MediaCmd API

Version 4.0

www.drastic.tv

May 18, 2023

Contents

Copyrights and Trademark Notices

General

Harris Corporation - Harris, and Leitch Technology Corp. are registered trademarks of Harris Corporation
Hewlett Packard Enterprise Company - OpenGL is a registered trademark and the OpenGL SC logo is a trademark of Hewlett Packard Enterprise Company
Hewlett Packard Group LLC - HP is a trademark of HP Hewlett Packard Group LLC.
Ikegami Electronics (USA) Inc. - EditCam is a registered trademark of Ikegami Electronics (USA) Inc.
Indiecam GmbH - IndieCam is a registered trademark of Indiecam GmbH
INOGENI Inc - INOGENI® is a Registered Trademark and TOGGLE is a Trademark of INOGENI Inc
Institute of Electrical and Electronics Engineers - IRE is a trademark of the Institute of Electrical and Electronics Engineers
INTEL CORPORATION - INTEL is a trademark of INTEL CORPORATION
International Business Machines Corporation ("IBM") - IBM® is a trademark owned by International Business Machines Corporation ("IBM") and might also be trademarked or a registered trademark in other countries
Interactive Effects, Inc. - Piranha is a registered trademark of Interactive Effects, Inc.
I**O Industries Ltd.** - IO Industries is a trademark of IO Industries Ltd.
Iteris, Inc. - Odetics is a registered trademark of Iteris, Inc.
JVC KENWOOD CORPORATION - JVC is a trademark of JVC KENWOOD CORPORATION
Kinefinity Inc. - KINEFINITY is a trademark of Kinefinity Inc.
L3Harris Technologies, Inc. - Louth is a trademark of L3Harris Technologies, Inc.
Linus Torvalds - Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.
Logitech International SA - LOGITECH is a trademark of Logitech International SA
Magic Lantern - Magic Lantern is a registered trademark of Magic Lantern
MAINCONCEPT GMBH - MAIN CONCEPT is a trademark of MAINCONCEPT GMBH
Marshall Electronics, Inc. - Marshall is a registered trademark of Marshall Electronics, Inc.
Matrox Electronic Systems, Ltd - Matrox and Matrox product names are registered trademarks and/or trademarks of Matrox Electronic Systems, Ltd.
MediaArea.net SARL - MediaInfo - Copyright© 2002-2013 MediaArea.net SARL. All rights reserved.
Meta Platforms, Inc - Facebook and Instagram are trademarks of Meta Platforms, Inc
Microsoft: Windows®, Video For Windows (VFW), DirectShow, Microsoft, Skype, Microsoft Azure, Microsoft Teams, Wave Mapper, Microsoft, Windows NT|2000|XP|XP Professional|Server 2003|Server 2008 |Server 2012, Windows 7, Windows 8, Media Player, Media Encoder, .Net, Internet Explorer, SQL Server 2005| 2008|2012|2014, Windows Media Technologies and Internet Explorer are trademarks of Microsoft Corporation.
MPEG LA - MPEG LA licenses patent pools covering essential patents required for use of the MPEG-2, MPEG-4, IEEE 1394, VC-1, ATSC, MVC, MPEG-2 Systems, AVC/H.264 and HEVC standards.
Nanjing Magewell Electronics Co. - Magewell™, ULTRA STREAM® and (the MAGEWELL Logo) are trademarks or registered trademarks of Nanjing Magewell Electronics Co.
Netflix, Inc. - Netflix is a registered trademark of Netflix, Inc.
NewTek, Inc. - NDI, TriCaster, 3Play, TalkShow, Video Toaster, LightWave 3D, and Broadcast Minds are registered trademarks of NewTek, Inc.
Nokia Corporation - OSPREY is a trademark owned by Nokia Corporation
NVIDIA Corporation - NVIDIA, the NVIDIA logo, NVIDIA Quadro, Rivermax, BlueField2, PhysX, and NVIDIA RTX are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and/or other countries
Object Matrix Limited - ObjectMatrix, and Object Matrix are registered trademarks of Object Matrix Limited
Omneon Video Networks, Inc - Omneon is a trademark of Omneon Video Networks, Inc

Wizards of OBS, LLC – UNIX, OBS, Open Broadcast Software, the OBS logo, and OBS Studio are trademarks of Wizards of OBS, LLC (The Company)

Wowza Media Systems, LLC - Wowza is a trademark of Wowza Media Systems, LLC

Xceed Software Inc. - Xceed DataGrid for JavaScript, Xceed Ultimate ListBox for Silverlight, Xceed DataGrid for Silverlight, Xceed DataGrid for WPF, Xceed Grid for .NET, Xceed Zip for .NET, Xceed Real-Time Zip for Silverlight, Xceed Upload for Silverlight, Xceed Zip Compression Library, Xceed FTP for .NET, Xceed Chart for .NET, Xceed Chart for ASP.NET, Xceed SmartUI for .NET, Xceed SmartUI, Xceed Encryption Library, Xceed Binary Encoding Library, Xceed Streaming Compression Library, Xceed Streaming Compression for .NET, Xceed Zip for .NET Compact Framework, Xceed Ultimate Suite, Xceed Data Manipulation Suite, Xceed Absolute Packager are trademarks of Xceed Software Inc.

Zapex Technologies - Zapex is a registered trademark of Zapex Technologies

Zhang Haijun - RYBOZEN is a trademark of Zhang Haijun

Ziflow Limited - Ziflow is a trademark of Ziflow Limited

ZLIB: The ZLIB Compressed Data Format Specification is Copyright (C) 1995-2013 Jean-Loup Gailly and Mark Adler.

Zoom Video Communications, Inc. - Zoom and the Zoom logo are trademarks of Zoom Video Communications, Inc.

x264 LLC: The product is manufactured by Drastic Technologies under license from x264 LLC.

LGPL: Portions of this product are licensed under LGPL, governed by the following license:

MPEG Disclaimers

MPEGLA MPEG2 Patent

ANY USE OF THIS PRODUCT IN ANY MANNER OTHER THAN PERSONAL USE THAT COMPLIES WITH THE MPEG-2 STANDARD FOR ENCODING VIDEO INFORMATION FOR PACKAGED MEDIA IS EXPRESSLY PROHIBITED WITHOUT A LICENSE UNDER APPLICABLE PATENTS IN THE MPEG-2 PATENT PORTFOLIO, WHICH LICENSE IS AVAILABLE FROM MPEG LA, LLC, 4600 S. Ulster Street, Suite 400, Denver, Colorado 80237 U.S.A.

MPEGLA MPEG4 VISUAL

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 VISUAL PATENT PORTFOLIO LICENSE FOR THE PERSONAL AND NON-COMMERCIAL USE OF A CONSUMER FOR (i) ENCODING VIDEO IN COMPLIANCE WITH THE MPEG-4 VISUAL STANDARD ("MPEG-4 VIDEO") AND/OR (ii) DECODING MPEG-4 VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL AND NON-COMMERCIAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION INCLUDING THAT RELATING TO PROMOTIONAL, INTERNAL AND COMMERCIAL USES AND LICENSING MAY BE OBTAINED FROM MPEG LA, LLC. SEE HTTP://WWW.MPEGLA.COM.

MPEGLA AVC

THIS PRODUCT IS LICENSED UNDER THE AVC PATENT PORTFOLIO LICENSE FOR THE PERSONAL USE OF A CONSUMER OR OTHER USES IN WHICH IT DOES NOT RECEIVE REMUNERATION TO (i) ENCODE VIDEO IN COMPLIANCE WITH THE AVC STANDARD ("AVC VIDEO") AND/OR (ii) DECODE AVC VIDEO THAT WAS ENCODED BY A CONSUMER ENGAGED IN A PERSONAL ACTIVITY AND/OR WAS OBTAINED FROM A VIDEO PROVIDER LICENSED TO PROVIDE AVC VIDEO. NO LICENSE IS GRANTED OR SHALL BE IMPLIED FOR ANY OTHER USE. ADDITIONAL INFORMATION MAY BE OBTAINED FROM MPEG LA, L.L.C. SEE HTTP://WWW.MPEGLA.COM.

MPEG4 SYSTEMS

THIS PRODUCT IS LICENSED UNDER THE MPEG-4 SYSTEMS PATENT PORTFOLIO LICENSE FOR ENCODING IN COMPLIANCE WITH THE MPEG-4 SYSTEMS STANDARD, EXCEPT THAT AN ADDITIONAL LICENSE AND PAYMENT OF ROYALTIES ARE NECESSARY FOR ENCODING IN CONNECTION WITH (i) DATA STORED OR REPLICATED IN PHYSICAL MEDIA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND/OR (ii) DATA WHICH IS PAID FOR ON A TITLE BY TITLE BASIS AND IS TRANSMITTED TO AN END USER FOR PERMANENT STORAGE AND/OR USE. SUCH ADDITIONAL LICENSE MAY BE OBTAINED FROM MPEG LA, LLC. SEE HTTP://WWW.MPEGLA.COM FOR ADDITIONAL DETAILS.

Limited Warranty and Disclaimers

Drastic Technologies Ltd (the Company) warrants to the original registered end user that the product will perform as stated below for a period of ninety (90) days from the date of licensing or; in the case of hardware, for a period matching the warranty period offered by the original manufacturer of said equipment.

Hardware and Media—The Product hardware components, if any, including equipment supplied but not manufactured by the Company but NOT including any third party equipment that has been substituted by the

Distributor or customer for such equipment (the "Hardware"), will be free from defects in materials and workmanship under normal operating conditions and use.

Warranty Remedies

Your sole remedies under this limited warranty are as follows:
Hardware and Media—The Company will either repair or replace (at its option) any defective Hardware component or part, or Software Media, with new or like new Hardware components or Software Media. Components may not be necessarily the same, but will be of equivalent operation and quality.

Software Updates

Except as may be provided in a separate agreement between Drastic Technologies and You, if any, Drastic Technologies is under no obligation to maintain or support the Software and Drastic Technologies has no obligation to furnish you with any further assistance, technical support, documentation, software, update, upgrades, or information of any nature or kind.

Restrictions and Conditions of Limited Warranty

This Limited Warranty will be void and of no force and effect if (i) Product Hardware or Software Media, or any part thereof, is damaged due to abuse, misuse, alteration, neglect, or shipping, or as a result of service or modification by a party other than the Company, or (ii) Software is modified without the written consent of the Company.

Limitations of Warranties

THE EXPRESS WARRANTIES SET FORTH IN THIS AGREEMENT ARE IN LIEU OF ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. No oral or written information or advice given by the Company, its distributors, dealers or agents, shall increase the scope of this Limited Warranty or create any new warranties.

Geographical Limitation of Warranty—This limited warranty is valid only within the country in which the Product is purchased/licensed.

Limitations on Remedies—YOUR EXCLUSIVE REMEDIES, AND THE ENTIRE LIABILITY OF Drastic Technologies Ltd WITH RESPECT TO THE PRODUCT, SHALL BE AS STATED IN THIS LIMITED WARRANTY. Your sole and exclusive remedy for any and all breaches of any Limited Warranty by the Company shall be the recovery of reasonable damages which, in the aggregate, shall not exceed the total amount of the combined license fee and purchase price paid by you for the Product.

Damages

Drastic Technologies Ltd SHALL NOT BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OR INABILITY TO USE THE PRODUCT, OR THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, EVEN IF THE COMPANY HAS BEEN ADVISED OF THE POSSIBILITY OF THOSE DAMAGES, OR ANY REMEDY PROVIDED FAILS OF ITS ESSENTIAL PURPOSE.

Further information regarding this limited warranty may be obtained by writing:
Drastic Technologies Ltd
523 The Queensway, Suite 201
Toronto, ON, M8V 1J7
Telephone: (416) 255-5636

Introduction

The Drastic MediaCmd SDK is the mechanism by which all the elements of Drastic's DDRs communicate with one another. 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 structure, 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 statuses (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

Installation depends on what features of MediaCmd your application requires. The main types will be DDR Control, Network DDR Control and VTR control:

DDR Control

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 yourself 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.

Network DDR Control

Generally network control will require either the DTMediaCmdNetRedist (or DTMediaCmdNetRedist64) installer or it will be using the direct source in C or Java, in which case no installation will be necessary.

VTR Control

The VTR Control will require one of four installers. Either the DTMediaCmdVtrDev or 64 bit development installer, or the DTMediaCmdVtrRedist (or DTMediaCmdVtr64Redist) installer. For end users, the redist must be used. Alternately, the DLL from the redist can be incorporated into your installer, so long as the compile redist is run (included) and if your application used the ActiveX method, the VVW.dll must be registered:
regsvr32 vvw.dll

Choosing An Access Method

The SDK access method you should use depends on what 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 – Direct On Server [deprecated]

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. Deprecated.
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 [deprecated]

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. Deprecated.
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. Real-time debug (Firefox). Works with most browsers.
Cons: Can be slow in some instances, JavaScript somewhat limited.
Setup: Connect to server with your 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 – Multi Platform

Type: Direct link to VVW.DLL
Pros: No ActiveX layer, code compatible with Linux, Irix, Mac OS-X.
Cons: Requires more configuration, normally via config.xml or library calls.
Setup: Link to vvw.lib, include vvwIf.h. Run appropriate redist on end user machines.
Issues: Normally configuration

Network DLL – no longer actively supported

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 vvwOpenLocal function to avoid QuickClip configuration issues. Requires a few DLLs 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: Your application 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.

SDK Structure

The location of the SDK directories will depend on the location you choose during the installation, but the directories within them 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
    - o \ActiveX – Examples that use the ActiveX control
    - o \Direct – Examples that link directly to DLLs
    - o \Java – Java based examples
    - o \HTTP – Ajax based examples (must use QuickClip HTTP server to run)

Main Documentation

*** PDF version of the MediaCmd Documentation
https://www.drastic.tv/images/documentation/mediacmd4api.pdf

Low Level Header Documentation

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

HTTP XML AJAX Documentation

*** Wiki 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
HTML	- mediacmd.xml	- XML I/O (w HTML components)
XML	- mediacmd.xml	- XML direct network connection

Direct Interface:

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

References

MediaCmd.h	- Internal mediacmd 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 Direct 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
Direct	vvw	Link vvw.dll / include vvwIF.h
XML		Via HTTP or Direct

- All pointers for ActiveX C++ and Direct 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 ActiveX/Direct, BOOL is the Windows.h BOOL definition
- szClipName has a maximum of 8 alphanumeric 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 your 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.

```
            import drastic.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 its current location.

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

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

This allows the VVW Interface 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*
Direct	long vvw EnableChannels (long lInternal0_31, long lInternal32_63, long lExternal64_95, long lExternal96_127, long lNetwork128_159, long lNetwork160_192)
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 ()
Direct	long vvwGetMaxChannels (long lChannel)
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)
Direct	*Not implemented*
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 ()
Direct	*Not implemented*
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 ()
Direct	Long vvwGetChannelName(long lChannel, char * szChanName)
XML	*Not required*

Get the name of the current channel. For Direct 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 ()
Direct	long vvwGetChannelType (long lChannel)
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*
Direct	long vvw ShowConfigDialog (long hWnd)
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 as the dialog only shows up on the local machine, and cannot be seen 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*
Direct	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*
Direct	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)
Direct	*Not implemented*
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 set up a persistent network connection in the VVW configuration 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()
Direct	*Not implemented*
XML	*Not required*

Disconnect from a previously connected server. For ActiveX and Direct access, please set up a persistent network connection in the VVW configuration utility.

Transport Operations

Play

ActiveX VB	vbx.Play () As Long
ActiveX C++	long pcx->Play ()
Java	long mci.Play ()
Direct	long vvwPlay (long lChannel)
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)
Direct	long vvwPlayAtSpeed (long lChannel, 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)
Direct	long vvwPlayFromTo (long lChannel, long lFrom, BOOL lTo, bool 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)
Direct	long vvwLoadClip (long lChannel, 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)
Direct	long vvwPlayClip (long lChannel, char * szClipName, BOOL 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)
Direct	long vvwPlayClipFromTo (long lChannel, char * szClipName, long lFrom, long lTo, BOOL 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)
Direct	long vvwPlayAtMs (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 the millisecond equivalent of a number of frames and playback will commence at that time. Be sure to 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 ()
Direct	long vvwFastForward (long lChannel)
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 ()
Direct	long vvwFastRewind (long lChannel)
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 ()
Direct	long vvwPause (long lChannel)
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)
Direct	long vvwSeek (long lChannel, 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)
Direct	long vvwSeekRelative (long lChannel, 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 offsets imply reverse.

Stop

ActiveX VB	vbx.Stop () As Long
ActiveX C++	long pcx-> Stop ()
Java	long mci. Stop ()
Direct	long vvwStop (long lChannel)
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 ()
Direct	long vvwRecord (long lChannel)
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)
Direct	long vvwRecordAtMs (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 the millisecond equivalent of a number of frames and the record will commence at that time. Be sure to 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)
Direct	long vwwRecordFromTo (long lChannel, 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)
Direct	Long vvwRecordStop (long lChannel, 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)
Direct	long vvwSetRecordPresets (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 ()
Direct	long vvwEject (long lChannel)
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)
Direct	long vvwTransfer (long lChannel, long lTargetChannel, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOL 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 time code 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 ()
Direct	long vvwUpdateStatus (long lChannel)
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 ()
Direct	long vvwGetState (long lChannel)
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 ()
Direct	long vvwGetFlags (long lChannel)
XML	*Not required*

Returns the current flags

cfDeferred = 1,	// 0x00000001 This is a delayed
cfOverideDeferred = 1 << 30,	// 0x40000000 Override 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 time code
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 an 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 ()
Direct	long vvwGetSpeed (long lChannel)
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 ()
Direct	long vvwGetPosition (long lChannel)
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 ()
Direct	long vvwGetLastMs (long lChannel)
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 ()
Direct	long vvwGetStart (long lChannel)
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 ()
Direct	long vvwGetEnd (long lChannel)
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 ()
Direct	long vvwGetClipName (long lChannel, char * sz8CharClipName)
XML	http://localhost/VVWXMLGetStatus?

Only supported in clip Mode. Returns the current clip name, if any. For Direct 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 ()
Direct	Long GetFileName (long lChannel, char * sz260CharFileName)
XML	Not supported

Returns the current file name, if any. For Direct access, the memory must be at least 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 ()
Direct	Long GetCurTC (long lChannel, char * sz14ByteTC)
XML	http://localhost/VVWXMLGetStatus?

Returns the current time code as a string (e.g. "00:01:00:00"). For Direct 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 ()
Direct	Long GetCurState (long lChannel, char * sz14ByteState)
XML	http://localhost/VVWXMLGetStatus?

Returns the current state as a string (e.g. "Play"). For Direct 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

GetNextClip

ActiveX VB	Vbx. GetNextClip (szLastClip As String) As String
ActiveX C++	BSTR pcx-> GetNextClip (BSTR szLastClip)
Java	String mci. GetNextClip (String szLastClip)
Direct	Char * vvwGetNextClip (long lChannel, 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 Direct 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)
Direct	Long vvwGetClipInfo (long lChannel, char * sz8CharClipName, long * lStart, long * lEnd, long * lVidEdit, long * lAudEdit, long * lInfEdit, char * sz260CharFileName)
XML	http://localhost/VVWXMLNextClip?

Returns 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 object'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*
Direct	Long vvwGetNextClipEx (long lChannel, char * szClipName, long * lCreation, long * lLastModification, long * lFileSize, long * lDiskFragments)
XML	*Not supported*

Returns 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)
Direct	Long vvwCopyClip (long lChannel, 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.

EDLResetToStart

ActiveX VB	Vbx. EDLResetToStart () As Long
ActiveX C++	Long pcx-> EDLResetToStart ()
Java	Long mci. EDLResetToStart ()
Direct	Long vvwEDLResetToStart (long lChannel)
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)
Direct	long vvwEDLGetEdit (long lChannel, long * lRecordIn, long * lPlayIn, long * lPlayOut, long * lVidEdit, long * lAudEdit, long * lInfEdit, char * sz8CharClipName, char * sz260CharFileName)
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 time code 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 successful 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 object'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>
```

