPE_FI_PTR_DATA is up in the DFRAME::dwType
member,
        * then this is the size of the data area.  It is set initially to
        * (DFRAME_MAX_EXTRA_DATA_SIZE+1) which is illegal.  If you have dwVitcAux!
=0
        * and dwCCData != 0 && dwCCData <= DFRAME_MAX_EXTRA_DATA_SIZE then
you
        * probably have a valid data area.  If the DFRAME_TYPE_FI_PTR_DATA in
        * DFRAME::dwType is available and up and the above conditions are met,
        * then you are definately looking at valid data.
        * ================================
        * Expanded to 64 bits to allow for pointers
        */
        size_t    dwVitcAux;
        //! Alternate name for dwVITCAux
#define pFI_DataAreaPtr                 dwVITCAux
        /** PRE V4!: Captured close caption data from line 21 or if
DFRAME_TYPE_FI_PTR_DATA
        * then this contains the size of the data area pointed to by dwVitcAux.
        * NOTE:  When the DFRAME_TYPE_FI_PTR_DATA is up in the DFRAME::dwType
member,
        * then this is the size of the data area.  It is set initially to
        * (DFRAME_MAX_EXTRA_DATA_SIZE+1) which is illegal.  If you have dwVitcAux!
=0
        * and dwCCData != 0 && dwCCData <= DFRAME_MAX_EXTRA_DATA_SIZE then
you
```

```
        * probably have a valid data area.  If the DFRAME_TYPE_FI_PTR_DATA in
        * DFRAME::dwType is available and up and the above conditions are met,
        * then you are definately looking at valid data.
        */
        DWORD           dwCCData;
        //! Alternate name for dwCCData
#define dwFI_DataAreaSize      dwCCData
} FRAME_INFO, * pFRAME_INFO;

/** Normal size of the pFI_DataAreaPtr
*/
#define DFRAME_MAX_EXTRA_DATA_SIZE       1024
//! This indicates that fi.dwVITCAux points to a data area and dwCCData contains its size
//// See below #define DFRAME_TYPE_FI_PTR_DATA    0x00008000

// Sends RP-215 data
#define DFRAME_DATAAREA_RPDPXHEADER    "DraStiCteCh-215"        //16th pos is 0
null term
#define DFRAME_DATAAREA_RPDPXHEADER_SIZE       16

//! Copy the frame info area, remove pointers
#define COPYFRAMEINFO(_ptrFiDst, _ptrFiSrc) \
        CopyMemoryFast(_ptrFiDst,_ptrFiSrc,sizeof(FRAME_INFO));      \
        (_ptrFiDst)->dwVitcAux = 0;     \
        (_ptrFiDst)->dwCCData = 0; \

////////////////////////////////////////////////////////////////
/**
* A channel memory area - This is the basis of the DSync system
* It is used to provide inter and intra channel information on
* current state in a timely fashion.  Most importantly, it is
* used for the ctTransfer command to move media to and from
* external vtrs.  This allows the transfering channel to
* get information on the other channel as quickly as possible.
* It was designed to operate effeciently when both channels
* are running on the same machine, but also to provide a single
* interface when a network or other transport is between
* the two channels.
*/
typedef struct /*tagDRASTIC_CHANNEL*/
{
        //! Last update time in ms per the local performance close (vsyncGetCurMs())
        DWORD           dwLastUpdate;
        //! The current ctl/ltc/vitc frame/userbits and other per frame data
        FRAME_INFO fi;
        //! The current state of the channel as a MEDIACMD structure
        MEDIACMD   mCmd;
        //! This will always point at one of the frame counts in the frame_info structure
above
```

```
        DWORD*          pdwFrame;
        //! A handle to used by the owner of this struct (usually its base class address)
        void*    pOwnerHandle;
        //! The channel ID (numeric 0-255) of this structure
        DWORD           dwChannelID;            // Channel identifier
} DRASTIC_CHANNEL, * pDRASTIC_CHANNEL;

//! Declare and initialize a DRASTIC_CHANNEL
#define DECLARE_DRASTIC_CHANNEL(__x_)   \
        { 0xFFFFFFFF, 0, 0, 0, 0, 0, 0, 0,                          \
        &__x_.dwFrame, 1, 3, ctStop, 0,                 \
        0, 0xFFFFFFFF, &__x_, 0};
//! Initialize a DRASTIC_CHANNEL
#define INIT_DRASTIC_CHANNEL(__x_)                                          \
        { ZeroMemory(&__x_, sizeof(DRASTIC_CHANNEL));     \
        __x_.pdwFrame = &__x_.dwFrame;                                      \
        __x_.ctCmd = ctStop;                                                \
        __x_.pOwnerHandle = &__x_; };
//! Initialize a memory area allocated as a DRASTIC_CHANNEL
#define INIT_PDRASTIC_CHANNEL(__x_)                                         \
        { ZeroMemory(__x_, sizeof(DRASTIC_CHANNEL));      \
        __x_->pdwFrame = &__x_->dwFrame;                            \
        __x_->ctCmd = ctStop;                                               \
        __x_->pOwnerHandle = &__x_; };

////////////////////////////////////////////////////////////////
/**
* This is the basis of all our clip handling.  If is used in
* tcspace, clipspace, clipctrl, edlxlat and most other clip
* handling areas.  It includes info on the clip, trim points,
* its position within a container, channels available,
* name, comment and reel.
*/
typedef struct /*tagDCLIP*/ {
        //! Clip In - user defined start of clip, trimmed from dwClipStart
        DWORD dwClipIn;
        //! Clip In - user defined end of clip, trimmed from dwClipStart, max == Clip End
(the outpoint is never included)
        DWORD dwClipOut;
        //! Clip Start - the actual physical start of a clip (normally 0)
        DWORD dwClipStart;
        //! Clip End - the actual physical end of a clip + 1 (the outpoint is never included)
        DWORD dwClipEnd;
        //! Clip Auxillary - used to denote picon frame, internal key frame, or other
dependant on clip type
        DWORD dwClipAux;
        //! A Numeric Clip or Reel ID - Normally a reel id, but could also be a take or
other info
        DWORD dwClipID;
```

```
        //! Available video channels bit array
        DWORD dwVidChan;
        //! Available audio channels bit array
        DWORD dwAudChan;
        //! Available information channels bit array
        DWORD dwInfChan;
        //! The Clip Name - For louth/odetics compatibility, this should be 8 characters +
0 terminator long
        char * szClipName;
        //! The full name - Usually the file name of the media on disk
        char * szName;
        //! User comment connected to this file.  Free form, may contain other key
information
        char * szComment;
        //! Reel ID or string denoting source tape or media
        char * szReel;
} DCLIP, * PDCLIP;

////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
#if 0 // commented out for docs !defined(__ICL) //|| defined(_NTDRIVER_)      // Intel
compiler does not support base structures
//: D_LNODE
#endif
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
/**
* The DFRAME - A container/marker for a frame of information in memory
*
*       This type holds the internal timing, memory allocation, frame flags
* and dlist elements for one frame of video, audio or information being
* processed by VVW or MEDIAREACTOR.  It should always be allocated
* by the PhysMem.DLL for maximum speed of memory manipulation.  It is
* used just about everywhere and should not be changed without extremely
* careful consideration.
*/
typedef struct /*tagDFRAME */
{
        //! Copy From D_LNODE - DO NOT MODIFY
        void* pPrev;    // Next lnode
        //! Copy From D_LNODE - DO NOT MODIFY
        void* pNext;    // Prev lnode
        //! Copy From D_LNODE - DO NOT MODIFY
        void* pList;    // Parent or List owner
        //! size of this structure (+ extra at end if required)
        DWORD dwSize;
        /**
        * Any combination of the DFRAME_ flags including
```

```
        * #DFRAME_TYPE_RECORD, #DFRAME_TYPE_PLAY, #DFRAME_TYPE_AUDIO,
#DFRAME_TYPE_VIDEO,
        * #DFRAME_PROGRESSIVE, #DFRAME_FIELD_INVERT,
#DFRAME_TIME_INVERT, #DFRAME_LARGE_AUDIO,
        * #DFRAME_NEW_FORMAT, #DFRAME_TYPE_KEYFRAME,
#DFRAME_TYPE_KEYFRAME_I, #DFRAME_TYPE_KEYFRAME_P,
        * #DFRAME_TYPE_KEYFRAME_B, #DFRAME_SKIP_FRAME
        */
        DWORD dwType;
//! This frame was recorded into (normally by AvHAL) see DFRAME::dwType
#define DFRAME_TYPE_RECORD                  0x00000001
//! This frame should be played out (normally by AvHAL) see DFRAME::dwType
#define DFRAME_TYPE_PLAY            0x00000002
//! This frame was aquired while in pause and is most likely a seek (normally by AvHAL)
see DFRAME::dwType
#define DFRAME_TYPE_PAUSE           (0x00000004 | DFRAME_TYPE_PLAY)    // Play +
Pause
//! This indicates PhysHeap did NOT allocate the dframe (not pheap or local heap)
#define DFRAME_TYPE_NOTPHYSHEAP0x00001000
//! This indicates that FRAME_INFO::dwVITCAux points to a data area and
FRAME_INFO::dwCCData contains its size see them for more info
#define DFRAME_TYPE_FI_PTR_DATA  0x00008000
//! This frame contains audio data see DFRAME::dwType
#define DFRAME_TYPE_AUDIO           0x00010000
//! This frame contains video data see DFRAME::dwType
#define DFRAME_TYPE_VIDEO           0x00020000
//! The contents of the frame are progressive (as opposed to interlaced) see
DFRAME::dwType
#define DFRAME_PROGRESSIVE                  0x00100000
//! The fields in the frame are inverted (jaggies) see DFRAME::dwType
#define DFRAME_FIELD_INVERT                 0x00200000
//! The fields in the frame are temporally inverts (jumps back and forth) see
DFRAME::dwType
#define DFRAME_TIME_INVERT                  0x00400000
//! The contents of the frame are inverted
#define DFRAME_ORIENTATION_INVERT                   0x00800000
//! The frame contains a large chunk of audio (allows for optimization in AvHAL) see
DFRAME::dwType
#define DFRAME_LARGE_AUDIO      0x01000000
//! This frame starts a new format that is different from the ones previous. Please get the
new format and adjust before displaying see DFRAME::dwType
#define DFRAME_NEW_FORMAT                   0x08000000
//! This frame is independant of other frames for decode see DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME        0x10000000
//! This frame is independant of other frames for decode (an MPEG I Frame) see
DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_I    0x10000000
//! This frame requires previous keyframe(s) (for MPEG a P Frame) see
DFRAME::dwType
```

```
#define DFRAME_TYPE_KEYFRAME_P   0x80000000
//! This frame requires more than one frame to decode (for MPEG a B Frame) see DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_B   0x20000000
//! This frame should be skipped (decoded, but not displayed) - Used to reach seek frame on a non key frame from key frame see DFRAME::dwType
#define DFRAME_SKIP_FRAME        0x40000000
        //! Number of repeats of this frame.  Uses to create slow motion effects, or save memory on still images
        DWORD dwReps;
        //! The external timing info for this frame (LTC/VITC/CTL timecode/userbits - See FRAME_INFO)
        FRAME_INFO fi;
        //! Internal - Count down for rep usage in slow motion (AvHAL exclusive)
        DWORD dwRepIndex;
        //! The VVW Speed (65520 based) at which this frame is supposed to play. Always forward?? Should be long probably.
        DWORD dwSpeed;
        //! Internal - The expected time that this frame may be safely released and deallocated.
        DWORD dwExpireMS;
        //! Used to determine the time to display the frame to the viewer, also dwTimeCaptured in AvHal::VfW
        DWORD dwPresentationMS;

        /**
        * Start - taken from VIDEOHDR, WAVEHEADER
        * This is actually a WAVEHDR - VideoHdr includes a dwTimeCaptured atfter
        *                                                    the dwunsigned chars used member!  Leave it alone
        *                                                    as we use the WAVEHDR directly with win32 wave...
        *                                                    functions
        */
        unsigned char * lpData;                 // Pointer to data buffer below
        /**
        * The maximum length of the buffer pointed to by DFRAME::lpData
        * This should set by PhysHeap.dll and left alone by the user.  If more
        * memory is required, allocate a new #DFRAME and copy the current
        * data into it.  Should always be aligned to disk sector size or
        * 4096 (whichever is greater).
        */
        DWORD dwBufferLength;       // MUST BE DWORD FOR WINDOWS Total length of buffer below
        /**
        * The current number of valid unsigned chars pointed to by DFRAME::lpData. Must be less
        * than DFRAME::dwBufferLength.  User adjustable as nec.
        */
```

```
        DWORD dwBytesUsed;          // MUST BE DWORD FOR WINDOWS unsigned
chars frame takes up in buffer below
                    // above - dwunsigned charsRecorded - in WAVEHEADER
        // insert - dwTimeCaptured - for VIDEOHDR only
        //! User var for VfW/Wave driver - internal - do not use
        DWORD_PTR dwUser;
        //! internal to AvHAL Set by the device driver to indicate it is finished with the
data buffer and is returning the buffer to the client. see DFRAME::dwFlags
#define DTVHDR_DONE                 0x00000001
        //! internal to AvHAL Indicates whether or not the buffer has been prepared for
use. See DVM_STREAM_PREPAREHEADER. see DFRAME::dwFlags
#define DTVHDR_PREPARED 0x00000002
        //! internal to AvHAL Set by the driver to indicate the buffer is in the driverís
buffer queue. see DFRAME::dwFlags
#define DTVHDR_INQUEUE      0x00000004
        //! internal to AvHAL Set by the device driver to indicate a key frame. see
DFRAME::dwFlags
#define DTVHDR_KEYFRAME     0x00000008
        // Audio Flags
        //! Set by the device driver to indicate that it is finished with the buffer and is
returning it to the application. see DFRAME::dwFlags
#define DTWHDR_DONE                 0x00000001
        //! internal to AvHAL Set by Windows to indicate that the buffer has been
prepared with the waveInPrepareHeader or waveOutPrepareHeader function. see
DFRAME::dwFlags
#define DTWHDR_PREPARED   0x00000002
        //! internal to AvHAL This buffer is the first buffer in a loop. This flag is used only
with output buffers. see DFRAME::dwFlags
#define DTWHDR_BEGINLOOP 0x00000004
        //! internal to AvHAL This buffer is the last buffer in a loop. This flag is used only
with output buffers. see DFRAME::dwFlags
#define DTWHDR_ENDLOOP    0x00000008
        //! internal to AvHAL Set by Windows to indicate that the buffer is queued for
playback. see DFRAME::dwFlags
#define DTWHDR_INQUEUE    0x00000010
        //! This is a single frame clip (implies progressive) that needs to be play for the
duration of the clip see DFRAME::dwFlags
#define _PDFRAMEFLAGS_CLIPSTILL             0x00000001     // Only one buffer in clip
        //! This is the first frame of a new clip see DFRAME::dwFlags
#define _PDFRAMEFLAGS_CLIPSTART             0x00000002     // Start of clip
        //! This is the last frame of the current clip see DFRAME::dwFlags
#define _PDFRAMEFLAGS_CLIPEND               0x00000004     // End of clip
        //! Alias for #_PDFRAMEFLAGS_CLIPSTART see DFRAME::dwFlags
#define _PDFRAMEFLAGS_FIRSTFRAME            0x00000002     // Start of clip
        //! Alias for #_PDFRAMEFLAGS_CLIPEND see DFRAME::dwFlags
#define _PDFRAMEFLAGS_LASTFRAME             0x00000004     // End of clip
        //! This Frame has been imposed if need be and watreramrked.
#define _PDFRAMEFLAGS_IMPOSED_MARKED            0x00000008     // End of clip
        //! Interleaved frame had already been split and it the first field part mask
```

```
#define _PDFRAMEFLAGS_FIELD_MARK_MASK 0x00030000
        //! If set, the first second bit is correct, if not then ignore even off bit
#define _PDFRAMEFLAGS_FIELD_HASMARK             0x00020000
        //! This is the first field, if the HASMARK is set
#define _PDFRAMEFLAGS_FIELD_FIRST               0x00000000
        //! This is the second field, if the HASMARK is set
#define _PDFRAMEFLAGS_FIELD_SECOND              0x00010000
        //! This is the second field, if the HASMARK is set
#define _PDFRAMEFLAGS_FIELD_INLINE              0x00040000
        //! Set as first field
#define _PDFRAMEFLAGS_FIELD_MARKFIRST           (0x00000000 |
_PDFRAMEFLAGS_FIELD_HASMARK)
        //! Set as second field
#define _PDFRAMEFLAGS_FIELD_MARKSECOND                  (0x00010000 |
_PDFRAMEFLAGS_FIELD_HASMARK)
        //! Set as two fields, one full field followed by the other ie not interlaced.
#define _PDFRAMEFLAGS_FIELD_MARK_TWO_FIELDS
        (_PDFRAMEFLAGS_FIELD_INLINE | _PDFRAMEFLAGS_FIELD_HASMARK)
        /**
        * Flags including external flags
        * #_PDFRAMEFLAGS_CLIPSTILL, #_PDFRAMEFLAGS_CLIPSTART,
#_PDFRAMEFLAGS_CLIPEND,
        * #_PDFRAMEFLAGS_FIRSTFRAME, #_PDFRAMEFLAGS_LASTFRAME
        * and AvHAL internal flags (stripped in AvHAL)
        * #DTWHDR_DONE, #DTWHDR_PREPARED, #DTWHDR_ENDLOOP,
        * #DTWHDR_INQUEUE and #DTVHDR_DONE, #DTVHDR_PREPARED,
        * #DTVHDR_INQUEUE, #DTVHDR_KEYFRAME
        */
        DWORD dwFlags;                          // Internal
        /** Loop counter for WAVEHDR
        */
        DWORD dwLoops;
        //! Internal - Do not use, hardware specific
        DWORD dwReserved[4];        // Hardware specfic
                // struct wavehdr_tag * lpNext - in WAVEHEADER
                // DWORD_PTR   reserved - in WAVEHEADER
        // End - taken from VIDEOHDR, WAVEHEADER
        //! Filler for alignment or extended user data
        WORD resdata;                   // Filler to 100 unsigned chars

} DFRAME, * PDFRAME;

//! The DWORD aligned size of a #DFRAME, used for more effecient memory allocations
#define SIZEOFDFRAME        (((sizeof(DFRAME) >> 2) + 1) << 2)

////////////////////////////////////////////////////////////////
// Channel position, size, name return
// This should be past in in place of PDFRAME to mfRead
// when requesting MF_CHAN_POSITION_SIZE
```

