

VW Interface Specification

©copyright 1995-2006, Drastic Technologies Ltd
Drastic Technologies
523 The Queensway, Suite 102
Toronto, ON, M8Y 1J7
CANADA
(416) 255 5636
(Fax) 255 8780
<http://www.drastictech.com>

Contents

VVW Interface Specification.....	1
Contents.....	2
Introduction.....	5
Components.....	9
References.....	10
Interface.....	11
JAVA.....	12
Channel Control.....	13
EnableChannels.....	13
GetMaxChannels.....	13
SetCurChannel.....	14
GetCurChannel.....	14
GetCurChannelName.....	14
Long vvwGetChannelName(long IChannel, char * szChanName).....	14
GetCurChannelType.....	15
long vvwGetChannelType (long IChannel).....	15
ShowConfigDialog.....	15
Network Specific.....	16
Connect.....	16
Disconnect.....	16
Transport Operations.....	17
Play.....	17
PlayAtSpeed.....	17
PlayFromTo.....	18
LoadClip.....	18
PlayClip.....	19
PlayClipFromTo.....	19
FastFoward.....	20
FastRewind.....	20
Pause.....	20
Seek.....	21
SeekRelative.....	21
Stop.....	21
Record.....	22
RecordFromTo.....	22
RecordStop (prepare record).....	22
SetRecordPresets.....	23
Eject.....	23
Special Commands.....	24
Transfer.....	24
Status Operation.....	25
UpdateStatus.....	25
GetState.....	26
GetFlags.....	26
GetSpeed.....	27

GetPosition.....	28
GetLastMs.....	28
GetStart.....	28
GetEnd.....	28
GetClipName.....	29
GetFileName.....	29
GetCurTC.....	29
GetCurState.....	30
Media Operations.....	31
Clip Mode Operations.....	31
GetNextClip.....	31
GetClipInfo.....	32
GetClipInfoEx.....	33
CopyClip.....	33
VTR Mode Operations.....	34
EDLResetToStart.....	34
EDLGetEdit.....	34
Shared Operations.....	36
GetLastChangeMs.....	36
Insert.....	37
Blank.....	38
Delete.....	39
Trim.....	40
Settings.....	41
ValueSupported.....	41
ValueGet.....	41
ValueSet.....	42
ValueSet2.....	42
Settings Format.....	42
Base Settings.....	43
ClipMode.....	43
Get/SetTCType.....	43
Get/SetTCSOURCE.....	43
Get/SetAutoMode.....	43
GetAvailablePresets.....	43
Audio Settings.....	44
Get/SetAudioInput.....	44
Get/SetAudioInputLevel.....	44
Get/SetAudioOutput.....	45
Get/SetAudioOutputLevel.....	45
GetAudioPeakRMS.....	45
Video Settings.....	45
Get/SetVideoInput.....	45
Get/SetVideoOutput.....	46
Get/SetVideoInputSetup.....	46
Get/SetVideoInputVideo.....	46
Get/SetVideoInputHue.....	47
Get/SetVideoInputChroma.....	47
Get/SetVideoTBCSetup.....	47

Get./SetVideoTBCVideo.....	47
Get/SetVideoTBCHue.....	47
Get/SetVideoTBCChroma.....	48
Get/SetVideoGenlock.....	48
Get/SetVideoGenlockSource.....	48
Error Log.....	48
vwwGetErrorLogMs.....	48
vwwSetErrorLog.....	48
vwwGetErrorLength.....	48
vwwGetError.....	48
System Settings.....	48
Get/SetCompressionRate.....	48
Get/SetSuperImpose.....	49
GetTotalTime.....	49
GetFreeTime.....	49
GetTotalStorage.....	49
GetFreeStorage.....	49
GetCurMs.....	49
GetChannelCapabilities.....	49
GetVWVersion.....	49
GetMRVersion.....	49
GetVWType.....	49
Utility Functions.....	50
FreeString.....	50
TCMaxFrame.....	50
TCToFrame.....	51
TCToString.....	52
VWSpeedToPercentage.....	52
PercentageToVWSpeed.....	52
Java Data Structure Definitions.....	53
ClipInfo.....	53
Data members	53
VTREditLine.....	53
GetValueMcmd.....	54
Data members	54
AvailabePresets.....	54
Appendix I – MediaCmd.h.....	55
Appendix II – MediaCmdX.h.....	55
Appendix III – VWWIF.h.....	55
Appendix IV – MediaCmdNet.h.....	72

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 VVW 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.

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)
- * ISpeed – 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 "VWV Interface Specification").

Installation

To properly work with the MediaCmd SDK, you should have a copy of the QuickClip software installed on your development system. Even if 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.

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

Type: Microsoft ActiveX/COM access method

Pros: Easy to program, 1:1 relationship with QuickClip/XO interface.

Cons: Uses same config as QuickClip/XO. Requires a local copy of QuickClip.

Setup: Register VWV.DLL using RegSvr32.exe in the QuickClip installation directory.

Issues: Difficult to use when communicating via TCP/IP within the same machine. Can be overcome by using the default pipe communication system, but this requires changes for remote network control.

Direct Link

Type: Direct link to VWV.DLL

Pros: No ActiveX layer, code compatible with Linux, Irix, Mac OS-X.

Cons: Uses default config from QuickClip/XO, application must be run in QuickClip directory. Requires a local copy of QuickClip.

Setup: Link to vvw.lib, include vvw.h. Copy application into the QuickClip directory before running

Issues: Needs access to VWV.dll and all its support DLLs/D1Xs. Still needs to be setup by LocalConfig.exe or QuickClip/XO

Network DLL

Type: Direct line to vvwNet2.dll

Pros: Consistent interface between local/remote and various OSs. Does not require a local copy of QuickClip.

Cons: Requires vvwNet2.dll and support dlls

Setup: Link to vvwNet2.lib, include vvwNet2.h. Copy dll set from SDK\bin directory with your application

Issues: Use the netOpenLocal function to avoid QuickClip configuration issues. Requires a few dlls to be added to you application installations. Does not run the client software automatically, so you application may need to start it, depending on what your application is doing.

Network Direct

Type: Direct compile of network sources in your app or your dll.

Pros: No extra dlls. Easy to customize and modify. Lots of comands already written.

Cons: You app needs to handle setup and may need to run QuickClip.exe/VVWServer.exe/QCRun.exe.

Setup: Copy source files from vvwNet2 into you project, modify and compile

Issues: Does not run the client software automatically, so you application may need to start it, depending on what your application is doing.

Manual

Type: Use the structures and defines to write your own communication and control layer.

Pros: This is required if you are using an unsupported development platform like PHP.

Cons: Everything has to be built and tested from the ground up.

Setup: None.

Issues: Unless you absolutely have to, this method is not recommended.