Media Operations – Shared Operations

GetLastChangeMs

ActiveX VB	vbx. GetLastChangeMs () As Long
ActiveX C++	long pcx-> GetLastChangeMs ()
Java	long mci. GetLastChangeMs ()
Direct	long vvwGetLastChangeMs (long lChannel)
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)
Direct	long vvwInsert (long lChannel, char * szClipName, char * szFileName, long lPosition, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOl 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)
Direct	long vvwBlank (long lChannel, char * szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOl 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)
Direct	long vvwDelete (long lChannel, char * szClipName, long lStart, long lEnd, long lVidEdit, long lAudEdit, long lInfEdit, BOOl 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)
Direct	long vvwTrim (long lChannel, long lPosition, long lStartOffset, long lEndOffset, long lVidEdit, long lAudEdit, long lInfEdit, BOOl 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)
Direct	long vvwValueSupported (long lChannel, 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)
Direct	long vvwValueGet (long lChannel, 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 object'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)
Direct	long vvwValueSet (long lChannel, 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)
Direct	long vvwValueSet2 (long lChannel, 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)
Direct	Long vvwGet*XXX* ()
	Long vvwSet*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 parameter to 9 (1001). You may also use the predefined audio channel definitions in the MEDIACMD structure (audChanX, where x is the audio channel – 1).

```
//! 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
//! 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 parameter to 9 (1001). You may also use the predefined audio channel definitions 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 parameter to 9 (1001). You may also use the predefined audio channel definitions 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 parameter to 9 (1001). You may also use the predefined audio channel definitions 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 superimpose 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*
Direct	Void vvwFreeString (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)
Direct	*Use TCXlat*
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)
Direct	*Use TCXlat*
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 string time code 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)
Direct	*Use TCXlat*
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)
Direct	*Not implemented*
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)
Direct	*Not implemented*
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 audio channels 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) 1998-2023 Drastic Technologies Ltd.  All Rights Reserved.
 * 523 The Queensway, Suite 201 Toronto ON M8Y 1J7
 * 416 255 5636  fax 255 8780
 * engineering@drastictech.com      http://www.drastic.tv
 *****************************************************************/

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

#ifndef _MEDIACMD_INCLUDED_H
#define _MEDIACMD_INCLUDED_H

#ifndef __midl
	// Make sure we include this for __x86_64__ and __i386__ (pointer size)
	#include "dtsystemtypes.h"
#endif

#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 Speed 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.
        * <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>
        */
        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
        * <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 locations 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 if 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
          /**
          * 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 vsynceGetCurMs()
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
#define cfInlayClock cfRipple
	/**
	* 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,
	/**
	* This flags tells the DDR that the command needs to span over the 24 hour mark
	*
	*
	* <HR>
```

```
        */
        cfOver24Hours = 1 << 26,
        /**
        * 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
        /**
        * For play commands that are tracking live sources.
        */
        cfIsLive = cfInvert,
        /**
        * 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 it is 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
        /**
        * Left this as a bit of a catch all, essentially if we rest the channel and then pause, when we have ‘match
output to clip’ on it resets
        * Use this for a more generic ignore cmd you can use the cmd alt for things to ignore
        * <BR> Caution:
        * <HR>
        */
        cfIgnore = 1 << 30,                   // 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
};

//! Spelling
#define cfOverideDeferred	cfOverrideDeferred

enum	cfIgnoreFlags	{
	ignoreSetup		= 1 << 0,
	ignorePresets	= 1 << 1,
#defineIgnoreAll 0xFFFFFFFFUL
};

// 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,
	//! Closed 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 'nothing' 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 - current tc
* <BR> MEDIACMD::dwVideoChannels - Always VITC time code frame value
* <BR> MEDIACMD::dwStart - Always VITC user bits
* <BR> MEDIACMD::dwAudioChannels - Always LTC time code frame value
* <BR> MEDIACMD::dwEnd - Always VITC user bits
* <BR> MEDIACMD::dwInfoChannels - Always absolute frame number
* <BR> MEDIACMD::lSpeed - Always VITC Aux setting (dwVitcAux)
* <HR>
*/
gsTc = 1,                              // Current internal TC (dwPosition)
/**
* Current internal user bits
* <BR>
* \li cmdType::ctSetValue
* <BR>  nada
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - current ub
* <BR> MEDIACMD::dwVideoChannels - Always VITC time code frame value
* <BR> MEDIACMD::dwStart - Always VITC user bits
* <BR> MEDIACMD::dwAudioChannels - Always LTC time code frame value
* <BR> MEDIACMD::dwEnd - Always VITC user bits
* <BR> MEDIACMD::dwInfoChannels - Always absolute frame number
* <BR> MEDIACMD::lSpeed - Always VITC Aux setting (dwVitcAux)
* <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 - current Ltc
* <BR> MEDIACMD::dwVideoChannels - Always VITC time code frame value
* <BR> MEDIACMD::dwStart - Always VITC user bits
* <BR> MEDIACMD::dwAudioChannels - Always LTC time code frame value
* <BR> MEDIACMD::dwEnd - Always VITC user bits
* <BR> MEDIACMD::dwInfoChannels - Always absolute frame number
* <BR> MEDIACMD::lSpeed - Always VITC Aux setting (dwVitcAux)
* <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 - current Lub
        * <BR> MEDIACMD::dwVideoChannels - Always VITC time code frame value
        * <BR> MEDIACMD::dwStart - Always VITC user bits
        * <BR> MEDIACMD::dwAudioChannels - Always LTC time code frame value
        * <BR> MEDIACMD::dwEnd - Always VITC user bits
        * <BR> MEDIACMD::dwInfoChannels - Always absolute frame number
        * <BR> MEDIACMD::lSpeed - Always VITC Aux setting (dwVitcAux)
        * <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 - current Vitc
        * <BR> MEDIACMD::dwVideoChannels - Always VITC time code frame value
        * <BR> MEDIACMD::dwStart - Always VITC user bits
        * <BR> MEDIACMD::dwAudioChannels - Always LTC time code frame value
        * <BR> MEDIACMD::dwEnd - Always VITC user bits
        * <BR> MEDIACMD::dwInfoChannels - Always absolute frame number
        * <BR> MEDIACMD::lSpeed - Always VITC Aux setting (dwVitcAux)
        * <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 - current Viub
        * <BR> MEDIACMD::dwVideoChannels - Always VITC time code frame value
        * <BR> MEDIACMD::dwStart - Always VITC user bits
        * <BR> MEDIACMD::dwAudioChannels - Always LTC time code frame value
        * <BR> MEDIACMD::dwEnd - Always VITC user bits
        * <BR> MEDIACMD::dwInfoChannels - Always absolute frame number
        * <BR> MEDIACMD::lSpeed - Always VITC Aux setting (dwVitcAux)
        * <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,
#GS_TCSOURCE_CLIP, #GS_TCSOURCE_IRIG
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - #GS_TCSOURCE_LTC, #GS_TCSOURCE_VITC, #GS_TCSOURCE_CTL,
#GS_TCSOURCE_CLIP, #GS_TCSOURCE_IRIG
        * <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,                              // 10 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,			// 20 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,		// 30
	/**
	* Get/Set closed captioning
	* <BR>
	* \li cmdType::ctSetValue
	* <BR>  MEDIACMD::dwPosition - Set as primary:  #GS_CC_DISABLE #GS_CC_CC1 #GS_CC_CC2 
#GS_CC_CC3 #GS_CC_CC4
	*			#GS_CC_TEXT1 #GS_CC_TEXT2 #GS_CC_TEXT3 #GS_CC_TEXT4 #GS_CC_XDS
	*			#GS_CC_708
	* <BR>  MEDIACMD::dwStart - Bitwise 'use these if available':  #GS_CC_DISABLE #GS_CC_CC1 
#GS_CC_CC2 #GS_CC_CC3 #GS_CC_CC4
	*			#GS_CC_TEXT1 #GS_CC_TEXT2 #GS_CC_TEXT3 #GS_CC_TEXT4 #GS_CC_XDS
	*			#GS_CC_708
	* <BR>MEDIACMD::dwEnd - If 1, display captions on VGA/Waveform Vector
	* \li cmdType::ctGetValue - not supported
	* <BR>  MEDIACMD::dwPosition - Current primary:  #GS_CC_DISABLE #GS_CC_CC1 #GS_CC_CC2 
#GS_CC_CC3 #GS_CC_CC4
	*			#GS_CC_TEXT1 #GS_CC_TEXT2 #GS_CC_TEXT3 #GS_CC_TEXT4 #GS_CC_XDS
	*			#GS_CC_708
	* <BR>  MEDIACMD::dwStart - Bitwise available types:  #GS_CC_DISABLE #GS_CC_CC1 #GS_CC_CC2 
#GS_CC_CC3 #GS_CC_CC4
	*			#GS_CC_TEXT1 #GS_CC_TEXT2 #GS_CC_TEXT3 #GS_CC_TEXT4 #GS_CC_XDS
	*			#GS_CC_708
	* <BR>MEDIACMD::dwEnd - If 1, displaying captions on VGA/Waveform Vector
	* <BR>MEDIACMD::arbID - String of last known characters
	* <HR>
	*/
	gsCCSetup,

	/**
	* Enable or disable all time code, or force time of day
	* <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>
	*/
	gsDisableTimecode,

	/**
	* Set the order of precedence for time code sources for the dwVitcFrame/Ub in fiInfo
	* <BR>
	* \li cmdType::ctSetValue
	* <BR>  MEDIACMD::dwPosition - Most favored type:  #GS_SOURCEPRECEDENCE_RP188_V, 
#GS_SOURCEPRECEDENCE_RP188_L, #GS_SOURCEPRECEDENCE_SMPTE, 
	*			#GS_SOURCEPRECEDENCE_TOD, #GS_SOURCEPRECEDENCE_VITC, 
#GS_SOURCEPRECEDENCE_IRIG, #GS_SOURCEPRECEDENCE_RP215, 
#GS_SOURCEPRECEDENCE_FRAMECOUNT
```

```
        * <BR>MEDIACMD::dwStart - 2nd most favored type:  see MEDIACMD::dwPostion for possible types
        * <BR>MEDIACMD::dwEnd - 3nd most favored type:  see MEDIACMD::dwPostion for possible types
        * <BR>MEDIACMD::dwVideoChannels - 4th most favored type:  see MEDIACMD::dwPostion for possible
types
        * <BR>MEDIACMD::dwAudioChannels - 5th most favored type:  see MEDIACMD::dwPostion for possible
types
        * <BR>MEDIACMD::dwInfoChannels - 6th most favored type:  see MEDIACMD::dwPostion for possible
types
        * <BR>MEDIACMD::lSpeed - 7th most favored type:  see MEDIACMD::dwPostion for possible types
        * \li cmdType::ctGetValue - Returns the current order of precedence
        * <BR>  MEDIACMD::dwPosition - Most favored type:  #GS_SOURCEPRECEDENCE_RP188_V,
#GS_SOURCEPRECEDENCE_RP188_L, #GS_SOURCEPRECEDENCE_SMPTE,
        *                         #GS_SOURCEPRECEDENCE_TOD, #GS_SOURCEPRECEDENCE_VITC,
#GS_SOURCEPRECEDENCE_IRIG, #GS_SOURCEPRECEDENCE_RP215,
#GS_SOURCEPRECEDENCE_FRAMECOUNT
        * <BR>MEDIACMD::dwStart - 2nd most favored type:  see MEDIACMD::dwPostion for possible types
        * <BR>MEDIACMD::dwEnd - 3nd most favored type:  see MEDIACMD::dwPostion for possible types
        * <BR>MEDIACMD::dwVideoChannels - 4th most favored type:  see MEDIACMD::dwPostion for possible
types
        * <BR>MEDIACMD::dwAudioChannels - 5th most favored type:  see MEDIACMD::dwPostion for possible
types
        * <BR>MEDIACMD::dwInfoChannels - 6th most favored type:  see MEDIACMD::dwPostion for possible
types
        * <BR>MEDIACMD::lSpeed - 7th most favored type:  see MEDIACMD::dwPostion for possible types
        * <HR>
        */
        gsVITCSourcePrecedence,

        /**
        * Set the order of precedence for time code sources for the dwLtcFrame/Ub in fiInfo
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - Most favored type:  #GS_SOURCEPRECEDENCE_RP188_V,
#GS_SOURCEPRECEDENCE_RP188_L, #GS_SOURCEPRECEDENCE_SMPTE,
        *                         #GS_SOURCEPRECEDENCE_TOD, #GS_SOURCEPRECEDENCE_VITC,
#GS_SOURCEPRECEDENCE_IRIG, #GS_SOURCEPRECEDENCE_RP215,
#GS_SOURCEPRECEDENCE_FRAMECOUNT
        * <BR>MEDIACMD::dwStart - 2nd most favored type:  see MEDIACMD::dwPostion for possible types
        * <BR>MEDIACMD::dwEnd - 3nd most favored type:  see MEDIACMD::dwPostion for possible types
        * <BR>MEDIACMD::dwVideoChannels - 4th most favored type:  see MEDIACMD::dwPostion for possible
types
        * <BR>MEDIACMD::dwAudioChannels - 5th most favored type:  see MEDIACMD::dwPostion for possible
types
        * <BR>MEDIACMD::dwInfoChannels - 6th most favored type:  see MEDIACMD::dwPostion for possible
types
        * <BR>MEDIACMD::lSpeed - 7th most favored type:  see MEDIACMD::dwPostion for possible types
        * \li cmdType::ctGetValue - Returns the current order of precedence
        * <BR>  MEDIACMD::dwPosition - Most favored type:  #GS_SOURCEPRECEDENCE_RP188_V,
#GS_SOURCEPRECEDENCE_RP188_L, #GS_SOURCEPRECEDENCE_SMPTE,
	*				#GS_SOURCEPRECEDENCE_TOD, #GS_SOURCEPRECEDENCE_VITC,
#GS_SOURCEPRECEDENCE_IRIG, #GS_SOURCEPRECEDENCE_RP215,
#GS_SOURCEPRECEDENCE_FRAMECOUNT
	* <BR>MEDIACMD::dwStart - 2nd most favored type:  see MEDIACMD::dwPostion for possible types
	* <BR>MEDIACMD::dwEnd - 3nd most favored type:  see MEDIACMD::dwPostion for possible types
	* <BR>MEDIACMD::dwVideoChannels - 4th most favored type:  see MEDIACMD::dwPostion for possible
types
	* <BR>MEDIACMD::dwAudioChannels - 5th most favored type:  see MEDIACMD::dwPostion for possible
types
	* <BR>MEDIACMD::dwInfoChannels - 6th most favored type:  see MEDIACMD::dwPostion for possible
types
	* <BR>MEDIACMD::lSpeed - 7th most favored type:  see MEDIACMD::dwPostion for possible types
	* <HR>
	*/
	gsLTCSourcePrecedence,

	/** Get the next clip value.  In clip mode, this returns a series of clips.  To get the first clip,
	* pass a NULL or all spaces string as the 8 character clip.  To get each subsequent clip, send back
	* the 8 character clip returned previously.  The full clip name will be in the MEDIACMD::arbID
	* starting at position [9] (right after the 8 character clip name)
	* <BR>
	* \li cmdType::ctSetValue
	* <BR>  - not supported
	* \li cmdType::ctGetValue - return the next clip name
	* <BR>MEDIACMD::dwPostion - not != #GS_NOT_SUPPORTED if clip is value, else end of list
	* <BR>MEDIACMD::dwStart - starting frame of clip
	* <BR>MEDIACMD::dwEnd - ending frame of clip (exclusive)
	* <BR>MEDIACMD::arbID - 8 char clip name, terminating 0, long name (starting at 9), terminating 0
	* <HR>
	*/
	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 than 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
        /**
        * Get set timecode edl comment info, later other things as well such as effects (cut wipe fade)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  Requires MEDIACMD::cfFlags set to affect stored clip info for each member
        * <BR>  MEDIACMD::arbID                          - String to set
        * <BR>  MEDIACMD::dwPosition          - Location on edl
        * <BR>  MEDIACMD::dwStart             - type to set (Comment / Effect)
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::arbID                          - Comment
        * <BR>  MEDIACMD::dwPosition          - Location on edl
        * <BR>  MEDIACMD::dwStart             - type set (Comment / Effect)
        * <HR>
        */
        gsTcClipInfo,                            // 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 metadata 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,

        /**
        * 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 its 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 its 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 its 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 - HIWORD=RMS channel +1, LOWORD=RMS channel +0 (range 0-65535)
        * <BR>MEDIACMD::dwEnd - HIWORD=Peak channel +1, LOWORD=Peak channel +0 (range 0-65535)
        * <BR>MEDIACMD::dwPosition - duplicates #MEDIACMD::dwStart
        * <HR>
        */
        gsAudWavePeakRMS,               // Current play or in peak|rms 0-(dwStart:HIWORD|LOWORD), 1-
(dwEnd:HIWORD|LOWORD)
        /**
        * 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,
	/**
	* Set an audio file to associate with video playback (hdr specific)
	* <BR>
	* \li cmdType::ctSetValue
	* <BR>MEDIACMD::arbID - audio filename
	* \li cmdType::ctGetValue
	* <BR>MEDIACMD::arbID - audio filename
	* <HR>
	*/
	gsAudPlaybackFile,	// 231
	/**
	* 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>
	*/
	gsAudLevelSelect,
	/**
	* Select if we want audio rms or loudness returned
	* <BR>
	* \li cmdType::ctSetValue
	* <BR>MEDIACMD::dwPosition - return type #GS_AUDIO_LEVEL_RMS, #GS_AUDIO_LEVEL_LOUDNESS
	* <BR>MEDIACMD::dwStart - return the meter mode  GS_EBU_MODE_NONE = 0,
GS_EBU_MODE_MOMENTARY, GS_EBU_MODE_SHORT_TERM, GS_EBU_MODE_INTEGRATED
	* <BR>MEDIACMD::dwEnd -  GS_EBU_SCALE_9 or GS_EBU_SCALE_18
   *						EBU_SCALE_9 range -18.0 LU to +9.0 LU (-41.0 LUFS to -14.0
LUFS), named ‘EBU +9 scale’
	*						EBU_SCALE_18 range -36.0 LU to +18.0 LU (-59.0 LUFS to -5.0
LUFS), named ‘EBU +18 scale’
	* \li cmdType::ctGetValue
	* <BR>MEDIACMD::dwPosition - selected type
	* <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)
	/**
	* Sets VGA playback and preview monitors to Anamorphic
	* <BR>
	* \li cmdType::ctSetValue
	* <BR>MEDIACMD::dwPosition - 1 = on, 0 = off
	* \li cmdType::ctGetValue
	* <BR>MEDIACMD::dwPosition - 1 = on, 0 = off
	* <HR>
	*/
	gsEnableSDAnamorphic,
	// 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)
        /**
        * Route Output video from to the target channel
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition - -1 for no routing vvwChannel # to target
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition          - Routing Channel targeted
        * <BR>MEDIACMD::lSpeed                       - Route Audio
        * <BR>MEDIACMD::dwVideoChannels  - Internal Use Only do not set
        * <BR>MEDIACMD::dwAudioChannels  - Internal Use Only do not set
        * <HR>
        */
        gsVideoInputRouting,   //Re-route video input from the selected channel to the target
        /**
        * 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 Dual Link, the out switches to dual link too.  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_SONY_HD_MXF,
	*			#VIDEOWRITETYPE_SONY_SR_MXF, #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_JP2K,
#VIDEOWRITETYPE_OP1a_MXF, #VIDEOWRITETYPE_DCP_MXF,
	*						#VIDEOWRITETYPE_TS,
#VIDEOWRITETYPE_MP4, #VIDEOWRITETYPE_FLASH, #VIDEOWRITETYPE_DNG
	* <BR>
	* \li cmdType::ctGetValue - See defines for file types
	* <HR>
	* <BR> MEDIACMD::dwPosition - #VIDEOWRITETYPE_AVI, #VIDEOWRITETYPE_MOV,
#VIDEOWRITETYPE_WMV, #VIDEOWRITETYPE_GEN, #VIDEOWRITETYPE_SONY_HD_MXF,
	*			#VIDEOWRITETYPE_SONY_SR_MXF, #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_JP2K,
#VIDEOWRITETYPE_OP1a_MXF, #VIDEOWRITETYPE_DCP_MXF,
	*						#VIDEOWRITETYPE_TS,
#VIDEOWRITETYPE_MP4, #VIDEOWRITETYPE_FLASH, #VIDEOWRITETYPE_DNG
	* <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 known fourcc
* <BR>
* \li cmdType::ctGetValue - See fccDef.h for known fourcc
* <HR>
*/
gsAlphaChromaSource,
/**
* RGBA BGRA ycbcr
* set to return fourcc code in dwPosition
* <BR>
* \li cmdType::ctSetValue - See fccDef.h for known fourcc
* <BR>
* \li cmdType::ctGetValue - See fccDef.h for known fourcc
* <HR>
*/
gsCompressionType,
/**
* <BR>
* \li cmdType::ctSetValue -
* <BR>
* \li cmdType::ctGetValue -
* <HR>
*/
gsVideoStandard,
/**
* Reset the channel to the new setup
* <BR>
* \li cmdType::ctSetValue - dwPositon = 0 for reset any other value to cancel
* <BR>
* \li cmdType::ctGetValue - returns TRUE if has changed FALSE if it hasn't
* <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,

        /**
        * 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_BAYER Raw Bayer frame ARRI LMP Weisscam Phantom
* <BR>#GS_COMPTYPE_H264 h264 I or IBP
* <BR>#GS_COMPTYPE_JPEG2000 - JPEG 2000
* <BR>#GS_COMPTYPE_CINEFORM_3D CineForm 3D Wavelet codec
* <BR>#GS_COMPTYPE_BGR BGR uncompressed
* <BR>#GS_COMPTYPE_HDCAM HDCam 10 bit 4:4:4/4:2:2 MPEG-4
* <BR>#GS_COMPTYPE_MPEG  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_CINEFORM CineForm normal 2D compression
* <BR>#GS_COMPTYPE_YCRCB_V210 10 Bit YCbCr
* <BR>#GS_COMPTYPE_RGB RGB 24 bit
* <BR>#GS_COMPTYPE_DNxHD Avid DNxHD
* <BR>#GS_COMPTYPE_AVCi AVC Intra Panasonic
* <BR>#GS_COMPTYPE_PRORES Apple ProRes
* <BR>#GS_COMPTYPE_BGRA_INVERT BGRA (TGA) vertically inverted
* <BR>#GS_COMPTYPE_DPX_YCBCR10 DPX YCbCr 10 bit
* <BR>#GS_COMPTYPE_ARGB ARGB
* <BR>#GS_COMPTYPE_RGBA RGBA
* <BR>#GS_COMPTYPE_ABGR Uncompressed A BGR - TIFF
* <BR>#GS_COMPTYPE_BGRA Uncompressed BGR A - BMP/TGA
* <BR>#GS_COMPTYPE_YCRCB_422 Uncompressed Y'CrCb 4:2:2 (DVS, VG)
* <BR>#GS_COMPTYPE_YCRCB_422A Uncompressed Y'CrCb 4:2:2A (DVS, Dual VG)
* <BR>#GS_COMPTYPE_YCRCB_444 Uncompressed Y'CrCb 4:4:4 (DVS, Dual VG)
* <BR>#GS_COMPTYPE_YCRCB_444A Uncompressed Y'CrCb 4:4:4A (DVS, Dual VG)
* <BR>#GS_COMPTYPE_STEREO Uncompressed Y'CrCb 4:4:4A (DVS, Dual VG) or 3D 8, 10, 30 or 32 bit
* <BR>#GS_COMPTYPE_YCRCB_420 Uncompressed Y'CrCb 4:2:0
* <BR>#GS_COMPTYPE_DPX_RGB10  DPX 10 bit rgb
* <BR>#GS_COMPTYPE_ALT    Use as generic alternative for use through AVCodec
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition -
* <BR>#GS_COMPTYPE_SOFTWARE    Software passed codec on main processor
* <BR>#GS_COMPTYPE_BAYER Raw Bayer frame ARRI LMP Weisscam Phantom
* <BR>#GS_COMPTYPE_H264 h264 I or IBP
* <BR>#GS_COMPTYPE_JPEG2000 - JPEG 2000
* <BR>#GS_COMPTYPE_CINEFORM_3D CineForm 3D Wavelet codec
* <BR>#GS_COMPTYPE_BGR BGR uncompressed
* <BR>#GS_COMPTYPE_HDCAM HDCam 10 bit 4:4:4/4:2:2 MPEG-4
* <BR>#GS_COMPTYPE_MPEG  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_CINEFORM CineForm normal 2D compression
	* <BR>#GS_COMPTYPE_YCRCB_V210 10 Bit YCbCr
	* <BR>#GS_COMPTYPE_RGB RGB 24 bit
	* <BR>#GS_COMPTYPE_DNxHD Avid DNxHD
	* <BR>#GS_COMPTYPE_AVCi AVC Intra Panasonic
	* <BR>#GS_COMPTYPE_PRORES Apple ProRes
	* <BR>#GS_COMPTYPE_BGRA_INVERT BGRA (TGA) vertically inverted
	* <BR>#GS_COMPTYPE_DPX_YCBCR10 DPX YCbCr 10 bit
	* <BR>#GS_COMPTYPE_ARGB ARGB
	* <BR>#GS_COMPTYPE_RGBA RGBA
	* <BR>#GS_COMPTYPE_ABGR Uncompressed A BGR - TIFF
	* <BR>#GS_COMPTYPE_BGRA Uncompressed BGR A - BMP/TGA
	* <BR>#GS_COMPTYPE_YCRCB_422 Uncompressed Y'CrCb 4:2:2 (DVS, VG)
	* <BR>#GS_COMPTYPE_YCRCB_422A Uncompressed Y'CrCb 4:2:2A (DVS, Dual VG)
	* <BR>#GS_COMPTYPE_YCRCB_444 Uncompressed Y'CrCb 4:4:4 (DVS, Dual VG)
	* <BR>#GS_COMPTYPE_YCRCB_444A Uncompressed Y'CrCb 4:4:4A (DVS, Dual VG)
	* <BR>#GS_COMPTYPE_STEREO Uncompressed Y'CrCb 4:4:4A (DVS, Dual VG) or 3D 8, 10, 30 or 32 bit
	* <BR>#GS_COMPTYPE_YCRCB_420 Uncompressed Y'CrCb 4:2:0
	* <BR>#GS_COMPTYPE_DPX_RGB10  DPX 10 bit rgb
	* <BR>#GS_COMPTYPE_ALT   Use as generic alternative for use through AVCodec
	* <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 kilobyte 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 kilobyte 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,
        *                       upper word is the type as defined in GS_SERIALPROTOCOLS
        *                       #GS_SERIALPROTOCOLS_SONY422, #GS_SERIALPROTOCOLS_ODETICS,
#GS_SERIALPROTOCOLS_VDCP,
        *                       #GS_SERIALPROTOCOLS_PROFILE, #GS_SERIALPROTOCOLS_HDCAMSR,
#GS_SERIALPROTOCOLS_NETSERIAL  << 8
        * \li cmdType::ctGetValue
        * <BR>MEDIACMD::dwPosition - Any unsigned short VTR ID - See Control key in registry docs and
LocalConfig.exe
        *                       upper word is the type as defined in GS_SERIALPROTOCOLS
        *                       #GS_SERIALPROTOCOLS_SONY422, #GS_SERIALPROTOCOLS_ODETICS,
#GS_SERIALPROTOCOLS_VDCP,
        *                       #GS_SERIALPROTOCOLS_PROFILE, #GS_SERIALPROTOCOLS_HDCAMSR,
#GS_SERIALPROTOCOLS_NETSERIAL  << 8
        * <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, SDTI, 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>
        */
        gsHDSDITransferType,            // Type of signal if using HDSDI bayer transfer
        /**
        * Clip list sort order, mostly for serial control
        * <BR>
        * \li cmdType::ctSetValue
```

```
	* <BR>MEDIACMD::dwPosition - Sort order (not all devices support all modes)
#GS_SORTORDER_FILENAME,
	*			#GS_SORTORDER_FILENAME_ASCENDING, #GS_SORTORDER_RECORDDATE,
#GS_SORTORDER_RECORDDATE_ASCENDING,
	*			#GS_SORTORDER_RENEWALDATE,
#GS_SORTORDER_RENEWALDATE_ASCENDING, #GS_SORTORDER_DURATION,
	*			#GS_SORTORDER_DURATION_ASCENDING, #GS_SORTORDER_TIMECODE,
#GS_SORTORDER_TIMECODE_ASCENDING
	* \li cmdType::ctGetValue
	* <BR>MEDIACMD::dwPosition - Sort order (not all devices support all modes)
#GS_SORTORDER_FILENAME,
	*			#GS_SORTORDER_FILENAME_ASCENDING, #GS_SORTORDER_RECORDDATE,
#GS_SORTORDER_RECORDDATE_ASCENDING,
	*			#GS_SORTORDER_RENEWALDATE,
#GS_SORTORDER_RENEWALDATE_ASCENDING, #GS_SORTORDER_DURATION,
	*			#GS_SORTORDER_DURATION_ASCENDING, #GS_SORTORDER_TIMECODE,
#GS_SORTORDER_TIMECODE_ASCENDING
	* <BR>  MEDIACMD::dwStart - Available sort modes
	* <HR>
	*/
	gsSortOrder,

	/**
	* 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
```

```
        /**
        * Color 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 not 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
        // Get current time code for each type
        /**
        * Get the current LTC (external input) time
        * <BR>
        * \li cmdType::ctSetValue - not supported
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - current time code
        * <BR> MEDIACMD::dwStart - current user bits
        * <BR> MEDIACMD::dwEnd - Current TC type
        * <HR>
        */
        gsGetTcLtc,
        /**
        * Get the current VITC (in vblank) time
        * <BR>
        * \li cmdType::ctSetValue - not supported
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - current time code
```

```
* <BR> MEDIACMD::dwStart - current user bits
* <BR> MEDIACMD::dwEnd - Current TC type
* <HR>
*/
gsGetTcVitc,
/**
* Get the current RP-188 time
* <BR>
* \li cmdType::ctSetValue - not supported
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - current RP-188 LTC time code
* <BR> MEDIACMD::dwStart - current RP-188 LTC user bits
* <BR> MEDIACMD::dwEnd - current RP-188 LTC TC type
* <BR> MEDIACMD::dwVideoChannels - current RP-188 VITC time code
* <BR> MEDIACMD::dwAudioChannels - current RP-188 VITC user bits
* <BR> MEDIACMD::dwInfoChannels - current RP-188 VITC TC type
* <HR>
*/
gsGetTcRP188,
/**
* Get the current RP-215 time and KEY
* <BR>
* \li cmdType::ctSetValue - not supported
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - current RP-215 LTC time code
* <BR> MEDIACMD::dwStart - current RP-215 LTC user bits
* <BR> MEDIACMD::dwEnd - current RP-215 LTC TC type
* <BR> MEDIACMD::dwVideoChannels - current RP-215 VITC time code
* <BR> MEDIACMD::dwAudioChannels - current RP-215 VITC user bits
* <BR> MEDIACMD::dwInfoChannels - current RP-215 VITC TC type
* <BR> MEDIACMD::lSpeed - Key length
* <BR> MEDIACMD::arbID - Key bytes
* <HR>
*/
gsGetTcRP215,
/**
* Get the current RP-215 time and INK
* <BR>
* \li cmdType::ctSetValue - not supported
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - current RP-215 LTC time code
* <BR> MEDIACMD::dwStart - current RP-215 LTC user bits
* <BR> MEDIACMD::dwEnd - current RP-215 LTC TC type
* <BR> MEDIACMD::dwVideoChannels - current RP-215 VITC time code
* <BR> MEDIACMD::dwAudioChannels - current RP-215 VITC user bits
* <BR> MEDIACMD::dwInfoChannels - current RP-215 VITC TC type
* <BR> MEDIACMD::lSpeed - Ink length
* <BR> MEDIACMD::arbID - Ink bytes
* <HR>
*/
gsGetTcRP215Ink,
        /**
        * Get the current IRIG time
        * <BR>
        * \li cmdType::ctSetValue - not supported
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - current IRIG time code
        * <BR> MEDIACMD::dwStart - current IRIG date/day
        * <BR> MEDIACMD::dwEnd - current IRIG TC type
        * <HR>
        */
        gsGetTcIRIG,
        /**
        * Get the current flags
        * <BR>
        * \li cmdType::ctSetValue - not supported
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - Active Format Description or -1
        * <BR> MEDIACMD::dwStart - Broadcast Flags or -1
        * <BR> MEDIACMD::dwEnd - Dolby Info or -1
        * <HR>
        */
        gsGetTcFlags,   // BCF AFD Dolby etc.
        /**
        * Record start type.  Record at time code, or from camera file naming source
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition - #GS_RECSTARTTYPE_NONE, #GS_RECSTARTTYPE_LTC,
#GS_RECSTARTTYPE_VITC, #GS_RECSTARTTYPE_TOD, #GS_RECSTARTTYPE_RED,
#GS_RECSTARTTYPE_ARRI, #GS_RECSTARTTYPE_SONY, #GS_RECSTARTTYPE_CANNON
        * <BR> MEDIACMD::dwStart - Time code or other flags
        * <BR> MEDIACMD::dwEnd - Optional duration
        * \li cmdType::ctGetValue
        * <BR> MEDIACMD::dwPosition - #GS_RECSTARTTYPE_NONE, #GS_RECSTARTTYPE_LTC,
#GS_RECSTARTTYPE_VITC, #GS_RECSTARTTYPE_TOD, #GS_RECSTARTTYPE_RED,
#GS_RECSTARTTYPE_ARRI, #GS_RECSTARTTYPE_SONY, #GS_RECSTARTTYPE_CANNON
        * <BR> MEDIACMD::dwStart - Time code or other flags
        * <BR> MEDIACMD::dwEnd - Optional duration
        * <HR>
        */
        gsRecordStartType, // Record auto start

        /*
        * 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 channel's 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 it 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 transferred
* \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 transferring
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - # of Clips transferring 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 transferring
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition - # of Clips transferring 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"Output 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 whatnot
        * <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::lSpeed - Top 32 bits of 64 bit 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::lSpeed - Top 32 bits of 64 bit pointer to the DTDraw structure to draw 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,
        /**
        * Load buffers (for opengl right now)
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  Load buffers now
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Get last load time
        * <HR>
        */
        gsMonitorLoadBuffers,
        /**
        * Change the wave format setup
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition - Set quality level -1 = auto, else 0~100
        * <BR>  MEDIACMD::dwStart - Set draw mode GS_WAVEVECTOR_WEIGHTED,
GS_WAVEVECTOR_MONO, GS_WAVEVECTOR_COLORIZE
        * <BR>  Load buffers now
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Get quality level
        * <HR>
        */
        gsWaveVectorQuality,
        /**
        * Get line/pixel data in 16 bit WORDs in SDI order
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition -
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Start pixel for return
        * <BR>  MEDIACMD::dwStart - Start line
        * <BR>  MEDIACMD::dwEnd - End line (exclusive)
        * <BR>  MEDIACMD::dwInfoChannels – End Pixel (width)
        * <BR>  MEDIACMD::arbID - Returned data as a series of WORDs in capture order
        * <BR>  MEDIACMD::lSpeed - Returned line pitch
        * <BR>  MEDIACMD::dwAudioChannels - Return data order and type GS_WAVEVECTOR_DATA_YCBCR,
GS_WAVEVECTOR_DATA_YCRCB, GS_WAVEVECTOR_DATA_RGBX, GS_WAVEVECTOR_DATA_GBRG
        * <HR>
        */
        gsWaveVectorGetData,
        /**
        * Save the current frame to a file on disk
        * <BR>
        * \li cmdType::ctSetValue
        * <BR>  MEDIACMD::dwPosition -
        * \li cmdType::ctGetValue
        * <BR>  MEDIACMD::dwPosition - Scale of image (percentage of original)
        * <BR>  MEDIACMD::arbID - format & target file name
        * <HR>
        */
        gsWaveVectorPreview,

        /** 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  #GS_DXRGB_OVERLAY
#GS_DXRGB_DIRECT #GS_DXYUV_OVERLAY #GS_DXYUV_DIRECT
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Allow DirectX
        * <BR>MEDIACMD::dwStart       - Bit Array for YUV, RGB and Overlay allow #GS_DXRGB_OVERLAY
#GS_DXRGB_DIRECT #GS_DXYUV_OVERLAY #GS_DXYUV_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 #GS_3DVGA_FLAG_SPLITVERT #GS_3DVGA_FLAG_LENTICULAR
        * <BR> #GS_3DVGA_FLAG_INVERT #GS_3DVGA_FLAG_FLIPLEFTVERT
#GS_3DVGA_FLAG_FLIPRIGHTVERT #GS_3DVGA_FLAG_FLIPLEFTHORIZ #GS_3DVGA_FLAG_FLIPRIGHTHORIZ
#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 #GS_3DVGA_FLAG_SPLITVERT #GS_3DVGA_FLAG_LENTICULAR
        * <BR> #GS_3DVGA_FLAG_INVERT #GS_3DVGA_FLAG_FLIPLEFTVERT
#GS_3DVGA_FLAG_FLIPRIGHTVERT #GS_3DVGA_FLAG_FLIPLEFTHORIZ #GS_3DVGA_FLAG_FLIPRIGHTHORIZ
#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                  - Wipe 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  - Mix 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                   - Vertical pixels to next line in grid
        * \li cmdType::ctSetValue
        * <BR>MEDIACMD::dwPosition  - Percentage size
        * <BR>MEDIACMD::dwStart       - Horizontal lines
        * <BR>MEDIACMD::dwEnd                   - Vertical 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,
	/**
	* Allow maximum number of channels to be set
	* <BR>
	* \li cmdType::ctGetValue
	* <BR>MEDIACMD::dwPosition  - # channels
	* \li cmdType::ctSetValue
	* <BR>MEDIACMD::dwPosition  - # channels
	* <HR>
	*/
	gsLimitAvailableChannels,
	/**
	* Zoom and pan the VGA (Overlay only right now) for bbReplay/Officiating<BR>
	* e.g. 1920<BR>
	* 1.0 = factor 1    1920<BR>
	* 2.0 = factor 655   960<BR>
	* 3.0 = factor 1310  640<BR>
	* 4.0 = factor 1965  240<BR>
	* <BR>
	* \li cmdType::ctGetValue
	* <BR>MEDIACMD::dwPosition  - Zoom level 1..65500 = 1..100
	* <BR>  MEDIACMD::dwStart - Pan X = 0..Width
	* <BR>  MEDIACMD::dwEnd - Pan Y = 0..Height
	* \li cmdType::ctSetValue
	* <BR>MEDIACMD::dwPosition  - Zoom level 1..65500 = 1..100
	* <BR>  MEDIACMD::dwStart - Pan X = 0..Width
	* <BR>  MEDIACMD::dwEnd - Pan Y = 0..Height
	* <HR>
	*/
	gsVGAZoomPan,
	/**
	* VEFXi 3D parameter set.  Immediate mode set (-1 position).  Get is always
	* immediate.  dwPosition can be set to set up parameters in the future.  Always
	* based on absolute frame count.
	* \li cmdType::ctGetValue
	* <BR>  MEDIACMD::arbID - dtVefxi3D structure from DTVideo.h
	* <BR>  MEDIACMD::dwStart - #GS_VEFXI3D_DISABLE, #GS_VEFXI3D_REALTIME,
#GS_VEFXI3D_PLAYBACK
	* \li cmdType::ctSetValue