```
//
typedef struct /*tagDFRAME */
{
        //! Copy From D_LNODE - DO NOT MODIFY
        void* pPrev;    // Next lnode
        //! Copy From D_LNODE - DO NOT MODIFY
        void* pNext;    // Prev lnode
        //! Copy From D_LNODE - DO NOT MODIFY
        void* pList;    // Parent or List owner
        //! Size of this structure, must be set or request will be rejected
        DWORD dwSize;          // MUST be sizeof(DFRAME)
        //! Data type
        DWORD dwType;
        //! This is a video frame
#define DPOSSIZENAME_VIDEO_FRAME                0x00000001
        //! This is a mono audio chunk
#define DPOSSIZENAME_MONO_AUDIO_FRAME           0x00000100
        //! This is a stereo audio chunk
#define DPOSSIZENAME_STEREO_AUDIO_FRAME         0x00000200
#define DPOSSIZENAME_QUAD_AUDIO_FRAME           0x00000400
#define DPOSSIZENAME_4_1_AUDIO_FRAME    0x00000800
#define DPOSSIZENAME_5_1_AUDIO_FRAME    0x00001000
#define DPOSSIZENAME_7_1_AUDIO_FRAME    0x00002000
#define DPOSSIZENAME_9_1_AUDIO_FRAME    0x00004000
#define DPOSSIZENAME_FRAME_MASK                 0x0000FFFF
        //! This frame contains audio data see DFRAME::dwType
// From #define DFRAME_TYPE_AUDIO            0x00010000
        //! 16 bit audio
#define DPOSSIZENAME_AUD_16_16_BIT              0x00100000
        //! 20 bit audio in 24
#define DPOSSIZENAME_AUD_20_24_BIT              0x00200000
        //! 24 bit audio in 24
#define DPOSSIZENAME_AUD_24_24_BIT              0x00400000
        //! 24/32 bit audio in 32
#define DPOSSIZENAME_AUD_24_32_BIT              0x00800000
        //! 32/32 bit audio in 32
#define DPOSSIZENAME_AUD_32_32_BIT              0x01000000
        //! Audio is big endian, else little endian
#define DPOSSIZENAME_AUD_BIGENDIAN_BIT0x00080000
        //! Just for completeness
//#define DPOSSIZENAME_AUD_LITTLEENDIAN_BIT     0x00000000
        //! This frame is independant of other frames for decode see DFRAME::dwType
// also used #define DFRAME_TYPE_KEYFRAME0x10000000
        //! This frame is independant of other frames for decode (an MPEG I Frame) see
DFRAME::dwType
// also used #define DFRAME_TYPE_KEYFRAME_I        0x10000000
        //! This frame requires previous keyframe(s) (for MPEG a P Frame) see
DFRAME::dwType
// also used #define DFRAME_TYPE_KEYFRAME_P        0x80000000
```

```
        //! This frame requires more than one frame to decode (for MPEG a B Frame)
see DFRAME::dwType
// also used #define DFRAME_TYPE_KEYFRAME_B        0x20000000
        //! This frame should be skipped (decoded, but not displayed) - Used to reach
seek frame on a non key frame from key frame see DFRAME::dwType
// also used #define DFRAME_SKIP_FRAME                0x40000000
        //! Number of repeats of this frame.  Uses to create slow motion effects, or save
memory on still images
        DWORD dwReps;
        //! The external timing info for this frame (LTC/VITC/CTL timecode/userbits - See
FRAME_INFO)
        FRAME_INFO fi;
        //
        // MUST MATCH PDFRAME UP TO THIS POINT TO ACCEPT FRAME INFO
        //
        //! Position
        __int64 u64Position;
        //! Size
        DWORD dwBytesUsed;
        //! Filename
        char szFileName[_MAX_PATH];
        //! End
} DPOSSIZENAME, * PDPOSSIZENAME;

//////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////

//! The size of the reserved area (in DWORDs) within #VVWSYSTEM, #VVWVIDEO,
#VVWAUDIO and #VVWINFO
#define _VVWXXX_RESERVED_SIZE     256
//! The size of the name area (in chars) within #VVWSYSTEM, #VVWVIDEO,
#VVWAUDIO and #VVWINFO
#define _VVWXXX_NAME_SIZE          256


#define dtstreamtypeVIDEO ( (DWORD)(unsigned char)('v') | ( (DWORD)(unsigned char)
('i') << 8 ) |     ( (DWORD)(unsigned char)('d') << 16 ) | ( (DWORD)(unsigned char)('s')
<< 24 ) )
#define dtstreamtypeAUDIO ( (DWORD)(unsigned char)('a') | ( (DWORD)(unsigned
char)('u') << 8 ) |        ( (DWORD)(unsigned char)('d') << 16 ) | ( (DWORD)(unsigned
char)('s') << 24 ) )

//////////////////////////////////////////////////////////////////
//
//        The system struct describes a file or harware system
//        as basically as possible.  For more in depth information
//        see the VVWAUDIO/VIDEO/INFO structs below.
```

```
//
// 03/13/98 Used By:
//                  Modules\AvHal
//                  Common\mfFile

//! The flag indicating the structure is a #VVWSYSTEM structure within a union
#define _VVW_IS_VVWSYSTEM           0x0000

/**
* The VVWSYSTEM structure holds the overview of a media stream including basic
* time (rate/scale), number of tracks, buffering info, msperframe, width/height
* of one or more video streams, maximum length in frames and (rate/scale).  It
* is binary compatible with the AVI sytem header, but translated as nec to OMF
* MOV or any other supported format.  Testing has not been thorough enough to
* entirely trust this structure, but it is a good place to start.
*/
typedef struct /*tagVVWSYSTEM*/ {
        // Stolen from MainAVIHeader
        /**
        * Number of microseconds for a frame duration.  Normally refers to video and
will
        * be set to values like 33367 (NTSC) or 40000 (PAL) or possibly 0 if not specified.
        * When the system structure is used in AvHAL or elsewhere internally, this value
        * must always be valid as some timing calculations are based on it.  As good
practice,
    * make sure it is correct for media files as well.
        * Related to VVWSYSTEM::dwRate and VVWSYSTEM::dwScale in that
        * == 1000000 / (VVWSYSTEM::dwRate / VVWSYSTEM::dwScale)
        */
        DWORD                   dwMicroSecPerFrame;   // frame display rate (or 0L)
        /**
        * The maximum unsigned chars per second this media will generate.  Should
assume all channels
        * active and in use.  Should be set by writer, but often isn't.  Not used internally,
        * so regard with some suspicion.
        */
    DWORD               dwMaxBytesPerSec;      // max. transfer rate
        /**
        * The size of pad used to write the file.  With bad writers, this is set to
0/512/1024/2048/4096
        * and then the file is written without any regard for its setting.  With decent
writers, it
        * is set to the disk sector size the writer is writing to and it is respected and
correct at
    * at least for video.  With our writers, it is set to 4096 (largest general multiple of
sector
        * size and most common on NTFS systems) and we write to 4096.  If the
granularity is bad,
```

```
        * we can usually still use sector aligned reads, by reading a little extra garbage
and ignoring
        * it, except for the last frame if we run out of file.
        */
        DWORD                   dwPaddingGranularity;  // pad to multiples of this size;
normally 2K.
        /**
        AVIFILEINFO_ flags such as
        \code
        AVIFILEINFO_HASINDEX                    0x00000010 - there is an index to the
frames included in the file
        AVIFILEINFO_MUSTUSEINDEX  0x00000020 - index is required, normally means
frames are out of order on disk
        AVIFILEINFO_ISINTERLEAVED  0x00000100 - includes multiple channels in same
file (usually audio and video)
        AVIFILEINFO_WASCAPTUREFILE          0x00010000 - from live source, may not
be optimally written in the heat of the moment (eg avicap)
        AVIFILEINFO_COPYRIGHTED             0x00020000 - early microsoft attempt at
screwing us.  not used to my knowledge
        \endcode
        */
    DWORD               dwFlags;                        // the ever-present flags
AVIFILEINFO_
        /**
        * This is the total number of actual video frames (counted as a frame count) of
the longest
        * video stream in the file.  If there is no video stream in the file, the audio will be
        * broken up into frames per (VVWSYSTEM::dwRate / VVWSYSTEM::dwScale)
and if they are set to
        * 0 then, we normally fill it with and NTSC value of frames by default.  This is the
place
        * we normally count on the be correct for conversion source length and clip
length in VVW.
        */
    DWORD               dwTotalFrames;          // # frames in file
        /**
        * The initial number of frames that should be loaded or, esp if using 'rec ' index
chunks, the
        * number of pre-buffer audio frames before video appears (usually 10~25).  We
usually ignore
        * this value as it will only help with extremely slow drives.  It doesn't even
appear to help
        * with 10bT network connections significantly.
        */
    DWORD               dwInitialFrames;
        /**
        * The total number of audio, video and information streams in the media file.
This does not
```

```
        * include channels within stereo audio streams which are considered to be one
channel.
        */
    DWORD               dwStreams;
        /**
        * Suggested buffer size is set by the writer and is supposed to be the size of the
largest
        * 'chunk' (audio or video) of data that will have to be read at once.  More
important in
        * systems where memory is at a premium.  With Win32 we only us this to
determine the correct
        * audio buffer size if it is greater than half a second of stereo audio at the audio
streams
        * sample rate and bit size.
        */
    DWORD               dwSuggestedBufferSize;
    /**
        * Basic or largest width of a video stream within this file.
        */
    DWORD               dwWidth;
    /**
        * Basic or largest height of a video stream within this file. Caution, for some
media
        * streams this value can be negative indicating the stream is vertically inverted.
If a
        * positive value is required, abs this before using.  In theory, this value should
        * always be positive, but it isn't.
        */
    DWORD               dwHeight;
    /**
        * As it says, reserved, do not touch.
        */
    DWORD               dwReserved[4];
        // End of MainAVIHeader

        // Extras for AVIFILEINFO
        /**
        * Capability flags from video for windows including:
        * \code
     AVIFILECAPS_CANREAD            0x00000001 - File may be read
     AVIFILECAPS_CANWRITE           0x00000002 - File may be written to
     AVIFILECAPS_ALLKEYFRAMES       0x00000010 - File contains frames that do not
require interframe data to decode (eg. JPEG)
     AVIFILECAPS_NOCOMPRESSION      0x00000020 - The frames in the file are
uncompressed
        * \endcode
        * Do not trust these flags.  Look at the #VVWVIDEO and #VVWAUDIO areas to
confirm types of streams.
        */
```

```
        DWORD                   dwCaps;                                 // AVIFILECAPS_
. . .
        /**
        * Scale - the frame divisor into VVWSYSTEM::dwRate to get the frame frame rate
        \code
  *   Scale   Rate    FrameRate   MsPerFrame
  *   1       60      60                  HD 60FPS (720p)
  *   1001    60000   59.94                HD 59.94FPS (720p)
  *   1       50      50                  HD 50FPS (720p)
  *   1       30      30                  HD 30FPS
  *   1       25      25          40       (PAL, 1080i)
  *   1001    30000   29.97       33.667    (NTSC, 1080i)
  *   1       24      24          41.667    (FILM, 1080, DCinema)
  *   1001    24000   23.98                23.98 psf/p
  *   66000   1000000 15.152      65.998    (NTSC->Multimedia)
  *   1       15      15          66.667    (MultiMedia)
  *   1       22050   22.050kHz   --        (Audio 22kHz 8 Bit Mono)
  *   4       176400  44.1kHz     --        (Audio 44.1kHz 16 Bit Stereo)
  *   2       48000   48kHz       --        (Audio 48kHz 16 Mono --OR-- 48kHz 8 Bit Stereo)
        \endcode
        */
        DWORD                   dwScale;                        // dwRate/dwScale =
samples per second
        /**
        * Rate - the frame second length divided by VVWSYSTEM::dwScale to get the
frame frame rate
        \code
  *   Scale   Rate    FrameRate   MsPerFrame
  *   1       60      60                  HD 60FPS (720p)
  *   1001    60000   59.94                HD 59.94FPS (720p)
  *   1       50      50                  HD 50FPS (720p)
  *   1       30      30                  HD 30FPS
  *   1       25      25          40       (PAL, 1080i)
  *   1001    30000   29.97       33.667    (NTSC, 1080i)
  *   1       24      24          41.667    (FILM, 1080, DCinema)
  *   1001    24000   23.98                23.98 psf/p
  *   66000   1000000 15.152      65.998    (NTSC->Multimedia)
  *   1       15      15          66.667    (MultiMedia)
  *   1       22050   22.050kHz   --        (Audio 22kHz 8 Bit Mono)
  *   4       176400  44.1kHz     --        (Audio 44.1kHz 16 Bit Stereo)
  *   2       48000   48kHz       --        (Audio 48kHz 16 Mono --OR-- 48kHz 8 Bit Stereo)
        \endcode
        */
        DWORD                   dwRate;                                 // as above
        /**
        * Length of the longest stream in samples. This could be audio or video samples,
but
        * will always relate to Rate/Scale above.  For streams with video, this is a frame
count
```

```
        * and should be the same as dwTotalFrames.
        */
        DWORD           dwLength;               // Lendth of file in
dwRate/dwScale units
        /**
        * The number of times this file has been edited.  Intended for tracing QuickTime
style
        * edits within an AVI file, but good editing capabilities where never added to
video
        * for windows, so this is always 0, 1 or a random number with little or no
meaning.
        */
        DWORD           dwEditCount;        // Number of streams added or
deleted
        /**
        * A marker for the time of media file or module that created this structure is
        * Free form, used to present to user when problems occur.
        */
        char            szFileType[_VVWXXX_NAME_SIZE];          // Descriptive
info
        /**
        * VVW and ME internal flags - Add Flags Here MF_TYPE_???, AVH_TYPE_???
        */
        DWORD       dwType;                                 // MF_TYPE_,
AVH_TYPE_
        /**
        * MediaFile Capability flags.  Not used currenly.
        */
        DWORD       dwMfCaps;                           // MF_CAP_, AVH_CAP_
        /**
        * Basic video standard.  Was PAL/NTSC, but should now us
#GS_SIGFORM_NTSC defines in mediacmd.h
        */
        DWORD       dwVidStandard;              // Best guess at PAL,NTSC,etc
        /**
        * Our internal flags including:
        * #DRFLAGS_ZERO_FIELD_DOMINANT, #DRFLAGS_FIRST_FIELD_DOMINANT,
#DRFLAGS_HAS_KEYFRAMES,
        * #DRFLAGS_FCC_MJPG_DIGISUITE, #DRFLAGS_FCC_MJPG_DCx0,
#DRFLAGS_FCC_MJPG_DSEDIT,
        * #DRFLAGS_FCC_MJPG_JPGDIB, #DRFLAGS_FCC_MJPG_JFIF,
        * #DRFLAGS_FCC_USE_INTERN, #DRFLAGS_FCC_USE_QT,
#DRFLAGS_FCC_USE_ICM,
        * #DRFLAGS_CODECPRIVATEDATA_AVI,
#DRFLAGS_CODECPRIVATEDATA_MOV, #DRFLAGS_CODECPRIVATEDATA_OMF,
        * #DTVVW_PREVIEW
        */
        DWORD dwDrFlags;
        //! Source File Type
```

```
        DWORD dwFileType;
        /**
        * Reserved.  Set to 0 on allocate and do not touch.
        */
        DWORD dwResDrastic;
} VVWSYSTEM, * pVVWSYSTEM;

// Macros       - See /Ass/Modules/MediaFile/Test Patterns/Test Patterns.cpp for
"Compile Test"
//! Set a #VVWSYSTEM structure pointer from #VVWVIDEO pointer, #VVWAUDIO
pointer, a granularity size and number of frames
#define VVWSYS_SET(__pvvwsys_, __pvvwvid_, __pvvwaud_, _granularity, _frames) { \
        (__pvvwsys_)->dwMicroSecPerFrame = (DWORD)(1000000.0 / ((double)
(__pvvwvid_)->dwRate / (double)(__pvvwvid_)->dwScale));    \
        (__pvvwsys_)->dwMaxBytesPerSec = ((__pvvwvid_)->biSizeImage *
(__pvvwvid_)->dwScale / (__pvvwvid_)->dwRate) + (__pvvwaud_)->nAvgBytesPerSec;
        \
        (__pvvwsys_)->dwPaddingGranularity = _granularity;   \
        (__pvvwsys_)->dwTotalFrames = _frames;      \
        (__pvvwsys_)->dwSuggestedBufferSize = (__pvvwvid_)-
>dwSuggestedBufferSize + (__pvvwaud_)->dwSuggestedBufferSize;    \
        (__pvvwsys_)->dwWidth = (__pvvwvid_)->biWidth;     \
        if((__pvvwvid_)->biHeight < 0) (__pvvwsys_)->dwHeight = -(LONG)
(__pvvwvid_)->biHeight; else (__pvvwsys_)->dwHeight = (__pvvwvid_)->biHeight;     \
        (__pvvwsys_)->dwScale = (__pvvwvid_)->dwScale;     \
        (__pvvwsys_)->dwRate = (__pvvwvid_)->dwRate;       \
        (__pvvwsys_)->dwLength= (__pvvwvid_)->dwLength;  \
        }
//! Set a #VVWSYSTEM structure pointer from #VVWVIDEO pointer, a granularity size
and number of frames
#define VVWSYS_SETVIDONLY(__pvvwsys_, __pvvwvid_, _granularity, _frames) {      \
        (__pvvwsys_)->dwMicroSecPerFrame = (DWORD)(1000000.0 / ((double)
(__pvvwvid_)->dwRate / (double)(__pvvwvid_)->dwScale));    \
        (__pvvwsys_)->dwMaxBytesPerSec = ((__pvvwvid_)->biSizeImage *
(__pvvwvid_)->dwScale / (__pvvwvid_)->dwRate);      \
        (__pvvwsys_)->dwPaddingGranularity = _granularity;   \
        (__pvvwsys_)->dwTotalFrames = _frames;      \
        (__pvvwsys_)->dwSuggestedBufferSize = (__pvvwvid_)-
>dwSuggestedBufferSize;        \
        (__pvvwsys_)->dwWidth = (__pvvwvid_)->biWidth;     \
        if((__pvvwvid_)->biHeight < 0) (__pvvwsys_)->dwHeight = -(LONG)
(__pvvwvid_)->biHeight; else (__pvvwsys_)->dwHeight = (__pvvwvid_)->biHeight;     \
        (__pvvwsys_)->dwScale = (__pvvwvid_)->dwScale;     \
        (__pvvwsys_)->dwRate = (__pvvwvid_)->dwRate;       \
        (__pvvwsys_)->dwLength= (__pvvwvid_)->dwLength;  \
        }
//! Set a #VVWSYSTEM structure pointer from #VVWAUDIO pointer, a granularity size
and number of frames
#define VVWSYS_SETAUDONLY(__pvvwsys_, __pvvwaud_, _granularity, _frames) {    \
```