SDK Structure

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

- * \BIN – Copies of the minimum dll set from a QuickClip installation.
- * \LIB – Libraries required to link the vvwNet2.dll, examples and your application
- * \INC – Header files required to compile vvwNet2.dll, examples and your application
- * \Src\vwNet2 – 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

<http://www.drastictech.com/manuals/VVW%20Interface%20Specification.pdf>

*** Online version of the MediaCmd Documentation

<http://www.drastictech.com/manuals/VVW%20Interface%20Specification.html>

Low Level Header Documentation

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

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

HTTP XML AJAX Documentation

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

Components

The VVW interfaces are designed to allow VVW 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
vwwXXX - Hardware driver components for VVW-1000 through 7000 series DDRs
vwwNET - Direct network sender to VVW or user component (TCP/IP)
vwwDSync - House clock/SMPTE LTC, Local GPI Control

Controlling Components:

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

Interface Types:

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

Do not use this interface type (being removed):

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

References

MediaCmd.h	- Internal media cmd structures on which this interface is based.
VWIF.h	- VW Direct Access (includes previous access method as well)
VWX.tlb	- MediaCmdX type library
MediaCmd.xml	- XML command set mirroring MediaCmd.h
CntrlVW_VB1	- Simple control from Visual Basic
CntrlVW_MFC1	- Simple control from Visual C++ with MFC
CntrlVW_Java	- Simple control from Visual J++/WFC
CntrlVW_Cmd	- Command line controller using the DLL methods
VTRControl	- Test app for VTR control
LocalConfig	- VW Setup
X-extdev.prm	- VTR plug in for Adobe Premiere
(QuickClipProSW	- Coming soon)

Interface

The following is assumed:

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

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

Nomenclature:

szClipName. Sz8CharClipName, etc

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

```
Char szClipName[9] = "";
```

// Include unused 1 char safety

```
Char pClipName = new char[9];  
Char szpClipName = malloc(9);
```

SzFileName, sz260FileName, etc

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

```
Char szClipName[261] = "";
```