	* <BR>  MEDIACMD::arbID - dtVefxi3D structure from DTVideo.h
	* <BR>  MEDIACMD::dwPosition - Frame this takes effect, -1 for immediate
	* <BR>  MEDIACMD::dwStart - #GS_VEFXI3D_DISABLE, #GS_VEFXI3D_REALTIME,
#GS_VEFXI3D_PLAYBACK
	* <HR>
	*/
	gsVEFXi3D,

	/**
	* 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 affected.  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 means it has 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 affected.  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 one 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
        /**
        * Get the last time a frame was drawn
        * <BR>
* \li cmdType::ctSetValue
* <BR>Not Supported
* \li cmdType::ctGetValue
* <BR>MEDIACMD::dwPosition - the current millisecond counter
* <BR>MEDIACMD::dwCmdAlt - the current millisecond counter
* <HR>
*/
gsLastDrawMs,                          // Last time a frame was drawn ms (dwPosition)

/*** 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,          // Save the spaces
        /**
        * Save the clip bin to disk using the current names
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - Optional new file name to save to
        * \li cmdType::ctGetValue
        * <BR> Not supported
        * <HR>
        */
        gsSaveClipSpaceToDisk,
        /**
        * Save the tcspace to disk using the current names
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::arbID - Optional new file name to save to
        * \li cmdType::ctGetValue
        * <BR> Not supported
        * <HR>
        */
        gsSaveTCSpaceToDisk,

        /**
        * 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 user's 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 user's 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 Ms
* <BR> MEDIACMD::dwStart              - frame Width
* <BR> MEDIACMD::dwEnd                         - Frame height
* <BR> MEDIACMD::lSpeed                        - Frame Pitch
* <BR> MEDIACMD::arbID                         - BYTE * for frame
*/
gsPreviewFrame,
/**
* Get a preview frame from AvHAL (usually via the network)
* This will be a full sized frame in whatever compression the board is set for
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition                    - Frame size
* <BR> MEDIACMD::arbID                         - Image Data
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition     - Frame Ms
* <BR> MEDIACMD::dwStart              - frame Width
* <BR> MEDIACMD::dwEnd                         - Frame height
* <BR> MEDIACMD::lSpeed                        - Frame Pitch
* <BR> MEDIACMD::arbID                         - BYTE * for frame
*/
gsCurrentFrame,
/**
* 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,

/**
*
* <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,

/**
* 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

	//
	// Commands for timeclock overlay

	/**
	* Set the Video Inlay enabled = 1 or disabled = 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::dwStart = This meld finished;
* <BR> MEDIACMD::dwEnd = This meld total;
* <BR> MEDIACMD::dwPosition                - Progress x / 1000
* <BR> MEDIACMD::dwInfoChannels            - is significant
* <BR> MEDIACMD::dwVideoChannels           - list item index if is significant
* <HR>
*/
gsExportClip,
/**
* Sets Export Directory
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID[0] = New directory Name;
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::arbID[0] = directory Name;
* <HR>
*/
gsExportClipDirectory,
/**
* Sets Export Audio File Source
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID[0] = New FileName
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::arbID[0] = FileName;
* <HR>
*/
gsExportAudioFileSourceName,
/**
* Sets Export FileName -> needs to be set before the export is sent.
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID[0] = New FileName
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::arbID[0] = FileName;
* <HR>
*/
gsExportFileName,
/**
* Sets Export Profile
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID[0] = New profile name NULL = system settings
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::arbID[0] = FileName;
* <HR>
*/
gsExportProfileName,
/**
* sets Export threshold limits
*
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition = thread threshold cpu;
* <BR> MEDIACMD::dwStart = thread sleep min;
* <BR> MEDIACMD::dwEnd = thread sleep max;
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition = thread threshold cpu;
* <BR> MEDIACMD::dwStart = thread sleep min;
* <BR> MEDIACMD::dwEnd = thread sleep max;
* <HR>
*/
gsExportThresholdLimits,
/**
* Sets Import Directory
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::arbID[0] = New directory Name;
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::arbID[0] = directory Name;
* <HR>
*/
gsImportClipDirectory,
/**
* Set File Length for segmentation support
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition = new segment length;
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition = segment length;
* <HR>
*/
gsFileSegmentSize,
/**
* Set Clip Info update frame rate
* <BR>
* \li cmdType::ctSetValue
* <BR> MEDIACMD::dwPosition = new update rate in frames;
*
* \li cmdType::ctGetValue
* <BR> MEDIACMD::dwPosition = update rate in frames;
* <HR>*/
```

```
	gsRecordFileUpdateFrames,

	/**
	* Set AvHal to monitor incoming frame rate to make ms adjustments
	* This always starts up as off, Hurricane sets this on on the dialog init
	* <BR>
	* \li cmdType::ctSetValue
	* <BR> MEDIACMD::dwPosition = on / off;
	*
	* \li cmdType::ctGetValue
	* <BR> MEDIACMD::dwPosition = not supported;
	* <HR>*/
	gsCheckFrameTimeStamp,

	//
	// 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
	/**
	* 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> MEDIACMD::cfFlags		- If 0x1A1A1A1A, call restart
	* <BR>
	* \li cmdType::ctGetValue Not supported
	* <BR>
	*/
	gsShutdownSystem,
	/**
	* Clean off the root, or the whole record drive
	* <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>
        */
        gsCleanRecordWipeDrive,
        /**
        * Initiate software install
        * <BR>
        * \li cmdType::ctSetValue
        * <BR> MEDIACMD::dwPosition              - Must be 0x2B2B2B2B
        * <BR> MEDIACMD::dwStart         - Must be 0x11111111
        * <BR> MEDIACMD::dwEnd                   - Must be 0x4E4E4E4E
        * <BR> MEDIACMD::arbID                   - Networkpath to install file (must have shared drive for
this)
        * <BR>
        * \li cmdType::ctGetValue Not supported
        * <BR>
        */
        gsInstallSystem,
        //
        // 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

//! Spelling error
#define gsAllowIndependantChanConfig        gsAllowIndependentChanConfig

//! 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
//@}

//! For cmdGetSetValue::gsVEFXi3D don't send commands
#define GS_VEFXI3D_DISABLE                  0
//! For cmdGetSetValue::gsVEFXi3D send command in next frame
#define GS_VEFXI3D_REALTIME                         1
//! For cmdGetSetValue::gsVEFXi3D send command when frame value comes up
#define GS_VEFXI3D_PLAYBACK                         2

// 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
//! For cmdGetSetValue::gsTcSource - Using irig natural (not converted) time code
#define GS_TCSOURCE_IRIG    8

//! 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

//! No CC cmdGetSetValue::gsCCSetup
#define GS_CC_DISABLE       0x0000
//! CC1 cmdGetSetValue::gsCCSetup
#define GS_CC_CC1           0x0001
//! CC2 cmdGetSetValue::gsCCSetup
#define GS_CC_CC2           0x0004
//! CC3 cmdGetSetValue::gsCCSetup
#define GS_CC_CC3           0x0008
//! CC4 cmdGetSetValue::gsCCSetup
#define GS_CC_CC4           0x0010
//! Text1 cmdGetSetValue::gsCCSetup
#define GS_CC_TEXT1         0x0020
//! Text2 cmdGetSetValue::gsCCSetup
#define GS_CC_TEXT2         0x0040
//! Text3 cmdGetSetValue::gsCCSetup
#define GS_CC_TEXT3         0x0080
//! Text4 cmdGetSetValue::gsCCSetup
#define GS_CC_TEXT4         0x0100
//! XDS cmdGetSetValue::gsCCSetup
#define GS_CC_XDS           0x0200
//! CEA 708 cmdGetSetValue::gsCCSetup
#define GS_CC_708           0x1000
//! CEA 708 cmdGetSetValue::gsCCSetup
#define GS_CC_SERVICE1      (0x0001 | GS_CC_708)
//! CEA 708 cmdGetSetValue::gsCCSetup
#define GS_CC_SERVICE2      (0x0004 | GS_CC_708)
//! CEA 708 cmdGetSetValue::gsCCSetup
#define GS_CC_SERVICE3      (0x0008 | GS_CC_708)
//! CEA 708 cmdGetSetValue::gsCCSetup
#define GS_CC_SERVICE4      (0x0010 | GS_CC_708)
```

```
/*
AFD '0000' indicates that information is not available and is undefined. Unless bar data is
available, DTV receivers and video equipment should interpret the active image area as being the
same as that of the coded frame.
AFD '0000', when accompanied by bar data, signals that the image's aspect ratio is narrower
than 16:9, but is not either 4:3 or 14:9. The bar data should be used to determine the extent of the
image.
AFD '0100', which should be accompanied by bar data, signals that the image's aspect ratio is
wider than 16:9, as is typically the case with widescreen features. The bar data should be used to
determine the height of the image.
Use of either '0010' or '0011' is not recommended in the ATSC television system. Values
'0001', '0101' through '0111', and '1100' are reserved.
*/
//! AFD cmdGetSetValue::gsGetTcFlags undefined
#define DTAFD_UNDEFINED                                                  0x0
// 0x1 reserved
// 0x2~0x3 not recommended
//! AFD cmdGetSetValue::gsGetTcFlags 0100 Aspect ratio greater than 16:9 (see below) Aspect ratio greater
than 16:9 (see below)
#define DTAFD_GREATER_THAN_16x9                                     0x4
// 0x5~0x7 reserved
//! AFD cmdGetSetValue::gsGetTcFlags '1000' 4:3 full frame image 16:9 full frame image
#define DTAFD_4x3_FULL_16x9_FULL                               0x8
//! AFD cmdGetSetValue::gsGetTcFlags '1001' 4:3 full frame image 4:3 pillarbox image
#define DTAFD_4x3_FULL_16x9_PILLAR                                  0x9
//! AFD cmdGetSetValue::gsGetTcFlags '1010' 16:9 letterbox image 16:9 full frame image
#define DTAFD_4x3_16LETTER_16x9_FULL                           0xA
//! AFD cmdGetSetValue::gsGetTcFlags '1011' 14:9 letterbox image 14:9 pillarbox image
#define DTAFD_4x3_14LETTER_16x9_14PILLAR                       0xB
// 0xC Reserved
//! AFD cmdGetSetValue::gsGetTcFlags '1101' 4:3 full frame image, alternative 14:9 center 4:3 pillarbox image,
alternative 14:9 center
#define DTAFD_4x3_FULL14CENTER_16x9_PILLAR14CENTER        0xD
//! AFD cmdGetSetValue::gsGetTcFlags '1110' 16:9 letterbox image, alternative 14:9 center 16:9 full frame
image, alternative 14:9 center
#define DTAFD_4x3_16LETTER15CENTER_16x9_FULL14CENTER      0xE
//! AFD cmdGetSetValue::gsGetTcFlags '1111' 16:9 letterbox image, alternative 4:3 center  16:9 full frame
image, alternative 4:3 center
#define DTAFD_4x3_16LETTER4CENTER_16x9_FULL4CENTER        0xF

//! RP-188 Ancillary time code video
#define GS_SOURCEPRECEDENCE_RP188_V     0x00000001
//! RP-188 Ancillary time code audio
#define GS_SOURCEPRECEDENCE_RP188_L     0x00000002
//! SMPTE LTC audio time code
#define GS_SOURCEPRECEDENCE_SMPTE       0x00000004
//! Time of day from computer clock (or GPS if available)
#define GS_SOURCEPRECEDENCE_TOD                 0x00000008
//! Time of day from VITC encoded line, or D-VITC in HD
#define GS_SOURCEPRECEDENCE_VITC        0x00000010
//! Time of day from IRIG RP-215 encode
```

```
#define GS_SOURCEPRECEDENCE_IRIG          0x00000020
//! Time of day from ILM style RP-215 A->LTC, V-VITC
#define GS_SOURCEPRECEDENCE_RP215         0x00000040
//! Use the record frame count
#define GS_SOURCEPRECEDENCE_FRAMECOUNT         0x10000000

//! Returns from cmdGetSetValue::gsTcClipInfo effect
#define GS_EDL_EFFECT                    0x00000001
//! Returns from cmdGetSetValue::gsTcClipInfo effect duration
#define GS_EDL_EFFECT_DUR  0x00000002
//! Returns from cmdGetSetValue::gsTcClipInfo comment
#define GS_EDL_COMMENT                   0x00000004
//! Returns from cmdGetSetValue::gsTcClipInfo edit number
#define GS_EDL_EDITNO                    0x00000008
//!
#define GS_EDL_FILENAME                  0x00000010

//! Set cmdGetSetValue::gsMetaDataReadWrite
#define METABASE_TYPE_UNKNOWN            0
//! Set cmdGetSetValue::gsMetaDataReadWrite
#define METABASE_TYPE_CHAR                         1
//! Set cmdGetSetValue::gsMetaDataReadWrite
#define METABASE_TYPE_INT                2

//! 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

//! Return RMS Levels with gsAudWavePeakRMS
#define GS_AUDIO_LEVEL_RMS                              0
//! Return Loudness Levels with gsAudWavePeakRMS
#define GS_AUDIO_LEVEL_LOUDNESS                 1

#define GS_EBU_MODE_NONE                        0
#define GS_EBU_MODE_MOMENTARY                   1
#define GS_EBU_MODE_SHORT_TERM                  2
#define GS_EBU_MODE_INTEGRATED                  3
#define GS_EBU_SCALE_9                                  0
```

```
#define GS_EBU_SCALE_18                                 1

//! 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 than 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 4x3 see #GS_ASPECT_RATIO_4x3
#define GS_MPEG_ASPECT_RATIO_4x3              GS_ASPECT_RATIO_4x3
//! MPEG aspect ratio 16x9 see #GS_ASPECT_RATIO_16x9
#define GS_MPEG_ASPECT_RATIO_16x9                     GS_ASPECT_RATIO_16x9
//! MPEG aspect ratio 2.21x1 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
//! Sony HDCAM SR MXF
#define VIDEOWRITETYPE_SONY_SR_MXF      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 - Panasonic 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 sends 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 - Panasonic 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 sends 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 sends YCbCr 8 unc to/from board, compression done in
MediaFile/PlugIn
#define VIDEOWRITETYPE_MP4                     0x02000000 //mpeg program/system
//! Flash video (264+mp3) - Note:  VVW sends 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_3840                     0x78    // Gap
#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))
// Quad HD
#define GS_SIGFORMSIZE_3840x2160  ((GS_SIGFORMSIZE_3840 << GS_SIGFORMSHIFT_HORIZONTAL) |
(GS_SIGFORMSIZE_2160 << GS_SIGFORMSHIFT_VERTICAL))
// 4K Quad
#define GS_SIGFORMSIZE_4096x2160  ((GS_SIGFORMSIZE_4096 << GS_SIGFORMSHIFT_HORIZONTAL) |
(GS_SIGFORMSIZE_2160 << 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 30) 3G Level B
#define GS_SIGFORM_1080_60_B                    (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_60)
//! 1920x1080P 59.94 (Dual 29.97) 3G Level B
#define GS_SIGFORM_1080_60X_B                   (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_59_94)
//! 1920x1080P 50 (Dual 25) 3G Level A
#define GS_SIGFORM_1080_50_B                    (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_INTERLACED | GS_SIGFORMFRAMERATE_50)

//! 1920x1080P 60 3G Level A
#define GS_SIGFORM_1080_60_A                    (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_60)
#define GS_SIGFORM_1080_60GS_SIGFORM_1080_60_A
//! 1920x1080P 59.94 3G Level A
```

```
#define GS_SIGFORM_1080_60X_A                (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_59_94)
#define GS_SIGFORM_1080_60X         GS_SIGFORM_1080_60X_A
//! 1920x1080P 50 3G Level A
#define GS_SIGFORM_1080_50_A                 (GS_SIGFORMSIZE_1920x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_50)
#define GS_SIGFORM_1080_50GS_SIGFORM_1080_50_A
//! 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)

#define GS_SIGFORM_DCIN_2048_30X(GS_SIGFORMSIZE_2048x1080 | GS_SIGFORMTYPE_PROGRESSIVE |
GS_SIGFORMFRAMERATE_29_97)
#define GS_SIGFORM_DCIN_2048_30         (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_30)
#define GS_SIGFORM_DCIN_2048_50         (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_50)
#define GS_SIGFORM_DCIN_2048_60X(GS_SIGFORMSIZE_2048x1080 | GS_SIGFORMTYPE_PROGRESSIVE |
GS_SIGFORMFRAMERATE_59_94)
#define GS_SIGFORM_DCIN_2048_60         (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_60)
#define GS_SIGFORM_DCIN_2048_48         (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_48)
```

```
/** 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)
#define GS_SIGFORM_DCIN_2048sf_30X         (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_29_97)
#define GS_SIGFORM_DCIN_2048sf_30          (GS_SIGFORMSIZE_2048x1080 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_30)

//! 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_2048_25            (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048_30X (GS_SIGFORMSIZE_2048x1556 | GS_SIGFORMTYPE_PROGRESSIVE |
GS_SIGFORMFRAMERATE_29_97)
//! gsGetSetValue::gsSignalFormat Film 2K
#define GS_SIGFORM_FILM_2048_30            (GS_SIGFORMSIZE_2048x1556 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_30)
```

```
//! 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 Quad HD 23.98 fps (4 x 1920x1080)
#define GS_SIGFORM_QUADHD_24X                               (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_23_98)
//! gsGetSetValue::gsSignalFormat Quad HD 23.98 sf fps (4 x 1920x1080)
#define GS_SIGFORM_QUADHDsf_24X                             (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_23_98)
//! gsGetSetValue::gsSignalFormat Quad HD 24 fps (4 x 1920x1080)
#define GS_SIGFORM_QUADHD_24                                (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Quad HD 24 sf fps (4 x 1920x1080)
#define GS_SIGFORM_QUADHDsf_24                              (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_24)
#define GS_SIGFORM_QUADHD_25                                (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
#define GS_SIGFORM_QUADHDsf_25                              (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_25)
#define GS_SIGFORM_QUADHD_30X                               (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_29_97)
#define GS_SIGFORM_QUADHDsf_30X                             (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_29_97)
#define GS_SIGFORM_QUADHD_30                                (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_30)
#define GS_SIGFORM_QUADHDsf_30                              (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_30)
#define GS_SIGFORM_QUADHD_48                                (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_48)
#define GS_SIGFORM_QUADHD_50                                (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_50)
#define GS_SIGFORM_QUADHD_60X                               (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_59_94)
#define GS_SIGFORM_QUADHD_60                                (GS_SIGFORMSIZE_3840x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_60)
//! gsGetSetValue::gsSignalFormat Quad 2K 23.98 fps (4 x 2048x1080)
#define GS_SIGFORM_4K_QUAD_24X                              (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_23_98)
//! gsGetSetValue::gsSignalFormat Quad 2K sf 23.98 fps (4 x 2048x1080)
#define GS_SIGFORM_4K_QUADsf_24X                    (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_23_98)
//! gsGetSetValue::gsSignalFormat Quad 2K 24 fps (4 x 2048x1080)
#define GS_SIGFORM_4K_QUAD_24                               (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_24)
//! gsGetSetValue::gsSignalFormat Quad 2K sf 24 fps (4 x 2048x1080)
#define GS_SIGFORM_4K_QUADsf_24                             (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_24)
#define GS_SIGFORM_4K_QUAD_25                               (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_25)
#define GS_SIGFORM_4K_QUADsf_25                             (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_25)
```

```
#define GS_SIGFORM_4K_QUAD_30X                          (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_29_97)
#define GS_SIGFORM_4K_QUADsf_30X                  (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_29_97)
#define GS_SIGFORM_4K_QUAD_30                           (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_30)
#define GS_SIGFORM_4K_QUADsf_30                         (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_SEGMENTEDFRAME | GS_SIGFORMFRAMERATE_30)
#define GS_SIGFORM_4K_QUAD_48                           (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_48)
#define GS_SIGFORM_4K_QUAD_50                           (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_50)
#define GS_SIGFORM_4K_QUAD_60X                          (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_59_94)
#define GS_SIGFORM_4K_QUAD_60                           (GS_SIGFORMSIZE_4096x2160 |
GS_SIGFORMTYPE_PROGRESSIVE | GS_SIGFORMFRAMERATE_60)
//! 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
//! Quad HD-SDI 3840 and 4096
#define GS_SIGFORM_SUPPORTS_QUADSDI       0x00001000              // 3840x2160
#define GS_SIGFORM_SUPPORTS_4K_QUAD 0x00002000                    // 4096x2160
//! 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)
//// 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
//! Panasonic 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

//! Not doing hd sdi transfer
#define GS_HDSDI_NONE                                     0x00000000
//! 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:  IO Industries
#define GS_HDSDIBAYER_IOI_SINGLE          0x00000010
//! Raw bayer HD-SDI:  IO Industries
#define GS_HDSDIBAYER_IOI_DOUBLE          0x00000020
//! 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)
```

```
//! HDCamSR SDTI
#define GS_HDSDTI_HDCAM_SR                              (GS_HDSDIBAYER_DUALLINKBIT | 0x00100000)

//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_FILENAME                           0x00000001
//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_FILENAME_ASCENDING                 0x00000002
//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_RECORDDATE                                 0x00000004
//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_RECORDDATE_ASCENDING   0x00000008
//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_RENEWALDATE                        0x00000010
//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_RENEWALDATE_ASCENDING 0x00000020
//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_DURATION                           0x00000040
//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_DURATION_ASCENDING                 0x00000080
//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_TIMECODE                           0x00000100
//! Sort order gsGetSetValue::gsSortOrder
#define GS_SORTORDER_TIMECODE_ASCENDING                 0x00000200

/** @{ */
/** 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 (cmdGetSetValue::gsSerialProtocols, cmdGetSetValue::gsVtrType)
#define GS_SERIALPROTOCOLS_SONY422              0x0001
//! Enable Odetics extensions (gsSerialProtocols)
#define GS_SERIALPROTOCOLS_ODETICS              0x0002
//! Enable VDCP Louth (cmdGetSetValue::gsSerialProtocols, cmdGetSetValue::gsVtrType)
#define GS_SERIALPROTOCOLS_VDCP                 0x0004
//! Enable Profile (cmdGetSetValue::gsSerialProtocols, cmdGetSetValue::gsVtrType)
#define GS_SERIALPROTOCOLS_PROFILE              0x0008
//! Enable HDCAMSR (cmdGetSetValue::gsSerialProtocols, cmdGetSetValue::gsVtrType)
#define GS_SERIALPROTOCOLS_HDCAMSR              0x0010
//! Enable network serial (cmdGetSetValue::gsSerialProtocols, cmdGetSetValue::gsVtrType)
#define GS_SERIALPROTOCOLS_NETSERIAL    0x1000

//! Auto record start (cmdGetSetValue::gsRecordStartType) none - don't do it
#define GS_RECSTARTTYPE_NONE                    0x00000000
//! Auto record start (cmdGetSetValue::gsRecordStartType) at dwPosition time from incoming LTC
#define GS_RECSTARTTYPE_LTC                             0x00000001
//! Auto record start (cmdGetSetValue::gsRecordStartType) at dwPosition time from incoming VITC
#define GS_RECSTARTTYPE_VITC                    0x00000002
//! Auto record start (cmdGetSetValue::gsRecordStartType) at dwPosition time from time of day
#define GS_RECSTARTTYPE_TOD                             0x00000004
//! Auto record start (cmdGetSetValue::gsRecordStartType) using red camera RP 188
#define GS_RECSTARTTYPE_RED                             0x00000008
//! Auto record start (cmdGetSetValue::gsRecordStartType) using arri camera RP 188
#define GS_RECSTARTTYPE_ARRI                    0x00000010
//! Auto record start (cmdGetSetValue::gsRecordStartType) using sony camera RP 188
#define GS_RECSTARTTYPE_SONY                    0x00000020
//! Auto record start (cmdGetSetValue::gsRecordStartType) using cannon XF and C300 (supports varicam)
camera RP 188
#define GS_RECSTARTTYPE_CANON                   0x00000040

//! 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 types mask
#define GS_WAVEAUDIO_MASK                               0x0000FE00
//! 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

#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

// General settings for all scopes
#define GS_WAVEVECTOR_FLAG_MASK                              0xFFF00000
#define GS_WAVEVECTOR_ALT_GRATICULE                0x40000000
#define GS_WAVEVECTOR_HIDE_GRATICULE   0x80000000
#define GS_WAVEVECTOR_HIDE_PICTURE                 0x00800000
#define GS_WAVEVECTOR_DRAWLINES                              0x00400000
// Vectorscope graticule settings
#define GS_WAVEVECTOR_HIDE75VECTOR                 0x01000000
#define GS_WAVEVECTOR_HIDE100VECTOR                0x02000000
#define GS_WAVEVECTOR_HIDEFLESHVECTOR0x04000000
#define GS_WAVEVECTOR_HIDEANGLES                   0x08000000
#define GS_WAVEVECTOR_HIDELUMASTICK                0x10000000
#define GS_WAVEVECTOR_HIDEMACBETH                  0x20000000
// Waveform graticule settings
#define GS_WAVEVECTOR_USESMPTESCALE                0x00000000
#define GS_WAVEVECTOR_USEFULLSCALE                 0x01000000
#define GS_WAVEVECTOR_WAVEG_10BIT                  0x02000000
#define GS_WAVEVECTOR_WAVEG_8BIT                   0x04000000
#define GS_WAVEVECTOR_WAVEG_IRE                              0x08000000
#define GS_WAVEVECTOR_WAVEG_MV                     0x10000000
// Picture settings
#define GS_WAVEVECTOR_PICT_CLEAN                   0x00000000
#define GS_WAVEVECTOR_PICT_SAFE                    0x01000000
#define GS_WAVEVECTOR_PICT_TITLE_SAFE   0x02000000

//! Weighted mode looks more like a traditional raster scope and expresses the number of pixels at a given value
by varying the brightness.
#define GS_WAVEVECTOR_WEIGHTED                     0x00000001
//! Mono displays every data point at full intensity, which can be useful in ensuring complete legality. With
weighted views it is possible to miss a small pixel region that is out of range
#define GS_WAVEVECTOR_MONO                                   0x00000002
//! Colorize replaces the traditional green in a scope with the actual color that it represents. For example, if
someone is wearing a bright red shirt, there will be a bright red streak where the scope renders those pixels
#define GS_WAVEVECTOR_COLORIZE                     0x00000004

//! Y Cb Y Cr
#define GS_WAVEVECTOR_DATA_YCBCR                   0x00000001
//! Y Cr Y Cb
#define GS_WAVEVECTOR_DATA_YCRCB                   0x00000002
//! Check 10 bit rgb order
#define GS_WAVEVECTOR_DATA_RGBX                              0x00000004
//! G B on first line, R G on the second
#define GS_WAVEVECTOR_DATA_GBRG                              0x00000010
//! G R on first line, B G on the second
#define GS_WAVEVECTOR_DATA_GRBG                              0x00000020
```

```
//! B G on first line, G R on the second
#define GS_WAVEVECTOR_DATA_BGGR                         0x00000040
//! R G on first line, G B on the second
#define GS_WAVEVECTOR_DATA_RGGB                         0x00000080

//! 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 images next to each other (squeeze horiz) (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_SIDEBYSIDE                             0x00000400
//! 3D VGA view arbitrary split (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_SPLIT                                  0x00000800
//! 3D VGA view squeeze 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 Checkerboard
#define GS_3DVGA_CHECKERBOARD                   0x00040000
//! 3D VGA Boxes (size based on mix)
#define GS_3DVGA_BOXES                                  0x00080000

//! 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_ADDGUIDE                  0x08000000
//! 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 left vertically (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_FLAG_FLIPLEFTVERT              0x40000000
//! 3D VGA view flag to flip right vertically (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_FLAG_FLIPRIGHTVERT             0x04000000
//! 3D VGA view flag to flip left horizontally (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_FLAG_FLIPLEFTHORIZ             0x20000000
//! 3D VGA view flag to flip right horizontally (for cmdGetSetValue::gsVga3DConfig)
#define GS_3DVGA_FLAG_FLIPRIGHTHORIZ    0x02000000
//! 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

//!     cmdGetSetValue::gsDTProjectToXml
#define GS_XML_FILENAME                                         1       //Set the XML File Name
//!     cmdGetSetValue::gsDTProjectToXml
#define GS_XML_CLIPFILE                                         2       //Set the project clipfilename
//!     cmdGetSetValue::gsDTProjectToXml
#define GS_XML_EDL                                              3       //Set the EDL name
//!     cmdGetSetValue::gsDTProjectToXml
#define GS_XML_STRING                                           4       //set a string in the project file
//!     cmdGetSetValue::gsDTProjectToXml
#define GS_XML_DWORD                                            5       //add a DWORD to the project
file
//!     cmdGetSetValue::gsDTProjectToXml
#define GS_XML_SAVE                                             6       //save the already open project
```

```
//!       cmdGetSetValue::gsDTProjectToXml
#define GS_XML_CHECK_OPEN                              7        //see if this project has conflicting EDLs
or clipfiles
//!       cmdGetSetValue::gsDTProjectToXml
#define GS_XML_OPEN_DELETE_FILES                 8        //Open the project while deleting same named
EDLs & clip logs
//!       cmdGetSetValue::gsDTProjectToXml
#define GS_XML_OPEN_IGNORE_FILES                 9        //Open the project while leaving same named
EDLs & clip logs
//!       cmdGetSetValue::gsDTProjectToXml
#define GS_ERROR_FILE_EXISTS                           -2       //Returned when a conflict occurs
(GS_XML_CHECK_OPEN)

//! 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
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_BBREPLAY                     0x00000040
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_COURTSIDECOACH          0x00000080
//! For cmdGetSetValue::gsApplicationID set to no specific application
#define GS_APP_HURRICANE               0x00000100

#define GS_SHUTDOWNAPPLICATION_POSITION 0x01010101
#define GS_SHUTDOWNAPPLICATION_START 0xA5A5A5A5
#define GS_SHUTDOWNAPPLICATION_END 0x5F5F5F5F

#define GS_SHUTDOWNSYSTEM_RESTART 0xA5A5A5A4
#define GS_SHUTDOWNSYSTEM_POSITION 0x11111111
#define GS_SHUTDOWNSYSTEM_START 0x5F5F5F5F
#define GS_SHUTDOWNSYSTEM_END 0xA5A5A5A5

#define GS_CLEANRECORDWIPE_ROOTDIR 0x0
#define GS_CLEANRECORDWIPE_WHOLEDRIVE 0x1
#define GS_CLEANRECORDWIPE_START 0x5F5F5F5F
#define GS_CLEANRECORDWIPE_END 0xA5A5A5A5

#define GS_INSTALLSYSTEM_POSITION 0x2B2B2B2B
#define GS_INSTALLSYSTEM_START 0x11111111
#define GS_INSTALLSYSTEM_END 0x4E4E4E4E
```

```
//! 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) as well as bayer types
#define GS_BITCOUNT_12                    0x20    /* 4:2:0 or bayer */
//! Supports Bayer types
#define GS_BITCOUNT_14                    0x40
//! Supports Bayer types
#define GS_BITCOUNT_16                    0x80

#define GS_FRAMEDROPMODE_NONE                             0x000000
//! Mask for varicam target frame rates
#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_VARICAM_MASK_TYPE 0x00F000
//! Original varicam
#define GS_FRAMEDROPMODE_VARICAM_PANASONIC 0x001000
//! Red frame valid varicam
#define GS_FRAMEDROPMODE_VARICAM_RED          0x002000
//! Userbit
#define GS_FRAMEDROPMODE_VARICAM_ARRI                     0x003000
//! Unkown
#define GS_FRAMEDROPMODE_VARICAM_SONY                     0x004000
//! 1080 varicam Cannon C300 (1080)
#define GS_FRAMEDROPMODE_VARICAM_CANNON                   0x005000
#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

#define GS_USE_THRESHOLD_DEFAULT                             0x00000000
#define GS_USE_THRESHOLD_PROCESS                             0x00000001
#define GS_USE_THRESHOLD_THREAD                              0x00000002
#define GS_USE_THRESHOLD_CYCLE_TIME                          0x00000004
#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 __x86_64__
        /** 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 structure 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 ever 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 __x86_64__
//#pragma comment(user, "MediaCMD is 32 bits")
//! 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
//#pragma comment(user, "MediaCMD is 64 bits")
#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>
<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>
  <BR> The structure:   MEDIACMD
  <BR> The commands:    #cmdType
  <BR> The Flags:       #cmdFlags
  <BR> Channels:        #cmdVidChan, #cmdAudChan, #cmdinf
  <BR> GetSet Commands  #cmdGetSetValue
  <BR>      Basic           cmdGetSetValue::gsTc
  <BR>      Clip            cmdGetSetValue::gsGetNextClip
  <BR>      Channel         cmdGetSetValue::gsAudChan
  <BR>      Audio           cmdGetSetValue::gsAudInSelect
  <BR>      Video General   cmdGetSetValue::gsVidFreeze
  <BR>      Video Input     cmdGetSetValue::gsVidInSelect
  <BR>      Video TBC       cmdGetSetValue::gsVidSetup
  <BR>      Video Output    cmdGetSetValue::gsVidOutSelect
  <BR>      Signal Type     cmdGetSetValue::gsSignalFormat
  <BR>      Compression     cmdGetSetValue::gsCompType
  <BR>      Storage         cmdGetSetValue::gsTotalStorageAvail
  <BR>      Control         cmdGetSetValue::gsLocal
  <BR>      Info            cmdGetSetValue::gsVVWVersion
  <BR>      Internal        cmdGetSetValue::gsSetHwnds
  <BR>      Mode+Info       cmdGetSetValue::gsChannelExist
  <BR>      Returns         #GS_NOT_SUPPORTED
  <BR>  Declaration Local   #DECLARE_MEDIACMD
  <BR>  Init Local          #INIT_MEDIACMD
  <BR>  Init Pointer        #INIT_PMEDIACMD
  <BR>  Sizes               #SIZEOF_MEDIACMD_BASE
  <BR>
</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 <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>