```
        (__pvvwsys_)->dwMicroSecPerFrame = (DWORD)(1000000.0 / ((double)
(__pvvwaud_)->dwRate / (double)(__pvvwaud_)->dwScale));  \
        (__pvvwsys_)->dwMaxBytesPerSec = (__pvvwaud_)->nAvgBytesPerSec;         \
        (__pvvwsys_)->dwPaddingGranularity = _granularity;   \
        (__pvvwsys_)->dwTotalFrames = _frames;      \
        (__pvvwsys_)->dwSuggestedBufferSize = (__pvvwaud_)-
>dwSuggestedBufferSize;        \
        (__pvvwsys_)->dwWidth = 0;   \
        (__pvvwsys_)->dwHeight = 0;  \
        (__pvvwsys_)->dwScale = (__pvvwaud_)->dwScale;    \
        (__pvvwsys_)->dwRate = (__pvvwaud_)->dwRate;      \
        (__pvvwsys_)->dwLength= (__pvvwaud_)->dwLength; \
        }
//! Clear a important 0 of a #VVWVIDEO structure pointer
#define VVWSYS_CLR(__pvvwsys_) {   \
        (__pvvwsys_)->dwFlags = 0;    \
        (__pvvwsys_)->dwInitialFrames = 0;    \
        (__pvvwsys_)->dwReserved[0] = 0;     \
        (__pvvwsys_)->dwReserved[1] = 0;     \
        (__pvvwsys_)->dwReserved[2] = 0;     \
        (__pvvwsys_)->dwReserved[3] = 0;     \
        (__pvvwsys_)->dwCaps = 0;    \
        (__pvvwsys_)->dwEditCount = 0;        \
        (__pvvwsys_)->dwType = 0;    \
        (__pvvwsys_)->dwMfCaps = 0; \
        (__pvvwsys_)->dwVidStandard = 0;     \
        (__pvvwsys_)->dwDrFlags = 0; \
        (__pvvwsys_)->dwResDrastic = 0;       \
        }

// Macros (sample/len should work for all types)- See /Ass/Modules/MediaFile/Test
Patterns/Test Patterns.cpp for "Compile Test"
//! This macro is incorrect.  @todo Find and remove
#define VVWXXX_SETSAMPLETOLENGTH(__pvvw_, _length) { (__pvvw_)->dwLength =
_length * (__pvvw_)->dwScale; }
//! This macro is incorrect.  @todo Find and remove
#define VVWXXX_GETSAMPLEFROMLENGTH(__pvvw_) ((__pvvw_)->dwLength /
(__pvvw_)->dwScale)

////////////////////////////////////////////////////////////////
//
// The video structure provides basic information on the
//        current video format of a channel either on the disk, on
//        the network or in hardware.
//
// 03/13/98 Used By:
//                Modules\AvHal
//                Common\mfFile
//
```

```
//! The flag indicating the structure is a #VVWVIDEO structure within a union
#define _VVW_IS_VVWVIDEO            0x0010

/**
* The video structure is a combination of a BITMAPINFOHEADER and a
* AVISTREAMINFO structure.  The top half should be treated as a BITMAPINOFHEADER
* and the bottom as a related but independant structure.  These are as they
* appear in an AVI file and are manipulated to fill other file types like
* OMF and MOV.
*/
typedef struct /*tagVVWVIDEO*/ {
        // First part is a BITMAPINFOHEADER
        //! Size of the BITMAPINFOHEADER portion of this structure
(sizeof(BITMAPINFOHEADER) + any VVWVIDEO::dwReserved used)
        DWORD      biSize;                                  // Size Of BITMAPINO +
Used dwReserved
        //! Width of the video frame
   LONG       biWidth;                                 // Width of bitmap
        //! Height of the video frame.  CAUTION:  For vertically inverted frames this
WILL be negative
   LONG       biHeight;                            // Height of bitmap (ALWAYS Positive in
VVW/MR)
        //! Number of RGB groups (like photoshop layers) - Always 1 for our purposes
   WORD       biPlanes;                            // Number of bitmaps (ALWAYS 1)
        //! Number of bits per pixel (eg. YUV422=16, RGB=24, RGBA=32)
   WORD       biBitCount;                              // Bitcount (24 RGB, 16 YUV)
(Alpha seperate)
        //! Compression - a fourcc usually, but not always equal to fccHandler.  Denotes
compression type of frame - see fccDef.h
   DWORD      biCompression;                       // Should be same as fccHandler
        //! Size of the image.  For uncompressed biWidth*abs(biHeight)*(biBitCount/8)
in unsigned chars.  For compressed, variable.
   DWORD      biSizeImage;                             // Size of bitmap (biPitch *
biHeight) + Alpha
        //! Horizontal picture elements per meter - normally 0
   LONG       biXPelsPerMeter;                     // Picture elements per meter
        //! Vertical picture elements per meter - normally 0
   LONG       biYPelsPerMeter;                     // Picture elements per meter
        //! For colour tables in dwReserved, the number of RGBQUAD elements used
   DWORD      biClrUsed;                               // Number of pallette entries
used in bitmap
        //! For colour tables in dwReserved, the number of RGBQUAD elements that are
critical to display (for windows pallette wars in < 256 colour mode)
   DWORD      biClrImportant;                      // Required entries to display bitmap
        /**
        * The dwReserved may hold many things.  Whatever it holds, the amount used
can be determined by
        * subracting sizeof(BITMAPINFOHEADER) from VVWVIDEO::biSize.  Here are
some possible uses
```

```
        *
         \code
                Table of struct RGBQUAD { unsigned char rgbBlue, rgbGreen, rgbRed,
rgbReserved; };
         \endcode
         \code
                typedef struct tagJPEGINFOHEADER {
                        // compression-specific fields
                        // these fields are defined for 'JPEG' and 'MJPG'
                        DWORD       JPEGSize;
                        DWORD       JPEGProcess;

                        // Process specific fields
                        DWORD       JPEGColorSpaceID;
                        DWORD       JPEGBitsPerSample;
                        DWORD       JPEGHSubSampling;
                        DWORD       JPEGVSubSampling;
                } JPEGINFOHEADER;
         \endcode
         \code
                typedef struct tagVIDEOINFOHEADER {

                        RECT            rcSource;          // The bit we really want to use
                        RECT            rcTarget;          // Where the video should go
                        DWORD           dwBitRate;         // Approximate bit data rate
                        DWORD           dwBitErrorRate;    // Bit error rate for this stream
                        REFERENCE_TIME  AvgTimePerFrame;   // Average time per
frame (100ns units)

                        BITMAPINFOHEADER bmiHeader;

                } VIDEOINFOHEADER;
         \endcode
         \code
                typedef struct tagMPEG1VIDEOINFO {

                        VIDEOINFOHEADER hdr;                  // Compatible with
VIDEOINFO
                        DWORD           dwStartTimeCode;      // 25-bit Group of
pictures time code

                // at start of data
                        DWORD           cbSequenceHeader;     // Length in unsigned
chars of bSequenceHeader
                        unsigned char           bSequenceHeader[1];     // Sequence
header including

                // quantization matrices if any
                } MPEG1VIDEOINFO;
```

```
\endcode
\code
    typedef struct tagAnalogVideoInfo {
        RECT            rcSource;          // Width max is 720, height varies
w/ TransmissionStd
        RECT            rcTarget;          // Where the video should go
        DWORD           dwActiveWidth;     // Always 720 (CCIR-601
active samples per line)
        DWORD           dwActiveHeight;    // 483 for NTSC, 575 for
PAL/SECAM
        REFERENCE_TIME  AvgTimePerFrame;   // Normal ActiveMovie
units (100 nS)
    } ANALOGVIDEOINFO;
\endcode
\code
    typedef struct tagMPEG2VIDEOINFO {
        VIDEOINFOHEADER2    hdr;
        DWORD               dwStartTimeCode;       //  ?? not used for
DVD ??
        DWORD               cbSequenceHeader;      // is 0 for DVD (no
sequence header)
        DWORD               dwProfile;             // use enum MPEG2Profile
        DWORD               dwLevel;               // use enum MPEG2Level
        DWORD               dwFlags;               // use AMMPEG2_*
defines.  Reject connection if undefined bits are not 0
        DWORD               dwSequenceHeader[1];   // DWORD instead
of unsigned char for alignment purposes

        //   For MPEG-2, if a sequence_header is included, the
sequence_extension

        //   should also be included
    } MPEG2VIDEOINFO;
\endcode
\code
    typedef struct tagVIDEOINFOHEADER2 {
        RECT              rcSource;
        RECT              rcTarget;
        DWORD             dwBitRate;
        DWORD             dwBitErrorRate;
        REFERENCE_TIME    AvgTimePerFrame;
        DWORD             dwInterlaceFlags;   // use AMINTERLACE_*
defines. Reject connection if undefined bits are not 0
        DWORD             dwCopyProtectFlags; // use
AMCOPYPROTECT_* defines. Reject connection if undefined bits are not 0
        DWORD             dwPictAspectRatioX; // X dimension of picture
aspect ratio, e.g. 16 for 16x9 display
        DWORD             dwPictAspectRatioY; // Y dimension of picture
aspect ratio, e.g.  9 for 16x9 display
```

```
                        DWORD           dwReserved1;        // must be 0; reject
connection otherwise
                        DWORD           dwReserved2;        // must be 0; reject
connection otherwise
                        BITMAPINFOHEADER    bmiHeader;
                    } VIDEOINFOHEADER2;
                \endcode
            */
            DWORD           dwReserved[_VVWXXX_RESERVED_SIZE];      // Pallette or
extended bitmap stucture per 98/NT/2K
            // End BITMAPINFOHEADER

            // Second part is a AVIStreamHeader (AVISTREAMINFO)
            //! For VVWVIDEO structure this is always streamtypeVIDEO == 'vids'
    DWORD fccType;                                              // streamtypeVIDEO,
streamtypeAUDIO, streamtypeMIDI, streamtypeTEXT
            //! Codec type, see fccDef.h  Normally the same as VVWVIDEO::biCompression
but not always
            DWORD fccHandler;                                   // Codec - should be
same as biCompression
            /*
            * AVISTREAMINFO_ flags such as
            \code
            AVISTREAMINFO_DISABLED          0x00000001
            AVISTREAMINFO_FORMATCHANGES     0x00010000
            \endcode
            */
            DWORD dwFlags;                                      // Pos.
AVISTREAMINFO_FORMATCHANGES or AVISTREAMINFO_DISABLED
            //! Not sure.  See VVWSYSTEM::dwCaps for possible interp if something is set.
MS Doc:  currently unused
            DWORD dwCaps;                                       //
            //! Priority of stream (<-MSDoc in relation to other streams in the file I suppose)
    WORD  wPriority;                                    //
            //! Language of stream (<-MSDoc but no language id defines)
            WORD  wLanguage;                                    //
            //! dwRate/dwScale = frame rate.  See VVSYSTEM::dwScale for more info and
table example
            DWORD dwScale;                                      //  1001  100  1
1  - dwRate / dwScale == frame rate
            //! dwRate/dwScale = frame rate.  See VVSYSTEM::dwRate for more info and
table example
            DWORD dwRate;                                       // 30000 2997 25
24
            /**
            * Delay in units per VVWVIDEO::dwRate/VVWVIDEO::dwScale (for video -
frames) for this
            * stream to start in the playback of the file.  NOTE AVI v1.0 and simple avi
readers
```

```
        * will choke or play incorrectly if this is not 0, so be careful.
        */
   DWORD dwStart;                                    // Starting sample per
dwRate/dwScale
        /**
        * Length of the video stream in units per
VVWVIDEO::dwRate/VVWVIDEO::dwScale (for video - frames)
        */
        DWORD dwLength;                                    // Length of
stream per dwRate/dwScale
        /**
        * Amount of audio in the file before video commences.  For offset files, typically
0.75 sec converted
        * to units per VVWVIDEO::dwRate/VVWVIDEO::dwScale.  For high end files,
always zero as audio
        * and video are sent without scew (except premiere, which uses 'rec ' chunks
and audio skew)
        */
        DWORD dwInitialFrames;                             // Audio skew, how much
startup audio in stream
        /**
        * Recommended buffer size based on the largest single chunk in the file.  Set by
        * writer, so often incorrect or 0.
        */
   DWORD dwSuggestedBufferSize;              // Largest chunk in stream, internally one
uncompressed frame
        /**
        * Quality used by the compressor.  Between 0 and 10,000 or -1 if default quality.
For some
        * compressors, the -1 can also mean the quality info is encoded into the frame or
in the
        * dwReserved or other private data area.
        */
        DWORD dwQuality;                                   // Codec compression
quality
        /**
        * Size, in unsigned chars, of a single data sample. If the value of this member is
zero, the samples
        * can vary in size and each data sample (such as a video frame) must be in a
separate chunk.
        * A nonzero value indicates that multiple samples of data can be grouped into a
single chunk
        * within the file. For video streams, this number is typically zero, although it can
be nonzero
        * if all video frames are the same size. For audio streams, this number should be
the same
        * as the nBlockAlign member of the WAVEFORMAT or WAVEFORMATEX structure
describing the audio.
        */
```

```
        DWORD dwSampleSize;                                     // Largest single sample
        /**
        * Dimensions of the video destination rectangle. The values represent the
coordinates of
        * upper left corner, the height, and the width of the rectangle.
        */
        RECT/*16*/  rcFrame;                                    // Frame dimensions
        //! Number of times the stream has been edited. The stream handler maintains
this count.
        DWORD dwEditCount;                                      // Number of time
stream has been edited
        //! Number of times the stream format has changed. The stream handler
maintains this count.
        DWORD dwFormatChangeCount;                              // Number of time format
has been changed
        //! Null-terminated string containing a description of the stream.
    char  szName[_VVWXXX_NAME_SIZE];        // Stream identifier
        // End of AVIStreamHeader  (AVISTREAMINFO)

        // Special pitch values
#define VVWVIDEO_720P_YCBCBR10   3456
#define VVWVIDEO_2048_YCBCBR10   5504
        //! The number of unsigned chars in a row of pixels.  Allows for unsigned
char/WORD/DWORD alignment of lines as nec for format
        LONG biPitch;                                           // Normally biWidth *
(biBitCount / 8) - Number of unsigned chars in a row
//
// Check mask bits for free areas
// note: fccmod depends on biCompression/fccHandler
//! Field dominance MASK for VVWVIDEO::dwDrFlags
#define MASK___DRFLAGS_FIELD                    0x00000001
//! Zero (Second) Field Dominant in VVWVIDEO::dwDrFlags
#define DRFLAGS_ZERO_FIELD_DOMINANT  0x00000000
//! First Field Dominant in VVWVIDEO::dwDrFlags
#define DRFLAGS_FIRST_FIELD_DOMINANT 0x00000001
//! Preview to QuickclipXO
#define DRFLAGS_NOT_QUICKCLIP                0x00000002
//! KeyFrame MASK for VVWVIDEO::dwDrFlags
#define MASK___DRFLAGS_KEYFRAME              0x00000010
//! Stream has key frames, else all key frames VVWVIDEO::dwDrFlags
#define DRFLAGS_HAS_KEYFRAMES                0x00000010
//! Frame type mask (1=interlaced,2=progressive,4=segmentedframe)
#define MASK___DRFLAGS_VTYPE                    0x00000700
//! Interlaced video frames
#define DRFLAGS_VTYPE_INTERLACED 0x00000100
//! Progressive video frames
#define DRFLAGS_VTYPE_PROGRESSIVE          0x00000200
//! Segmented Frame video frames
#define DRFLAGS_VTYPE_SEGMENTEDFRAME 0x00000400
```

```
//! Is opening for a compression (0== decompression)
#define DRFLAGS_IS_COMPRESS                         0x000001000
//! Fourcc modifiers MASK for VVWVIDEO::dwDrFlags
#define MASK___FCCMODIFIERS                         0x00FF0000
//! Stream is DigiSuite MJPG see VVWVIDEO::dwDrFlags
#define DRFLAGS_FCC_MJPG_DIGISUITE      0x00000000    // resolve 'mjpg' types
//! Stream is Miro DC50 MJPG see VVWVIDEO::dwDrFlags
#define DRFLAGS_FCC_MJPG_DCx0           0x00010000
//! Stream is DigiSuite Edit MJPG see VVWVIDEO::dwDrFlags
#define DRFLAGS_FCC_MJPG_DSEDIT         0x00020000
//! Stream is MJPG MS-Dib variant see VVWVIDEO::dwDrFlags
#define DRFLAGS_FCC_MJPG_JPGDIB         0x00040000
//! Stream is JFIF jpeg see VVWVIDEO::dwDrFlags
#define DRFLAGS_FCC_MJPG_JFIF           0x00080000
//! Stream should use internal codecs see VVWVIDEO::dwDrFlags
#define DRFLAGS_FCC_USE_INTERN          0x00100000    // which codec dup to
use
//! Stream should use quicktime codecs see VVWVIDEO::dwDrFlags
#define DRFLAGS_FCC_USE_QT                          0x00400000
//! Stream should use windows icm/vfw codecs see VVWVIDEO::dwDrFlags
#define DRFLAGS_FCC_USE_ICM                         0x00800000
//! Private Data MASK see VVWVIDEO::dwDrFlags
#define MASK__CODECPRIVATEDATA          0x0F000000
//! Private data format is AVI see VVWVIDEO::dwDrFlags
#define DRFLAGS_CODECPRIVATEDATA_AVI 0x01000000             // Std AVI
VVWVIDEO
//! Private data format is MOV see VVWVIDEO::dwDrFlags
#define DRFLAGS_CODECPRIVATEDATA_MOV 0x02000000             // ImageDesc in
VVWVIDEO (as above)
//! Private data format is OMF see VVWVIDEO::dwDrFlags
#define DRFLAGS_CODECPRIVATEDATA_OMF 0x04000000             // OMF Util
//! MASK Room for current DTVVW_PREVIEW and more if nec see
VVWVIDEO::dwDrFlags
#define MASK___PREVIEW                              0xF0000000
        /**
        * Our internal flags including:
        * #DRFLAGS_ZERO_FIELD_DOMINANT, #DRFLAGS_FIRST_FIELD_DOMINANT,
#DRFLAGS_HAS_KEYFRAMES,
        * #DRFLAGS_FCC_MJPG_DIGISUITE, #DRFLAGS_FCC_MJPG_DCx0,
#DRFLAGS_FCC_MJPG_DSEDIT,
        * #DRFLAGS_FCC_MJPG_JPGDIB, #DRFLAGS_FCC_MJPG_JFIF,
        * #DRFLAGS_FCC_USE_INTERN, #DRFLAGS_FCC_USE_QT,
#DRFLAGS_FCC_USE_ICM,
        * #DRFLAGS_CODECPRIVATEDATA_AVI,
#DRFLAGS_CODECPRIVATEDATA_MOV, #DRFLAGS_CODECPRIVATEDATA_OMF,
        * #DTVVW_PREVIEW
        */
        DWORD dwDrFlags;                                    // Our internal flags
        //! Source File Type
```

```
        DWORD dwFileType;
//! Reserved, init to zero and leave alone
        DWORD dwResDrastic;                                     // Always

} VVWVIDEO, * pVVWVIDEO;

