# MediaCmd HTTP/Ajax Interface



www.drastictech.com

## Table of Contents

## *Overview*

Drastic's MediaCMD API is a standard methods

There are three major types of commands available from the DDR http interface:

1. Simple string commands and returns
2. Full MediaCmd Ajax/XML access
3. Special XML access commands

There

## *Simple string commands and returns (deprecated)*

The simple commands do not have XML returns, like the MediaCmd Ajax/XML. They can be used with a VVWPassThrough call to load or re-load a particular web page. Status commands return simple strings that can be use to be displayed directly.

## Simple Transport (deprecated)

| | |
|---|---|
| play | Play forward at normal play speed |
| play=%d | Play at a particular speed<br>65520 = normal forward play<br>-65520 = reverse play<br>32760 = play at half (50%) speed<br>655200 = 10 times forward play speed |
| stop | Stop playback and pass through the input, if it exists |
| pause | Pause playback and show a still frame |
| record | Start recording as soon as possible |
| seek=%d | Seek or cue to a particular location<br>+5:00 move forward from the current location 5 seconds<br>-01:00:00;00 move backward one hour from current location<br>1:21:00:00 go to 1 hour and 21 minutes<br>1800 go to 1800 frames (1:00:00) |
| setin=%d | Set the in point |
| setout=%d | Set the out point |
| setpos=%d | Set a new position |

## Modifiers and Setup (deprecated)

| | |
|---|---|
| page=%s | The Next page to load |
| channel=%d | Channel for this command to target, overrides the current channel |
| mode=%d | Sets the DDR's mode:<br>Clip Mode = 0<br>VTR/Conform Mode = 1 |
| type=%d | General type number |
| setchannel=%d | Set a new current or default channel |
| load=%s | Load a new clip, using clip name, in Clip Mode |
| delete=%s | Delete a clip, using the clip name |
| remove=%s | Remove a clip, using the clip name, without deleting it from the disk |
| delete=%s | Delete a clip, using a clip name. Removes it from the disk. |
| insert=%s | Add a new clip, using a file path and name |

## Status Requests

| | |
|---|---|
| VVWGetState | Returns a string describing the current state:<br>play, pause, record, stop |
| VVWGetPosition | Returns the current time code:<br>01:00:00:00 |
| VVWGetFrame | Returns the current frame location:<br>108000 |
| VVWGetLastChangeMs | Returns a millisecond counter indicating the last time the current clip or VTR/Conform space changed |
| VVWGetErrorLogMs | Returns a millisecond counter indicating the last time a new error log entry was added |

## *Full MediaCmd Ajax/XML Access*

This access methods allows our javascript or php application to access all the same functions used by Drastic's GUIs from an html interface. The basic form of the commands is:

http://localhost:1080/VVWXMLMediaCmd?<mediacmd>

The <mediacmd> is a series of ampersand delimited (&) commands and modifiers. Normally this command will be sent via an HTTPObject, and will return synchronously or asynchronously an standard XML return that can be parsed. To send a command in Ajax/Javascript, you will first need to instantiate a HTTPObject to send it through. Here is a HTTPObject instantiation that will work in most browsers:

```
// Create an HttpObject
function getHTTPObject()
{
	var xmlhttp;

	/*@cc_on
		@if (@_jscript_version >= 5)
			try
			{
				xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
			}catch (e)
			{
				try
				{
					xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
				} catch (E)
				{
					xmlhttp = false;
				}
			}
		@else
			xmlhttp = false;
		@end @*/

	if (!xmlhttp && typeof XMLHttpRequest != 'undefined')
	{
		try
		{
			xmlhttp = new XMLHttpRequest();
		} catch (e)
		{
			xmlhttp = false;
		}
	}
```

```
        return xmlhttp;
}

// Instantiate the various HTTP Objects
var _xmlHttp = getHTTPObject(); // Create the HTTP Object
```

Once the HTTPObject is instantiated into a variable, the variable (_xmlHttp in this case) can be used to call the DDR and send and receive MediaCmds. These commands can be sent synchronously (the command will complete and return the XML immediately) or asynchronously (the command will process, but return immediately. Later a callback will be called with the return XML data). Either way the return will be the same.