```
MEDIACMD::dwStructSize = 28<BR>
MEDIACMD::ctCmd = #ctPlay<BR>
<BR>
\b Simple \b Pause <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 28<BR>
MEDIACMD::ctCmd = #ctPause<BR>
<BR>
\b Simple \b Stop <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 28<BR>
MEDIACMD::ctCmd = #ctStop<BR>
<BR>
\b Play \b From-To <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctPlay<BR>
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd<BR>
MEDIACMD::dwStart = 1800           // 1 minute in frames NTSC NDF<BR>
MEDIACMD::dwEnd = 2100                   // 1 minute 10 seconds NTSC NDF<BR>
<BR>
\b Play \b DMC \b (Play \b at \b speed) <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 36<BR>
MEDIACMD::ctCmd = #ctPlay<BR>
MEDIACMD::cfFlags = #cfUseSpeed<BR>
MEDIACMD::lSpeed = 32760                 // Half play speed forward<BR>
<BR>
\b Record \b Edit <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctRecord<BR>
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd | #cfUsePresets<BR>
MEDIACMD::dwVideoChannels = 0x01         // Record V<BR>
MEDIACMD::dwAudioChannels = 0x03         // Record A1 & A2 (AA)<BR>
MEDIACMD::dwInfoChannels = 0x00<BR>
MEDIACMD::dwStart = 3000<BR>
MEDIACMD::dwEnd = 3200<BR>
<BR>
\b Play \b Two \b Segments <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctPlay<BR>
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd<BR>
MEDIACMD::dwStart = 2200<BR>
MEDIACMD::dwEnd = 2500<BR>
<BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctPlay<BR>
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd | #cfDeferred<BR>
```