// Macros - See /Ass/Modules/MediaFile/Test Patterns/Test Patterns.cpp for "Compile
Test"
//! Set VVWVIDEO::biPitch member to 1 byte alignment based on VVWVIDEO::biWidth
and VVWVIDEO::biBitCount
#define VVWVID_PITCHALIGN1(__pvvwvid_)  { (__pvvwvid_)->biPitch =
(((__pvvwvid_)->biWidth * (__pvvwvid_)->biBitCount+  7) / 8 ); }
//! Set VVWVIDEO::biPitch member to 4 byte (DWORD) alignment based on
VVWVIDEO::biWidth and VVWVIDEO::biBitCount
#define VVWVID_PITCHALIGN4(__pvvwvid_)  { (__pvvwvid_)->biPitch =
(((__pvvwvid_)->biWidth * (__pvvwvid_)->biBitCount+ 31) / 32) * 4; }
//! Set VVWVIDEO::biPitch member to 8 byte alignment based on VVWVIDEO::biWidth
and VVWVIDEO::biBitCount
#define VVWVID_PITCHALIGN8(__pvvwvid_)  { (__pvvwvid_)->biPitch =
(((__pvvwvid_)->biWidth * (__pvvwvid_)->biBitCount+ 63) / 64) * 8; }
//! Set VVWVIDEO::biPitch member to 16 byte alignment based on VVWVIDEO::biWidth
and VVWVIDEO::biBitCount
#define VVWVID_PITCHALIGN16(__pvvwvid_) { (__pvvwvid_)->biPitch =
(((__pvvwvid_)->biWidth * (__pvvwvid_)->biBitCount+127) / 128) * 16; }
//! Set VVWVIDEO::biPitch member to alignment specified based on VVWVIDEO::biWidth
and VVWVIDEO::biBitCount
#define VVWVID_PITCHALIGNANY(__pvvwvid_, _align_1_4_8_16) { (__pvvwvid_)-
>biPitch = (((__pvvwvid_)->biWidth * (__pvvwvid_)->biBitCount+((_align_1_4_8_16 *
8) - 1)) / (_align_1_4_8_16 * 8)) * _align_1_4_8_16; }
//! Set pitch to 1-0 alignment ?
#define VVWVID_PITCHMODULO(__pvvwvid_) ((__pvvwvid_)->biPitch - ((__pvvwvid_)-
>biWidth * (__pvvwvid_)->biBitCount + 7) / 8)
//! Set the VVWVIDEO::biSizeImage based on VVWVIDEO::biWidth,
VVWVIDEO::biHeight and VVWVIDEO::biBitCount
#define VVWVID_SIZEIMAGE(__pvvwvid_) { if(!(__pvvwvid_)->biPitch)
VVWVID_PITCHALIGN1((__pvvwvid_)); (__pvvwvid_)->biSizeImage = (__pvvwvid_)-
>biPitch * abs((__pvvwvid_)->biHeight); }
//! Set the VVWVIDEO::dwSuggestedBufferSize based on VVWVIDEO::biWidth,
VVWVIDEO::biHeight and VVWVIDEO::biBitCount
#define VVWVID_SUGGESTEDBUFFERSIZE(__pvvwvid_) { if(!(__pvvwvid_)->biPitch)
VVWVID_PITCHALIGN1((__pvvwvid_)); (__pvvwvid_)->dwSuggestedBufferSize =
(__pvvwvid_)->biPitch * (abs((__pvvwvid_)->biHeight) + (abs((__pvvwvid_)->biHeight)
>> 2)); }
//! Set the VVWVIDEO::dwRate and VVWVIDEO::dwScale for 30000/1001 (NTSC)
#define VVWVID_RATESCALE_NTSC_AVI(__pvvwvid_) { (__pvvwvid_)->dwScale =
1001; (__pvvwvid_)->dwRate = 30000; }
//! Set the VVWVIDEO::dwRate and VVWVIDEO::dwScale for 2997/100 (NTSC)
#define VVWVID_RATESCALE_NTSC(__pvvwvid_) { (__pvvwvid_)->dwScale = 100;
(__pvvwvid_)->dwRate = 2997; }
```

```
//! Set the VVWVIDEO::dwRate and VVWVIDEO::dwScale for 25/1 (PAL)
#define VVWVID_RATESCALE_PAL(__pvvwvid_) { (__pvvwvid_)->dwScale = 1;
(__pvvwvid_)->dwRate = 25; }
//! Set the VVWVIDEO::dwRate and VVWVIDEO::dwScale for 24/1 (FILM)
#define VVWVID_RATESCALE_FILM(__pvvwvid_) { (__pvvwvid_)->dwScale = 1;
(__pvvwvid_)->dwRate = 24; }
//! Basic setup of a #VVWVIDEO pointer from width, height, align, codec, scale, rate and
length in frames
#define VVWVID_SET(__pvvwvid_, _bitcount, _width, _height, _pitch_align_1_4_8_16,
_fcccodec, _scale, _rate, _frames) {     \
        (__pvvwvid_)->biBitCount = _bitcount; \
        if((LONG)_height < 0) { (__pvvwvid_)->biHeight = -(LONG)_height; } else
{ (__pvvwvid_)->biHeight = (LONG)_height; }  \
        (__pvvwvid_)->biWidth = _width;        \
        (__pvvwvid_)->biCompression = _fcccodec;     \
        VVWVID_PITCHALIGNANY(__pvvwvid_, _pitch_align_1_4_8_16);         \
        VVWVID_SIZEIMAGE((__pvvwvid_));    \
        VVWVID_SUGGESTEDBUFFERSIZE((__pvvwvid_));       \
        (__pvvwvid_)->dwScale = _scale; (__pvvwvid_)->dwRate = _rate;      \
        VVWXXX_SETSAMPLETOLENGTH((__pvvwvid_), _frames);       \
        (__pvvwvid_)->fccHandler = (__pvvwvid_)->biCompression;    \
        (__pvvwvid_)->rcFrame.top = 0;         \
        (__pvvwvid_)->rcFrame.bottom = abs((__pvvwvid_)->biHeight);         \
        (__pvvwvid_)->rcFrame.left = 0;         \
        (__pvvwvid_)->rcFrame.right = (__pvvwvid_)->biWidth;         \
        }
//! Clean and do basic setup of a #VVWVIDEO pointer from width, height, align, codec,
scale, rate and length in frames
#define VVWVID_SETCLR(__pvvwvid_, _bitcount, _width, _height,
_pitch_align_1_4_8_16, _fcccodec, _scale, _rate, _frames) {     \
        VVWVID_SET((__pvvwvid_), _bitcount, _width, _height, _pitch_align_1_4_8_16,
_fcccodec, _scale, _rate, _frames); \
        (__pvvwvid_)->biSize = sizeof(BITMAPINFOHEADER);   \
        (__pvvwvid_)->biPlanes = 1;    \
        (__pvvwvid_)->biXPelsPerMeter = 0;    \
        (__pvvwvid_)->biYPelsPerMeter = 0;    \
        (__pvvwvid_)->biClrUsed = 0;  \
        (__pvvwvid_)->biClrImportant = 0;      \
        ZeroMemory((__pvvwvid_)->dwReserved, _VVWXXX_RESERVED_SIZE);          \
        (__pvvwvid_)->fccType = dtstreamtypeVIDEO; \
        (__pvvwvid_)->dwFlags = 0;    \
        (__pvvwvid_)->dwCaps = 0;     \
        (__pvvwvid_)->wPriority = 0;   \
        (__pvvwvid_)->wLanguage = 0;         \
        (__pvvwvid_)->dwStart = 0;     \
        (__pvvwvid_)->dwInitialFrames = 0;     \
        (__pvvwvid_)->dwQuality = 0xFFFFFFFF;        \
        (__pvvwvid_)->dwSampleSize = 0;       \
        (__pvvwvid_)->dwEditCount = 0;        \
```

```
        (__pvvwvid_)->dwFormatChangeCount = 0;     \
        ZeroMemory((__pvvwvid_)->szName, _VVWXXX_NAME_SIZE); \
        (__pvvwvid_)->dwDrFlags = 0; \
        (__pvvwvid_)->dwResDrastic = 0;       \
        }

////////////////////////////////////////////////////////////////
//
// The audio structure provides basic information on the
//      current audio format of a channel either on the disk, on
//      the network or in hardware.
//
// 03/13/98 Used By:
//              Modules\AvHal
//              Common\mfFile
//
//! The flag indicating the structure is a #VVWAUDIO structure within a union
#define _VVW_IS_VVWAUDIO            0x0100

/**
* The audio structure is a combination of a WAVEFORMATEX and a
* AVISTREAMINFO structure.  The top half should be treated as a WAVEFORMATEX
* and the bottom as a related but independant structure.  These are as they
* appear in an AVI file and are manipulated to fill other file types like
* OMF and MOV.
*/
typedef struct /*tagVVWAUDIO*/ {
        //! PCM Wave Type, see fccDef.h for other possible types
#define DTWAVE_FORMAT_PCM                                   1
#define DTWAVE_FORMAT_EXTENSIBLE        0xFFFE

        //! Format of audio data, uncompressed == #DTWAVE_FORMAT_PCM (windows
WAVE_FORMAT_PCM) == 1
   WORD       wFormatTag;                          /* format type (normally
WAVE_FORMAT_PCM == 1) */
        //! Number of channels in this stream.  Usually 1 or 2
   WORD       nChannels;                           /* number of channels (i.e.
mono=1, stereo=2...) */
        //! Number of samples per second.  eg. 44100, 22050, 48000 etc
   DWORD      nSamplesPerSec;                      /* sample rate (i.e. 44100,
48000. 22050) */
        //! Average unsigned chars Per Second (see also VVWAUDIO::dwRate) ==
#nBlockAlign * #nSamplesPerSec
   DWORD      nAvgBytesPerSec;             /* for buffer estimation (nSamplesPerSec
* nBlockAlign) */
        //! Size of a sample group.  == to #nChannels * (#wBitsPerSample / 8) eg.
Stereo 16bit = 4, Mono 16bit = 2, Stereo 8Bit = 2
   WORD       nBlockAlign;                         /* block size of data (nChannels +
((wBitsPerSample + 7) / 8)) */
```

```
        //! Number of bits per sample, normally 8 or 16
    WORD        wBitsPerSample;                 /* number of bits per sample of mono
data (Normally 8, 16, 20, 24 or 32) */
        //! Size of the dwReserved area used.  For PCM this will be zero.  For other
compressors, it may be anything < 256 DWORDs
    WORD        cbSize;                                 /* the count in unsigned chars of
the size of */
                                    /* extra information (after cbSize) */
        //! Extra info storage for audio codecs.
        DWORD           dwReserved[_VVWXXX_RESERVED_SIZE];      // Some drivers
req more spave
        // End of WAVEFORMATEX

        // Second part is a AVIStreamHeader (AVISTREAMINFO)
        //! For VVWAUDIO structure this is always streamtypeAUDIO == 'auds'
    DWORD fccType;                                              // streamtypeVIDEO,
streamtypeAUDIO, streamtypeMIDI, streamtypeTEXT
        //! Codec type, see fccDef.h  Normally the same as VVWAUDIO::wFormatTag but
not always
        DWORD fccHandler;                                       // Codec - should be
same as biCompression
        /*
        * AVISTREAMINFO_ flags such as
        \code
        AVISTREAMINFO_DISABLED              0x00000001
        AVISTREAMINFO_FORMATCHANGES     0x00010000
        \endcode
        */
        DWORD dwFlags;                                          // Pos.
AVISTREAMINFO_FORMATCHANGES or AVISTREAMINFO_DISABLED
        //! Not sure.  See VVWSYSTEM::dwCaps for possible interp if something is set.
MS Doc:  currently unused
        DWORD dwCaps;                                           //
        //! Priority of stream (<-MSDoc in relation to other streams in the file I suppose)
    WORD  wPriority;                                    //
        //! Language of stream (<-MSDoc but no language id defines)
        WORD  wLanguage;                                        //
        //! For PCM == to VVWAUDIO::nBlockAlign.  See VVSYSTEM::dwScale for more
info and table example
        DWORD dwScale;                                          //  1001  100  1
1  - dwRate / dwScale == frame rate
        //! For PCM == to VVWAUDIO::nAvgBytesPerSec.  See VVSYSTEM::dwRate for
more info and table example
        DWORD dwRate;                                           // 30000 2997 25
24
        /**
        * Delay in units per VVWAUDIO::dwRate/VVWAUDIO::dwScale (for video -
frames) for this
```

```
        * stream to start in the playback of the file.  NOTE AVI v1.0 and simple avi
readers
        * will choke or play incorrectly if this is not 0, so be careful.
        */
   DWORD dwStart;                                  // Starting sample per
dwRate/dwScale
        /**
        * Length of the video stream in units per
VVWAUDIO::dwRate/VVWAUDIO::dwScale (for video - frames)
        */
        DWORD dwLength;                                  // Length of
stream per dwRate/dwScale
        /**
        * Amount of audio in the file before video commences.  For offset files, typically
0.75 sec converted
        * to units per VVWAUDIO::dwRate/VVWAUDIO::dwScale.  For high end files,
always zero as audio
        * and video are sent without scew (except premiere, which uses 'rec ' chunks
and audio skew)
        */
        DWORD dwInitialFrames;                           // Audio skew, how much
startup audio in stream
        /**
        * Recommended buffer size based on the largest single chunk in the file.  Set by
        * writer, so often incorrect or 0.
        */
   DWORD dwSuggestedBufferSize;          // Largest chunk in stream, internally one
uncompressed frame
        /**
        * Quality used by the compressor.  Between 0 and 10,000 or -1 if default quality.
For some
        * compressors, the -1 can also mean the quality info is encoded into the frame or
in the
        * dwReserved or other private data area.
        */
        DWORD dwQuality;                                 // Codec compression
quality
        /**
        * Size, in unsigned chars, of a single data sample. If the value of this member is
zero, the samples
        * can vary in size and each data sample (such as a video frame) must be in a
separate chunk.
        * A nonzero value indicates that multiple samples of data can be grouped into a
single chunk
        * within the file. For video streams, this number is typically zero, although it can
be nonzero
        * if all video frames are the same size. For audio streams, this number should be
the same
```

```
        * as the nBlockAlign member of the WAVEFORMAT or WAVEFORMATEX structure
describing the audio.
        */
        DWORD dwSampleSize;                                     // Largest single sample
        /**
        * NOT USED BY VVWAUDIO. Dimensions of the video destination rectangle. The
values represent
        * the coordinates of upper left corner, the height, and the width of the rectangle.
        */
        RECT/*16*/  rcFrame;                                    // Frame dimensions
        //! Number of times the stream has been edited. The stream handler maintains
this count.
        DWORD dwEditCount;                                      // Number of time
stream has been edited
        //! Number of times the stream format has changed. The stream handler
maintains this count.
        DWORD dwFormatChangeCount;                              // Number of time format
has been changed
        //! Null-terminated string containing a description of the stream.
    char  szName[_VVWXXX_NAME_SIZE];        // Stream identifier
        // End of AVIStreamHeader  (AVISTREAMINFO)

        //! No used currenly
        LONG biUnused;                                                  //
        /**
        * Our internal flags including:
        * #DRFLAGS_HAS_KEYFRAMES,
        * #DRFLAGS_FCC_USE_INTERN, #DRFLAGS_FCC_USE_QT,
#DRFLAGS_FCC_USE_ICM,
        * #DRFLAGS_CODECPRIVATEDATA_AVI,
#DRFLAGS_CODECPRIVATEDATA_MOV, #DRFLAGS_CODECPRIVATEDATA_OMF,
        * #DTVVW_PREVIEW
        */
        DWORD dwDrFlags;                                        // Our internal flags
        //! Source File Type
        DWORD dwFileType;
//! Reserved, init to zero and leave alone
        DWORD dwResDrastic;                                     // Always

} VVWAUDIO, * pVVWAUDIO;

/**
* Recalculate audio structure:  Uses VVWAUDIO::nChannels,
VVWAUDIO::wBitsPerSample and VVWAUDIO::nSamplesPerSec
* to caclulate the VVWAUDIO::nBlockAlign, VVWAUDIO::nAvgBytesPerSec,
VVWAUDIO::dwSuggestedBufferSize,
* VVWAUDIO::dwSampleSize, VVWAUDIO::dwScale and VVWAUDIO::dwRate members -
Only valid of PCM
* uncompressed audio streams.
```

```
* NOTE:  nBlockAlign = nChannels * ((wBitsPerSample + 7) >> 3)  gives the minimum container size
* for the bit size.  In our write and intenal cases, we always use 32 bits for 20 and 24. This calc
* return 24 for 20 and 24.  We need this for the read side, so make sure any write size stuff is caught.
*/
#define VVWAUD_RECALC(__pvvwaud_) {       \
        if((__pvvwaud_)->wFormatTag == DTWAVE_FORMAT_PCM) { \
                (__pvvwaud_)->nBlockAlign = ((__pvvwaud_)->nChannels * (((__pvvwaud_)->wBitsPerSample + 7) >> 3));          \
                (__pvvwaud_)->nAvgBytesPerSec = (__pvvwaud_)->nBlockAlign * (__pvvwaud_)->nSamplesPerSec;        \
                (__pvvwaud_)->dwSuggestedBufferSize = (__pvvwaud_)->nAvgBytesPerSec >> 1;        \
                (__pvvwaud_)->dwSampleSize = (__pvvwaud_)->nBlockAlign;  \
                (__pvvwaud_)->dwScale = (__pvvwaud_)->nBlockAlign;          \
                (__pvvwaud_)->dwRate = (__pvvwaud_)->nAvgBytesPerSec;   \
        }       \
        }
//! Set the #VVWAUDIO structure pointer from formattag, channels, samples per sec, bits per sample and number of samples
#define VVWAUD_SET(__pvvwaud_, _formattag, _channels, _samplespersec, _bitspersample, _samples) {     \
        (__pvvwaud_)->wFormatTag = _formattag;      \
        (__pvvwaud_)->fccHandler = _formattag;      \
        (__pvvwaud_)->nChannels = _channels;        \
        (__pvvwaud_)->nSamplesPerSec = _samplespersec;     \
        (__pvvwaud_)->wBitsPerSample = _bitspersample;      \
        (__pvvwaud_)->dwLength = _samples;\
        VVWAUD_RECALC((__pvvwaud_));       \
        }
//! Clean and set the #VVWAUDIO structure pointer from formattag, channels, samples per sec, bits per sample and number of samples
#define VVWAUD_SETCLR(__pvvwaud_, _formattag, _channels, _samplespersec, _bitspersample, _samples) {     \
        VVWAUD_SET((__pvvwaud_), _formattag, _channels, _samplespersec, _bitspersample, _samples); \
        ZeroMemory((__pvvwaud_)->dwReserved, _VVWXXX_RESERVED_SIZE);          \
        (__pvvwaud_)->fccType = dtstreamtypeAUDIO;          \
        (__pvvwaud_)->dwFlags = 0;   \
        (__pvvwaud_)->dwCaps = 0;    \
        (__pvvwaud_)->wPriority = 0;  \
        (__pvvwaud_)->wLanguage = 0;        \
        (__pvvwaud_)->dwStart = 0;    \
        (__pvvwaud_)->dwInitialFrames = 0;    \
        (__pvvwaud_)->dwQuality = 0xFFFFFFFF;       \
        (__pvvwaud_)->dwEditCount = 0;       \
        (__pvvwaud_)->dwFormatChangeCount = 0;    \
```