// Include unused 1 char safety

```
Char pClipName = new char[261];
```

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

Time code strings

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

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

JAVA

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

```
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 VVWInterface to return multiple values in the defined class structure.

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

Channel Control

EnableChannels

ActiveX VB	vbx.EnableChannels (IInternal0_31 As Long, IInternal32_63 As Long, IExternal64_95 As Long, IExternal96_127 As Long, INetwork128_159 As Long, INetwork160_192 As Long) As Long
ActiveX C++	long pcx->EnableChannels (long IInternal0_31, long IInternal32_63, long IExternal64_95, long IExternal96_127, long INetwork128_159, long INetwork160_192)
Java	<i>Not Available</i>
Dll	long vvw EnableChannels (long IInternal0_31, long IInternal32_63, long IExternal64_95, long IExternal96_127, long INetwork128_159, long INetwork160_192)
Unix	<i>Not Available</i>
XML	<i>Not Available</i>

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

GetMaxChannels

ActiveX VB	vbx.GetMaxChannels () As Long
ActiveX C++	long pcx->GetMaxChannels ()
Java	long mci.GetMaxChannels ()
Dll	long vvwGetMaxChannels (long IChannel)
Unix	long GetMaxChannels ()
XML	<i>Not implemented</i>

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 IChannel parameter for the DLL interface.

SetCurChannel

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

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

Returns 0 or an error code

GetCurChannel

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

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

GetCurChannelName

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

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

GetCurChannelType

ActiveX VB	vbv.GetCurChannelType () As Long
ActiveX C++	long pcx->GetCurChannelType ()
Java	long mci.GetCurChannelType ()
Dll	long vvwGetChannelType (long IChannel)
Unix	long GetCurChannelType ()
XML	<i>Not supported</i>

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

VVW_CHANATYPE_HOUSE	0x1	1
VVW_CHANATYPE_INTERNAL	0x2	2
VVW_CHANATYPE_VTR_DDR	0x4	4
VVW_CHANATYPE_UNKNOWN	0xFFFFFFFF	-1

ShowConfigDialog

ActiveX VB	vbv.ShowConfigDialog (hWnd as Long) As Long
ActiveX C++	long pcx->ShowConfigDialog (long hWnd)
Java	<i>Not Available</i>
Dll	long vvw ShowConfigDialog (long hWnd)
Unix	<i>Not Available</i>
XML	<i>Not Available</i>

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

Network Specific

Connect

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

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

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

Disconnect

ActiveX VB	<i>Not implemented</i>
ActiveX C++	<i>Not implemented</i>
Java	boolean mci.Disconnect()
Dll	<i>Not implemented</i>
Unix	Long Disconnect()
XML	<i>Not required</i>

Disconnect from a previously connected server. For ActiveX and Dll access, please setup 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 ()
Dll	long vvwPlay (long IChannel)
Unix	long Play ()
XML	http://localhost/VVWXMLMediaCmd?Play

Play at normal speed.

Returns 0 if successful, else an error code.

PlayAtSpeed

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

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

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

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

Returns 0 if successful, else an error code.

PlayClip

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

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

Returns 0 if successful, else an error code.

PlayClipFromTo

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

FastForward

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

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

FastRewind

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

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

Pause

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

Stop playback and display the current frame.
Returns 0 if successful, else an error code.

Seek

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

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 IFrame, the seek is complete.

Returns 0 if successful, else an error code.

SeekRelative

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

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

Stop

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

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

Returns 0 if successful, else an error code.

Record

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

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

Returns 0 if successful, else an error code.

RecordFromTo

ActiveX VB	vbx.RecordFromTo (IFrom As Long, ITo As Long) As Long
ActiveX C++	long pcx-> RecordFromTo (long IFrom, long ITo)
Java	long mci. RecordFromTo (long IFrom, long ITo)
Dll	long vvwRecordFromTo (long IChannel, long IFrom, long ITo)
Unix	long RecordFromTo (long IFrom, long ITo)
XML	http://localhost/VVWXMLMediaCmd?Record&start=IStart&end=IEnd

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 IFrom - 1.

Returns 0 if successful, else an error code.

RecordStop (prepare record)

ActiveX VB	Vbx.RecordStop (szClipName As String, IDuration As Long) As Long
ActiveX C++	Long pcx-> RecordStop (BSTR szClipName, long IDuration)
Java	Long mci. RecordStop (String szClipName, long IDuration)
Dll	Long vvwRecordStop (long IChannel, char * szClipName, long IDuration)
Unix	Long RecordStop (char * szClipName, long IDuration)
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 IDuration 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 (IVidEdit As Long, IAudEdit As Long IInfEdit As Long) As Long
ActiveX C++	long pcx-> SetRecordPresets (long IVidEdit, long IAudEdit, long IInfEdit)
Java	long mci. SetRecordPresets (long IVidEdit, long IAudEdit, long IInfEdit)
Dll	long vvwSetRecordPresets (long IVidEdit, long IAudEdit, long IInfEdit)
Unix	long SetRecordPresets (long IVidEdit, long IAudEdit, long IInfEdit)
XML	http://localhost/VVXMLMediaCmd?Record &videochannels=IVidEdit&audiochannels=IAudEdit &infochannels=IInfEdit

Set the channels to record. Using -1 values implies that the Preset should be set to all available channels. Record Presets will remain set until the user changes them. Returns 0 if successful, else an error code.

Eject

ActiveX VB	vbx.Eject () As Long
ActiveX C++	long pcx-> Eject ()
Java	long mci. Eject ()
Dll	long vvwEject (long IChannel)
Unix	long Eject ()
XML	http://localhost/VVXMLMediaCmd?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	vb.Transfer (ITargetChannel As Long, IPosition As Long, IStart As Long, IEnd As Long, IVidEdit As Long, IAudEdit As Long, IInfEdit As Long, szClipName As String, fToTape As Boolean) As Long
ActiveX C++	long pcx-> Transfer (long ITargetChannel, long IPosition, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fToTape)
Java	long mci. Transfer (long ITargetChannel, long IPosition, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, Boolean fToTape)
DII	long vvwTransfer (long IChannel, long ITargetChannel, long IPosition, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fToTape)
Unix	long Transfer (long ITargetChannel, long IPosition, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, VVWBOOL fToTape)
XML	http://localhost/VVWXMLMediaCmd?Transfer &channel=ITargetChannel&position=IPosition &start=IStart&end=IEnd&videochannels=IVidEdit &audiochannels=IAudEdit&infochannels=IInfEdit &Flags=Invert

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

Returns 0 if successful, else an error code.

Status Operation

UpdateStatus

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

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

Returns 0 if successful, else an error code.

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

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

GetState

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

Returns the current state

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

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

GetFlags

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

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

GetSpeed

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

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

$$D1Speed = ((double)VVWSpeed / 65520.0)$$

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

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

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

Typical VVW speeds (note speeds are linear):

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

GetPosition

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

Returns the current position if the cfUsePosition flag is set, otherwise invalid.

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

GetLastMs

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

Returns the millisecond time the last status occurred (time of the last vertical blank).

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

GetStart

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

Returns the current start or in point if the cfUseStart flag is set.

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

GetEnd

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

Return the current end point or out point if cfUseEnd is set.

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

GetClipName

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

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

GetFileName

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

Returns the current file name, if any. For dll 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 ()
Dll	Long GetCurTC (long IChannel, char * sz14ByteTC)
Unix	Char * GetCurTC ()
XML	http://localhost/VVWXMLGetStatus?

Returns the current time code as a string (e.g. "00:01:00:00"). For dll access, the memory must always be at least 15 bytes long (14 byte time code plus id + NULL) and is always ANSI.

XML: See <MediaCmd> root element, <Positon> sub-element (text)

GetCurState

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

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

Media Operations

Clip Mode Operations

GetNextClip

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

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

Returns 0 if successful, else an error code.

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

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

GetClipInfo

ActiveX VB	Vbx. GetClipInfo (szClipName As String, ByRef IStart As Long, ByRef IEnd As Long, ByRef IVidEdit As Long, ByRef IAudEdit As Long, ByRef IInfEdit As Long, szFileName As String) As Long