To send a command synchronously (return after processing) without using the return:

```
function play()
{
        // Build the URL to connect to
        var url = "VVWXMLMediaCmd?Play”;
        // Open a connection to the server
        _xmlHttp.open("GET", url, false);    // indicates sync call
        // Send the request
        _xmlHttp.send(null);
}
```

For a command that is sent synchronously, but the return needs to be processed, the call is very similar:

```
function getClipMode()
{
        // Build the URL to connect to
        var url = "VVWXMLMediaCmd?GetValue&position=0&cmdalt=ClipMode&Flags=-1”;
        // Open a connection to the server
        _xmlHttp.open("GET", url, false);    // indicates sync call
        // Send the request
        _xmlHttp.send(null);
        // Get the clip mode out of the XML response
        var xmlobject = _xmlHttpMode.responseXML;
        if(xmlobject == null) {
                return;
        }
        // Get MediaCmd return (in XML)
        var mCmd = xmlobject.getElementsByTagName("MediaCmd");
        if(mCmd[0])
        {
                // Return the current mode
                return mCmd[0].getElementsByTagName("Position")[0].getAttribute("Value")
```

```
        }
        return "errorValue";
}
```

A typical XML return would look like this:

<insert mediacmd XML return here>

Often, to maximize user responsiveness, or to allow for long command to process, commands need to be sent asynchronously. The asynchronous version of the command is essentially the same as the synchronous with processing version, except the send and return are divided into separate functions:

```
function getClipMode()
{
        // Build the URL to connect to
        var url = "VVWXMLMediaCmd?GetValue&position=0&cmdalt=ClipMode&Flags=-1";
        // Open a connection to the server
        _xmlHttp.open("GET", url, false);    // indicates sync call
        // Setup a function for the server to run when it's done
        _xmlHttp.onreadystatechange = updateClipMode;
        // Send the request
        _xmlHttp.send(null);
}
// A response to an 'Mode' request has been received
function updateMode()
        if (_xmlHttpMode.readyState == 4)
        {
                // A complete response has been received
                // Get the clip mode out of the XML response
                var xmlobject = _xmlHttpMode.responseXML;
                if(xmlobject == null) {
                        return;
                }
                // Get MediaCmd return (in XML)
                var mCmd = xmlobject.getElementsByTagName("MediaCmd");
                if(mCmd[0])
                {
                        // Return the current mode
                        return mCmd[0].getElementsByTagName("Position")[0].getAttribute("Value")
                }
        }
        return "errorValue";
}
```

## VVWXMLMediaCmd main commands

The first parameter of the VVWXMLMediaCmd? (following the question mark) must be be one of the following commands:

- Stop – Full stop/all stop/e to e
- Pause – Pause on current frame, seek or load
- Play – Play, either at normal speed or shuttle speeds. May also load and seek.
- Record – Record to the disk or tape
- RecStop – Prepare for a record
- Eject – Eject the current tape or media
- Transfer – Transfer to/from an internal channel and an external channel
- Insert – Insert media into the clip bin or time code space
- Blank – Remove media from the clip bin or time code space
- Delete – Delete media from the storage and blank it
- Trim – Alter a clip or time code space edit
- ChanSelect – Change the currently selected channels
- GetState – Get the current channel state
- SetState – Set the current channel state
- GetValue – Get a setup value
- ValueSupported – See if a setup value is supported
- SetValue – Change a setup value
- Error – Report an error
- Terminate – Kill the current operation
- Abort – Abort the current operation

## VVWXMLMediaCmd modifiers

With these commands a number of modifiers are available. Each modifier must be separated by an ampersand (&) on the command line.

- channel=%d – specify the channel this command should be sent to
- position=%s – set the position element for a command
    - 1:00:00:00 – go to one hour
    - +5 – go forward from the current location 5 frames
    - -5:00 – go backward from the current location 5 seconds
    - 1800 – go to one minutes (specified as 1800 frames, not drop frame time code0
- start=%s – set the start element (see position for format)
- end=%s – set the end element (see position for format)
- speed=%d – set the speed element for a command
    - 65520 – normal forward play (100%)
    - -65520 – reverse play
    - 32760 – half play speed (50%)
    - -655200 – 10 times reverse speed
    - 0 - pause (no play)
- timems – millisecond time for the command
- cmdalt – set the cmdalt element of the mediacmd
- videochannels – which video channels to use (bitwise)
- audiochannels – which audio channels to use (bitwise)
- infochannels – which information channels to use (bitwise)
- clipid – 8 character clip identifier
- filename – filename for the command
- string – sting to be used in the command
- There are a number of flags that may be used, just like the elements above
    - Deferred – wait for previous command to complete before new this command
    - OverrideDeferred – override a previous deferred command
    - Loop – Loop whole clip, or a start/end subset
    - AllIDs – Command should affect all available clip ids
    - NoClipFiles – Ignore clip space clips