```
        ZeroMemory((__pvvwaud_)->szName, _VVWXXX_NAME_SIZE);\
        (__pvvwaud_)->dwDrFlags = 0;          \
        (__pvvwaud_)->dwResDrastic = 0;       \
        }

////////////////////////////////////////////////////////////////
//
// The info structure provides basic information on the
//      current information channels available.  For now, this
//      is mostly LTC in file, SmpteX LTC info, current time
//      and data, but it leaves from for other VITC readers,
//      close caption decode and comment.
//
// 03/13/98 Used By:
//              Common\mfFile
//
//! The flag indicating the structure is a #VVWINFO structure within a union
#define _VVW_IS_VVWINFO                         0x1000
/** Numeric values for all the metadata information types available in MR and VVW
*/
enum vvwInfoMetaTypes {
        //! see VVWINFO::szFileName
        vvwiFileName,
        //! see VVWINFO::szNativeLocator
        vvwiNativeLocator,
        //! see VVWINFO::szUniversalName
        vvwiUniversalName,
        //! see VVWINFO::szIP
        vvwiIP,
        //! see VVWINFO::szSourceLocator
        vvwiSourceLocator,

        //! see VVWINFO::szChannel
        vvwiChannel,
        //! see VVWINFO::szChannelName
        vvwiChannelName,
        //! see VVWINFO::szChannelDescription
        vvwiChannelDescription,
        //! see VVWINFO::szTitle
        vvwiTitle,
        //! see VVWINFO::szSubject
        vvwiSubject,
        //! see VVWINFO::szCategory
        vvwiCategory,                   // <-- 10
        //! see VVWINFO::szKeywords
        vvwiKeywords,
        //! see VVWINFO::szRatings
        vvwiRatings,
        //! see VVWINFO::szComments
```

```
vvwiComments,
//! see VVWINFO::szOwner
vvwiOwner,
//! see VVWINFO::szEditor
vvwiEditor,
//! see VVWINFO::szSupplier
vvwiSupplier,
//! see VVWINFO::szSource
vvwiSource,
//! see VVWINFO::szProject
vvwiProject,
//! see VVWINFO::szStatus
vvwiStatus,
//! see VVWINFO::szAuthor
vvwiAuthor,                                    // <-- 20
//! see VVWINFO::szRevisionNumber
vvwiRevisionNumber,
//! see VVWINFO::szProduced
vvwiProduced,
//! see VVWINFO::szAlbum
vvwiAlbum,
//! see VVWINFO::szArtist
vvwiArtist,
//! see VVWINFO::szComposer
vvwiComposer,
//! see VVWINFO::szCopyright
vvwiCopyright,
//! see VVWINFO::szCreationData
vvwiCreationData,
//! see VVWINFO::szDescription
vvwiDescription,
//! see VVWINFO::szDirector
vvwiDirector,
//! see VVWINFO::szDisclaimer
vvwiDisclaimer,                        // <-- 30
//! see VVWINFO::szEncodedBy
vvwiEncodedBy,
//! see VVWINFO::szFullName
vvwiFullName,
//! see VVWINFO::szGenre
vvwiGenre,
//! see VVWINFO::szHostComputer
vvwiHostComputer,
//! see VVWINFO::szInformation
vvwiInformation,
//! see VVWINFO::szMake
vvwiMake,
//! see VVWINFO::szModel
vvwiModel,
```

```
//! see VVWINFO::szOriginalArtist
vvwiOriginalArtist,
//! see VVWINFO::szOriginalFormat
vvwiOriginalFormat,
//! see VVWINFO::szPerformers
vvwiPerformers,                         // <-- 40
//! see VVWINFO::szProducer
vvwiProducer,
//! see VVWINFO::szProduct
vvwiProduct,
//! see VVWINFO::szSoftware
vvwiSoftware,
//! see VVWINFO::szSpecialPlaybackRequirements
vvwiSpecialPlaybackRequirements,
//! see VVWINFO::szTrack
vvwiTrack,
//! see VVWINFO::szWarning
vvwiWarning,
//! see VVWINFO::szURLLink
vvwiURLLink,
//! see VVWINFO::szEditData1
vvwiEditData1,
//! see VVWINFO::szEditData2
vvwiEditData2,
//! see VVWINFO::szEditData3
vvwiEditData3,                  // <-- 50
//! see VVWINFO::szEditData4
vvwiEditData4,
//! see VVWINFO::szEditData5
vvwiEditData5,
//! see VVWINFO::szEditData6
vvwiEditData6,
//! see VVWINFO::szEditData7
vvwiEditData7,
//! see VVWINFO::szEditData8
vvwiEditData8,
//! see VVWINFO::szEditData9
vvwiEditData9,
//! see VVWINFO::szVersionString
vvwiVersionString,
//! see VVWINFO::szManufacturer
vvwiManufacturer,
//! see VVWINFO::szLanguage
vvwiLanguage,
//! see VVWINFO::szFormat
vvwiFormat,                             // <-- 60
//! see VVWINFO::szInputDevice
vvwiInputDevice,
//! see VVWINFO::szDeviceModelNum
```

```
    vvwiDeviceModelNum,
    //! see VVWINFO::szDeviceSerialNum
    vvwiDeviceSerialNum,
    //! see VVWINFO::szReel
    vvwiReel,
    //! see VVWINFO::szShot
    vvwiShot,
    //! see VVWINFO::szTake
    vvwiTake,
    //! see VVWINFO::szSlateInfo
    vvwiSlateInfo,
    //! see VVWINFO::szFrameAttribute
    vvwiFrameAttribute,
    //! see VVWINFO::szEpisode
    vvwiEpisode,
    //! see VVWINFO::szScene
    vvwiScene,                                  // <-- 70
    //! see VVWINFO::szDailyRoll
    vvwiDailyRoll,
    //! see VVWINFO::szCamRoll
    vvwiCamRoll,
    //! see VVWINFO::szSoundRoll
    vvwiSoundRoll,
    //! see VVWINFO::szLabRoll
    vvwiLabRoll,
    //! see VVWINFO::szKeyNumberPrefix
    vvwiKeyNumberPrefix,
    //! see VVWINFO::szInkNumberPrefix
    vvwiInkNumberPrefix,
    //! see VVWINFO::szPictureIcon
    vvwiPictureIcon,
    //! see VVWINFO::szProxyFile
    vvwiProxyFile,

    vvwiEND_OF_STRINGS,

    vvwiNumericStart = 0x1000,
    //! see VVWINFO::dwTimeCode
    vvwiTimeCode,
    //! see VVWINFO::dwUserBits
    vvwiUserBits,
    //! see VVWINFO::dwVITCTimeCode
    vvwiVITCTimeCode,
    //! see VVWINFO::dwVITCUserBits
    vvwiVITCUserBits,
    //! see VVWINFO::dwVITCLine3
    vvwiVITCLine3,
    //! see VVWINFO::dwPosterFrame
    vvwiPosterFrame,
```

```
//! see VVWINFO::dwAFrame
vvwiAFrame,
//! see VVWINFO::dwAspectRatio
vvwiAspectRatio,
//! see VVWINFO::dwOriginalRate
vvwiOriginalRate,
//! see VVWINFO::dwOriginalScale
vvwiOriginalScale,
//! see VVWINFO::dwConversions
vvwiConversions,
//! see VVWINFO::dwVersionNumber
vvwiVersionNumber,
//! see VVWINFO::dwFileSize
vvwiFileSize,
//! see VVWINFO::dwFileDate
vvwiFileDate,
//! see VVWINFO::dwFileTime
vvwiFileTime,
//! see VVWINFO::dwSequenceNumber
vvwiSequenceNumber,
//! see VVWINFO::dwTotalStreams
vvwiTotalStreams,
//! see VVWINFO::dwTotalLength
vvwiTotalLength,
//! see VVWINFO::dwFilmManufacturerCode
vvwiFilmManufacturerCode,
//! see VVWINFO::dwFilmTypeCode
vvwiFilmTypeCode,
//! see VVWINFO::dwWhitePoint
vvwiWhitePoint,
//! see VVWINFO::dwBlackPoint
vvwiBlackPoint,
//! see VVWINFO::dwBlackGain
vvwiBlackGain,
//! see VVWINFO::dwBreakPoint
vvwiBreakPoint,
//! see VVWINFO::dwGamma1000
vvwiGamma1000,
//! see VVWINFO::dwTagNumber
vvwiTagNumber,
//! see VVWINFO::dwFlags
vvwiFlags,
//! see VVWINFO::dwTimeCodeType
vvwiTimeCodeType,
//! see VVWINFO::dwLTCTimeCodeType
vvwiLTCTimeCodeType,
//! see VVWINFO::dwVITCTimeCodeType
vvwiVITCTimeCodeType,
//! see VVWINFO::dwProdDate
```

```
        vvwiProdDate,
        //End: v3.0
        //! see VVWINFO::dwUniqueID
        vvwiUniqueID,

        vvwiEND_OF_DWORD_V2,
        // Add elements here
        //VVVID STRUCT
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoWidth = 0x10000,
        //! XML tag name for width
#define VVWINFOTAG_woVideoWidth                 "Width"
#define VVWINFODESC_woVideoWidth            "Width"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoHeight,
        //! XML tag name for height
#define VVWINFOTAG_woVideoHeight            "Height"
#define VVWINFODESC_woVideoHeight               "Height"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoPlanes,
        //! XML tag name for planes
#define VVWINFOTAG_woVideoPlanes            "Planes"
#define VVWINFODESC_woVideoPlanes           "Planes"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoBitCount,
        //! XML tag name for bit count
#define VVWINFOTAG_woVideoBitCount              "BitCount"
#define VVWINFODESC_woVideoBitCount             "BitCount"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoCompression,
        //! XML tag name for compression (fourcc)
#define VVWINFOTAG_woVideoCompression       "Compression"
#define VVWINFODESC_woVideoCompression      "Compression"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoSizeImage,
        //! XML tag name for size of the image in unsigned chars
#define VVWINFOTAG_woVideoSizeImage             "SizeImage"
#define VVWINFODESC_woVideoSizeImage        "SizeImage"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoXPelsPerMeter,
        //! XML tag name for X pels per meter
#define VVWINFOTAG_woVideoXPelsPerMeter     "XPelsPerMeter"
#define VVWINFODESC_woVideoXPelsPerMeter"XPelsPerMeter"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoYPelsPeMeter,
        //! XML tag name for Y pels per meter
#define VVWINFOTAG_woVideoYPelsPerMeter     "YPelsPerMeter"
#define VVWINFODESC_woVideoYPelsPerMeter"YPelsPerMeter"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
```

```
        vvwiVideoClrUsed,
        //! XML tag name for color elements used
#define VVWINFOTAG_woVideoClrUsed               "ClrUsed"
#define VVWINFODESC_woVideoClrUsed              "ClrUsed"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoClrImportant,
        //! XML tag name for
#define VVWINFOTAG_woVideoClrImportant          "ClrImportant"
#define VVWINFODESC_woVideoClrImportant         "ClrImportant"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoReserved,
        //! XML tag name for reserved array
#define VVWINFOTAG_woVideoReserved              "Reserved"
#define VVWINFODESC_woVideoReserved             "Reserved"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoFccType,
        //! XML tag name for four cc type (video/audio)
#define VVWINFOTAG_woVideoFccType               "FccType"
#define VVWINFODESC_woVideoFccType              "FccType"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoFccHandler,
        //! XML tag name for four cc handler
#define VVWINFOTAG_woVideoFccHandler            "FccHandler"
#define VVWINFODESC_woVideoFccHandler           "FccHandler"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoFlags,
        //! XML tag name for flags
#define VVWINFOTAG_woVideoFlags                 "Flags"
#define VVWINFODESC_woVideoFlags                "Flags"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoCaps,
        //! XML tag name for capabilities
#define VVWINFOTAG_woVideoCaps                  "Caps"
#define VVWINFODESC_woVideoCaps                 "Caps"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoPriority,
        //! XML tag name for priority
#define VVWINFOTAG_woVideoPriority              "Priority"
#define VVWINFODESC_woVideoPriority             "Priority"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoLanguage,
        //! XML tag name for language
#define VVWINFOTAG_woVideoLanguage              "Language"
#define VVWINFODESC_woVideoLanguage             "Language"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoScale,
        //! XML tag name for scale (fps = rate / scale)
#define VVWINFOTAG_woVideoScale                 "Scale"
#define VVWINFODESC_woVideoScale                "Scale"
```

```
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoRate,
        //! XML tag name for rate (fps = rate / scale)
#define VVWINFOTAG_woVideoRate                  "Rate"
#define VVWINFODESC_woVideoRate                 "Rate"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
    vvwiVideoStart,
        //! XML tag name for start frame
#define VVWINFOTAG_woVideoStart                 "Start"
#define VVWINFODESC_woVideoStart            "Start"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoLength,
        //! XML tag name for the length in frames
#define VVWINFOTAG_woVideoLength            "Length"
#define VVWINFODESC_woVideoLength               "Length"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoInitialFrames,
        //! XML tag name for number of initial frames to load
#define VVWINFOTAG_woVideoInitialFrames     "InitialFrames"
#define VVWINFODESC_woVideoInitialFrames  "InitialFrames"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
    vvwiVideoSuggestedBufferSize,
        //! XML tag name for suggested maximum buffer size
#define VVWINFOTAG_woVideoSuggestedBufferSize   "SuggestedBufferSize"
#define VVWINFODESC_woVideoSuggestedBufferSize  "SuggestedBufferSize"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoQuality,
        //! XML tag name for quality
#define VVWINFOTAG_woVideoQuality           "Quality"
#define VVWINFODESC_woVideoQuality              "Quality"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoSampleSize,
        //! XML tag name for recommended sample size
#define VVWINFOTAG_woVideoSampleSize        "SampleSize"
#define VVWINFODESC_woVideoSampleSize       "SampleSize"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoEditCount,
        //! XML tag name for number of edits done on this file
#define VVWINFOTAG_woVideoEditCount             "EditCount"
#define VVWINFODESC_woVideoEditCount        "EditCount"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoFormatChangeCount,
        //! XML tag name for number of format changes
#define VVWINFOTAG_woVideoFormatChangeCount     "FormatChangeCount"
#define VVWINFODESC_woVideoFormatChangeCount  "FormatChangeCount"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoPitch,
        //! XML tag name for video line pitch
#define VVWINFOTAG_woVideoPitch                 "Pitch"
```

```
#define VVWINFODESC_woVideoPitch                "Pitch"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoDrFlags,
        //! XML tag name for internal drastic flags
#define VVWINFOTAG_woVideoDrFlags               "DrFlags"
#define VVWINFODESC_woVideoDrFlags                      "DrFlags"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoFileType,
        //! XML tag name for drastic 'mft' file type
#define VVWINFOTAG_woVideoFileType                      "FileType"
#define VVWINFODESC_woVideoFileType                     "FileType"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiVideoResDrastic,
        //! XML tag name for reserved drastic array of DWORDs
#define VVWINFOTAG_woVideoResDrastic            "ResDrastic"
#define VVWINFODESC_woVideoResDrastic           "ResDrastic"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiAudioType,
        //! XML tag
#define VVWINFOTAG_woAudioType                          "AudioType"
#define VVWINFODESC_woAudioType                         "AudioType"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiAudioChannels,
        //! XML tag
#define VVWINFOTAG_woAudioChannels                      "AudioChannels"
#define VVWINFODESC_woAudioChannels                     "AudioChannels"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiAudioFrequency,
        //! XML tag
#define VVWINFOTAG_woAudioFrequency                     "AudioFrequency"
#define VVWINFODESC_woAudioFrequency            "AudioFrequency"
        //! INTERNAL: Auto generated for XML output from #VVWVIDEO/#VVWAUDIO
        vvwiAudioBits,
        //! XML tag
#define VVWINFOTAG_woAudioBits                          "AudioBits"
#define VVWINFODESC_woAudioBits                         "AudioBits"
        //char  szName[_VVWXXX_NAME_SIZE];      // Stream identifier
        //RECT/*16*/  rcFrame;                                   // Frame
dimensions
        vvwiLastElementPlus1
        // DO NOT ADD ANYTHING BELOW vvwiLastElementPlus1
};

// Total number of accessible metadata elements
#define VVWINFO_TOTAL_ITEMS (vvwiEND_OF_STRINGS + (vvwiEND_OF_DWORD_V2
- vvwiNumericStart))

/**
* The info structure is used to hold the metadata for one clip
```

```
* It is a combination of a custom info identifiers from all the file
* types we currently support.  It is always written as a side bar
* xml file, and also insert into/read from the actual file whenever
* possible.  NOTE:  There are other fields included in the xml that
* are not contained here.  There are contained in #VVWSYSTEM, #VVWVIDEO
* or #VVWAUDIO.
* <BR>
*
* Meta Data Rules (needs updating)<BR>
*
* <UL>
* <LI><i><b>--- LOCATOR STRINGS</b></i>
* <LI> File Name (base name+ext)
* <LI> Native Locator
* <LI> Universal Name Convention Locator
* <LI> IP
* <LI> Source Locator (Universal)
* </UL>
* <UL>
* <LI><i><b>--- Inferred</b></i>
* <LI> Channel
* <LI> Channel Length
* <LI> Channel Name
* <LI> Channel Description
* <LI> Channel Poster Frame
* </UL>
* <UL>
* <LI><i><b>--- STRINGS</b></i>
* <LI> Title                    W (AAF:Name) (TIF:DocumentName)
* <LI> Subject                  W
* <LI> Category                       W
* <LI> Keywords                       W
* <LI> Rating                   AVI
* <LI> Comments                       W QT
* <LI> Owner                    AVI
* <LI> Editor                   AVI
* <LI> Supplier                       AVI
* <LI> Source                   W
* <LI> Project                  AVI
* <LI> Status<BR>               AVI
*       Other, Final, Proof, Review, Edit, In Progress, Draft, Preliminary, New Normal
* <LI> Author                   W QT
* <LI> Revision Number          W
* <LI> Produced                       AVI
* <LI> Album                     QT
* <LI> Artist                    QT TIF
* <LI> Composer                        QT
* <LI> Copyright                 QT TIF
* <LI> Creation Data    QT
```