ActiveX C++	Long pcx-> GetClipInfo (BSTR szLastClip, long * IStart, long * IEnd, long * IVidEdit, long * IAudEdit, long * IInfEdit, BSTR * szFileName)
Java	Long mci. GetClipInfo (mci.ClipInfo clipData)
Dll	Long vvwGetClipInfo (long IChannel, char * sz8CharClipName, long * IStart, long * IEnd, long * IVidEdit, long * IAudEdit, long * IInfEdit, char * sz260CharFileName)
Unix	Long GetClipInfo (char * sz8CharClipName, long * IStart, long * IEnd, long * IVidEdit, long * IAudEdit, long * IInfEdit, char * sz260CharFileName)
XML	http://localhost/VVWXMLNextClip?

Returns the basic information from szClip. The information is located in IStart, IEnd, IVidEdit, IAudEdit 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, IStart, IEnd, IVidEdit, IAudEdit, IInfEdit, 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	vb. GetClipInfoEx (szClipName As String, ByRef ICreation As Long, ByRef ILastModification As Long, ByRef IFileSize As Long, ByRef IDiskFragments) As Long
ActiveX C++	Long pcx-> GetClipInfoEx (BSTR szClipName, long * ICreation, long * ILastModification, long * IFileSize, long * IDiskFragments)
Java	<i>Not Implemented</i>
Dll	Long vvwGetNextClipEx (long IChannel, char * szClipName, long * ICreation, long * ILastModification, long * IFileSize, long * IDiskFragments)
Unix	Long GetNextClipEx (char * szClipName, long * ICreation, long * ILastModification, long * IFileSize, long * IDiskFragments)
XML	Not supported

Returns the extended information from szClip. The information is located in IStart, IEnd, IVidEdit, IAudEdit 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	vb. CopyClip (szSourceClip As String, szDestClip As String, IStart As Long, IEnd As Long) As long
ActiveX C++	Long pcx-> CopyClip (BSTR szSourceClip, BSTR szDestClip, long IStart, long IEnd)
Java	Long mci. CopyClip (String szSourceClip, String szDestClip, long IStart, long IEnd)
Dll	Long vvwCopyClip (long IChannel, char * szSourceClip, char * szDestClip, long IStart, long IEnd)
Unix	Long CopyClip (char * szSourceClip, char * szDestClip, long IStart, long IEnd)
XML	Not supported

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

Returns 0 if successful, else an error code.

VTR Mode Operations

EDLResetToStart

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

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

EDLGetEdit

ActiveX VB	Vbx. EDLGetEdit (ByRef IRecordIn As Long, ByRef IPlayIn As Long, ByRef IPlayOut As Long, ByRef IVidEdit As Long, ByRef IAudEdit As Long, ByRef IInfEdit As Long, ByRef szClipName As String, ByRef szFileName As Long) As Long
ActiveX C++	Long pcx-> EDLGetEdit (long * IRecordIn, long * IPlayIn, long * IPlayOut, long * IVidEdit, long * IAudEdit, long * IInfEdit, BSTR * szClipName, BSTR * szFileName)
Java	long mci. EDLGetEdit (VTREditLine editInfo)
Dll	long vvwEDLGetEdit (long IChannel, long * IRecordIn, long * IPlayIn, long * IPlayOut, long * IVidEdit, long * IAudEdit, long * IInfEdit, char * sz8CharClipName, char * sz260CharFileName)
Unix	Long EDLGetEdit (long * IRecordIn, long * IPlayIn, long * IPlayOut, long * IVidEdit, long * IAudEdit, long * IInfEdit, char * szClipName, char * szFileName)
XML	http://localhost/VVWXMLInfo?position=0 &videochannels=0&audiochannels=0&infochannels=0

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

Returns 0 if successfule else an Error code.

Java: This method requires the user to instantiate a new object of type VTREditLine. The IRecordIn, IPlayIn, IPlayOut, IVidEdit, IAudEdit, IInfEdit, szClipName, and szFileName values are returned in the object's instance variables of the same name.

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

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

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

Shared Operations

GetLastChangeMs

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

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

Insert

ActiveX VB	vbx. Insert (szClipName As String, szFileName As String, IPosition As Long, IStart As Long, IEnd As Long, IVidEdit As Long, IAudEdit As Long, IInfEdit As Long, fRipple As Boolean) As Long
ActiveX C++	long pcx-> Insert (BSTR szClipName, BSTR szFileName, long IPosition, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fRipple)
Java	long mci. Insert (String szClipName, String szFileName, long IPosition, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, Boolean fRipple)
Dll	long vvwInsert (long IChannel, char * szClipName, char * szFileName, long IPosition, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fRipple)
Unix	long Insert (char * szClipName, char * szFileName, long IPosition, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, VVWBOOL fRipple)
XML	http://localhost/VVWXMLMediaCmd?Insert &ClipID=szClipName&position=IPosition &start=IStart&end=IEnd&videochannels=IVidEdit &audiochannels=IAudEdit&infochannels=IInfEdit &Flags=Ripple

Internal Channels Only. Do not use yet.

Blank

ActiveX VB	vbx. Blank (szClipName As String, IStart As Long, IEnd As Long, IVidEdit As Long, IAudEdit As Long, IInfEdit As Long, fRipple As Boolean) As Long
ActiveX C++	long pcx-> Blank (BSTR szClipName, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fRipple)
Java	long mci. Blank (String szClipName, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, Boolean fRipple)
Dll	long vvwBlank (long IChannel, char * szClipName, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fRipple)
Unix	long Blank (char * szClipName, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, VVWBOOI fRipple)
XML	http://localhost/VVXMLMediaCmd?Blank &ClipID=szClipName&position=IPosition &start=IStart&end=IEnd&videochannels=IVidEdit &audiochannels=IAudEdit&infochannels=IInfEdit &Flags=Ripple

Internal Channels Only. Do not use yet.

Delete

ActiveX VB	vbx. Delete (szClipName As String, IStart As Long, IEnd As Long, IVidEdit As Long, IAudEdit As Long, IInfEdit As Long, fRipple As Boolean) As Long
ActiveX C++	long pcx-> Delete (BSTR szClipName, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fRipple)
Java	long mci. Delete (String szClipName, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, Boolean fRipple)
Dll	long vvwDelete (long IChannel, char * szClipName, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fRipple)
Unix	long Delete (char * szClipName, long IStart, long IEnd, long IVidEdit, long IAudEdit, long IInfEdit, VVWBOOI fRipple)
XML	http://localhost/VVXMLMediaCmd?Delete &ClipID=szClipName&position=IPosition &start=IStart&end=IEnd&videochannels=IVidEdit &audiochannels=IAudEdit&infochannels=IInfEdit &Flags=Ripple

Internal Channels Only. Do not use yet.

Trim

ActiveX VB	vbx. Trim (IPosition As Long, IStartOffset As Long, IEndOffset As Long, IVidEdit As Long, IAudEdit As Long, IInfEdit As Long, fRipple As Boolean) As Long
ActiveX C++	long pcx-> Trim (long IPosition, long IStartOffset, long IEndOffset, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fRipple)
Java	long mci. Trim (long IPosition, long IStartOffset, long IEndOffset, long IVidEdit, long IAudEdit, long IInfEdit, Boolean fRipple)
Dll	long vvwTrim (long IChannel, long IPosition, long IStartOffset, long IEndOffset, long IVidEdit, long IAudEdit, long IInfEdit, BOOL fRipple)
Unix	long Trim (long IPosition, long IStartOffset, long IEndOffset, long IVidEdit, long IAudEdit, long IInfEdit, VVWBOOL fRipple)
XML	http://localhost/VVXMLMediaCmd?Trim &position=IPosition&start=IStartOffset &end=IEndOffset&videochannels=IVidEdit &audiochannels=IAudEdit&infochannels=IInfEdit &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	vb. ValueSupported (IValueType As Long) As Long
ActiveX C++	long pcx-> ValueSupported (long IValueType)
Java	long mci. ValueSupported (long IValueType)
Dll	long vvwValueSupported (long IChannel, long IValueType)
Unix	long ValueSupported (long IValueType)
XML	http://localhost/ValueSupported&cmdalt=valuetype &position=IValueType

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

ValueGet

ActiveX VB	vb. ValueGet (IValueType As Long, ByRef IMin As Long, ByRef IMax As Long) As Long
ActiveX C++	long pcx-> ValueGet (long IValueType, long * pMin, long * pMax)
Java	long mci. ValueGet (GetValueMcmd mCmdValues)
Dll	long vvwValueGet (long IChannel, long IValueType, long * pMin, long * pMax)
Unix	long ValueGet (long IValueType, long * pMin, long * pMax)
XML	http://localhost/GetValue&cmdalt=valuetype &position=IValueType

Returns the current setting for a get/set value.

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

ValueSet

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

Sets the get/set value to setting.

ValueSet2

ActiveX VB	vbx. ValueSet2 (IValueType As Long, ISetting As Long, IStart As Long, IEnd As Long, IVidChannel As Long, IAudChannel As Long, IInfChannel As Long) As Long
ActiveX C++	long pcx-> ValueSet2 (long IValueType, long ISetting, long IStart, long IEnd, long IVidChannel, long IAudChannel, long IInfChannel)
Java	long mci. ValueSet2 (long IValueType, long ISetting, long IStart, long IEnd, long IVidChannel, long IAudChannel, long IInfChannel)
Dll	long vvwValueSet2 (long IChannel, long IValueType, long ISetting, long IStart, long IEnd, long IVidChannel, long IAudChannel, long IInfChannel)
Unix	long ValueSet2 (long IValueType, long ISetting, long IStart, long IEnd, long IVidChannel, long IAudChannel, long IInfChannel)
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. GetXXX () As Long
	Vbx. SetXXX (ISetting As Long) As Long
ActiveX C++	Long pcx-> GetXXX ()
	Long pcx-> SetXXX (long ISetting)
Java	Long mci. GetXXX ()
	Long mci. SetXXX (long ISetting)
Dll	Long vvwGetXXX ()
	Long vvwSetXXX (long ISetting)
Unix	Long GetXXX ()
	Long SetXXX (long ISetting)

Base Settings

ClipMode

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 IVidEdit, IAudEdit, IInfEdit

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.

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=ISetting &videochannels=0&audiochannels=IAudChannels&infochannels=0)

Get/SetAudioOutput

Get the current audio Output – See Get/SetAudioInput

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

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

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

Get/SetAudioOutputLevel

Get the current audio output level.

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

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

(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsAudOutputLevel&position=ISetting &videochannels=0&audiochannels=IAudChannels&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 HDS DI 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_SD TI 0x400
//! No video available or no configurable settings (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_NONE 0
(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidInSelect
&position=ISetting)