```
MEDIACMD::dwStart = 5300<BR>
MEDIACMD::dwEnd = 5450<BR>
<BR>
\b Play \b A \b Named \b Clip <BR>
Char szName = "C:\Adir\Afile.OMF"<BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60 + strlen(szName)<BR>
MEDIACMD::ctCmd = #ctPlay<BR>
MEDIACMD::cfFlags = #cfUseClipID<BR>
strcpy( MEDIACMD::arbID , szName)<BR>
<BR>
\b Seek \b To \b 1 \b Minute <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 48<BR>
MEDIACMD::ctCmd = #ctPause<BR>
MEDIACMD::cfFlags = #cfUsePosition<BR>
MEDIACMD::dwPosition = 1800            // Frames NTSC NDF<BR>
<BR>
\b Step \b Reverse \b 1 \b Frame <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 48<BR>
MEDIACMD::ctCmd = #ctPause<BR>
MEDIACMD::cfFlags = #cfUsePositionOffset<BR>
MEDIACMD::dwPosition = (DWORD) -1<BR>
<BR>
\b Record \b A \b 1 \b Minute \b Named \b Clip <BR>
Char szName = "C:\Adir\ArecordFile.MOV"<BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60 + strlen(szName)<BR>
MEDIACMD::ctCmd = #ctRecord<BR>
MEDIACMD::cfFlags = #cfUseClipID | #cfUseEnd<BR>
MEDIACMD::dwEnd = 1800<BR>
strcpy( MEDIACMD::arbID , szName)<BR>
 <BR>
\b Record \b An \b Odetics \b or \b Louth \b Clip \b (ext) <BR>
(Note:  The stop command is required for accuracy on some DDRs)<BR>
Char szName = "THISCLIP"       // Max 8 char<BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60 + strlen(szName)<BR>
MEDIACMD::ctCmd = #ctStop              // Record ready<BR>
MEDIACMD::cfFlags = #cfUseClipID<BR>
strcpy( MEDIACMD::arbID , szName)<BR>
<BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60 + strlen(szName)<BR>
MEDIACMD::ctCmd = #ctRecord<BR>
MEDIACMD::cfFlags = #cfUseClipID<BR>
strcpy( MEDIACMD::arbID , szName)<BR>
<BR>
\b Eject \b The \b Current \b Media <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
```

```
MEDIACMD::dwStructSize = 28<BR>
MEDIACMD::ctCmd = #ctEject<BR>
<BR>
\b Setup \b The \b Video \b To \b Nominal <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctSetValue<BR>
MEDIACMD::dwCmdAlt = gsVidSetup<BR>
MEDIACMD::dwStart = 0<BR>
<BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctSetValue<BR>
MEDIACMD::dwCmdAlt = gsVidVideo<BR>
MEDIACMD::dwStart = 0<BR>
<BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctSetValue<BR>
MEDIACMD::dwCmdAlt = gsVidHue<BR>
MEDIACMD::dwStart = 0<BR>
<BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctSetValue<BR>
MEDIACMD::dwCmdAlt = gsVidChroma<BR>
MEDIACMD::dwStart = 0<BR>
<BR>
\b Check \b The \b Device \b Type <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctGetValue<BR>
MEDIACMD::dwCmdAlt = gsVtrType<BR>
MEDIACMD::dwStart = 0<BR>
(Returns:  VTR/DDR/Media type in .dwStart)<BR>
<BR>
\b Change \b The \b Default \b TC \b Type \b To \b VITC <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctSetValue<BR>
MEDIACMD::dwCmdAlt = gsVitcTc<BR>
MEDIACMD::dwStart = GS_DEFAULT<BR>
<BR>
\b Transfer \b From \b External \b VTR \b To \b Internal \b Channel <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctTransfer<BR>
MEDIACMD::cfFlags = #cfUsePosition | #cfUseStart | #cfUseEnd | #cfUsePresets<BR>
MEDIACMD::dwVideoChannels = 0x01          // Capture Video<BR>
MEDIACMD::dwAudioChannels = 0x00          // No Audio<BR>
MEDIACMD::dwInfoChannels = 0x00<BR>
```

```
MEDIACMD::dwCmdAlt = hExternalChannel       // The VTR channel<BR>
MEDIACMD::dwPosition = 2100                              // Where to record to<BR>
MEDIACMD::dwStart = 10850                                // Source In on VTR<BR>
MEDIACMD::dwEnd = 11000                                           // Source Out on VTR<BR>
<BR>
\b Insert \b Media \b From \b File \b Over \b Current <BR>
Char szName = "C:\Adir\ArecordFile.MOV"<BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60 + strlen(szName)<BR>
MEDIACMD::ctCmd = #ctInsert<BR>
MEDIACMD::cfFlags = #cfUsePosition | #cfUseStart | #cfUseEnd | #cfUseClipID<BR>
MEDIACMD::dwPosition = 2100                   // Insert @ on target channel<BR>
MEDIACMD::dwStart = 0                                    // Start on source file<BR>
MEDIACMD::dwEnd = 180                                    // End on source file<BR>
strcpy( MEDIACMD::arbID , szName)    // Name of source file<BR>
<BR>
\b Delete \b A \b Section \b Of \b Media \b (No \b Hole) <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60<BR>
MEDIACMD::ctCmd = #ctDelete<BR>
MEDIACMD::cfFlags = #cfUseStart | #cfUseEnd | #cfRipple      // Remove #cfRipple to leave hole<BR>
MEDIACMD::dwStart = 140500                               // Start on target media<BR>
MEDIACMD::dwEnd = 150010                                 // End on target media<BR>
<BR>
\b Trim \b A \b Clip <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 60 <BR>
MEDIACMD::ctCmd = #ctTrim<BR>
MEDIACMD::cfFlags = #cfUsePosition | #cfUseStart | #cfUseEnd <BR>
MEDIACMD::dwPosition = 2100                              // Clip @ position<BR>
MEDIACMD::dwStart = +32                                  // Clip 32 frames from start<BR>
MEDIACMD::dwEnd = (DWORD) -12                            // Clip 12 frame from end<BR>
 <BR>
\b Terminate \b Session \b (Restart \b At \b Default) <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 28<BR>
MEDIACMD::ctCmd = #ctTerminate<BR>
<BR>
\b Abort \b Current \b Command <BR>
MEDIACMD::dwCmdID = #MEDIACMD_CURRENT<BR>
MEDIACMD::dwStructSize = 28<BR>
MEDIACMD::ctCmd = #ctAbort<BR>
<BR>
```

* \section mediacmdsamplertns MEDIACMD Sample Returns
 Sample Returns

```
MEDIACMD::dwCmdID == #MEDIACMD_CURRENT <BR>
MEDIACMD::dwStructSize == #SIZEOF_MEDIACMD <BR>
MEDIACMD::ctCmd == cmdType::ctPlay <BR>
MEDIACMD::cfFlags == 0 <BR>
```

\b Normal \b Play \b (100% \b Play \b Speed) <BR>

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

* \section mediacmdddrsetup MEDIACMD DDR Setup
 DDR Setup

DDR setup is achieved using cmdType::ctGetValue and cmdType::ctSetValue. Basically, the cmdType::ctGetValue will
return the current setting in MEDIACMD::dwPosition and the bit wise available settings in MEDIACMD::dwStart. To
change the setting, send one of the supported bit wise settings in MEDIACMD::dwPosition using cmdType::ctSetValue.
Here is a list of the main setup commands:<br>
\li Video Input - cmdGetSetValue::gsVidInSelect <br>settings: GS_VIDSELECT_???
\li Main Signal Format - cmdGetSetValue::gsSignalFormat <br>settings: GS_SIGFORM_???
\li File Format - cmdGetSetValue::gsVideoEncodeFormat <br>settings: VIDEOWRITETYPE_??? <br>note: When you change the file format
you will need to re populate the compression types and the bit depths. The compression type may no longer exist
in the new file type, but the DDR will pick a new default in this case
\li Compression - cmdGetSetValue::gsCompType <br>settings: GS_COMPTYPE_??? <br>note: When you change the compression type, you
will need to re-populate the bit depths, as they will have changed
\li Bit Depth/Count - cmdGetSetValue::gsCompChBitCount <br>settings: GS_BITCOUNT_#
\li HDSDI Transfer/Camera - cmdGetSetValue::gsHDSDITransferType <br>settings: GS_HDSDIBAYER_???
\li Analog Monitor/Up/Down Convert - gsVidAnalogMonitorMethod <br>settings: GS_ANALOGMONITORMETHOD_???
\li Up Convert HD Type - cmdGetSetValue::gsVidAnalogMonitorHDType <br>settings: GS_VIDSELECT_???
\li Down Convert SD Type - cmdGetSetValue::gsVidAnalogMonitorSDType <br>settings: GS_VIDSELECT_???
\li Set # Audio Channels - cmdGetSetValue::gsAudChan <br>settings: Bitwise audio channels
\li Audio Input - cmdGetSetValue::gsAudInSelect <br>settings: GS_AUDSELECT_???
\li Audio Bits - cmdGetSetValue::gsAudInputBitRate <br>settings: 16, 20, 24, 32
\li Audio File Type - cmdGetSetValue::gsAudioEncodeFormat <br>settings: AUDIOWRITETYPE_
\li Audio Monitor Pair - cmdGetSetValue::gsAudMonitorSelect <br>settings: Bitwise audio pair
\li Enable/Disable Genlock/Reference - cmdGetSetValue::gsVidOutGenlock <br>settings: 0/1
\li Select Genlock/Reference Source - cmdGetSetValue::gsVidOutGenlockSource <br>settings: GS_LOCKSRC_???
 */

/*! \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 - 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).

 * \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<br>
Type: Microsoft ActiveX/COM access method<br>
Pros: Easy to program, 1:1 relationship with QuickClip/XO interface.<br>
Cons: Uses same config as QuickClip/XO. Requires a local copy of QuickClip.<br>
Setup: Register VVW.DLL using RegSvr32.exe in the QuickClip installation directory.<br>
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.<br>
<br>
Direct Link<br>
Type: Direct link to VVW.DLL<br>
Pros: No ActiveX layer, code compatible with Linux, Irix, Mac OS-X.<br>
Cons: Uses default config from QuickClip/XO, application must be run in QuickClip
directory. Requires a local copy of QuickClip.<br>
Setup: Link to vvw.lib, include vvw.h. Copy application into the QuickClip directory
before running<br>
Issues: Needs access to VVW.dll and all its support DLLs/D1Xs. Still needs to be setup by
LocalConfig, DDRConfig or QuickClip/XO<br>
<br>
HTTP Ajax/SOAP<br>
Type: XML command interchange with the DDR internal HTTP server, optionally wrapped as SOAP<br>
Pros: Standard, multi language, multi platform.<br>
Cons: Not as fast or efficient as direct connection.<br>
Setup: Standard on version 3 and 4 DrasticDDRs<br>
Issues: Soap is optional, contact Drastic<br>
<br>
Command Line App<br>
Type: Network connection, command line sender<br>
Pros: Standard, multi platform (windows, os-x, linux).<br>
Cons: Single command or command list, not interactive.<br>
Setup: Downloadable.<br>

Issues: Command line.<br>
<br>
Network DLL<br>
Type: Direct line to vvwNet2.dll<br>
Pros: Consistent interface between local/remote and various OSs. Does not require a local copy of QuickClip.<br>
Cons: Requires vvwNet2.dll and support dlls<br>
Setup: Link to vvwNet2.lib, include vvwNet2.h. Copy dll set from SDK/bin directory with your application<br>
Issues: Use the netOpenLocal function to avoid QuickClip configuration issues. Requires a few DLLs 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.<br>
<br>
Network Direct<br>
Type: Direct compile of network sources in your app or your DLL.<br>
Pros: No extra DLLs. Easy to customize and modify. Lots of commands already written.<br>
Cons: Your app needs to handle setup and may need to run QuickClip.exe/VVWServer.exe/QCRun.exe.<br>
Setup: Copy source files from vvwNet2 into your project, modify and compile<br>
Issues: Does not run the client software automatically, so your application may need to start it, depending on what your application is doing.<br>
<br>
Manual<br>
Type: Use the structures and defines to write your own communication and control layer.<br>
Pros: This is required if you are using an unsupported development platform like PHP.<br>
Cons: Everything has to be built and tested from the ground up.<br>
Setup: None.<br>
Issues: Unless you absolutely have to, this method is not recommended.<br>
<br>

 * \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 - 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
    o /ActiveX - Examples that use the ActiveX control
    o /Direct - Examples that link directly to DLLs
    o /Java - Java based examples
    o /HTTP - AJAX based examples (must use QuickClip HTTP server to run)

 * \section maindoclink_sec Main Documentation Links

*** PDF version of the MediaCmd Documentation