```
* <LI> Description                QT AAF (TIF:ImageDescription)
* <LI> Director                   QT
* <LI> Disclaimer                 QT
* <LI> Encoded By                 QT
* <LI> Full Name                  QT
* <LI> Genre                      QT
* <LI> Host Computer   QT TIF
* <LI> Information                QT
* <LI> Make                               QT TIF
* <LI> Model                      QT TIF
* <LI> Original Artist    QT
* <LI> Original Format  QT
* <LI> Original Source  QT
* <LI> Performers                 QT
* <LI> Producer                           QT
* <LI> Product                    QT
* <LI> Software                           QT TIF
* <LI> Special Playback Requirements QT
* <LI> Track                      QT
* <LI> Warning                            QT
* <LI> URL Link                           QT
* <LI> Edit Data 1..9    QT
* <LI> Version String     AAF
* <LI> Manufacturer               AAF
* <LI> Language                           DublinCore.org
* <LI> Format                     DublinCore.org
* <LI>
* </UL>
* <UL>
* <LI><i><b>Numeric Entries</b></i>
* <LI> Timecode (LTC)
* <LI> UserBits (LTC)
* <LI> VITC Timecode
* <LI> VITC Userbits
* <LI> VITC Line 3
* <LI> Poster Frame
* <LI> 'A' Frame
* <LI> Aspect Ratio
* <LI> OrginalScale
* <LI> OrginalRate
* <LI> Conversions
* <LI> Version Number  AAF
* <LI> File Size (64 Bits)
* <LI> File Date                  TIF
* <LI> File Time                  TIF
* <LI> Sequence Number       AVI
* </UL>
*/
typedef struct /*tagVVWINFO*/ {
```

```
        /**
        * File Name and Extension (no path info)
        * Created
        * <BR> XML tag \<FileName\>
        */
        char * szFileName;
        //! XML tag name for VVWINFO::szFileName
#define VVWINFOTAG_szFileName       "FileName"              // # 1
#define VVWINFODESC_szFileName      "FileName"              // # 1

        /**
        * Native path + file + ext (C:/Test/Junk.avi)
        * Created
        * <BR> XML tag \<NativeLocator\>
        */
        char * szNativeLocator;
        //! XML tag name for VVWINFO::szNativeLocator
#define VVWINFOTAG_szNativeLocator "NativeLocator"
#define VVWINFODESC_szNativeLocator         "NativeLocator"
        /**
        * UNC (/TestBox/C/Test/Junk.avi)
        * Created
        * <BR> XML tag \<UniversalFileName\>
        */
        char * szUniversalName;
        //! XML tag name for VVWINFO::szUniversalLocator
#define VVWINFOTAG_szUniversalName          "UniversalLocator"
#define VVWINFODESC_szUniversalName         "UniversalLocator"
        /**
        * IP Address (192.168.0.3)
        * Created
        * <BR> XML tag \<TCP-IPAddress\>
        */
        char * szIP;
        //! XML tag name for VVWINFO::szIP
#define VVWINFOTAG_szIP     "TCP-IPAddress"
#define VVWINFODESC_szIP    "TCP-IPAddress"
        /**
        * Source locator (either szUnversalName, szNativeLocator, szFileName available
in this order)
        * Created
        * <BR> XML tag \<SourceLocator\>
        */
        char * szSourceLocator;
        //! XML tag name for VVWINFO::szSourceLocator
#define VVWINFOTAG_szSourceLocator          "SourceLocator"
#define VVWINFODESC_szSourceLocator         "SourceLocator"

        /**
```

```
        * Channel identifier
        * <BR> XML tag \<ChannelIdentifier\>
        */
        char * szChannel;
        //! XML tag name for VVWINFO::szChannelIdentifier
#define VVWINFOTAG_szChannel        "ChannelIdentifier"
#define VVWINFODESC_szChannel       "ChannelIdentifier"
        /**
        * Name of channel
        * <BR> XML tag \<ChannelName\>
        */
        char * szChannelName;
        //! XML tag name for VVWINFO::szChannelName
#define VVWINFOTAG_szChannelName "ChannelName"
#define VVWINFODESC_szChannelName         "ChannelName"
        /**
        * Description of channel
        * <BR> XML tag \<ChannelDescription\>
        */
        char * szChannelDescription;
        //! XML tag name for VVWINFO::szChannelDescription
#define VVWINFOTAG_szChannelDescription    "ChannelDescription"
#define VVWINFODESC_szChannelDescription   "ChannelDescription"
        /**
        * Title
        * <BR> XML tag \<Title\>
        */
        char * szTitle;
        //! XML tag name for VVWINFO::szTitle
#define VVWINFOTAG_szTitle  "Title"
#define VVWINFODESC_szTitle "Title"
        /**
        * Subject
        * <BR> XML tag \<Subject\>
        */
        char * szSubject;
        //! XML tag name for VVWINFO::szSubject
#define VVWINFOTAG_szSubject        "Subject"        // # 10
#define VVWINFODESC_szSubject       "Subject"        // # 10
        /**
        * Category of media (AVI/Mov Format)
        * <BR> XML tag \<Category\>
        */
        char * szCategory;
        //! XML tag name for VVWINFO::szCategory
#define VVWINFOTAG_szCategory       "Category"
#define VVWINFODESC_szCategory      "Category"
        /**
        * Search keywords
```

```
        * <BR> XML tag \<Keywords\>
        */
        char * szKeywords;
        //! XML tag name for VVWINFO::szKeywords
#define VVWINFOTAG_szKeywords      "Keywords"
#define VVWINFODESC_szKeywords     "Keywords"
        /**
        * Parental Ratings
        * <BR> XML tag \<Ratings\>
        */
        char * szRatings;
        //! XML tag name for VVWINFO::szRatings
#define VVWINFOTAG_szRatings       "Ratings"
#define VVWINFODESC_szRatings      "Ratings"
        /**
        * Free form comments
        * <BR> XML tag \<Comments\>
        */
        char * szComments;
        //! XML tag name for VVWINFO::szComments
#define VVWINFOTAG_szComments      "Comments"
#define VVWINFODESC_szComments     "Comments"
        /**
        * ??? - This was (sz)SequenceNumber which was in conflict
        * with (dw)SequenceNumber below.  Leave empty for compatibility
        * <BR> XML tag \<DoNotUse\>
        */
        char * szDoNotUse;      //SequenceNumber;
        //! XML tag name for VVWINFO::szDoNotUse
#define VVWINFOTAG_szDoNotUse      "DoNotUse"
#define VVWINFODESC_szDoNotUse     "DoNotUse"
        /**
        * REG Default
        * Owner of content
        * <BR> XML tag \<Owner\>
        */
        char * szOwner;
        //! XML tag name for VVWINFO::szOwner
#define VVWINFOTAG_szOwner         "Owner"
#define VVWINFODESC_szOwner        "Owner"
        /**
        * REG Default
        * Editor of file
        * <BR> XML tag \<Editor\>
        */
        char * szEditor;
        //! XML tag name for VVWINFO::szEditor
#define VVWINFOTAG_szEditor"Editor"
#define VVWINFODESC_szEditor       "Editor"
```

```
        /**
        * REG Default
        * Supplier
        * <BR> XML tag \<Supplier\>
        */
        char * szSupplier;
        //! XML tag name for VVWINFO::szSupplier
#define VVWINFOTAG_szSupplier       "Supplier"
#define VVWINFODESC_szSupplier      "Supplier"
        /**
        * Source of file (VTR, Betacam, MPEG, Satellite)
        * <BR> XML tag \<Source\>
        */
        char * szSource;
        //! XML tag name for VVWINFO::szSource
#define VVWINFOTAG_szSource         "Source"
#define VVWINFODESC_szSource        "Source"
        /**
        * REG Default
        * Project Name
        * <BR> XML tag \<Project\>
        */
        char * szProject;
        //! XML tag name for VVWINFO::szProject
#define VVWINFOTAG_szProject        "Project"       // # 20
#define VVWINFODESC_szProject       "Project"       // # 20
        /**
        * Status
        * <BR> XML tag \<Status\>
        */
        char * szStatus;
        //! XML tag name for VVWINFO::szStatus
#define VVWINFOTAG_szStatus         "Status"
#define VVWINFODESC_szStatus        "Status"
        /**
        * REG Default
        * Author's name
        * <BR> XML tag \<Author\>
        */
        char * szAuthor;
        //! XML tag name for VVWINFO::szAuthor
#define VVWINFOTAG_szAuthor         "Author"
#define VVWINFODESC_szAuthor        "Author"
        /**
        * Revision number as string
        * <BR> XML tag \<RevisionNumber\>
        */
        char * szRevisionNumber;
        //! XML tag name for VVWINFO::szRevisionNumber
```

```
#define VVWINFOTAG_szRevisionNumber        "RevisionNumber"
#define VVWINFODESC_szRevisionNumber       "RevisionNumber"
        /**
        * Producer company name
        * <BR> XML tag \<Produced\>
        */
        char * szProduced;
        //! XML tag name for VVWINFO::szProduced
#define VVWINFOTAG_szProduced       "Produced"
#define VVWINFODESC_szProduced      "Produced"
        /**
        * Album Name
        * <BR> XML tag \<Album\>
        */
        char * szAlbum;
        //! XML tag name for VVWINFO::szAlbum
#define VVWINFOTAG_szAlbum          "Album"
#define VVWINFODESC_szAlbum         "Album"
        /**
        * Artist Name
        * <BR> XML tag \<Artist\>
        */
        char * szArtist;
        //! XML tag name for VVWINFO::szArtist
#define VVWINFOTAG_szArtist "Artist"
#define VVWINFODESC_szArtist          "Artist"
        /**
        * Composer's Name
        * <BR> XML tag \<Composer\>
        */
        char * szComposer;
        //! XML tag name for VVWINFO::szComposer
#define VVWINFOTAG_szComposer       "Composer"
#define VVWINFODESC_szComposer      "Composer"
        /**
        * REG Default
        * Copyright message
        * <BR> XML tag \<Copyright\>
        */
        char * szCopyright;
        //! XML tag name for VVWINFO::szCopyright
#define VVWINFOTAG_szCopyright      "Copyright"
#define VVWINFODESC_szCopyright     "Copyright"
        /**
        * Creation Data
        * <BR> XML tag \<CreationData\>
        */
        char * szCreationData;
        //! XML tag name for VVWINFO::szCreationData
```

```
#define VVWINFOTAG_szCreationData  "CreationData"
#define VVWINFODESC_szCreationData          "CreationData"
        /**
        * Description of contents
        * <BR> XML tag \<Description\>
        */
        char * szDescription;
        //! XML tag name for VVWINFO::szDescription
#define VVWINFOTAG_szDescription    "Description"    // # 30
#define VVWINFODESC_szDescription   "Description"    // # 30
        /**
        * Director's Name
        * <BR> XML tag \<Director\>
        */
        char * szDirector;
        //! XML tag name for VVWINFO::szDirector
#define VVWINFOTAG_szDirector       "Director"
#define VVWINFODESC_szDirector      "Director"
        /**
        * Disclaimer
        * <BR> XML tag \<Disclaimer\>
        */
        char * szDisclaimer;
        //! XML tag name for VVWINFO::szDisclaimer
#define VVWINFOTAG_szDisclaimer     "Disclaimer"
#define VVWINFODESC_szDisclaimer    "Disclaimer"
        /**
        * REG Default
        * Encoders Name
        * <BR> XML tag \<Encoded By\>
        */
        char * szEncodedBy;
        //! XML tag name for VVWINFO::szEncodedBy
#define VVWINFOTAG_szEncodedBy    "EncodedBy"
#define VVWINFODESC_szEncodedBy   "EncodedBy"
        /**
        * Full name of media
        * <BR> XML tag \<FullName\>
        */
        char * szFullName;
        //! XML tag name for VVWINFO::szFullName
#define VVWINFOTAG_szFullName       "FullName"
#define VVWINFODESC_szFullName      "FullName"
        /**
        * Genre (type) of content
        * <BR> XML tag \<Genre\>
        */
        char * szGenre;
        //! XML tag name for VVWINFO::szGenre
```

```
#define VVWINFOTAG_szGenre"Genre"
#define VVWINFODESC_szGenre         "Genre"
        /**
        * Computer hosting content
        * <BR> XML tag \<HostComputer\>
        */
        char * szHostComputer;
        //! XML tag name for VVWINFO::szHostComputer
#define VVWINFOTAG_szHostComputer          "HostComputer"
#define VVWINFODESC_szHostComputer         "HostComputer"
        /**
        * Free form information
        * <BR> XML tag \<Information\>
        */
        char * szInformation;
        //! XML tag name for VVWINFO::szInformation
#define VVWINFOTAG_szInformation    "Information"
#define VVWINFODESC_szInformation  "Information"
        /**
        * Make - not used
        * <BR> XML tag \<Make\>
        */
        char * szMake;
        //! XML tag name for VVWINFO::szMake
#define VVWINFOTAG_szMake "Make"
#define VVWINFODESC_szMake          "Make"
        /**
        * Not Used
        * <BR> XML tag \<Model\>
        */
        char * szModel;
        //! XML tag name for VVWINFO::szModel
#define VVWINFOTAG_szModel"Model"
#define VVWINFODESC_szModel         "Model"
        /**
        * Orginal artist's name
        * <BR> XML tag \<OriginalArtist\>
        */
        char * szOriginalArtist;
        //! XML tag name for VVWINFO::szOriginalArtist
#define VVWINFOTAG_szOriginalArtist  "OriginalArtist"  // # 40
#define VVWINFODESC_szOriginalArtist"OriginalArtist"  // # 40
        /**
        * Original format of content
        * <BR> XML tag \<OriginalFormat\>
        */
        char * szOriginalFormat;
        //! XML tag name for VVWINFO::szOriginalFormat
#define VVWINFOTAG_szOriginalFormat         "OriginalFormat"
```

```
#define VVWINFODESC_szOriginalFormat        "OriginalFormat"
        /**
        * Performers in content
        * <BR> XML tag \<Performers\>
        */
        char * szPerformers;
        //! XML tag name for VVWINFO::szPerformers
#define VVWINFOTAG_szPerformers     "Performers"
#define VVWINFODESC_szPerformers    "Performers"
        /**
        * Producer's Name
        * <BR> XML tag \<Producer\>
        */
        char * szProducer;
        //! XML tag name for VVWINFO::szProducer
#define VVWINFOTAG_szProducer       "Producer"
#define VVWINFODESC_szProducer      "Producer"
        /**
        * Targeted associated product
        * <BR> XML tag \<Product\>
        */
        char * szProduct;
        //! XML tag name for VVWINFO::szProduct
#define VVWINFOTAG_szProduct        "Product"
#define VVWINFODESC_szProduct       "Product"
        /**
        * Software in use (DEF:  "MediaReactor")
        * <BR> XML tag \<Software\>
        */
        char * szSoftware;
        //! XML tag name for VVWINFO::szSoftware
#define VVWINFOTAG_szSoftware       "Software"
#define VVWINFODESC_szSoftware      "Software"
        /**
        * Special playback requirements
        * <BR> XML tag \<SpecialPlaybackRequirements\>
        */
        char * szSpecialPlaybackRequirements;
        //! XML tag name for VVWINFO::szSpecialPlaybackRequirements
#define VVWINFOTAG_szSpecialPlaybackRequirements "SpecialPlaybackRequirements"
#define VVWINFODESC_szSpecialPlaybackRequirements
        "SpecialPlaybackRequirements"
        /**
        * Track Name
        * <BR> XML tag \<Track\>
        */
        char * szTrack;
        //! XML tag name for VVWINFO::szTrack
#define VVWINFOTAG_szTrack "Track"
```

```
#define VVWINFODESC_szTrack          "Track"
        /**
        * Warning info
        * <BR> XML tag \<Warning\>
        */
        char * szWarning;
        //! XML tag name for VVWINFO::szWarning
#define VVWINFOTAG_szWarning         "Warning"
#define VVWINFODESC_szWarning        "Warning"
        /**
        * REG Default
        * Link to site (http://www.drastictech.com)
        * <BR> XML tag \<URL\>
        */
        char * szURLLink;
        //! XML tag name for VVWINFO::szURLLink
#define VVWINFOTAG_szURLLink         "URL"
#define VVWINFODESC_szURLLink        "URL"
        /**
        * User Data
        * <BR> XML tag \<EditData1\>
        */
        char * szEditData1;
        //! XML tag name for VVWINFO::szEditData1
#define VVWINFOTAG_szEditData1       "EditData1"              // # 50
#define VVWINFODESC_szEditData1      "EditData1"
        /**
        * User Data
        * <BR> XML tag \<EditData2\>
        */
        char * szEditData2;
        //! XML tag name for VVWINFO::szEditData2
#define VVWINFOTAG_szEditData2       "EditData2"
#define VVWINFODESC_szEditData2      "EditData2"
        /**
        * User Data
        * <BR> XML tag \<EditData3\>
        */
        char * szEditData3;
        //! XML tag name for VVWINFO::szEditData3
#define VVWINFOTAG_szEditData3       "EditData3"
#define VVWINFODESC_szEditData3      "EditData3"
        /**
        * User Data
        * <BR> XML tag \<EditData4\>
        */
        char * szEditData4;
        //! XML tag name for VVWINFO::szEditData4
#define VVWINFOTAG_szEditData4       "EditData4"
```

```
#define VVWINFODESC_szEditData4     "EditData4"
        /**
        * User Data
        * <BR> XML tag \<EditData5\>
        */
        char * szEditData5;
        //! XML tag name for VVWINFO::szEditData5
#define VVWINFOTAG_szEditData5      "EditData5"
#define VVWINFODESC_szEditData5     "EditData5"
        /**
        * User Data
        * <BR> XML tag \<EditData6\>
        */
        char * szEditData6;
        //! XML tag name for VVWINFO::szEditData6
#define VVWINFOTAG_szEditData6      "EditData6"
#define VVWINFODESC_szEditData6     "EditData6"
        /**
        * User Data
        * <BR> XML tag \<EditData7\>
        */
        char * szEditData7;
        //! XML tag name for VVWINFO::szEditData7
#define VVWINFOTAG_szEditData7      "EditData7"
#define VVWINFODESC_szEditData7     "EditData7"
        /**
        * User Data
        * <BR> XML tag \<EditData8\>
        */
        char * szEditData8;
        //! XML tag name for VVWINFO::szEditData8
#define VVWINFOTAG_szEditData8      "EditData8"
#define VVWINFODESC_szEditData8     "EditData8"
        /**
        * User Data
        * <BR> XML tag \<EditData9\>
        */
        char * szEditData9;
        //! XML tag name for VVWINFO::szEditData9
#define VVWINFOTAG_szEditData9      "EditData9"
#define VVWINFODESC_szEditData9     "EditData9"
        /**
        * Version string
        * <BR> XML tag \<VersionString\>
        */
        char * szVersionString;
        //! XML tag name for VVWINFO::szVersionString
#define VVWINFOTAG_szVersionString  "VersionString"
#define VVWINFODESC_szVersionString         "VersionString"
```