```

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=ISetting)

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=ISetting)

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=ISetting)

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=ISetting)

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=ISetting)

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 Video in the GetValueMcmd object.

(XML: localhost/XMLMediaCmd?SetValue&cmdalt=gsVidVideo&position=ISetting)

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=ISetting)

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=ISetting)

Get/SetVideoGenlock

Turn the house/reference lock on or off

Get/SetVideoGenlockSource

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

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

GetTotalTime

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

GetFreeTime

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

GetTotalStorage

Returns the total storage connected in megabytes.

GetFreeStorage

Returns the amount of available storage for recording in megabytes.

GetCurMs

Get the current millisecond time.

GetChannelCapabilities

Get the available commands for a channel.

GetVWVersion

Returns the version string of the VW subsystem.

GetMRVersion

Returns the version string of the MediaReactor subsystem.

GetVWType

Returns the type string of the VW channel.

Utility Functions

FreeString

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

Free a string value returned by the channel.

TCMaxFrame

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

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

TCToFrame

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

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

Flags:

```

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

TC2_FTYPE_FIELD          0x10000000 // Field based (else frame)

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

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

```

TCToString

ActiveX VB	vb. TCToString (ITC As Long, IFlags As Long) As String
ActiveX C++	BSTR pcx-> TCToString (long ITC, long IFlags)
Java	String mci. TCToString (long ITC, long IFlags)
Dll	<i>Use TCXlat</i>
Unix	Char * TCToString (long ITC, long IFlags)
XML	<i>Not supported</i>

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

VVWSpeedToPercentage

ActiveX VB	vb. VVWSpeedToPercentage (IVVWSpeed As Long) As Double
ActiveX C++	Double pcx-> VVWSpeedToPercentage (long IVVWSpeed)
Java	Double mci. VVWSpeedToPercentage (long IVVWSpeed)
Dll	<i>Not implemented</i>
Unix	Double VVWSpeedToPercentage (long IVVWSpeed)
XML	<i>Not supported</i>

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

PercentageToVVWSpeed

ActiveX VB	vb. PercentageToVVWSpeed (double ddPercentageSpeed) As Long
ActiveX C++	Long pcx-> PercentageToVVWSpeed (double ddPercentageSpeed)
Java	Long mci. PercentageToVVWSpeed (double ddPercentageSpeed)
Dll	<i>Not implemented</i>
Unix	Long PercentageToVVWSpeed (double ddPercentageSpeed)
XML	<i>Not supported</i>

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

Java Data Structure Definitions

ClipInfo

Java	<pre>MediaCmdIF.ClipInfo clipData = new MediaCmdIF.ClipInfo(szClipName)</pre>
------	---

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 `IStart`: the in point of the inquired clip.

long `IEnd`: the out point of the inquired clip.

long `IvidEdit`: number of video channels available for the requested clip.

long `IaudEdit`: number of audio channels available for the inquired clip.

long `IinfEdit`: number of info channels available for the inquired clip.

String `szFileName`: file name of the requested clip.

VTREditLine

Java	<pre>MediaCmdIF.VTREditLine editInfo = new MediaCmdIF.VTREditLine();</pre>
------	--

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 `IRecordIn`: time code on the VTR edit line.

long `IPlayIn`: the in point of the VTR edit line.

long `IPlayOut`: the out point of the VTR edit line.

long `IvidEdit`: number of video channels available for the VTR edit line.

long `IaudEdit`: number of audio channels available for the VTR edit line.

long `IinfEdit`: 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	<code>MediaCmdIF.GetValueMcmd mCmdValues = new MediaCmdIF.GetValueMcmd(setValueType, pMin, pMax)</code>
------	---

For use with the `mci.ValueGet` method. Must construct this data structure with a valid `IValueType`. 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 IValueType`: the get/set value. Used by the `mci.ValueGet` method to determine which setting to retrieve.

`long pMin`: stores the min value for a get/set value if instantiated with a value different than `-1`.

`long pMax`: stores the max value for a get/set value if instantiated with a value different than `-1`.

AvailablePresets

Java	<code>MediaCmdIF.AvailablePresets presets = new MediaCmdIF.AvaiblePresets()</code>
------	--

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

Data members

`long pVidEdit`: supported video presets for a channel

`long pAudEdit`: supported audio presets for a channel

`long pInfEdit`: supported info presets for a channel

Appendix I – MediaCmd.h

Please see included file MediaCmd.chm (Doxygen)

Appendix II – MediaCmdX.h

Please see help file MediaCmd.chm (JavaDoc)

Appendix III – VVWIF.h

```
/**
 * Convert a 0..3 channel to 0, 64, 65536
 * Mostly internal
 * Undocumented
 */
void* __stdcall VvwChannelToHandle(long lChannel);

/**
 * Convert a 0, 64, 65536 to 0..3
 * Mostly internal
 * Undocumented
 */
long __stdcall VvwHandleToChannel(void* hVvw);

/**
 Enable or disable channels based on the bit arrays 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, pdetics or Louth protocol. The remaining channels
 are for controlling other devices through the network. Please note that a network channel
 controls all the channels on the network server box, so disabling one network connection
 may disable more than one channel. Always call GetMaxChannels() after setting the bits to
 make sure all the channels you expect exist actually exist. This should be the first call
 made to the activeX component.
 */
long __stdcall VvwEnableChannels(long lInternal0_31,
                                long lInternal32_63,
                                long lExternal64_95,
                                long lExternal96_127,
                                long lNetwork128_159,
                                long lNetwork160_191);
```

```

/**
 * Release memory allocated to channels
 */
long __stdcall VwReleaseChannels(void);
/**
 Returns the maximum number of channels available for control. Channels start at 0 and
 end at max_channels-1. This return is one greater than the largest value available for
 SetCurChannel(), GetCurChannel() and the lChannel parameter for the DLL interface.
 */
long __stdcall VwGetMaxChannels(void);

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

/**
 Returns the basic type of the channel (VTR, Internal, User, House)
      VW_CHAN_TYPE_HOUSE          0x1          1
      VW_CHAN_TYPE_INTERNAL       0x2          2
      VW_CHAN_TYPE_VTR_DDR        0x4          4
      VW_CHAN_TYPE_UNKNOWN        0xFFFFFFFF -1
 */
long __stdcall VwGetChannelType(long lChannel);

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

//
// Transport
//

/**
 Play at normal speed.
      Returns 0 if successful, else an error code.
 */
long __stdcall VwPlay(long lChannel);

/**
 Play at a particular VW speed. VW 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 VW speeds using the
 PercentageToVwSpeed() function. For speed calculations please see GetSpeed() below.