- NoTCSpaces – Ignore conform space files
- IsShuttle – The command should be interpreted as a shuttle, even for normal play
- UsingCurrent – Use the current start/end/position
- UseFrameCount – Use the absolute frame count, not the time code values
- Fields – Use fields, if not a progressive signal formats
- Ripple – When removing a file, ripple the following files back
- Trigger – Wait for a trigger
- Preview – Doing a preview, not a full play
- Test – Don't do the command, just see if it exists
- NoReturn – Don't return any information from the command

## VVWXMLMediaCmd Examples

VVWXMLMediaCmd?play
– Normal play

VVWXMLMediaCmd?play&speed=32760
– Play at 50% forward speed

VVWXMLMediaCmd?play&speed=-65520
– Play at 100% reverse play speed

VVWXMLMediaCmd?play&start=1:00&end=4:00&loop
– Play from one second to four seconds in a loop

VVWXMLMediaCmd?pause
– Pause the channel

VVWXMLMediaCmd?stop
– Stop (e to e passthrough) the channel

VVWXMLMediaCmd?pause&position=1:00:00
– Seek to one minute

VVWXMLMediaCmd?record&clipid=newrec&end=5:00
– Record a new file name 'newrec' which will be five seconds long

## Dealing with Picon Images

Server Mode, clip: Kroatien, file: KroatienMovie.mov

http://localhost/VVWXMLMediaCmd?SetValue&cmdalt=1000000&clipid=Kroatien&position=200
– Make a new picon from frame 200 of the clip Kroatien
– result name: KroatienMovie.picon.jpg

http://localhost/VVWXMLMediaCmd?GetValue&cmdalt=1000000&clipid=Kroatien&position=ffffffff
– Return the actual file name of the picon file (char elem 9)
– result name: Kroatien.picon.jpg

http://localhost/VVWXMLMediaCmd?
GetValue&cmdalt=1000000&clipid=Kroatien&position=4294967295
– Return the size of the picon file in the Position elements
– result: dwPosition = 7900

http://localhost/VVWXMLMediaCmd?GetValue&cmdalt=1000000&clipid=Kroatien&position=1
– Return the actual bytes of data for the JPEG picon frame in arbID
– result: Not available in HTTP, have to use C/C++

http://localhost/VVWXMLMediaCmd?
SetValue&cmdalt=1000000&filename=V:\Media\KroatienMovie.mov&position=100
– Make a new picon from frame 100 without associating it with the clip
– result name: KroatienMovie.picon.jpg
(not normally used, conflicts with vtr tape mode picon)

VTR Tape Mode, Time line 00:00:01:00 Kroatien.mov?

http://localhost/VVWXMLMediaCmd?
SetValue&cmdalt=1000000&filename=V:\Media\Kroatien.mov&position=1000
– Make a new picon from the frame at position 1000, default for file
– result name: Kroatien.picon.jpg

http://localhost/VVWXMLMediaCmd?
GetValue&cmdalt=1000000&filename=V:\Media\Kroatien.mov&position=ffffffff
– Return the actual file name of the picon file (char elem 9)
– result name: Kroatien.picon.jpg

http://localhost/VVWXMLMediaCmd?
GetValue&cmdalt=1000000&filename=V:\Media\Kroatien.mov&position=4294967295
– Return the size of the picon file in the Position elements
– result: dwPosition = 7900

http://localhost/VVWXMLMediaCmd?
GetValue&cmdalt=1000000&filename=V:\Media\Kroatien.mov&position=1
– Return the actual bytes of data for the JPEG picon frame in arbID
– result: Not available in HTTP, have to use C/C++

## *Special XML Access Commands*

XML Returns. These are to be used with Ajax/DOM pages.

http://localhost/VVWXMLGetStatus
– Returns an XML package including state, speed, position start and end points.

http://localhost/VVWXMLNextClip
– Returns an XML package with all the clip information. Used to retrieve the clip bin information

http://localhost/VVWXMLClipInfo
– Returns an XML package with all the clip information. Used to retrieve information on a specific clip

http://localhost/VVWXMLEDLState
– Used in conjunction with VVWXMLEDLInfo to retrieve the time code space edits. The command will always be VVWXMLEDLState?
position=#&videochannels=#&audiochannels=#&infochannels=#.

http://localhost/VVWXMLEDLInfo
– Used in conjunction with VVWXMLEDLState to retrieve the time code space edits.