```
        /**
        * Manufacturer (DEF: Drastic Technologies Ltd)
        * <BR> XML tag \<Manufacturer\>
        */
        char * szManufacturer;
        //! XML tag name for VVWINFO::szManufacturer
#define VVWINFOTAG_szManufacturer "Manufacturer" // # 60
#define VVWINFODESC_szManufacturer          "Manufacturer" // # 60
        /**
        * Language
        * <BR> XML tag \<Language\>
        */
        char * szLanguage;
        //! XML tag name for VVWINFO::szLanguage
#define VVWINFOTAG_szLanguage      "Language"
#define VVWINFODESC_szLanguage     "Language"
        /**
        * File format
        * <BR> XML tag \<Format\>
        */
        char * szFormat;
        //! XML tag name for VVWINFO::szFormat
#define VVWINFOTAG_szFormat        "Format"
#define VVWINFODESC_szFormat       "Format"
        /**
        * Input device (Cin/DPix) telecine name (64 char)
        * <BR> XML tag \<InputDevice\>
        */
        char * szInputDevice;
        //! XML tag name for VVWINFO::szInputDevice
#define VVWINFOTAG_szInputDevice   "InputDevice"
#define VVWINFODESC_szInputDevice  "InputDevice"
        /**
        * Input device Model (32 char)
        * <BR> XML tag \<DeviceModelNum\>
        */
        char * szDeviceModelNum;
        //! XML tag name for VVWINFO::szDeviceModelNum
#define VVWINFOTAG_szDeviceModelNum                "DeviceModelNum"
#define VVWINFODESC_szDeviceModelNum               "DeviceModelNum"
        /**
        * Input device serial number (32 char)
        * <BR> XML tag \<DeviceSerialNum\>
        */
        char * szDeviceSerialNum;
        //! XML tag name for VVWINFO::szDeviceSerialNum
#define VVWINFOTAG_szDeviceSerialNum       "DeviceSerialNum"
#define VVWINFODESC_szDeviceSerialNum      "DeviceSerialNum"
        /**
```

```
        * Reel this was recorded to
        * <BR> XML tag \<Reel\>
        */
        char * szReel;
        //! XML tag name for VVWINFO::szReel
#define VVWINFOTAG_szReel                       "Reel"
#define VVWINFODESC_szReel                      "Reel"
        /**
        * Shot name (or number in string)
        * <BR> XML tag \<Shot\>
        */
        char * szShot;
        //! XML tag name for VVWINFO::szShot
#define VVWINFOTAG_szShot                       "Shot"
#define VVWINFODESC_szShot                      "Shot"
        /**
        * Take name (or number in string)
        * <BR> XML tag \<Take\>
        */
        char * szTake;
        //! XML tag name for VVWINFO::szTake
#define VVWINFOTAG_szTake                       "Take"
#define VVWINFODESC_szTake                      "Take"
        /**
        * Info from the original slate (200 char)
        * <BR> XML tag \<SlateInfo\>
        */
        char * szSlateInfo;
        //! XML tag name for VVWINFO::szSlateInfo
#define VVWINFOTAG_szSlateInfo                  "SlateInfo"
#define VVWINFODESC_szSlateInfo                 "SlateInfo"
        /**
        * Default frame attribute string for Cin/DPix (32 char)
        * <BR> XML tag \<FrameAttribute\>
        */
        char * szFrameAttribute;
        //! XML tag name for VVWINFO::szFrameAttibute
#define VVWINFOTAG_szFrameAttribute             "FrameAttribute"        // # 70
#define VVWINFODESC_szFrameAttribute            "FrameAttribute"        // # 70
        /**
        * RP-215 epsiode name
        */
        char * szEpisode;
        //! XML tag name for VVWINFO::szEpisode
#define VVWINFOTAG_szEpisode                    "Episode"
#define VVWINFODESC_szEpisode                   "Episode"
        /**
        * RP-215 scene
        */
```

```
        char * szScene;
        //! XML tag name for VVWINFO::szScene
#define VVWINFOTAG_szScene                      "Scene"
#define VVWINFODESC_szScene                             "Scene"
        /**
        * RP-215 daily roll namd/id
        */
        char * szDailyRoll;
        //! XML tag name for VVWINFO::szDailyRoll
#define VVWINFOTAG_szDailyRoll                  "DailyRoll"
#define VVWINFODESC_szDailyRoll                 "DailyRoll"
        /**
        * RP-215 camera roll namd/id
        */
        char * szCamRoll;
        //! XML tag name for VVWINFO::szCamRoll
#define VVWINFOTAG_szCamRoll                    "CamRoll"
#define VVWINFODESC_szCamRoll                   "CamRoll"
        /**
        * RP-215 sound roll namd/id
        */
        char * szSoundRoll;
        //! XML tag name for VVWINFO::szSoundRoll
#define VVWINFOTAG_szSoundRoll                  "SoundRoll"
#define VVWINFODESC_szSoundRoll                 "SoundRoll"
        /**
        * RP-215 lab roll namd/id
        */
        char * szLabRoll;
        //! XML tag name for VVWINFO::szLabRoll
#define VVWINFOTAG_szLabRoll                    "LabRoll"
#define VVWINFODESC_szLabRoll                   "LabRoll"
        /**
        * Prefix (0 feet) of the key number from the film
        */
        char * szKeyNumberPrefix;
        //! XML tag name for VVWINFO::szKeyNumberPrefix
#define VVWINFOTAG_szKeyNumberPrefix    "KeyNumberPrefix"
#define VVWINFODESC_szKeyNumberPrefix   "KeyNumberPrefix"
        /**
        * Prefix (0 feet) of the ink number from the film
        */
        char * szInkNumberPrefix;
        //! XML tag name for VVWINFO::szInkNumberPrefix
#define VVWINFOTAG_szInkNumberPrefix    "InkNumberPrefix"
#define VVWINFODESC_szInkNumberPrefix   "InkNumberPrefix"
        /**
        * Small jpg frame capture
        */
```

```
        char * szPictureIcon;
        //! XML tag name for VVWINFO::szPictureIcon
#define VVWINFOTAG_szPictureIcon            "PictureIcon"
#define VVWINFODESC_szPictureIcon           "PictureIcon"
        /**
        * Low res proxy file
        */
        char * szProxyFile;
        //! XML tag name for VVWINFO::szProxyFile
#define VVWINFOTAG_szProxyFile                      "ProxyFile"     // # 80
#define VVWINFODESC_szProxyFile                     "ProxyFile"     // # 80
        /**
        * TBD
        */
        char * szNU_81;
        //! XML tag name for VVWINFO::szNU_81
#define VVWINFOTAG_szNU_81                              "NotUsed_81"
//#define VVWINFODESC_                                  ""
        /**
        * TBD
        */
        char * szNU_82;
        //! XML tag name for VVWINFO::szNU_82
#define VVWINFOTAG_szNU_82                              "NotUsed_82"
//#define VVWINFODESC_                                  ""
        /**
        * TBD
        */
        char * szNU_83;
        //! XML tag name for VVWINFO::szNU_83
#define VVWINFOTAG_szNU_83                              "NotUsed_83"
//#define VVWINFODESC_                                  ""
        /**
        * TBD
        */
        char * szNU_84;
        //! XML tag name for VVWINFO::szNU_84
#define VVWINFOTAG_szNU_84                              "NotUsed_84"
//#define VVWINFODESC_                                  ""
        /**
        * TBD
        */
        char * szNU_85;
        //! XML tag name for VVWINFO::szNU_85
#define VVWINFOTAG_szNU_85                              "NotUsed_85"
//#define VVWINFODESC_                                  ""
        /**
        * TBD
        */
```

```
        char * szNU_86;
        //! XML tag name for VVWINFO::szNU_86
#define VVWINFOTAG_szNU_86                      "NotUsed_86"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_87;
        //! XML tag name for VVWINFO::szNU_87
#define VVWINFOTAG_szNU_87                      "NotUsed_87"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_88;
        //! XML tag name for VVWINFO::szNU_88
#define VVWINFOTAG_szNU_88                      "NotUsed_88"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_89;
        //! XML tag name for VVWINFO::szNU_89
#define VVWINFOTAG_szNU_89                      "NotUsed_89"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_90;
        //! XML tag name for VVWINFO::szNU_90
#define VVWINFOTAG_szNU_90                      "NotUsed_90"   // # 90
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_91;
        //! XML tag name for VVWINFO::szNU_91
#define VVWINFOTAG_szNU_91                      "NotUsed_91"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_92;
        //! XML tag name for VVWINFO::szNU_92
#define VVWINFOTAG_szNU_92                      "NotUsed_92"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
```

```
        char * szNU_93;
        //! XML tag name for VVWINFO::szNU_93
#define VVWINFOTAG_szNU_93                      "NotUsed_93"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_94;
        //! XML tag name for VVWINFO::szNU_94
#define VVWINFOTAG_szNU_94                      "NotUsed_94"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_95;
        //! XML tag name for VVWINFO::szNU_95
#define VVWINFOTAG_szNU_95                      "NotUsed_95"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_96;
        //! XML tag name for VVWINFO::szNU_96
#define VVWINFOTAG_szNU_96                      "NotUsed_96"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_97;
        //! XML tag name for VVWINFO::szNU_97
#define VVWINFOTAG_szNU_97                      "NotUsed_97"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_98;
        //! XML tag name for VVWINFO::szNU_98
#define VVWINFOTAG_szNU_98                      "NotUsed_98"
//#define VVWINFODESC_                          ""
        /**
        * TBD
        */
        char * szNU_99;
        //! XML tag name for VVWINFO::szNU_99
#define VVWINFOTAG_szNU_99                      "NotUsed_99"   // # 99
//#define VVWINFODESC_                          ""

        /**
        * Special reserved place holder for end of strings
```

```
        */
        DWORD           dwReservedMustBe0;

        /**
        * Starting time code (LTC)
        * <BR> XML tag \<TimeCode\>
        */
        DWORD dwTimeCode;
        //! XML tag name for VVWINFO::dwTimeCode
#define VVWINFOTAG_dwTimeCode     "TimeCode"
#define VVWINFODESC_dwTimeCode    "TimeCode"
        /**
        * Starting user bits (LTC)
        * <BR> XML tag \<UserBits\>
        */
        DWORD dwUserBits;
        //! XML tag name for VVWINFO::dwUserBits
#define VVWINFOTAG_dwUserBits       "UserBits"
#define VVWINFODESC_dwUserBits      "UserBits"
        /**
        * Starting VITC time code
        * <BR> XML tag \<VITCTimeCode\>
        */
        DWORD dwVITCTimeCode;
        //! XML tag name for VVWINFO::dwVITCTimeCode
#define VVWINFOTAG_dwVITCTimeCode         "VITCTimeCode"
#define VVWINFODESC_dwVITCTimeCode        "VITCTimeCode"
        /**
        * Starting VITC user bits
        * <BR> XML tag \<VITCUserBits\>
        */
        DWORD dwVITCUserBits;
        //! XML tag name for VVWINFO::dwVITCUserBits
#define VVWINFOTAG_dwVITCUserBits"VITCUserBits"
#define VVWINFODESC_dwVITCUserBits          "VITCUserBits"
        /**
        * Extra VITC data
        * <BR> XML tag \<VITCExtraData\>
        */
        DWORD dwVITCLine3;
        //! XML tag name for VVWINFO::dwVICTLine3
#define VVWINFOTAG_dwVITCLine3    "VITCExtraData"
#define VVWINFODESC_dwVITCLine3   "VITCExtraData"
        /**
        * Poster (picon) frame number
        * <BR> XML tag \<PosterFrame\>
        */
        DWORD dwPosterFrame;
        //! XML tag name for VVWINFO::dwPosterFrame
```

```
#define VVWINFOTAG_dwPosterFrame "PosterFrame"
#define VVWINFODESC_dwPosterFrame          "PosterFrame"
        /**
        * A-Frame if 2/3 else 0
        * <BR> XML tag \<A-Frame\>
        */
        DWORD dwAFrame;
        //! XML tag name for VVWINFO::dwAFrame
#define VVWINFOTAG_dwAFrame        "A-Frame"
#define VVWINFODESC_dwAFrame       "A-Frame"
        /**
        * Upperword/LowerWord (e.g. 0x00040003 = 4/3)
        * <BR> XML tag \<AspectRatio\>
        */
        DWORD dwAspectRatio;
        //! XML tag name for VVWINFO::dwAspectRatio
#define VVWINFOTAG_dwAspectRatio  "AspectRatio"
#define VVWINFODESC_dwAspectRatio "AspectRatio"
        /**
        * Original file rate
        * <BR> XML tag \<OriginalFileRate\>
        */
        DWORD dwOriginalRate;
        //! XML tag name for VVWINFO::dwOriginalRate
#define VVWINFOTAG_dwOriginalRate  "OriginalRate"
#define VVWINFODESC_dwOriginalRate          "OriginalRate"
        /**
        * Original file scale
        * <BR> XML tag \<OriginalFileScale\>
        */
        DWORD dwOriginalScale;
        //! XML tag name for VVWINFO::dwOriginalScale
#define VVWINFOTAG_dwOriginalScale "OriginalScale"
#define VVWINFODESC_dwOriginalScale         "OriginalScale"
        /**
        * Number of conversion
        * <BR> XML tag \<TotalConversions\>
        */
        DWORD dwConversions;
        //! XML tag name for VVWINFO::dwTotalConversions
#define VVWINFOTAG_dwConversions  "TotalConversions"
#define VVWINFODESC_dwConversions"TotalConversions"
        /**
        * Version number
        * <BR> XML tag \<VersionNumber\>
        */
        DWORD dwVersionNumber;
        //! XML tag name for VVWINFO::dwVersionNumber
#define VVWINFOTAG_dwVersionNumber        "VersionNumber"
```

```
#define VVWINFODESC_dwVersionNumber       "VersionNumber"
        /**
        * Size of file
        * <BR> XML tag \<FileSize\>
        */
        DWORD dwFileSize;
        //! XML tag name for VVWINFO::dwFileSize
#define VVWINFOTAG_dwFileSize        "FileSize"
#define VVWINFODESC_dwFileSize       "FileSize"
        /**
        * Date stamp of file
        * <BR> XML tag \<FileDate\>
        */
        DWORD dwFileDate;
        //! XML tag name for VVWINFO::dwFileDate
#define VVWINFOTAG_dwFileDate        "FileDate"
#define VVWINFODESC_dwFileDate       "FileDate"
        /**
        * Time of file
        * <BR> XML tag \<FileTime\>
        */
        DWORD dwFileTime;
        //! XML tag name for VVWINFO::dwFileTime
#define VVWINFOTAG_dwFileTime        "FileTime"
#define VVWINFODESC_dwFileTime       "FileTime"
        /**
        * Sequnce number
        * <BR> XML tag \<SequenceNumber\>
        */
        DWORD dwSequenceNumber;
        //! XML tag name for VVWINFO::dwSequenceNumber
#define VVWINFOTAG_dwSequenceNumber      "SequenceNumber"
#define VVWINFODESC_dwSequenceNumber     "SequenceNumber"
        /**
        * Total number of streams in file
        * <BR> XML tag \<TotalStreams\>
        */
        DWORD dwTotalStreams;
        //! XML tag name for VVWINFO::dwTotalStreams
#define VVWINFOTAG_dwTotalStreams"TotalStreams"
#define VVWINFODESC_dwTotalStreams         "TotalStreams"
        /**
        * Total frames of longest stream in file
        * <BR> XML tag \<TotalLength\>
        */
        DWORD dwTotalLength;
        //! XML tag name for VVWINFO::dwTotalLength
#define VVWINFOTAG_dwTotalLength  "TotalLength"
#define VVWINFODESC_dwTotalLength"TotalLength"
```

```
        /**
        * Film manufacturer's code
        * <BR> XML tag \<FileManufacturerCode\>
        */
        DWORD dwFilmManufacturerCode;
        //! XML tag name for VVWINFO::dwFilmManufacturerCode
#define VVWINFOTAG_dwFilmManufacturerCode       "FilmManufacturerCode"
#define VVWINFODESC_dwFilmManufacturerCode      "FilmManufacturerCode"
        /**
        * Film type code
        * <BR> XML tag \<FileTypeCode\>
        */
        DWORD dwFilmTypeCode;
        //! XML tag name for VVWINFO::dwFilmTypeCode
#define VVWINFOTAG_dwFilmTypeCode       "FilmTypeCode"
#define VVWINFODESC_dwFilmTypeCode      "FilmTypeCode"
        /**
        * Log captured White point (also CCIR white for 8/10 YCbCr)
        * <BR> XML tag \<WhitePoint\>
        */
        DWORD dwWhitePoint;
        //! XML tag name for VVWINFO::dwWhitePoint
#define VVWINFOTAG_dwWhitePoint   "WhitePoint"
#define VVWINFODESC_dwWhitePoint  "WhitePoint"
        /**
        * Log captured Black point (also CCIR black for 8/10 YCbCr)
        * <BR> XML tag \<BlackPoint\>
        */
        DWORD dwBlackPoint;
        //! XML tag name for VVWINFO::dwBlackPoint
#define VVWINFOTAG_dwBlackPoint    "BlackPoint"
#define VVWINFODESC_dwBlackPoint   "BlackPoint"
        /**
        * Log capture Black gain
        * <BR> XML tag \<BlackGain\>
        */
        DWORD dwBlackGain;
        //! XML tag name for VVWINFO::dwBlackGain
#define VVWINFOTAG_dwBlackGain     "BlackGain"
#define VVWINFODESC_dwBlackGain    "BlackGain"
        /**
        * Log captured break point
        * <BR> XML tag \<BreakPoint\>
        */
        DWORD dwBreakPoint;
        //! XML tag name for VVWINFO::dwBreakPoint
#define VVWINFOTAG_dwBreakPoint   "BreakPoint"
#define VVWINFODESC_dwBreakPoint  "BreakPoint"
        /**
```

```
        * Gamma * 1000 (e.g. 1.7 == 17000)
        * <BR> XML tag \<Gamma1000\>
        */
        DWORD dwGamma1000;
        //! XML tag name for VVWINFO::dwGamma1000
#define VVWINFOTAG_dwGamma1000 "Gamma1000"
#define VVWINFODESC_dwGamma1000         "Gamma1000"
        /**
        * TBD
        */
        DWORD dwTagNumber;
        //! XML tag name for VVWINFO::dwTagNumber
#define VVWINFOTAG_dwTagNumber  "TagNumber"
#define VVWINFODESC_dwTagNumber "TagNumber"
        /**
        * TBD
        */
        DWORD dwFlags;
        //! XML tag name for VVWINFO::dwFlags
#define VVWINFOTAG_dwFlags"Flags"
#define VVWINFODESC_dwFlags         "Flags"
        /**
        * Time code type for the counter/ctl
        * NDF/30, DF/29, PAL/24, FILM/24, NTSCFILM/23
        */
        DWORD dwTimeCodeType;
        //! XML tag name for VVWINFO::dwTimeCodeType
#define VVWINFOTAG_dwTimeCodeType                   "TimeCodeType"
#define VVWINFODESC_dwTimeCodeType                  "TimeCodeType"
        /**
        * LTC Time code type for the counter/ctl
        * NDF/30, DF/29, PAL/24, FILM/24, NTSCFILM/23
        */
        DWORD dwLTCTimeCodeType;
        //! XML tag name for VVWINFO::dwLTCTimeCodeType
#define VVWINFOTAG_dwLTCTimeCodeType    "LTCTimeCodeType"
#define VVWINFODESC_dwLTCTimeCodeType  "LTCTimeCodeType"
        /**
        * VITC Time code type for the counter/ctl
        * NDF/30, DF/29, PAL/24, FILM/24, NTSCFILM/23
        */
        DWORD dwVITCTimeCodeType;
        //! XML tag name for VVWINFO::dwVITCTimeCodeType
#define VVWINFOTAG_dwVITCTimeCodeType  "VITCTimeCodeType"
#define VVWINFODESC_dwVITCTimeCodeType"VITCTimeCodeType"
        /**
        * RP-215 ProdDate 4 bytes
        */
        DWORD dwProdDate;
```