```

```

        Returns 0 if successful, else an error code.
*/
long __stdcall VwPlayAtSpeed(long lChannel, long lVwSpeed);

/**
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

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

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

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

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

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

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

```

```

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

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

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

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

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

/**
Record from a frame value to a frame value. As with editing systems, the 'from' point is
included and will be recorded but the to point is NOT included and will not be recorded.
This means that the last frame recorded will be lFrom - 1.
Returns 0 if successful, else an error code.
*/
long __stdcall VwRecordFromTo(long lChannel, long lFrom, long lTo);

/**
Clip Mode Only. Set the clip name and length of time to record in frames. The record will
not actually start until Record() is called. If the lDuration is set to -1 the record will
continue until Stop() is called or the channel runs out of space.
Returns 0 if successful, else an error code.

```

```

*/
Long__stdcallVwRecordStop(long lChannel,char * sz8CharClipName, long lDuration);

/**
Set the channelsto record.Using -1values implies that the Presets shouldbe set to all
availablechannels.Record Presetswill remain set untilthe user changes them.
Returns0 ifsuccessful,elsean errorcode.
*/
Long__stdcallVwSetRecordPresets(long lChannel, long lVidEdit, long lAudEdit, long lInfEdit);

/**
Ejectthe currentmedia ifit is removable. Normallyonlyused withVTRs.
Returns0 ifsuccessful,elsean errorcode.
*/
Long__stdcallVwEject(long lChannel);

/**
* MediaCmd directaccess
*/
Long__stdcallVwMediaCmd(long lChannel, void* pMediaCmd);

//
// SpecialCommands (lChannelmust be vtrtype, lTargetChannelmust be internaltype)
//
/**
Transfermedia from one channelto another. Onlysupportedby VTR channels. Currently
only implementedforVTR to internalchannelor internalchannelsto VTR channels. To
recordmedia from a VTR, the fToTape shouldbe false,to recordmedia onto a VTR the
fToTape shouldbe true.The startand end pointare from the playbackdevice.The editwill
occur at the currenttimecode locationon the recorder.
Returns0 ifsuccessful,elsean errorcode.
*/
Long__stdcallVwTransfer(long lChannel, long lTargetChannel, long lPosition, long lStart, long
lEnd, long lVidEdit, long lAudEdit, long lInfEdit, char * szClipName, BOOL fToTape);

/**
Retrieve the currentstatus from the controlleddevice. The status is automaticallyupdated
by the interface, but thiscallensures that the status is currentwhen you are checkingit.
Returns0 ifsuccessful,elsean errorcode.
*/
Long__stdcallVwUpdateStatus(long lChannel);

/**
Returns the currentstate

ctStop      0          // Stop allaction
ctPause     1          // Pause, Seek
ctPlay      2          // Play at specifiedspeed (includespause)
ctRecord    3          // Record at specifiedspeed

```

```

ctRecStop      4          // Stop ready for recording
ctEject        5          // Eject the current media
ctError        17         // An error has occurred
ctAbort        19         // Abort any queued commands
*/

```

```

long __stdcall VfwGetState(long lChannel);

```

```

/**

```

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 ttc
cfUseEnd= 1 << 10,           // 0x00000400 End command as specified
cfUseEndOffset= 1 << 11,     // 0x00000800 End is an offset from current ttc
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 media cmd is required
*/

```

```

long __stdcall VfwGetFlags(long lChannel);

```

```

/**

```

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

$D1Speed = ((double)VwSpeed / 65520.0)$

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

$Dpercent = (((double)VwSpeed * 100.0) / 65520.0)$
 $= ((double)VwSpeed / 655.2)$

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

Typical Vw speeds (note speeds are linear):

Pause	0%	0
Play	100%	65520
HalfPlay	50%	32760
Rev Play-100%		-65520
Rev 2 x Play-200%		131040
10 x Play	1000%	655200
Max Play	90000%	5896800
Max Rev	-90000%	-5896800

*/

`long __stdcall VwGetSpeed(long lChannel);`

/**

Returns the current position if the cfUsePosition flag is set, otherwise invalid.

*/

`long __stdcall VwGetPosition(long lChannel);`

/**

Returns the millisecond time the last status occurred (time of the last vertical blank).

*/

`long __stdcall VwGetLastMs(long lChannel);`

/**

Returns the current start or in point if the cfUseStart flag is set.

*/

`long __stdcall VwGetStart(long lChannel);`

/**

Return the current end point or out point if cfUseEnd is set.

*/

`long __stdcall VwGetEnd(long lChannel);`

/**

Only supported in clip mode. Returns the current clip name, if any. For dll access, the memory must be at least 9 bytes long (8 character bytes + NULL) and is always ANSI.

*/

`long __stdcall VwGetClipName(long lChannel, char * sz8CharClipName);`

/**

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

*/

`long __stdcall VwGetFileName(long lChannel, char * sz260CharFileName);`

```

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

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

//
// MediaOperation
//

// ClipMode
/**
ClipMode Only. Returns the next clip identifier. To get the first clip, szLastClip should be
an empty string. Once the last clip available has been returned, GetNextClip will return an
error or NULL for unix/dll access. Please note: For unix/dll access, the
sz8CharLastClipCurClip memory area is used for the new clip. The previous clip name is
therefore lost and the memory is not allocated by the vwv.
Returns 0 if successful, else an error code.
*/
char * __stdcall VwvGetNextClip(long lChannel, 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 filename respectively.
Returns 0 if successful, else an error code.
*/
long __stdcall VwvGetClipInfo(long lChannel, char * sz8CharClipName, long * lStart, long *
lEnd, long * lVidEdit, long * lAudEdit, long * lInfEdit, char * szFileName);

/**
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.
*/
long __stdcall VwvGetNextClipEx(long lChannel, char * sz8CharClipName, long * lCreation,
long * lLastModification, long * lFileSize, long * lDiskFragments);

/**
Create a virtual copy of a clip, changing the in and out points if necessary. To use the
whole clip, set lStart to 0 and the end to -1.
Returns 0 if successful, else an error code.
*/

```

```

*/
long__stdcallvwCopyClip(long lChannel,char * szSourceClip,char * szDestClip,long lStart,
long lEnd);