```
http://www.drastic.tv/images/sdk/mediacmd/mediacmd4api.pdf

 * \section lowlevellink_sec Low Level Header Documentation Links

*** Doxygen PDF file version of the main MediaCmd headers
http://www.drastic.tv/images/sdk/mediacmd/mediacmd4apilowlevel.pdf

 * \section htmlajaxlink_sec HTTP XML AJAX Documentation Links

*** PDF for HTTP/AJAX and PHP MediaCmd
http://www.drastic.tv/images/sdk/mediacmd/mediacmd4ajaxhttp.pdf

 * \section commandlinelink_sec Command Line Documentation Links

*** PDF for Commmand Line MediaCmd
http://www.drastic.tv/images/sdk/mediacmd/mediacmd4cmdline.pdf

 * \section olderapi_sec Older Version 3 documentes

http://www.drastic.tv/images/sdk/mediacmd/mediacmd3api.pdf
http://www.drastic.tv/images/sdk/mediacmd/mediacmd3api.chm
 *
 *
 */

/**
' Licensing Flags
*/
#endif //_MEDIACMD_INCLUDED_H
```

Appendix III – VvwTypes.h

```
/***********************************************************************
 * $Id$:
 *
 * $HeadURL$:
 * $Author$:
 * $Revision$:
 * $Date$:
 *
 * Copyright (c) 1998-2023 Drastic Technologies Ltd.  All Rights Reserved.
 * 523 The Queensway, Suite 201 Toronto ON M8V 1Y7
 * phone (416) 255 5636  fax (416) 255 8780
 * engineering@drastictech.com      http://www.drastic.tv
 ***********************************************************************/

#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 metadata
* 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 field one (uses 2 bytes)
#define FRAMEINFO_DATA_F1_EIA608		0x00000001
	//! Data is EIA-608B SD closed caption data field two (uses 2 bytes)
#define FRAMEINFO_DATA_F2_EIA608		0x00000002
	//! Data is EIA-708 HD closed caption data (uses remaining bytes = minus the above)
#define FRAMEINFO_DATA_EIA708			0x00000100
	//! Data is RP-215 KLV data
#define FRAMEINFO_DATA_RP215_KLV		0x00001000
	//! Data is raw line with info per SMPTE 436 VBI Frame Element
#define FRAMEINFO_DATA_VBI_FRAME_ELEM	0x0004000
	//! Data is ARRI T-Link header
#define FRAMEINFO_DATA_ARRI_HEADER				0x0010000
	//! Frame includes ARRI T-Link header
#define FRAMEINFO_FRAME_INCLUDES_ARRI_HEADER			0x0020000
	/** 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  8192
        /** 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.
        * Version 4 Mark 2:  Need enough room for VBI FRAME ELEMENT data from
        * 436.  Max we will allow is 3 lines of 10 bit hd 1920x1080.  The
        * mapping is always Y only, so worst case is 1920 samples / 3 samples
        * per dword (at 10 bit) * 4 bytes per dword = 7680 bytes.  Round
        * up to 8K
        */
        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 definitely 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 definitely 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 transferring channel to
* get information on the other channel as quickly as possible.
* It was designed to operate efficiently 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 dependent 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 (not #DFRAME_TYPE_UNCTYPE_MASK):
        * #DFRAME_TYPE_RECORD, #DFRAME_TYPE_PLAY, #DFRAME_TYPE_PAUSE,
        * #DFRAME_TYPE_UNCOMPRESSED, #DFRAME_TYPE_UNCTYPE_MASK,
#DFRAME_TYPE_UNCTYPE_YCBCR8,
        * #DFRAME_TYPE_UNCTYPE_YUY2, #DFRAME_TYPE_UNCTYPE_V210, #DFRAME_TYPE_UNCTYPE_BGR,
        * #DFRAME_TYPE_UNCTYPE_BGRA, #DFRAME_TYPE_UNCTYPE_DPX10
        * #FRAME_TYPE_NOTPHYSHEAP, #DFRAME_TYPE_INCLUDES_HEADER
        * #DFRAME_TYPE_AUDIO, #DFRAME_TYPE_VIDEO,
        * #DFRAME_PROGRESSIVE, #DFRAME_FIELD_INVERT, #DFRAME_TIME_INVERT,
#DFRAME_ORIENTATION_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 acquired 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 frame has already been decompressed.  If avhal is in a compressed mode, bypass the decompression.
see DFRAME::dwType
#define DFRAME_TYPE_UNCOMPRESSED                0x00000008
//! Uncompressed type mask, if DFRAME_TYPE_UNCOMPRESSED is true see DFRAME::dwType
```

```
#define DFRAME_TYPE_UNCTYPE_MASK                0x000000F0
//! Uncompressed YCbCr 8 bit UYVY/yuv2 see DFRAME::dwType
#define DFRAME_TYPE_UNCTYPE_YCBCR8              0x00000000
//! Uncompressed YUY2 8 bit see DFRAME::dwType
#define DFRAME_TYPE_UNCTYPE_YUY2                0x00000010
//! Uncompressed V210 10 bit YCbCr see DFRAME::dwType
#define DFRAME_TYPE_UNCTYPE_V210                0x00000020
//! Windows BGR 8 bit see DFRAME::dwType
#define DFRAME_TYPE_UNCTYPE_BGR                 0x00000030
//! Windows BGRA 8 bit see DFRAME::dwType
#define DFRAME_TYPE_UNCTYPE_BGRA                0x00000040
//! DPX 10 bit RGB see DFRAME::dwType
#define DFRAME_TYPE_UNCTYPE_DPX10               0x00000050
//! HD-SDI Arri D21 bayer double buffer
#define DFRAME_TYPE_UNCTYPE_ARRID21             0x00000060
//! HD-SDI Arri Alexa bayer double buffer
#define DFRAME_TYPE_UNCTYPE_ARRIALEXA  0x00000070
//! 3D Stereo, first frame/right eye inverted vertically
#define DFRAME_TYPE_RIGHT1ST_INVERT_VERT        0x00000100
//! 3D Stereo, first frame/right eye inverted horizontally
#define DFRAME_TYPE_RIGHT1ST_INVERT_HORIZ       0x00000200
//! 3D Stereo, second frame/left eye inverted vertically
#define DFRAME_TYPE_LEFT2ND_INVERT_VERT                 0x00000400
//! 3D Stereo, second frame/left eye inverted horizontally
#define DFRAME_TYPE_LEFT2ND_INVERT_HORIZ        0x00000800
//! This indicates PhysHeap did NOT allocate the dframe (not pheap or local heap) see DFRAME::dwType
#define DFRAME_TYPE_NOTPHYSHEAP                 0x00001000
//! This indicates the header is includes with the video data (e.g. ARRI from Alexa) see DFRAME::dwType
#define DFRAME_TYPE_INCLUDES_HEADER             0x00002000
//! This indicates that FRAME_INFO::dwVITCAux points to a data area and FRAME_INFO::dwCCData contains its
size see them for more info see DFRAME::dwType
#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
        // 0x00040000
        // 0x00080000
//! 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
//! AUDIO:  The frame contains a large chunk of audio (allows for optimization in AvHAL) see DFRAME::dwType
//#define DFRAME_LARGE_AUDIO                            0x01000000
//! VIDEO 3D:  If the codec (e.g. cineform) has flipped the eyes, don't reflip
#define DFRAME_TYPE_EYES_ARE_FLIPPED    0x01000000
        // 0x02000000
```

```
//! The frame is from the previous gop.  Important when seeking in open gop mpeg
#define DFRAME_PREGOP_FRAME                                   0x04000000
//! 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 independent of other frames for decode see DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME                        0x10000000
//! This frame is independent of other frames for decode (an MPEG I Frame) see DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_I                      0x10000000
//! 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
//! This frame requires previous keyframe(s) (for MPEG a P Frame) see DFRAME::dwType
#define DFRAME_TYPE_KEYFRAME_P                      0x80000000
//! Mask for key types
#define DFRAME_TYPE_KEYFRAME_MASK                   (DFRAME_TYPE_KEYFRAME_I|
DFRAME_TYPE_KEYFRAME_B|DFRAME_TYPE_KEYFRAME_P)
        //! 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 driver's 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 played 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 watermarked.
#define _PDFRAMEFLAGS_IMPOSED_MARKED  0x00000008     // End of clip
        //! This frame needs to be made black (default frame) in MediaFile
```

```
#define _PDFRAMEFLAGS_PLEASE_BLACK                    0x00000080
        //! Read only valid frames (HDR dropped) (default frame) in MediaFile
#define _PDFRAMEFLAGS_ENSURE_VALID                    0x00000100
        //! 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)
        //! Used in dwFlags - dup above, should condense at some point
#define DT_TOP_FIELD                   0x10000000
        //! Used in dwFlags - dup above, should condense at some point
#define DT_BOTTOM_FIELD                0x20000000
        //! Used in dwFlags - dup above, should condense at some point
#define DT_FRAME                       0x30000000
//! This frame contains video data from a different channels input source see DFRAME::dwType
#define _PDFRAMEFLAGS_LIVE_VIDEO                      0x40000000
        /**
        * 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 specific
                // 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 efficient 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
        //! Is this file type currently recording
#define DPOSSIZENAME_RECORDING                    0x00000004
        //! This frame needs to be made black (default frame) in MediaFile
#define DPOSSIZENAME_PLEASE_BLACK                 _PDFRAMEFLAGS_PLEASE_BLACK        //
        0x00000080
        //! 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_AUDIO_MASK                   (DPOSSIZENAME_MONO_AUDIO_FRAME|
DPOSSIZENAME_STEREO_AUDIO_FRAME|DPOSSIZENAME_STEREO_AUDIO_FRAME|
DPOSSIZENAME_QUAD_AUDIO_FRAME|DPOSSIZENAME_4_1_AUDIO_FRAME|
DPOSSIZENAME_5_1_AUDIO_FRAME| DPOSSIZENAME_7_1_AUDIO_FRAME|
DPOSSIZENAME_9_1_AUDIO_FRAME)
#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 compressed
#define DPOSSIZENAME_AUD_COMPRESSED		0x02000000
	//! Audio is big endian, else little endian
#define DPOSSIZENAME_AUD_BIGENDIAN_BIT0x00080000
	//! Just for completeness
//#define DPOSSIZENAME_AUD_LITTLEENDIAN_BIT	0x00000000
	//! This frame is independent of other frames for decode see DFRAME::dwType
// also used #define DFRAME_TYPE_KEYFRAME0x10000000
	//! This frame is independent 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
	//! Send this in if you just need the filename (e.g. scratch)
#define DPOSSIZENAME_FILENAME_ONLY		0x40000000		// Same as
DFRAME_SKIP_FRAME
	//! 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 hardware 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 system 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 an NTSC value of frames by default.  This is the place
        * we normally count on to 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 were 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 currently.
        */
        DWORD           dwMfCaps;                               // MF_CAP_,
 AVH_CAP_
        /**
        * Basic video standard.  Was PAL/NTSC, but should now use #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 independent 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 separate)
        //! 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 palette entries used in bitmap
        //! For colour tables in dwReserved, the number of RGBQUAD elements that are critical to display (for
windows palette wars in < 256 color mode)
   DWORD      biClrImportant;                        // Required entries to display bitmap
        /**
        * The dwReserved may hold many things.  Whatever it holds, the amount used can be determined by
        * subtracting 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];        // Palette or extended bitmap structure
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 skew (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
#define VVWVIDEO_4096_YCBCBR10   11008
#define VVWVIDEO_YCBCR10_PITCH(__width__) ((( __width__ % 48 == 0 ) ? __width__ : (((__width__ / 48 )
+ 1) * 48)) * 8 / 3)
	//! 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
//! Normal NTSC PAL field dominance for 720 CCIR VVWVIDEO::dwDrFlags
#define DRFLAGS_LOWER_FIELD1_DOMINANT		0x00000000
//! Old 640 square pixel dominance in VVWVIDEO::dwDrFlags
#define DRFLAGS_UPPER_FIELD2_DOMINANT  0x00000001
//! Preview to QuickclipXO
#define DRFLAGS_NOT_QUICKCLIP			0x00000002
//! Field dominance MASK for VVWVIDEO::dwDrFlags
#define MASK___DRFLAGS_TEMPORAL			0x00000004
//! The first FULL line (line 2) is temporally first VVWVIDEO::dwDrFlags
#define DRFLAGS_LOWER_TEMPORAL_FIRST   0x00000000
//! The half line is temporally first VVWVIDEO::dwDrFlags
#define DRFLAGS_UPPER_TEMPORAL_FIRST    0x00000004
//! 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 (NT3SC)
#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;       \
        (__pvvwvid_)->dwLength = _frames;    \
	/*(__pvvwvid_)->fccHandler = (__pvvwvid_)->biCompression;*/		\
	(__pvvwvid_)->rcFrame.top = 0;		\
	(__pvvwvid_)->rcFrame.bottom = abs((__pvvwvid_)->biHeight);		\
	(__pvvwvid_)->rcFrame.left = 0;		\
	(__pvvwvid_)->rcFrame.right = (__pvvwvid_)->biWidth;		\
	}
//	VVWXXX_SETSAMPLETOLENGTH((__pvvwvid_), _frames);	WRONG WRONG WRONG

//! 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_)->fccHandler = (__pvvwvid_)->biCompression; \
	(__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 independent 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 space
        // 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 skew (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)

        //! Not used currently
        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 internal cases, we always use 32 bits for 20 and 24.  This calc
* returns 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_)->cbSize = 0;	\
	(__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_)->biUnused = 0; \
	(__pvvwaud_)->dwDrFlags = 0;	\
	(__pvvwaud_)->dwFileType = 0;	\
	(__pvvwaud_)->dwResDrastic = 0;	\
	(__pvvwaud_)->rcFrame.top = 0;	\
	(__pvvwaud_)->rcFrame.bottom = 0;	\
	(__pvvwaud_)->rcFrame.left = 0;	\
	(__pvvwaud_)->rcFrame.right = 0;	\
	}


/** Special wave header for quickly creating wave file
* Simply fill in, write and then add audio samples afterward
* and then update the sizes
*/
typedef struct {
//!The canonical WAVE format starts with the RIFF header:
#define DTWAVEHDR_RIFF	0x46464952
/** 0	4   ChunkID	Contains the letters "RIFF" in ASCII form
				(0x52494646 big-endian form).
				*/
```

```
        DWORD u32ChunkID;
/** 4        4   ChunkSize        36 + SubChunk2Size, or more precisely:
                          4 + (8 + SubChunk1Size) + (8 + SubChunk2Size)
                          This is the size of the rest of the chunk
                          following this number.  This is the size of the
                          entire file in bytes minus 8 bytes for the
                          two fields not included in this count:
                          ChunkID and ChunkSize.
                          */
        DWORD u32ChunkSize;
#define DTWAVEHDR_WAV      0x45564157
/** 8        4   Format           Contains the letters "WAVE"
                          (0x57415645 big-endian form).
                          */
   DWORD u32Format;
/**
The "WAVE" format consists of two subchunks: "fmt " and "data":
The "fmt " subchunk describes the sound data's format:
*/
#define DTWAVEHDR_FMT
/** 12       4   Subchunk1ID      Contains the letters "fmt "
                          (0x666d7420 big-endian form).
                          */
        DWORD u32SubChunk1ID;
/** 16       4   Subchunk1Size    16 for PCM.  This is the size of the
                          rest of the Subchunk which follows this number.
                          */
        DWORD u32SubChunk1Size;
        //! 20 2 Format of audio data, uncompressed == #DTWAVE_FORMAT_PCM (windows
WAVE_FORMAT_PCM) == 1
   WORD      wFormatTag;                    /* format type (normally WAVE_FORMAT_PCM == 1) */
        //! 22 2 Number of channels in this stream.  Usually 1 or 2
   WORD      nChannels;                     /* number of channels (i.e. mono=1, stereo=2...) */
        //! 24 4 Number of samples per second.  eg. 44100, 22050, 48000 etc
   DWORD     nSamplesPerSec;                /* sample rate (i.e. 44100, 48000. 22050) */
        //! 28 4 Average unsigned chars Per Second (see also VVWAUDIO::dwRate) == #nBlockAlign *
#nSamplesPerSec
   DWORD     nAvgBytesPerSec;          /* for buffer estimation (nSamplesPerSec * nBlockAlign) */
        //! 32 2 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))
*/
        //! 34 2 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) */
        //! 36 2 Size of the dwReserved area used.  For PCM this will be zero.  For other compressors, it may be
anything < 256 DWORDs
   // Not for PCM   WORD      cbSize;                    /* the count in unsigned chars of the
/** The "data" subchunk contains the size of the data and the actual sound:
*/
#define DTWAVEHDR_DATA
```

```
/** 36        4   Subchunk2ID      Contains the letters "data"
                               (0x64617461 big-endian form).
                               */
        DWORD u32SubChunk2ID;
/** 40        4   Subchunk2Size    == NumSamples * NumChannels * BitsPerSample/8
                               This is the number of bytes in the data.
                               You can also think of this as the size
                               of the read of the subchunk following this
                               number.
                               */
        DWORD u32SubChunk2Size;
        // Data goes here
//44        *   Data             The actual sound data.
} DTDIRECT_WAVEHDR, * pDTDIRECT_WAVEHDR;

/////////////////////////////////////////////////////////////////
//
// 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 room for other VITC readers,
//        closed 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

/** Structure to pass image information from the file to the render.  Always add at end.
 */
typedef struct {
        //! How much do we have, effective version
        DWORD           dwStructSize;
        //! SMPTE_UMID
        DWORD           ardwSMPTE_UMID[8];
        //! Base curve type (may be overridden below, but simple version here
        DWORD           dwCurveType;
        //! Gamma value
        float     fGamma;
        //! Bayer white balance offset
        float     fR;
        //! Bayer white balance offset
        float     fG1;
        //! Bayer white balance offset
        float     fG2;
        //! Bayer white balance offset
        float     fB;
        //! As shot white - X
        float   fAsShotWhiteX;
        //! As shot white - Y
```

```
          float   fAsShotWhiteY;
          //! As shot Z
#define fAsShotWhiteZ (1 - fAsShotWhiteX - fAsShotWhiteY)
          //! White balance
          DWORD           dwWhiteBalance;
          //!
          DWORD           dwExposure;
          //!
          DWORD           dwBlackLevel;
          //!
          DWORD           dwWhiteLevel;
          //! Horizontal flip
          DWORD           dwHorizontalFlip;
          //! Vertical flip
          DWORD           dwVerticalFlip;
          //! Stereo Horizontal flip
          DWORD           dwStereoHorizontalFlip;
          //! Stereo Vertical flip
          DWORD           dwStereoVerticalFlip;
          //! Exposure time in micro seconds (10^-6 seconds)
          DWORD           dwExposureTime;
          //! Shutter angle in degree * 1000. Shutter Angle = (ExposureTime * SensorFps)/1E9 * 360.0
          DWORD           dwShuttleAngle;
          //! Camera position
          DWORD           dwX;
          //! Camera position
          DWORD           dwY;
          //! Camera position
          DWORD           dwZ;
          //! Camera position
          DWORD           dwPan;
          //! Camera position
          DWORD           dwTilt;
          //! Camera position
          DWORD           dwRoll;
          //! Do we have a custom matrix
          bool     fCustomMatrix;
          /** Colour Matrix
          // OUT RED - R G B
          pvIII->matrix[0][0];     R
          pvIII->matrix[0][1];     G
          pvIII->matrix[0][2];     B
          pvIII->matrix[0][3];     A
          // OUT GREEN - R G B
          pvIII->matrix[1][0];     R
          pvIII->matrix[1][1];     G
          pvIII->matrix[1][2];     B
          pvIII->matrix[1][3];     A
          //
 OUT BLUE - R G B
          pvIII->matrix[2][0];     R
```