```
        //! XML tag name for VVWINFO::dwProdDate
#define VVWINFOTAG_dwProdDate     "ProdDate"
#define VVWINFODESC_dwProdDate    "ProdDate"
        //End: v3.0
        /**
        * TBD
        */
        DWORD dwUniqueID;
        //! XML tag name for VVWINFO::dwUniqueID
#define VVWINFOTAG_dwUniqueID     "UniqueID"
#define VVWINFODESC_dwUniqueID    "UniqueID"
        /**
        * TBD
        */
        DWORD dwNU_33;
        //! XML tag name for VVWINFO::dwNU_33
#define VVWINFOTAG_dwNU_33        "NumericNotUsed_33"
//#define VVWINFODESC_            ""
        /**
        * TBD
        */
        DWORD dwNU_34;
        //! XML tag name for VVWINFO::dwNU_34
#define VVWINFOTAG_dwNU_34        "NumericNotUsed_34"
//#define VVWINFODESC_            ""
        /**
        * TBD
        */
        DWORD dwNU_35;
        //! XML tag name for VVWINFO::dwNU_35
#define VVWINFOTAG_dwNU_35        "NumericNotUsed_35"
//#define VVWINFODESC_            ""
        /**
        * TBD
        */
        DWORD dwNU_36;
        //! XML tag name for VVWINFO::dwNU_36
#define VVWINFOTAG_dwNU_36        "NumericNotUsed_36"
//#define VVWINFODESC_            ""
        /**
        * TBD
        */
        DWORD dwNU_37;
        //! XML tag name for VVWINFO::dwNU_37
#define VVWINFOTAG_dwNU_37        "NumericNotUsed_37"
//#define VVWINFODESC_            ""
        /**
        * TBD
        */
```

```
        DWORD dwNU_38;
        //! XML tag name for VVWINFO::dwNU_38
#define VVWINFOTAG_dwNU_38      "NumericNotUsed_38"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_39;
        //! XML tag name for VVWINFO::dwNU_39
#define VVWINFOTAG_dwNU_39      "NumericNotUsed_39"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_40;
        //! XML tag name for VVWINFO::dwNU_40
#define VVWINFOTAG_dwNU_40      "NumericNotUsed_40"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_41;
        //! XML tag name for VVWINFO::dwNU_41
#define VVWINFOTAG_dwNU_41      "NumericNotUsed_41"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_42;
        //! XML tag name for VVWINFO::dwNU_42
#define VVWINFOTAG_dwNU_42      "NumericNotUsed_42"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_43;
        //! XML tag name for VVWINFO::dwNU_43
#define VVWINFOTAG_dwNU_43      "NumericNotUsed_43"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_44;
        //! XML tag name for VVWINFO::dwNU_44
#define VVWINFOTAG_dwNU_44      "NumericNotUsed_44"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
```

```
        DWORD dwNU_45;
        //! XML tag name for VVWINFO::dwNU_45
#define VVWINFOTAG_dwNU_45      "NumericNotUsed_45"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_46;
        //! XML tag name for VVWINFO::dwNU_46
#define VVWINFOTAG_dwNU_46      "NumericNotUsed_46"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_47;
        //! XML tag name for VVWINFO::dwNU_47
#define VVWINFOTAG_dwNU_47      "NumericNotUsed_47"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_48;
        //! XML tag name for VVWINFO::dwNU_48
#define VVWINFOTAG_dwNU_48      "NumericNotUsed_48"
//#define VVWINFODESC_          ""
        /**
        * TBD
        */
        DWORD dwNU_49;
        //! XML tag name for VVWINFO::dwNU_49
#define VVWINFOTAG_dwNU_49      "NumericNotUsed_49"
//#define VVWINFODESC_          ""

        //! Source File Type
        DWORD dwFileType;

        //! Set to 0 on allocate and leave alone
        DWORD dwResDrastic;                             // Always

} VVWINFO, * pVVWINFO;
//To write changes if anyhting has change din the metadata
#define DT_META_DATA_CHANGED 0x01
////////////////////////////////////////////////////////////////
//! Flag for mediafile/avhal to get audio dframe
#define GetAudio        0x00000000          // Use audio driver
//! Flag for mediafile/avhal to get video dframe
#define GetVideo        0x00000001          // Use video driver
//! Flag for mediafile/avhal to get info (VVWINFO)
#define GetInf          0x00000002
```

```
//! Flag for mediafile/avhal to put audio dframe
#define PutAudio        GetAudio                // As Above
//! Flag for mediafile/avhal to put audio dframe
#define PutVideo        GetVideo                // As Above
//! Flag for mediafile/avhal to get info (VVWINFO)
#define PutInf          GetInf
//! Flag for getting second copy or other non queue critical frames
#define GetNoFail       0x00000100
//! Flag for avhal to get the current #FRAME_INFO
#define GetCurrent      0x00000000              // Return current info
//! Flag for avhal to get the last queued #FRAME_INFO
#define GetQueued       0x00000010              // Return max queued info
//! Flag for avhal to get the last queued #FRAME_INFO
#define GetCurrentField         0x00000020              // Return max queued info
//! Flag for avhal to get the last queued #FRAME_INFO
#define IsSingleField   0x00000040              // Return max queued info
//! Flag for avhal to get the vbi video queued #FRAME_INFO
#define GetVBIVideo     0x00000080              // Return vbi video info
//! Flag to get CTL even if VITC/LTC selected
#define GetCTLTrack     0x00000200              // Return vbi video info

#define DT_TOP_FIELD                0x10000000
#define DT_BOTTOM_FIELD             0x20000000
#define DT_FRAME                    0x30000000
#define DT_WATERMARKED              0x04000000

#ifndef RC_INVOKED
#pragma pack()
#endif

#else //defined(_VVW_TYPES_HAVE_ALREADY_BEEN_INCLUDED)

        #ifdef _VVW_TYPES_DEFINE_META_DATA_TAG_NAME_ARRAY_DRASTIC
                static char * gsszMetaDataTagName[] =
                {
                (char*)VVWINFOTAG_szFileName,       //"FileName"              // # 1
                (char*)VVWINFOTAG_szNativeLocator,  //"NativeLocator"
                (char*)VVWINFOTAG_szUniversalName,          //"UniversalLocator"
                (char*)VVWINFOTAG_szIP,     //"TCP-IPAddress"
                (char*)VVWINFOTAG_szSourceLocator, //"SourceLocator"
                (char*)VVWINFOTAG_szChannel,        //"ChannelIdentifier"
                (char*)VVWINFOTAG_szChannelName, //"ChannelName"
                (char*)VVWINFOTAG_szChannelDescription,     //"ChannelDescription"
                (char*)VVWINFOTAG_szTitle,  //"Title"
                (char*)VVWINFOTAG_szSubject,        //"Subject"       // # 10
                (char*)VVWINFOTAG_szCategory,       //"Category"
                (char*)VVWINFOTAG_szKeywords,       //"Keywords"
                (char*)VVWINFOTAG_szRatings,        //"Ratings"
                (char*)VVWINFOTAG_szComments,       //"Comments"
```

```
        (char*)VVWINFOTAG_szDoNotUse,       //"DoNotUser"
        (char*)VVWINFOTAG_szOwner,//"Owner"
        (char*)VVWINFOTAG_szEditor, //"Editor"
        (char*)VVWINFOTAG_szSupplier,       //"Supplier"
        (char*)VVWINFOTAG_szSource,         //"Source"
        (char*)VVWINFOTAG_szProject,        //"Project"       // # 20
        (char*)VVWINFOTAG_szStatus, //"Status"
        (char*)VVWINFOTAG_szAuthor,//"Author"
        (char*)VVWINFOTAG_szRevisionNumber,       //"RevisionNumber"
        (char*)VVWINFOTAG_szProduced,       //"Produced"
        (char*)VVWINFOTAG_szAlbum, //"Album"
        (char*)VVWINFOTAG_szArtist,  //"Artist"
        (char*)VVWINFOTAG_szComposer,       //"Composer"
        (char*)VVWINFOTAG_szCopyright,      //"Copyright"
        (char*)VVWINFOTAG_szCreationData,   //"CreationData"
        (char*)VVWINFOTAG_szDescription,    //"Description"  // # 30
        (char*)VVWINFOTAG_szDirector,       //"Director"
        (char*)VVWINFOTAG_szDisclaimer,     //"Disclaimer"
        (char*)VVWINFOTAG_szEncodedBy,      //"EncodedBy"
        (char*)VVWINFOTAG_szFullName,       //"FullName"
        (char*)VVWINFOTAG_szGenre, //"Genre"
        (char*)VVWINFOTAG_szHostComputer, //"HostComputer"
        (char*)VVWINFOTAG_szInformation,    //"Information"
        (char*)VVWINFOTAG_szMake,  //"Make"
        (char*)VVWINFOTAG_szModel, //"Model"
        (char*)VVWINFOTAG_szOriginalArtist,   //"OriginalArtist"          // # 40
        (char*)VVWINFOTAG_szOriginalFormat,//"OriginalFormat"
        (char*)VVWINFOTAG_szPerformers,     //"Performers"
        (char*)VVWINFOTAG_szProducer,       //"Producer"
        (char*)VVWINFOTAG_szProduct,        //"Product"
        (char*)VVWINFOTAG_szSoftware,       //"Software"
        (char*)VVWINFOTAG_szSpecialPlaybackRequirements,
//"SpecialPlaybackRequirements"
        (char*)VVWINFOTAG_szTrack,  //"Track"
        (char*)VVWINFOTAG_szWarning,        //"Warning"
        (char*)VVWINFOTAG_szURLLink,        //"URL"
        (char*)VVWINFOTAG_szEditData1,      //"EditData1"             // # 50
        (char*)VVWINFOTAG_szEditData2,      //"EditData2"
        (char*)VVWINFOTAG_szEditData3,      //"EditData3"
        (char*)VVWINFOTAG_szEditData4,      //"EditData4"
        (char*)VVWINFOTAG_szEditData5,      //"EditData5"
        (char*)VVWINFOTAG_szEditData6,      //"EditData6"
        (char*)VVWINFOTAG_szEditData7,      //"EditData7"
        (char*)VVWINFOTAG_szEditData8,      //"EditData8"
        (char*)VVWINFOTAG_szEditData9,      //"EditData9"
        (char*)VVWINFOTAG_szVersionString,  //"VersionString"
        (char*)VVWINFOTAG_szManufacturer,  //"Manufacturer"          // # 60
        (char*)VVWINFOTAG_szLanguage,       //"Language"
        (char*)VVWINFOTAG_szFormat,         //"Format"
```

```
        (char*)VVWINFOTAG_szInputDevice,    //"InputDevice"
        (char*)VVWINFOTAG_szDeviceModelNum,        //"DeviceModelNum"
        (char*)VVWINFOTAG_szDeviceSerialNum,        //"DeviceSerialNum"
        (char*)VVWINFOTAG_szReel,    //"Reel"
        (char*)VVWINFOTAG_szShot,   //"Shot"
        (char*)VVWINFOTAG_szTake,   //"Take"
        (char*)VVWINFOTAG_szSlateInfo,        //"SlateInfo"
        (char*)VVWINFOTAG_szFrameAttribute ,          //"FrameAttribute"
// # 70
        (char*)VVWINFOTAG_szEpisode                 ,          //"Episode"
        (char*)VVWINFOTAG_szScene                   ,          //"Scene"
        (char*)VVWINFOTAG_szDailyRoll               ,          //"DailyRoll"
        (char*)VVWINFOTAG_szCamRoll                 ,          //"CamRoll"
        (char*)VVWINFOTAG_szSoundRoll               ,          //"SoundRoll"
        (char*)VVWINFOTAG_szLabRoll           ,          //"LabRoll"
        (char*)VVWINFOTAG_szKeyNumberPrefix         ,
//"KeyNumberPrefix"
        (char*)VVWINFOTAG_szInkNumberPrefix         ,
//"InkNumberPrefix"
        (char*)VVWINFOTAG_szPictureIcon             ,          //"PictureIcon"
        (char*)VVWINFOTAG_szProxyFile               ,          //"ProxyFile"
// # 80
        (char*)VVWINFOTAG_szNU_81                   ,          //"NotUsed_81"
        (char*)VVWINFOTAG_szNU_82                   ,          //"NotUsed_82"
        (char*)VVWINFOTAG_szNU_83                   ,          //"NotUsed_83"
        (char*)VVWINFOTAG_szNU_84                   ,          //"NotUsed_84"
        (char*)VVWINFOTAG_szNU_85                   ,          //"NotUsed_85"
        (char*)VVWINFOTAG_szNU_86                   ,          //"NotUsed_86"
        (char*)VVWINFOTAG_szNU_87                   ,          //"NotUsed_87"
        (char*)VVWINFOTAG_szNU_88                   ,          //"NotUsed_88"
        (char*)VVWINFOTAG_szNU_89                   ,          //"NotUsed_89"
        (char*)VVWINFOTAG_szNU_90                   ,          //"NotUsed_90"
// # 90
        (char*)VVWINFOTAG_szNU_91                   ,          //"NotUsed_91"
        (char*)VVWINFOTAG_szNU_92                   ,          //"NotUsed_92"
        (char*)VVWINFOTAG_szNU_93                   ,          //"NotUsed_93"
        (char*)VVWINFOTAG_szNU_94                   ,          //"NotUsed_94"
        (char*)VVWINFOTAG_szNU_95                   ,          //"NotUsed_95"
        (char*)VVWINFOTAG_szNU_96                   ,          //"NotUsed_96"
        (char*)VVWINFOTAG_szNU_97                   ,          //"NotUsed_97"
        (char*)VVWINFOTAG_szNU_98                   ,          //"NotUsed_98"
        (char*)VVWINFOTAG_szNU_99                   ,          //"NotUsed_99"
// # 99
        (char*)VVWINFOTAG_dwTimeCode,      //"TimeCode"
        (char*)VVWINFOTAG_dwUserBits,        //"UserBits"
        (char*)VVWINFOTAG_dwVITCTimeCode,          //"VITCTimeCode"
        (char*)VVWINFOTAG_dwVITCUserBits, //"VITCUserBits"
        (char*)VVWINFOTAG_dwVITCLine3,     //"VITCExtraData"
        (char*)VVWINFOTAG_dwPosterFrame,  //"PosterFrame"
```

```
        (char*)VVWINFOTAG_dwAFrame,         //"A-Frame"
        (char*)VVWINFOTAG_dwAspectRatio,    //"AspectRatio"
        (char*)VVWINFOTAG_dwOriginalRate,   //"OriginalRate"
        (char*)VVWINFOTAG_dwOriginalScale,  //"OriginalScale"
        (char*)VVWINFOTAG_dwConversions,    //"TotalConversions"
        (char*)VVWINFOTAG_dwVersionNumber,          //"VersionNumber"
        (char*)VVWINFOTAG_dwFileSize,       //"FileSize"
        (char*)VVWINFOTAG_dwFileDate,       //"FileDate"
        (char*)VVWINFOTAG_dwFileTime,       //"FileTime"
        (char*)VVWINFOTAG_dwSequenceNumber,     //"SequenceNumber"
        (char*)VVWINFOTAG_dwTotalStreams, //"TotalStreams"
        (char*)VVWINFOTAG_dwTotalLength,    //"TotalLength"
        (char*)VVWINFOTAG_dwFilmManufacturerCode,
    //"FilmManufacturerCode"
        (char*)VVWINFOTAG_dwFilmTypeCode,           //"FilmTypeCode"
        (char*)VVWINFOTAG_dwWhitePoint,     //"WhitePoint"
        (char*)VVWINFOTAG_dwBlackPoint,     //"BlackPoint"
        (char*)VVWINFOTAG_dwBlackGain,      //"BlackGain"
        (char*)VVWINFOTAG_dwBreakPoint,     //"BreakPoint"
        (char*)VVWINFOTAG_dwGamma1000,  //"Gamma1000"
        (char*)VVWINFOTAG_dwTagNumber,  //"TagNumber"
        (char*)VVWINFOTAG_dwFlags, //"Flags"
        (char*)VVWINFOTAG_dwTimeCodeType,           //"NumericNotUsed_28"
        (char*)VVWINFOTAG_dwLTCTimeCodeType,    //"NumericNotUsed_29"
        (char*)VVWINFOTAG_dwVITCTimeCodeType,   //"NumericNotUsed_30"
        (char*)VVWINFOTAG_dwProdDate,       //"NumericNotUsed_31"
    //End: v3.0
        (char*)VVWINFOTAG_dwUniqueID,       //"NumericNotUsed_32"
        (char*)VVWINFOTAG_dwNU_33,          //"NumericNotUsed_33"
        (char*)VVWINFOTAG_dwNU_34,          //"NumericNotUsed_34"
        (char*)VVWINFOTAG_dwNU_35,          //"NumericNotUsed_35"
        (char*)VVWINFOTAG_dwNU_36,          //"NumericNotUsed_36"
        (char*)VVWINFOTAG_dwNU_37,          //"NumericNotUsed_37"
        (char*)VVWINFOTAG_dwNU_38,          //"NumericNotUsed_38"
        (char*)VVWINFOTAG_dwNU_39,          //"NumericNotUsed_39"
        (char*)VVWINFOTAG_dwNU_40,          //"NumericNotUsed_40"
        (char*)VVWINFOTAG_dwNU_41,          //"NumericNotUsed_41"
        (char*)VVWINFOTAG_dwNU_42,          //"NumericNotUsed_42"
        (char*)VVWINFOTAG_dwNU_43,          //"NumericNotUsed_43"
        (char*)VVWINFOTAG_dwNU_44,          //"NumericNotUsed_44"
        (char*)VVWINFOTAG_dwNU_45,          //"NumericNotUsed_45"
        (char*)VVWINFOTAG_dwNU_46,          //"NumericNotUsed_46"
        (char*)VVWINFOTAG_dwNU_47,          //"NumericNotUsed_47"
        (char*)VVWINFOTAG_dwNU_48,          //"NumericNotUsed_48"
        (char*)VVWINFOTAG_dwNU_49,          //"NumericNotUsed_49"
    };
    #endif //_VVW_TYPES_DEFINE_META_DATA_TAG_NAME_ARRAY_DRASTIC
#endif //!defined(_VVW_TYPES_HAVE_ALREADY_BEEN_INCLUDED)
```

Appendix IV – VVWMediaCmd.bas

The Visual Basic version of the MediaCMD interface. Available on request from Drastic.

Appendix V – MEDIACMD.java

The Java version of the MediaCMD interface. Available on request from Drastic.