// VTR Mode
/**
Reset the edl returns in VTR mode to the first element of the list.
*/
long__stdcallvwEDLResetToStart(long lChannel);

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

// Shared
/**
Returns the millisecond time of the last change in the transfer queue
*/
long__stdcallvwGetLastChangeXferMs(long lChannel);
/**
Returns the millisecond time of the last change in the current mode (clip or vtr).
*/
long__stdcallvwGetLastChangeMs(long lChannel);

/**
*/
long__stdcallvwInsert(long lChannel,char * szClipName, char * szFileName, long lPosition,
long lStart,long lEnd, long lVidEdit,long lAudEdit,long lInfEdit,BOOL fRipple);

/**
*/
long__stdcallvwBlank (long lChannel,char * szClipName, long lStart,long lEnd, long
lVidEdit,long lAudEdit,long lInfEdit,BOOL fRipple);

/**
*/
long__stdcallvwDelete(long lChannel,char * szClipName, long lStart,long lEnd, long
lVidEdit,long lAudEdit,long lInfEdit,BOOL fRipple);

/**
*/

```

```

long__stdcallvwTrim (long lChannel, long lPosition, long lStartOffset, long lEndOffset, long
lVidEdit, long lAudEdit, long lInfEdit, BOOL fRipple);

//
// Settings
//
/**
Returns the supported attributes of a get/set value (gsClipMode, gsTcSource, etc) or -1 for
not supported.
*/
long__stdcallvwValueSupported(long lChannel, long lValueType);

/**
Returns the current setting for a get/set value.
*/
long__stdcallvwValueGet(long lChannel, long lValueType, long * pMin, long * pMax);

/**
Sets the get/set value to setting.
*/
long__stdcallvwValueSet(long lChannel, long lValueType, long lSetting);

/**
Sets the get/set value to setting with extended parameters. Please set unused parameters
to NULL.
*/
long__stdcallvwValueSet2(long lChannel, long lValueType, long lSetting, long lStart, long
lEnd, long lVidChan, long lAudChan, long lInfChan);

/**
Calls valuexxx with gsClipMode. If equal to 1 then the channel is in clip mode, if 0 the
channel is in VTR mode.
*/
long__stdcallvwGetClipMode(long lChannel);

/**
Calls valuexxx with gsClipMode. If equal to 1 then the channel is in clip mode, if 0 the
channel is in VTR mode.
*/
long__stdcallvwSetClipMode(long lChannel, long lSetting);

/**
Calls valuexxx with gsTcType (drop frame, non drop frame, pal).
#TC2_TCTYPE_FILM, #TC2_TCTYPE_NDF, #TC2_TCTYPE_DF, #TC2_TCTYPE_PAL, #TC2_TCTYPE_50,
#TC2_TCTYPE_5994, #TC2_TCTYPE_60, #TC2_TCTYPE_NTSCFILM, #TC2_TCTYPE_2398,
#TC2_TCTYPE_100
*/
long__stdcallvwGetTcType(long lChannel);

/**
Calls valuexxx with gsTcType (drop frame, non drop frame, pal).

```

```

#TC2_TCTYPE_FILM,#TC2_TCTYPE_NDF,#TC2_TCTYPE_DF,#TC2_TCTYPE_PAL,#TC2_TCTYPE_50,
#TC2_TCTYPE_5994,#TC2_TCTYPE_60,#TC2_TCTYPE_NTSCFILM,#TC2_TCTYPE_2398,
#TC2_TCTYPE_100
*/
long__stdcallvwSetTcType(long lChannel, long lSetting);

/**
CallsvaluexxxwithgsTcSource (VITC,LTC,Control,Clip).
#GS_TCSOURCE_LTC,#GS_TCSOURCE_VITC,#GS_TCSOURCE_CTLor #GS_TCSOURCE_CLIP
*/
long__stdcallvwGetTcSource(long lChannel);
/**
CallsvaluexxxwithgsTcSource (VITC,LTC,Control,Clip).
#GS_TCSOURCE_LTC,#GS_TCSOURCE_VITC,#GS_TCSOURCE_CTLor #GS_TCSOURCE_CLIP
*/
long__stdcallvwSetTcSource(long lChannel, long lSetting);

/**
CallsvaluexxxwithgsAutoMode. Requiredforplaylists,deferredcommands and auto edit
commands on VTRs.
*/
long__stdcallvwGetAutoMode(long lChannel);
/**
CallsvaluexxxwithgsAutoMode. Requiredforplaylists,deferredcommands and auto edit
commands on VTRs.
*/
long__stdcallvwSetAutoMode(long lChannel, long lSetting);

/**
ADD FUNCTIONS lVidEdit, lAudEdit, lInfEdit
Returns the supported audio, video and info presets for a channel.
*/
long__stdcallvwGetAvailablePresets(long lChannel, long * p lVidEdit, long * p lAudEdit, long *
p lInfEdit);

/**
ADD FUNCTION lAudIn
Get the current audio input.
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
#GS_AUDSELECT_BALANCED_10 #GS_AUDSELECT_BALANCED_4 #GS_AUDSELECT_SPDIF
#GS_AUDSELECT_AES_EBU
#GS_AUDSELECT_EMBEDDED
*/
long__stdcallvwGetAudioInput(long lChannel);
/**
ADD FUNCTION lAudIn
Set the current audio input.
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4

```

```

#GS_AUDSELECT_BALANCED_10 #GS_AUDSELECT_BALANCED_4 #GS_AUDSELECT_SPDIF
#GS_AUDSELECT_AES_EBU
#GS_AUDSELECT_EMBEDDED
*/
long__stdcallvwSetAudioInput(long lChannel, long lSetting);

/**
Get the current audio input level
*/
long__stdcallvwGetAudioInputLevel(long lChannel);
/**
Get the current audio input level
*/
long__stdcallvwSetAudioInputLevel(long lChannel, long lSetting);

/**
Get the current audio output - See Get/SetAudioInput
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
#GS_AUDSELECT_BALANCED_10 #GS_AUDSELECT_BALANCED_4 #GS_AUDSELECT_SPDIF
#GS_AUDSELECT_AES_EBU
#GS_AUDSELECT_EMBEDDED
*/
long__stdcallvwGetAudioOutput(long lChannel);
/**
Set the current audio output - See Get/SetAudioInput
#GS_AUDSELECT_UNBALANCED_10 #GS_AUDSELECT_UNBALANCED_4
#GS_AUDSELECT_BALANCED_10 #GS_AUDSELECT_BALANCED_4 #GS_AUDSELECT_SPDIF
#GS_AUDSELECT_AES_EBU
#GS_AUDSELECT_EMBEDDED
*/
long__stdcallvwSetAudioOutput(long lChannel, long lSetting);