```
        pvIII->matrix[2][1];     G
        pvIII->matrix[2][2];     B
        pvIII->matrix[2][3];     A
        */
        float     matrix[4][4];
        //! Do we have a custom matrix
        bool      fCustomMatrix2;
        float     matrix2[4][4];
        //!
        char      szLookFile[32];
        //! Do we have a forward matrix
        bool      fUserMatrix1;
        /** User Matrix
        */
        float     matrixUser1[4][4];
        //! Do we have a forward matrix
        bool      fUserMatrix2;
        /** User Matrix
        */
        float     matrixUser2[4][4];
        //! Is the hue valid
        bool    HasHue;
        //! Hue -180..180
        float     fHue;
        //! Is the bright valid
        bool      HasBrightness;
        //! Brightness 0..1.0
        float     fBrightness;
        //! Is the contrast valid
        bool      HasContrast;
        //! Contrast 0..1.0
        float     fContrast;
        //! Is the chroma valid
        bool      HasChroma;
        //! Chroma 0..1.0
        float     fChroma;
        //
        bool      GammaRValid;
        float     fGammaR;
        bool      GammaBValid;
        float     fGammaB;
        //
        bool      PedestalValid;
    float fPedestalR;                   // [-1.0, 1.0], neutral 0.0;
                                                          // 1.0 means shift by the maximum pixel value
    float fPedestalG;                   // after gamma offset
    float fPedestalB;
        //
        bool      GainValid;
        float     fGainR;
        float     fGainG;
        float    fGainB;
        //
        bool     MaxValid;
        float    fMaxR;
        float    fMaxG;
        float    fMaxB;
        //
        bool     ToneValid;
   int32_t TonePoints;
   float fTone[32*2];              // up to 32  points  + 0.0,0.0  1.0,1.0
                                                               // defining a LUT using spline curves
        DWORD          dwCalibration1;
        DWORD          dwCalibration2;
        DWORD          dwUserCalibration1;
        DWORD          dwUserCalibration2;
#define DTCALIBRATION_UNKNOWN    0 //= Unknown
#define DTCALIBRATION_DAYLIGHT 1 //= Daylight
#define DTCALIBRATION_FLOURESCENT 2 //= Fluorescent
#define DTCALIBRATION_TUNGSTEN 3 //= Tungsten (incandescent light)
#define DTCALIBRATION_FLASH 4 //= Flash
#define DTCALIBRATION_FINE_WEATHER 9 //= Fine weather
#define DTCALIBRATION_CLOUDY_WEATHER 10 //= Cloudy weather
#define DTCALIBRATION_SHADE 11 //= Shade
#define DTCALIBRATION_DAYLIGHT_FLUORESCENT 12 //= Daylight fluorescent (D 5700 - 7100K)
#define DTCALIBRATION_DAY_WHITE_FLUORESCENT 13 //= Day white fluorescent (N 4600 - 5400K)
#define DTCALIBRATION_COOL_WHITE_FLUORESCENT 14 //= Cool white fluorescent (W 3900 - 4500K)
#define DTCALIBRATION_WHITE_FLUORESCENT 15 //= White fluorescent (WW 3200 - 3700K)
#define DTCALIBRATION_STD_LIGHT_A 17 //= Standard light A
#define DTCALIBRATION_STD_LIGHT_B 18 //= Standard light B
#define DTCALIBRATION_STD_LIGHT_C 19 //= Standard light C
#define DTCALIBRATION_D55 20 //= D55
#define DTCALIBRATION_D65 21 //= D65
#define DTCALIBRATION_D75 22 //= D75
#define DTCALIBRATION_D50 23 //= D50
#define DTCALIBRATION_ISO_STUDIO 24 //= ISO studio tungsten
#define DTCALIBRATION_OTHER 255 //= Other light source
#define DTCALIBRATION_DIRECT 256
        // Special case
#define DTCALIBRATION_CAMERATORGB        4096
#define DTCALIBRATION_RGBTOFINAL4097
        DWORD          dwFlags;
#define DTIMAGEINFO_TYPE_MASK               0x0000FF00
#define DTIMAGEINFO_TYPE_IS_AATON           0x00000100
#define DTIMAGEINFO_TYPE_IS_ONECAM          0x00000200
        //! if there is a linearization table and it is dword size
        DWORD * pdwLinearizationTable;
        //! if there is a linearization table and it is word size
        WORD * pwLinearizationTable;
        //! Number of elements in the table
        DWORD dwLinearizationElems;
} vvwInfImageInfo, * pvvwInfImageInfo;
```

```
//! Invalid setting, ignore it
#define VVWINF_INVALID                                  (-1)
//! dwCurveType
#define VVWINF_CURVETYPE_UNKNOWN        0
#define VVWINF_CURVETYPE_LINEAR         1
#define VVWINF_CURVETYPE_LOG            2
//!

/** 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,
//!
vvwiCustomMetadataBlockPointer,
//!
vvwiImageInfo,
//!
vvwiUMID,
//
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,
//!
vvwiCustomMetadataBlockType,
vvwiCustomMetadataBlockSize,
vvwiNorthSouthEastWest,
vvwiLatitude,
vvwiLongitude,
vvwiExposure,
vvwiRedGain,
vvwiBlueGain,
        vvwiWhiteBalance,

        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
        vvwiVideoYPelsPerMeter,
        //! 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))

//!
#define DT_CustomMetadataBlockType_CINE             0x00000001
#define DT_CustomMetadataBlockType_DPX              0x00000002
#define DT_CustomMetadataBlockType_ILLEGAL          0xFFFFFFFF
#define DT_CustomMetadataBlockType_NONE             0x00000000

/**
* 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>
*
* Metadata 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.drastic.tv)
        * <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
        /**
        * CustomMetadataBlockPointer
        */
        char * szCustomMetadataBlockPointer;
        //! XML tag name for VVWINFO::szCustomMetadataBlockPointer
#define VVWINFOTAG_szCustomMetadataBlockPointer             "CustomMetadataBlockPointer"
#define VVWINFODESC_szCustomMetadataBlockPointer            "CustomMetadataBlockPointer"
        /**
        * Pointer to ImageInfo structure
        */
        char * szImageInfo;
        //! XML tag name for VVWINFO::szImageInfo
#define VVWINFOTAG_szImageInfo                          "ImageInfo"
#define VVWINFODESC_ImageInfo                           "ImageInfo"
        /**
        * The SMPTE 330M Unique Material Identifier (UMID) is a standard for providing a stand-alone method
for generating a unique label designed to be used to attach to media files and streams.

There are two types of UMIDs:

The Basic UMID contains the minimal components necessary for the unique identification (the essential
metadata) The length of the basic UMID is 32 octets.

The Extended UMID provides information on the creation time and date, recording location and the name of the
organization and the maker as well as the components of the Basic UMID. The length of the Extended UMID is
```

64 octets. This data may be parsed to extract specific information produced at the time it was generated or simply used as a unique label.

```
        */
        char * szUMID;
        //! XML tag name for VVWINFO::szUMID
#define VVWINFOTAG_szUMID                      "UMID"
#define VVWINFODESC_UMID                       "UMID"
        /**
        * 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>
        * Date as YYYYMMDD (BCD: 1 digit -> 4 bits).
        * <BR> XML tag \<FileDate\>
        */
        DWORD dwFileDate;
        //! XML tag name for VVWINFO::dwFileDate
#define VVWINFOTAG_dwFileDate         "FileDate"
#define VVWINFODESC_dwFileDate        "FileDate"
        /**
        * Time of file <br>
        * Time as HHMMSScc (BCD: 1 digit -> 4 bits) The last byte 'cc' are centiseconds.
        * <BR> XML tag \<FileTime\>
        */
        DWORD dwFileTime;
        //! XML tag name for VVWINFO::dwFileTime
#define VVWINFOTAG_dwFileTime         "FileTime"
#define VVWINFODESC_dwFileTime        "FileTime"
        /**
        * Sequence 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"
        /**
        * Custom metadata block type
        */
        DWORD dwCustomMetadataBlockType;
        //! XML tag name for VVWINFO::dwCustomMetadataBlockType
#define VVWINFOTAG_dwCustomMetadataBlockType              "CustomMetadataBlockType"
#define VVWINFODESC_dwCustomMetadataBlockType             "CustomMetadataBlockType"
        /**
        * Custom metadata block size
        */
        DWORD dwCustomMetadataBlockSize;
        //! XML tag name for VVWINFO::dwCustomMetadataBlockSize
#define VVWINFOTAG_dwCustomMetadataBlockSize              "CustomMetadataBlockSize"
#define VVWINFODESC_dwCustomMetadataBlockSize             "CustomMetadataBlockSize"
        /**
        * GPS coordinates - North=2/South=0, East=1/West=0
        */
        DWORD dwNorthSouthEastWest;
        //!
 XML tag name for VVWINFO::dwNorthSouthEastWest
#define VVWINFOTAG_dwNorthSouthEastWest "NorthSouthEastWest"
#define VVWINFODESC_dwNorthSouthEastWest           "NorthSouthEastWest"
        /**
        * Latitude value in decimal degrees * 10000000  (uses bit 1 for north/south)
        */
        DWORD dwLatitude;
        //! XML tag name for VVWINFO::dwLatitude
#define VVWINFOTAG_dwLatitude       "Latitude"
#define VVWINFODESC_dwLatitude      "Latitude"
        /**
        * Longitude value in decimal degrees * 10000000
        */
        DWORD dwLongitude;
        //! XML tag name for VVWINFO::dwLongitude
#define VVWINFOTAG_dwLongitude              "Longitude"
#define VVWINFODESC_dwLongitude             "Longitude"
        /**
        * Camera Exposure
        */
        DWORD dwExposure;
        //! XML tag name for VVWINFO::dwExposure
#define VVWINFOTAG_dwExposure       "Exposure"
#define VVWINFODESC_dwExposure              "Exposure"
        /**
        * Red gain value times 1000
        */
        DWORD dwRedGain;
        //! XML tag name for VVWINFO::dwRedGain
#define VVWINFOTAG_dwRedGain        "RedGain"
#define VVWINFODESC_dwRedGain               "RedGain"
        /**
        * Blue gain value times 1000
        */
        DWORD dwBlueGain;
        //! XML tag name for VVWINFO::dwBlueGain
#define VVWINFOTAG_dwBlueGain       "BlueGain"
#define VVWINFODESC_dwBlueGain              "BlueGain"
        /**
        * White balance
        */
        DWORD dwWhiteBalance;
        //! XML tag name for VVWINFO::dwWhiteBalance
#define VVWINFOTAG_dwWhiteBalance           "WhiteBalance"
#define VVWINFODESC_dwWhiteBalance                  "WhiteBalance"
        /**
        * 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 anything has changed in 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

#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_szCustomMetadataBlockPointer   ,        //"NotUsed_81"
(char*)VVWINFOTAG_szImageInfo            ,        //"NotUsed_82"
(char*)VVWINFOTAG_szUMID                 ,        //"UMID"
(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,           //
                (char*)VVWINFOTAG_dwLTCTimeCodeType,     //
                (char*)VVWINFOTAG_dwVITCTimeCodeType,   //
                (char*)VVWINFOTAG_dwProdDate,       //
        //End: v3.0
                (char*)VVWINFOTAG_dwUniqueID,       //
                (char*)VVWINFOTAG_dwCustomMetadataBlockType,     //
                (char*)VVWINFOTAG_dwCustomMetadataBlockSize,      //
                (char*)VVWINFOTAG_dwNorthSouthEastWest,  //
                (char*)VVWINFOTAG_dwLatitude,        //
                (char*)VVWINFOTAG_dwLongitude,      //
                (char*)VVWINFOTAG_dwExposure,       //"Exposure"
                (char*)VVWINFOTAG_dwRedGain,        //"RedGain"
                (char*)VVWINFOTAG_dwBlueGain,       //"BlueGain"
                (char*)VVWINFOTAG_dwWhiteBalance, //"WhiteBalance"
                (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 – vvwIF.h

```
/***********************************************************************
* $Id$:
*
* $HeadURL$:
* $Author$:
* $Revision$:
* $Date$:
*
* Copyright (c) 1998-2023 Drastic Technologies Ltd.  All Rights Reserved.
* 523 The Queensway, Suite 201 Toronto ON M8V 1Y7
* 416 255 5636  fax 255 8780
* engineering@drastictech.com      http://www.drastic.tv
***********************************************************************/

/** @NOTE This is an SDK file, so it needs to stay clean and being able to
compile independently of the main source tree.  Please test any changes
under Linux, macOS and Windows.
*/
#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 <unistd.h>
#endif

typedef uint32_t VVWIFOPAQUE;

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

uint32_t __stdcall vvwCmdIF(VVWIFOPAQUE vvwChannel, PMEDIACMD pCmd);

#ifndef BUILDING_VVW_DLL
long __stdcall vvwOpeningChannel();
#define COMPILING_DIRECT_MEDIACMD_SOURCE

#ifdef BUILD_FOR_QTCREATOR

DWORD __stdcall vvwInitQT(void);

VVWHANDLE __stdcall vvwOpenQT(DWORD dwChannel);
```

```
uint32_t __stdcall vvwCmdQT(VVWHANDLE hVvw, PMEDIACMD pCmd);

uint32_t __stdcall vvwCloseQT(VVWHANDLE hVvw);

DWORD __stdcall vvwQuitQT(void);

DWORD __stdcall vvwIsInProcessQT(void);

long __stdcall vvwIsLocal(VVWIFOPAQUE vvwChannel);

#endif

/**
Private open for the direct source MediaCMD API access
*/
uint32_t __stdcall vvwOpenNetworkChannel(char * szAddress, uint32_t dwPort, uint32_t dwChannel);
/**
Private close for the direct source MediaCMD API access
*/
uint32_t __stdcall vvwCloseNetworkChannel(VVWIFOPAQUE vvwChannel);
/**
* Check that we are still conneted through the network
*/
uint32_t __stdcall vvwCheckNetworkChannel(VVWIFOPAQUE vvwChannel);
/**
* Process is exiting
*/
uint32_t __stdcall vvwQuitNetwork();
#endif

/**
Get the current millisecond time.
@param lChannel the mediacmd target being controlled (internal, external, network)
@return the current millisecond counter for that channel
*/
uint32_t __stdcall vvwGetCurMs(VVWIFOPAQUE vvwChannel);

/**
* Convert a 0..3 channel to 0, 64, 65536
* Mostly internal
* Undocumented
*/
VVWHANDLE __stdcall vvwChannelToHandle(VVWIFOPAQUE vvwChannel);

/**
* Convert a 0, 64, 65536 to 0..3
* Mostly internal
* Undocumented
*/
uint32_t __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.
*/
uint32_t __stdcall vvwEnableChannels(uint32_t lInternal0_31,
        uint32_t lInternal32_63,
        uint32_t lExternal64_95,
        uint32_t lExternal96_127,
        uint32_t lNetwork128_159,
        uint32_t lNetwork160_191);
/**
* Release memory allocated to channels
*/
uint32_t __stdcall vvwReleaseChannels(void);
/**
Has the vvw sub system been opened
*/
uint32_t __stdcall vvwIsOpen(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.
*/
uint32_t __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 Direct interface.
*/
uint32_t __stdcall vvwGetMaxInternalChannels(void);
/**
Returns the maximum number of network channels available for control.  Channels start at 128 and end at 256 –
1.  This return is one greater than the largest value available for SetCurChannel(), GetCurChannel() and the
lChannel parameter for the Direct interface.
*/
uint32_t __stdcall vvwGetMaxNetworkChannels(void);
/**
Get the name of the current channel.  For Direct 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).
*/
uint32_t __stdcall vvwGetChannelName(VVWIFOPAQUE vvwChannel, 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
*/
uint32_t __stdcall vvwGetChannelType(VVWIFOPAQUE vvwChannel);

/**
Setting this will cause the cfUseFrameCount Flag to be set on transport commands
This will cause the interface to ignore LTC/VITC offsets and use absolute Timecode values.
Mostly for ease of use in TC Space when still running as a LTC / VITC timecode source.
*/
BOOL __stdcall vvwSetUseAbsoluteTC(BOOL bUseAbsoluteTC);
/**
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 as the dialog only shows up on the local machine, and
cannot be seen through the network.
*/
uint32_t __stdcall vvwShowConfigDialog(VVWIFOPAQUE vvwChannel, uint32_t hWnd);

//
// Transport
//

/**
Wait for the channel to reach a state (ctPlay, ctRecord, etc).  Will wait
up to 500 milliseconds
@param lChannel the mediacmd target being controlled (internal, external, network)
@param ctCmd the command state we will wait for (ctPlay, ctRecord, ctPause, etc)
@return if state is reached, return 0.  If it timed out waiting, return -1
*/
uint32_t __stdcall vvwWaitForState(VVWIFOPAQUE vvwChannel, uint32_t ctCmd, uint32_t lPosition);

/**
Play at normal speed.
Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwPlay(VVWIFOPAQUE vvwChannel);

/**
* Set the channel into play jumping a amount of frames at a time
* for no offset, set lOffset to 0.  For no position set it to -1.  For now start time, set dwMS to -1.
*/
uint32_t __stdcall vvwPlayOffsetAt(VVWIFOPAQUE vvwChannel, uint32_t dwPosition, long lOffset, uint32_t
dwMS);

/**
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.
*/
uint32_t __stdcall vvwPlayAtSpeed(VVWIFOPAQUE vvwChannel, uint32_t lVVWSpeed, uint32_t lEnd);

/**
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.
*/
uint32_t __stdcall vvwPlayFromTo(VVWIFOPAQUE vvwChannel, uint32_t lFrom, uint32_t lTo, int fDeferred, int fLoop);

/**
Clip Mode Only. Load a clip into the channel and display the lStartFrame.
Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwLoadClip(VVWIFOPAQUE vvwChannel, char * sz8CharClipName, uint32_t lStartFrame);

/**
Clip Mode Only. Switch to a different clip without changing play/pause mode.
Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwSwitchClip(VVWIFOPAQUE vvwChannel, char * sz8CharClipName, uint32_t 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.
*/
uint32_t __stdcall vvwPlayClip(VVWIFOPAQUE vvwChannel, 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.
*/
uint32_t __stdcall vvwPlayClipFromTo(VVWIFOPAQUE vvwChannel, char * sz8CharClipName, uint32_t lFrom, uint32_t lTo, int fDeferred);

/**
Set the channel into its fastest possible forward motion state.
Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwFastForward(VVWIFOPAQUE vvwChannel);

```
/**
Set the channel into its fastest possible reverse motion state.
        Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwFastRewind(VVWIFOPAQUE vvwChannel);

/**
Stop playback and display the current frame.
        Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwPause(VVWIFOPAQUE vvwChannel);

/**
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.
*/
uint32_t __stdcall vvwSeek(VVWIFOPAQUE vvwChannel, uint32_t lFrame);

/**
Seek a certain number of frames from the current position.  Positive offsets imply forward direction, negative
offset imply reverse.
*/
uint32_t __stdcall vvwSeekRelative(VVWIFOPAQUE vvwChannel, uint32_t 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.
*/
uint32_t __stdcall vvwStop(VVWIFOPAQUE vvwChannel);

/**
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.
*/
uint32_t __stdcall vvwRecord(VVWIFOPAQUE vvwChannel);

/**
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.
*/
uint32_t __stdcall vvwRecordFromTo(VVWIFOPAQUE vvwChannel, uint32_t lFrom, uint32_t 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.
*/
uint32_t __stdcall vvwRecordStop(VVWIFOPAQUE vvwChannel, char * sz8CharClipName, uint32_t 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.
*/
uint32_t __stdcall vvwRecordStopFileName(VVWIFOPAQUE vvwChannel, char * sz8CharClipName, char * sz256CharFileName, uint32_t 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.
*/
uint32_t __stdcall vvwSetRecordPresets(VVWIFOPAQUE vvwChannel, uint32_t lVidEdit, uint32_t lAudEdit, uint32_t lInfEdit);

/**
Reset Record file and/or delete all files found in the record folder
	Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwCleanRecordWipeDrive(VVWIFOPAQUE vvwChannel, uint32_t lWipeFolder);

/**
Eject the current media if it is removable. Normally only used with VTRs.
	Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwEject(VVWIFOPAQUE vvwChannel);

/**
* MediaCmd direct access
*/
uint32_t __stdcall vvwMediaCmd(VVWIFOPAQUE vvwChannel, 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.
*/
uint32_t __stdcall vvwTransfer(VVWIFOPAQUE vvwChannel, uint32_t lTargetChannel, uint32_t lPosition, uint32_t lStart, uint32_t lEnd, uint32_t lVidEdit, uint32_t lAudEdit, uint32_t 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.
*/
uint32_t __stdcall vvwUpdateStatus(VVWIFOPAQUE vvwChannel);

/**
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
*/
uint32_t __stdcall vvwGetState(VVWIFOPAQUE vvwChannel);

/**
        Returns the current flags

cfDeferred = 1,                         // 0x00000001 This is a delayed
cfOverrideDeferred = 1 << 30,// 0x40000000 Override 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
*/
uint32_t __stdcall vvwGetFlags(VVWIFOPAQUE vvwChannel);

/**
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
*/
uint32_t __stdcall vvwGetSpeed(VVWIFOPAQUE vvwChannel);

/**
Returns the current position if the cfUsePosition flag is set, otherwise invalid.
*/
uint32_t __stdcall vvwGetPosition(VVWIFOPAQUE vvwChannel);

/**
Returns the millisecond time the last status occurred (time of the last vertical blank).
*/
uint32_t __stdcall vvwGetLastMs(VVWIFOPAQUE vvwChannel);

/**
Returns the current start or in point if the cfUseStart flag is set.
*/
uint32_t __stdcall vvwGetStart(VVWIFOPAQUE vvwChannel);

/**
Return the current end point or out point if cfUseEnd is set.
*/
uint32_t __stdcall vvwGetEnd(VVWIFOPAQUE vvwChannel);

/**
Only supported in clip Mode. Returns the current clip name, if any.  For Direct access, the memory must be at
least 9 bytes uint32_t (8 character bytes + NULL) and is always ANSI.
*/
```

```
uint32_t __stdcall vvwGetClipName(VVWIFOPAQUE vvwChannel, char * sz8CharClipName);

/**
Returns the current file name, if any.  For Direct access, the memory must be at least 261 bytes LONG (260
bytes max path + NULL) and is always ANSI.
*/
uint32_t __stdcall vvwGetFileName(VVWIFOPAQUE vvwChannel, char * sz260CharFileName);

/**
Returns the current time code as a string (e.g. "00:01:00:00").  For Direct access, the memory must always be at
least 15 bytes uint32_t (14 byte time code plus id + NULL) and is always ANSI.
*/
uint32_t __stdcall vvwGetCurTC(VVWIFOPAQUE vvwChannel, char * sz14ByteTC);