Here is a basic EDL retrieval session:

Call··· ·Position··· ·Start··· ·End ··· ·V ··· ·A ··· ·I ··· ·File Name··· ·Comment
VVWXMLEDLInfo··· ·0···
0··· ·0··· ·0···
Restart list at 0
return info··· ·0··· ·0··· ·300··· ·1··· ·2··· ·0··· ·file1.mov··· ·10 sec VA2 from file1
VVWXMLEDLState··· ·0···
0··· ·0··· ·0···
First state sent in above
return state··· ·0···
1··· ·2··· ·0···
Used clip channels to pass back into Info
VVWXMLEDLInfo··· ·0···
1··· ·2··· ·0···
Copy of the return of VVWXMLEDLState above
return info··· ·0··· ·0··· ·150··· ·0··· ·1··· ·0··· ·file2.wav··· ·5 sec A1 from file2
VVWXMLEDLState··· ·0···
1··· ·2··· ·0···
Use the return of the last VVWXMLEDLState
return state··· ·0···
1··· ·3··· ·0···
These are the channels used so far
VVWXMLEDLInfo··· ·0···
1··· ·3··· ·0···
Copy of the return of VVWXMLEDLState above

return info··· ·150··· ·150··· ·210··· ·0··· ·1··· ·0··· ·file3.wav··· ·2 sec A1 from file3
VVWXMLEDLState··· ·0···
1··· ·3··· ·0···
Use the return of the last VVWXMLEDLState
return state··· ·150···
0··· ·1··· ·0···
All edits completed before 150

Take the MEDIACMD struct returned from VVWXMLEDLState and find the next active clip. For the first clip in the time line, send all zeroes. Other than the first call, all calls should include the position/channel bits from the previous VVWXMLEDLState call and (other than first call) VVWXMLEDLState should be called immediately before VVWXMLEDLInfo .

http://localhost/VVWXMLNextDirEntry
– Used to retrieve the directory structure.
Takes 2 parameters:
The base directory you are getting the listing for
The last directory entry returned

Assuming you had a directory structure that looked like this:

\Record\
\Record\Test.wav
\Record\Test.avi
\OfflineMedia\
\OfflineMedia\EmptyDir\
\OfflineMedia\retry.doc
\OfflineMedia\big.tga
\LocalMedia\AnotherDir\
\LocalMedia\test.aiff

The first call would only include the parameter '\'
http://localhost/VVWXMLNextDirEntry?\
Returns: <locator>\Record</locator>
This will return the first FileDir XML structure that will include the first locator. To get the next item, return the same base path plus the new locator.
http://localhost/VVWXMLNextDirEntry?\?\Record
Returns: <locator>\OfflineMedia</locator>
http://localhost/VVWXMLNextDirEntry?\?\OfflineMedia
Returns: <locator>\LocalMedia</locator>
http://localhost/VVWXMLNextDirEntry?\?\LocalMedia
Returns: <locator>END OF LIST</locator>
To descend into a sub directory, use the sub directory as the base path. To see what is in \Record
http://localhost/VVWXMLNextDirEntry?\Record\
Returns: <locator>\Record\..</locator>
http://localhost/VVWXMLNextDirEntry?\Record\&\Record\..
Returns: <locator>\Record\Test.wav</locator>
http://localhost/VVWXMLNextDirEntry?\Record\&\Record\Test.wav
Returns: <locator>\Record\Test.avi</locator>

http://localhost/VVWXMLNextDirEntry?\Record\&\Record\Test.avi
Returns: <locator>END OF LIST</locator>

http://localhost/VVWXMLFileInfo
– Used to retrieve information on a specific file.

http://localhost/VVWXMLGetErrorMsg&#
– Used to return one error message from the current list. The first call will not include an error number (just VVWXMLGetErrorMsg). This will return an ErrorNumber to use to get the next message (VVWXMLGetErrorMsg&202 for instance), as will each subsequent call. When all the error messages have been returned, it will return an ErrorNumber of -1.