/**
Get the current audio output level.
*/
long__stdcallvwGetAudioOutputLevel(long lChannel);
/**
Get the current audio output level.
*/
long__stdcallvwSetAudioOutputLevel(long lChannel, long lSetting);

/**
Returns the RMS and Peak audio levels of the input (stop/record) or output (play/pause).
*/
long__stdcallvwGetAudioPeakRMS(long lChannel, long lAudEdit, long * p1Peaks);

/**
Get the current video input.
#GS_VIDSELECT_COMPOSITE, #GS_VIDSELECT_COMPOSITE_2, #GS_VIDSELECT_SVIDEO,

```

```

#GS_VIDSELECT_COMPONENT_YUV,#GS_VIDSELECT_COMPONENT_YUV_M2,
#GS_VIDSELECT_COMPONENT_YUV_SMPTE,#GS_VIDSELECT_COMPONENT_RGB,
#GS_VIDSELECT_D1_SERIAL,#GS_VIDSELECT_D1_PARALLEL,#GS_VIDSELECT_SDTI,
#GS_VIDSELECT_NONE
*/
long__stdcallvwGetVideoInput(long lChannel);
/**
Set the current video input.
#GS_VIDSELECT_COMPOSITE,#GS_VIDSELECT_COMPOSITE_2,#GS_VIDSELECT_SVIDEO,
#GS_VIDSELECT_COMPONENT_YUV,#GS_VIDSELECT_COMPONENT_YUV_M2,
#GS_VIDSELECT_COMPONENT_YUV_SMPTE,#GS_VIDSELECT_COMPONENT_RGB,
#GS_VIDSELECT_D1_SERIAL,#GS_VIDSELECT_D1_PARALLEL,#GS_VIDSELECT_SDTI,
#GS_VIDSELECT_NONE
*/
long__stdcallvwSetVideoInput(long lChannel,long lSetting);

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

/**
Get the current video input's 'Setup' TBC setting.
*/
long__stdcallvwGetVideoInputSetup(long lChannel);
/**
Set the current video input's 'Setup' TBC setting.
*/
long__stdcallvwSetVideoInputSetup(long lChannel,long lSetting);
/**
Get the current video input's 'Video' TBC setting.
*/
long__stdcallvwGetVideoInputVideo(long lChannel);
/**

```

```

Set the current video input's 'video' TBC setting.
*/
Tong__stdcallvwSetVideoInputVideo(long lChannel, long lSetting);
/**
Get the current video input's 'Hue' TBC setting.
*/
Tong__stdcallvwGetVideoInputHue(long lChannel);
/**
Set the current video input's 'Hue' TBC setting.
*/
Tong__stdcallvwSetVideoInputHue(long lChannel, long lSetting);
/**
Get the current video input's 'Chroma' TBC setting.
*/
Tong__stdcallvwGetVideoInputChroma(long lChannel);
/**
Set the current video input's 'Chroma' TBC setting.
*/
Tong__stdcallvwSetVideoInputChroma(long lChannel, long lSetting);

/**
Get the current global TBC's 'Setup' setting.
*/
Tong__stdcallvwGetVideoTBCSetup(long lChannel);
/**
Set the current global TBC's 'Setup' setting.
*/
Tong__stdcallvwSetVideoTBCSetup(long lChannel, long lSetting);
/**
Get the current global TBC's 'Video' setting.
*/
Tong__stdcallvwGetVideoTBCVideo(long lChannel);
/**
Set the current global TBC's 'Video' setting.
*/
Tong__stdcallvwSetVideoTBCVideo(long lChannel, long lSetting);
/**
Get the current global TBC's 'Hue' setting.
*/
Tong__stdcallvwGetVideoTBCHue(long lChannel);
/**
Set the current global TBC's 'Hue' setting.
*/
Tong__stdcallvwSetVideoTBCHue(long lChannel, long lSetting);
/**
Get the current global TBC's 'Chroma' setting.
*/
Tong__stdcallvwGetVideoTBCChroma(long lChannel);
/**

```

```

Set the current globalTBC's 'chroma' setting.
*/
long__stdcallvwSetVideoTBCchroma(long lChannel, long lSetting);

/**
Turn the house/reference lock on or off
*/
long__stdcallvwGetVideoGenlock(long lChannel);
/**
Turn the house/reference lock on or off
*/
long__stdcallvwSetVideoGenlock(long lChannel, long lSetting);

/**
Set the genlock source to input or external reference
*/
long__stdcallvwGetVideoGenlockSource(long lChannel);
/**
Set the genlock source to input or external reference
*/
long__stdcallvwSetVideoGenlockSource(long lChannel, long lSetting);

/**
Get the current compression rate
*/
long__stdcallvwGetCompressionRate(long lChannel);
/**
Set the current compression rate
*/
long__stdcallvwSetCompressionRate(long lChannel, long lSetting);

/**
Get the status of the super imposed time code overlay
*/
long__stdcallvwGetSuperImpose(long lChannel);
/**
Set the status of the super imposed time code overlay
*/
long__stdcallvwSetSuperImpose(long lChannel, long lSetting);
/**
Get the ms time the last error was added to the error log
*/
long__stdcallvwGetErrorLogMs();
/**
Set the error log pointer to the message you want
*/
long__stdcallvwSetErrorLog(long lSetting);
/**
Get the length of the current error string

```

```

*/
long __stdcall VwGetErrorLength(long* pLastError, long* pErrorLength);
/**
Get the current error. Sets pointer to the next one automatically
*/
long __stdcall VwGetError(long* pLastError, long* pSeverity, char* szError);

/**
Returns the total number of frames of storage available at current compression rate if the
storage space was empty.
*/
long __stdcall VwGetTotalTime(long lChannel);
/**
Returns the remaining number of frames of storage available at current compression rate.
*/
long __stdcall VwGetFreeTime(long lChannel);
/**
Returns the total storage connected in megabytes.
*/
long __stdcall VwGetTotalStorage(long lChannel);
/**
Returns the amount of available storage for recording in megabytes.
*/
long __stdcall VwGetFreeStorage(long lChannel);

/**
Get the current millisecond time.
*/
long __stdcall VwGetCurMs(long lChannel);

/**
Get the available commands for a channel.
*/
long __stdcall VwGetChannelCapabilities(long lChannel);

/**
Returns the version string of the Vw subsystem.
*/
char* __stdcall VwGetVwVersion();

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

/**
Returns the type string of the Vw channel.
*/
char* __stdcall VwGetVwType(long lChannel);

```

```
/**  
Get the name of a picon file from the media file's name  
*/  
long__stdcallVwvGetPiconName(char* szFileName);
```

```
//  
// Utility  
//  
/**  
Free a string value returned by the channel.  
*/  
void__stdcallVwvFreeString(char* szString);
```

Appendix IV – MediaCmdNet.h

Now available for Linux and Win32 as share library/DLL.