/**
Returns the current state as a string (e.g. "Play"). For Direct access, the memory must always be at least 15
bytes uint32_t (14 byte state + NULL) and is always ANSI.
*/
uint32_t __stdcall vvwGetCurState(VVWIFOPAQUE vvwChannel, 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 Direct access.  Please note:
For Direct 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(VVWIFOPAQUE vvwChannel, char * sz8CharLastClipCurClip);

/**
Returns 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.
*/
uint32_t __stdcall vvwGetClipInfo (VVWIFOPAQUE vvwChannel, char * sz8CharClipName, uint32_t * lStart,
uint32_t * lEnd, uint32_t * lVidEdit, uint32_t * lAudEdit, uint32_t * lInfEdit, char * szFileName);

/**
Sets the metadata for szClip.
        Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwSetMetaData (VVWIFOPAQUE vvwChannel, char * sz8CharClipName, uint32_t
vvwiInfoRequest, uint32_t nValue, char * szValue);

/**
```

```
Retuns the metadata from szClip.
	Returns 0 if successful, else an error code.
*/
uint32_t __stdcall vvwGetMetaData (VVWIFOPAQUE vvwChannel, char * sz8CharClipName, char *
sz260CharFileName, uint32_t vvwiInfoRequest, char * szValue);

/**
Returns 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.
*/
uint32_t __stdcall vvwGetNextClipEx (VVWIFOPAQUE vvwChannel, char * sz8CharClipName, uint32_t * lCreation,
uint32_t * lLastModification, uint32_t * lFileSize, uint32_t * 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.
*/
uint32_t __stdcall vvwCopyClip (VVWIFOPAQUE vvwChannel, char * szSourceClip, char * szDestClip, uint32_t
lStart, uint32_t lEnd);

// VTR Mode
/**
Reset the EDL returns in VTR mode to the first element of the list.
*/
uint32_t __stdcall vvwEDLResetToStart(VVWIFOPAQUE vvwChannel);

/**
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 successful else an Error code.
*/
uint32_t __stdcall vvwEDLGetEdit (VVWIFOPAQUE vvwChannel, uint32_t * lRecordIn, uint32_t * lPlayIn, uint32_t
* lPlayOut, uint32_t * lVidEdit, uint32_t * lAudEdit, uint32_t * lInfEdit, char * szClipName, char * szFileName,
BOOL bClipInfo);

/**
Set Comment
*/
uint32_t __stdcall vvwEDLSetExtendedInfo (VVWIFOPAQUE vvwChannel, uint32_t lType, uint32_t lRecordIn,
uint32_t lValue, char * sz260Comment, uint32_t lDuration);
/**
Get Comment
*/
uint32_t __stdcall vvwEDLGetExtendedInfo (VVWIFOPAQUE vvwChannel, uint32_t lType, uint32_t lRecordIn,
char * sz260Comment, uint32_t * plExtra);

// Shared
/**
```

```
Returns the millisecond time of the last change in the transfer queue
*/
uint32_t __stdcall vvwGetLastChangeXferMs(VVWIFOPAQUE vvwChannel);
/**
Returns the millisecond time of the last change in the current mode (clip or vtr).
*/
uint32_t __stdcall vvwGetLastChangeMs(VVWIFOPAQUE vvwChannel, uint32_t lClipSpace, uint32_t * lNumClips);

/**
*/
uint32_t __stdcall vvwInsert (VVWIFOPAQUE vvwChannel, char * szClipName, char * szFileName, uint32_t
lPosition, uint32_t lStart, uint32_t lEnd, uint32_t lVidEdit, uint32_t lAudEdit, uint32_t lInfEdit, int fRipple);

/**
*/
uint32_t __stdcall vvwBlank (VVWIFOPAQUE vvwChannel, char * szClipName, uint32_t lStart, uint32_t lEnd,
uint32_t lVidEdit, uint32_t lAudEdit, uint32_t lInfEdit, int fRipple);

/**
*/
uint32_t __stdcall vvwBlankAllClipIds (VVWIFOPAQUE vvwChannel);
/**
*/
uint32_t __stdcall vvwDelete (VVWIFOPAQUE vvwChannel, char * szClipName, uint32_t lStart, uint32_t lEnd,
uint32_t lVidEdit, uint32_t lAudEdit, uint32_t lInfEdit, int fRipple);

/**
*/
uint32_t __stdcall vvwTrim (VVWIFOPAQUE vvwChannel, uint32_t lPosition, uint32_t lStartOffset, uint32_t
lEndOffset, uint32_t lVidEdit, uint32_t lAudEdit, uint32_t lInfEdit, int fRipple);

//
// Settings
//
/**
Returns the supported attributes of a get/set value  (gsClipMode, gsTcSource, etc) or –1 for not supported.
*/
uint32_t __stdcall vvwValueSupported(VVWIFOPAQUE vvwChannel, uint32_t lValueType);

/**
Returns the current setting for a get/set value.
*/
uint32_t __stdcall vvwValueGet(VVWIFOPAQUE vvwChannel, uint32_t lValueType, uint32_t * plMin, uint32_t *
plMax);

/**
Sets the get/set value to setting.
*/
uint32_t __stdcall vvwValueSet(VVWIFOPAQUE vvwChannel, uint32_t lValueType, uint32_t lSetting);

/**
```

Sets the get/set value to setting with extended parameters. Please set unused parameters to NULL.
*/
uint32_t __stdcall vvwValueSet2(VVWIFOPAQUE vvwChannel, uint32_t lValueType, uint32_t lSetting, uint32_t lStart, uint32_t lEnd, uint32_t lVidChan, uint32_t lAudChan, uint32_t lInfChan);

/**
Calls ValueXXX with gsClipMode. If equal to 1 then the channel is in clip mode, if 0 the channel is in VTR mode.
*/
uint32_t __stdcall vvwGetClipMode(VVWIFOPAQUE vvwChannel);
/**
Calls ValueXXX with gsClipMode. If equal to 1 then the channel is in clip mode, if 0 the channel is in VTR mode.
*/
uint32_t __stdcall vvwSetClipMode(VVWIFOPAQUE vvwChannel, uint32_t 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
*/
uint32_t __stdcall vvwGetTCType(VVWIFOPAQUE vvwChannel);
/**
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
*/
uint32_t __stdcall vvwSetTCType(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Calls ValueXXX with gsTcSource (VITC, LTC, Control, Clip).
#GS_TCSOURCE_LTC, #GS_TCSOURCE_VITC, #GS_TCSOURCE_CTL or #GS_TCSOURCE_CLIP
*/
uint32_t __stdcall vvwGetTCSource(VVWIFOPAQUE vvwChannel);
/**
Calls ValueXXX with gsTcSource (VITC, LTC, Control, Clip).
#GS_TCSOURCE_LTC, #GS_TCSOURCE_VITC, #GS_TCSOURCE_CTL or #GS_TCSOURCE_CLIP
*/
uint32_t __stdcall vvwSetTCSource(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Calls ValueXXX with gsAutoMode. Required for play lists, deferred commands and auto edit commands on VTRs.
*/
uint32_t __stdcall vvwGetAutoMode(VVWIFOPAQUE vvwChannel);
/**
Calls ValueXXX with gsAutoMode. Required for play lists, deferred commands and auto edit commands on VTRs.
*/
uint32_t __stdcall vvwSetAutoMode(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
ADD FUNCTIONS lVidEdit, lAudEdit, lInfEdit
    Returns the current audio, video and info presets for a channel.

```
*/
uint32_t __stdcall vvwGetCurrentPresets(VVWIFOPAQUE vvwChannel, uint32_t * plVidEdit, uint32_t * plAudEdit,
uint32_t * plInfEdit);
/**
ADD FUNCTIONS lVidEdit, lAudEdit, lInfEdit
          Returns the supported audio, video and info presets for a channel.
*/
uint32_t __stdcall vvwGetAvailablePresets(VVWIFOPAQUE vvwChannel, uint32_t * plVidEdit, uint32_t *
plAudEdit, uint32_t * 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
*/
uint32_t __stdcall vvwGetAudioInput(VVWIFOPAQUE vvwChannel);
/**
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
*/
uint32_t __stdcall vvwSetAudioInput(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Get the current audio input level
*/
uint32_t __stdcall vvwGetAudioInputLevel(VVWIFOPAQUE vvwChannel);
/**
Get the current audio input level
*/
uint32_t __stdcall vvwSetAudioInputLevel(VVWIFOPAQUE vvwChannel, uint32_t 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
*/
uint32_t __stdcall vvwGetAudioOutput(VVWIFOPAQUE vvwChannel);
/**
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
*/
uint32_t __stdcall vvwSetAudioOutput(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Get the current audio output level.
*/
uint32_t __stdcall vvwGetAudioOutputLevel(VVWIFOPAQUE vvwChannel);
/**
Get the current audio output level.
*/
uint32_t __stdcall vvwSetAudioOutputLevel(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Returns the RMS and Peak audio levels of the input (stop/record) or output (play/pause).
*/
uint32_t __stdcall vvwGetAudioPeakRMS(VVWIFOPAQUE vvwChannel, uint32_t lAudEdit, uint32_t * 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
*/
uint32_t __stdcall vvwGetVideoInput(VVWIFOPAQUE vvwChannel);
/**
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
*/
uint32_t __stdcall vvwSetVideoInput(VVWIFOPAQUE vvwChannel, uint32_t 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
*/
uint32_t __stdcall vvwGetVideoOutput(VVWIFOPAQUE vvwChannel);
/**
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
*/
uint32_t __stdcall vvwSetVideoOutput(VVWIFOPAQUE vvwChannel, uint32_t lSetting);
```

```
/**
Get the current video input’s ‘Setup’ TBC setting.
*/
uint32_t __stdcall vvwGetVideoInputSetup(VVWIFOPAQUE vvwChannel);
/**
Set the current video input’s ‘Setup’ TBC setting.
*/
uint32_t __stdcall vvwSetVideoInputSetup(VVWIFOPAQUE vvwChannel, uint32_t lSetting);
/**
Get the current video input’s ‘Video’ TBC setting.
*/
uint32_t __stdcall vvwGetVideoInputVideo(VVWIFOPAQUE vvwChannel);
/**
Set the current video input’s ‘Video’ TBC setting.
*/
uint32_t __stdcall vvwSetVideoInputVideo(VVWIFOPAQUE vvwChannel, uint32_t lSetting);
/**
Get the current video input’s ‘Hue’ TBC setting.
*/
uint32_t __stdcall vvwGetVideoInputHue(VVWIFOPAQUE vvwChannel);
/**
Set the current video input’s ‘Hue’ TBC setting.
*/
uint32_t __stdcall vvwSetVideoInputHue(VVWIFOPAQUE vvwChannel, uint32_t lSetting);
/**
Get the current video input’s ‘Chroma’ TBC setting.
*/
uint32_t __stdcall vvwGetVideoInputChroma(VVWIFOPAQUE vvwChannel);
/**
Set the current video input’s ‘Chroma’ TBC setting.
*/
uint32_t __stdcall vvwSetVideoInputChroma(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Get the current global TBC’s ‘Setup’ setting.
*/
uint32_t __stdcall vvwGetVideoTBCSetup(VVWIFOPAQUE vvwChannel);
/**
Set the current global TBC’s ‘Setup’ setting.
*/
uint32_t __stdcall vvwSetVideoTBCSetup(VVWIFOPAQUE vvwChannel, uint32_t lSetting);
/**
Get the current global TBC’s ‘Video’ setting.
*/
uint32_t __stdcall vvwGetVideoTBCVideo(VVWIFOPAQUE vvwChannel);
/**
Set the current global TBC’s ‘Video’ setting.
*/
uint32_t __stdcall vvwSetVideoTBCVideo(VVWIFOPAQUE vvwChannel, uint32_t lSetting);
```

```
/**
Get the current global TBC's 'Hue' setting.
*/
uint32_t __stdcall vvwGetVideoTBCHue(VVWIFOPAQUE vvwChannel);
/**
Set the current global TBC's 'Hue' setting.
*/
uint32_t __stdcall vvwSetVideoTBCHue(VVWIFOPAQUE vvwChannel, uint32_t lSetting);
/**
Get the current global TBC's 'Chroma' setting.
*/
uint32_t __stdcall vvwGetVideoTBCChroma(VVWIFOPAQUE vvwChannel);
/**
Set the current global TBC's 'Chroma' setting.
*/
uint32_t __stdcall vvwSetVideoTBCChroma(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Turn the house/reference lock on or off
*/
uint32_t __stdcall vvwGetVideoGenlock(VVWIFOPAQUE vvwChannel);
/**
Turn the house/reference lock on or off
*/
uint32_t __stdcall vvwSetVideoGenlock(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Set the genlock source to input or external reference
*/
uint32_t __stdcall vvwGetVideoGenlockSource(VVWIFOPAQUE vvwChannel);
/**
Set the genlock source to input or external reference
*/
uint32_t __stdcall vvwSetVideoGenlockSource(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Get the current compression rate
*/
uint32_t __stdcall vvwGetCompressionRate(VVWIFOPAQUE vvwChannel);
/**
Set the current compression rate
*/
uint32_t __stdcall vvwSetCompressionRate(VVWIFOPAQUE vvwChannel, uint32_t lSetting);

/**
Get the status of the superimposed time code overlay
*/
uint32_t __stdcall vvwGetSuperImpose(VVWIFOPAQUE vvwChannel);
/**
Set the status of the superimposed time code overlay
*/
```

```
uint32_t __stdcall vvwSetSuperImpose(VVWIFOPAQUE vvwChannel, uint32_t lSetting);
/**
Get the ms time the last error was added to the error log
*/
uint32_t __stdcall vvwGetErrorLogMs();
/**
Set the error log pointer to the message you want
*/
uint32_t __stdcall vvwSetErrorLog(uint32_t lSetting);
/**
Get the number of current errors
*/
uint32_t __stdcall vvwGetNumberOfErrors(uint32_t * plErrors);
/**
Get the length of the current error string
*/
uint32_t __stdcall vvwGetErrorLength(uint32_t * plLastError, uint32_t * plErrorLength);
/**
Get the current error.  Sets pointer to the next one automatically
*/
uint32_t __stdcall vvwGetError(uint32_t * plLastError, uint32_t * 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.
*/
uint32_t __stdcall vvwGetTotalTime(VVWIFOPAQUE vvwChannel);
/**
Returns the remaining number of frames of storage available at current compression rate.
*/
uint32_t __stdcall vvwGetFreeTime(VVWIFOPAQUE vvwChannel);
/**
Returns the total storage connected in megabytes.
*/
uint32_t __stdcall vvwGetTotalStorage(VVWIFOPAQUE vvwChannel);
/**
Returns the amount of available storage for recording in megabytes.
*/
uint32_t __stdcall vvwGetFreeStorage(VVWIFOPAQUE vvwChannel);

/**
Get the available commands for a channel.
*/
uint32_t __stdcall vvwGetChannelCapabilities(VVWIFOPAQUE vvwChannel);

/**
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(VVWIFOPAQUE vvwChannel);

/**
Get the name of a file from the media file's name
*/
uint32_t __stdcall vvwGetPiconName(char * szFileName);

/**
Start playback at a specified MS
*/
uint32_t __stdcall vvwPlayAtMs(VVWIFOPAQUE vvwChannel, uint32_t lMs);
/**
Start playback at a specified MS with start and end point
*/
uint32_t __stdcall vvwPlayFromToAtMs(VVWIFOPAQUE vvwChannel, uint32_t lMs, uint32_t lStart, uint32_t lEnd);
/**
Start Recording at a specified MS
*/
uint32_t __stdcall vvwRecordAtMs(VVWIFOPAQUE vvwChannel, uint32_t lMs, uint32_t lStart, uint32_t lEnd);
/**
Get the last preview ms from the preview queue
*/
uint32_t __stdcall vvwGetLastPreviewMs(VVWIFOPAQUE vvwChannel);
/***
Export to XML functions
*/
uint32_t __stdcall vvwXMLProjectSave(char * lpszProjectName);
/* Check for file conflicts*/
uint32_t __stdcall vvwXMLProjectCheckOpen(char * lpszProjectName);
/* Open the project*/
uint32_t __stdcall vvwXMLProjectOpen(char * lpszProjectName, int bDeleteConflictingFiles);
/*** Get number of backups on disk
*/
uint32_t __stdcall vvwGetNumberOfBackUps(VVWIFOPAQUE vvwChannel, uint32_t * pdwBackUps, uint32_t
dwClipMode);
/***
Revert to specified backup
*/
uint32_t __stdcall vvwSetBackUpNumber(VVWIFOPAQUE vvwChannel, uint32_t dwBackUp, uint32_t
dwClipMode);

uint32_t __stdcall vvwGetLicenseOptions(VVWIFOPAQUE vvwChannel, uint32_t * pdwLicenseValid, uint32_t *
pdwOptions);
```

```
#ifndef COMPILING_DIRECT_MEDIACMD_SOURCE
/***
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
*/
uint32_t __stdcall vvwChannel2File(VVWIFOPAQUE vvwChannel, char * lpszFileName, uint32_t * plFileType,
uint32_t * plCompression, uint32_t * plBitCount,
          uint32_t * plWidth, uint32_t * plHeight,
          uint32_t * plRate, uint32_t * 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
*/
#define C2V_MATCH_NONE                                                          0
#define C2V_MATCH_INTERNAL_SIGNALFORMAT                               1
#define C2V_MATCH_INTERNAL_FILEFORMAT                       2
#define C2V_MATCH_INTERNAL_COMPRESSION                               4
#define C2V_MATCH_INTERNAL_ALL
          (C2V_MATCH_INTERNAL_SIGNALFORMAT | C2V_MATCH_INTERNAL_FILEFORMAT |
C2V_MATCH_INTERNAL_COMPRESSION)

uint32_t __stdcall vvwChannel2Video(VVWIFOPAQUE vvwChannel, uint32_t * plFileType, uint32_t *
plCompression, uint32_t * plBitCount,
          uint32_t * plWidth, uint32_t * plHeight,
          uint32_t * plRate, uint32_t * plScale, DWORD dwMatchFlags);
#endif

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


#ifndef COMPILING_DIRECT_MEDIACMD_SOURCE
//
// MetaData DataBase Access
//
/**
*/

/**
```

```
// Get database name
*/
uint32_t __stdcall vvwMetaBaseGetName(char * szDataBase);

/**
// Open a database file
*/
uint32_t __stdcall vvwMetaBaseOpen(char * szDataBase);

/**
// Create a new database file
*/
uint32_t __stdcall vvwMetaBaseCreate(char * szDataBase);

/**
// Close previously opened database
*/
uint32_t __stdcall vvwMetaBaseClose();

/**
// Get number of metadata file items in database (table count)
//  return -1 if not open or table doesn't exist
*/
uint32_t __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
*/
uint32_t __stdcall vvwMetaBaseFileName(uint32_t nIndex, char * szFileName, char * szClipName, char *
szUniqueID, char * szMetaFilter, BOOL bUseAnd, BOOL bCountDBs);

/**
// Remove a file item
//  return -1 if failed
*/
uint32_t __stdcall vvwMetaBaseFileRemove(char * szFileName, char * szClipName, char * szUnique);

/**
// Change file item name
//  return -1 if failed
*/
uint32_t __stdcall vvwMetaBaseFileRename(char * szUniqueID, char * szNewFileName);

/**
// Add new file item
//  return -1 if already exist
*/
uint32_t __stdcall vvwMetaBaseFileAdd(char * szFileName, char * szClipName, uint32_t lAddDefaults, uint32_t
lIgnoreGUID);
```

```
/**
* Get the index of given tag name, or -1 if !exist
*/
uint32_t __stdcall vvwMetaBaseTagIndex(char * szFileName, char * szMetaDataTag);

/**
*
*/
uint32_t __stdcall vvwOutputReport(char * szFileName, uint32_t nExportType);

/**
*
*/
uint32_t __stdcall vvwMetaBaseGetAllTags(char * szFileName, char * szMetaDataValue);
/**
// Get metadata item and its value for specified file item
//  return eof to determine if we reached the end
*/
uint32_t __stdcall vvwMetaBaseGetTag(char * szFileName, uint32_t nIndex, char * szMetaDataTag, char *
szMetaDataValue, uint32_t * plMetaDataValue, uint32_t * plMetaDataType);

/**
// Set metadata item and its value for specified file item
*/
uint32_t __stdcall vvwMetaBaseSetTag(char * szFileName, char * szMetaDataTag, char * szMetaDataValue,
uint32_t lMetaDataValue, uint32_t lMetaDataType);

uint32_t __stdcall vvwMetaBaseSavePicon(char * szDataBase, char * szPlay, char * szPicon);

/**
// Set metadata item and its value for specified file item
*/
uint32_t __stdcall vvwMetaBaseDefaultTags(char * szTable, char * szFileName);
uint32_t __stdcall vvwMetaBaseXOSTags(char * szTable);
uint32_t __stdcall vvwMetaBaseSMPTETags(char * szTable, char * szFileName);

/**
// Handle metabase commands
*/
uint32_t __stdcall vvwMetaBaseCmd(PMEDIACMD pCmd);
uint32_t __stdcall vvwMetaBaseDeletePicons();
uint32_t __stdcall vvwMetaBaseSetReplayMark(char * szFullTable, char * szMetaTag, char * szMetaVal, uint32_t
lPosition);
uint32_t __stdcall vvwMetaBaseGetReplayMark(char * szFullTable, char * szMetaTag, char * szMetaVal, uint32_t
* 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);

#endif

#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 V – MEDIACMD.java

The Java version of the MediaCMD interface. Available on request from Drastic.