Net-X-Code

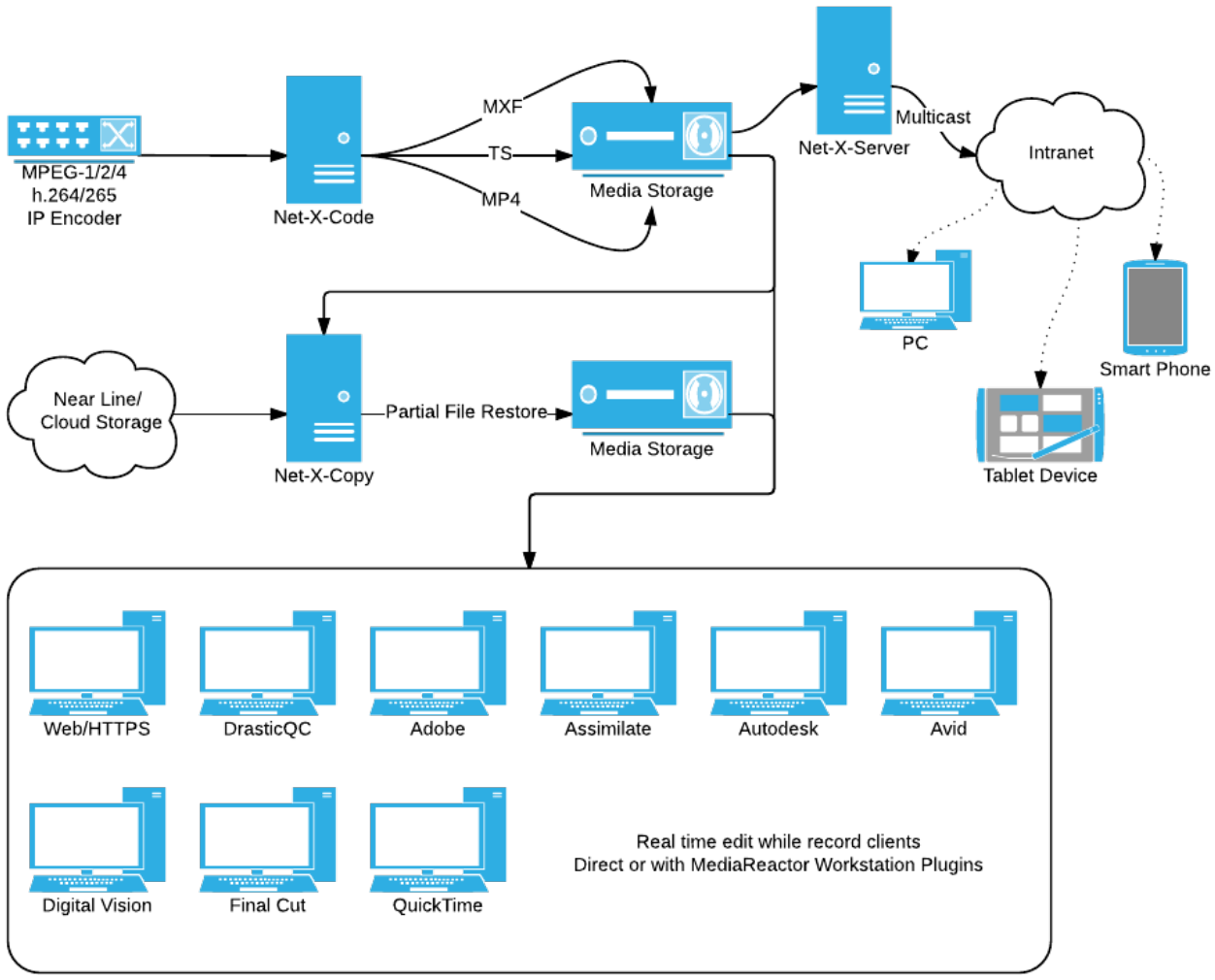
API Guide

Jan 18th, 2018

Table of Contents

Introduction

Net-X-Code is both a series of interconnected applications (Net-X-Code, Net-X-Cmd, Net-X-Copy, Net-X-Streamer, DrasticQC and MediaReactor Workstation Plugins). The various applications auto-detect and join user-defined enterprise groups on one or more servers within a network. Once connected, the controller/user can configure the system in real time via the HTTP/RESTful interface. The major components are diagrammed and listed below:



For the latest Net-X-Code information, please see:

http://www.net-x-code.com

Theory of Operation

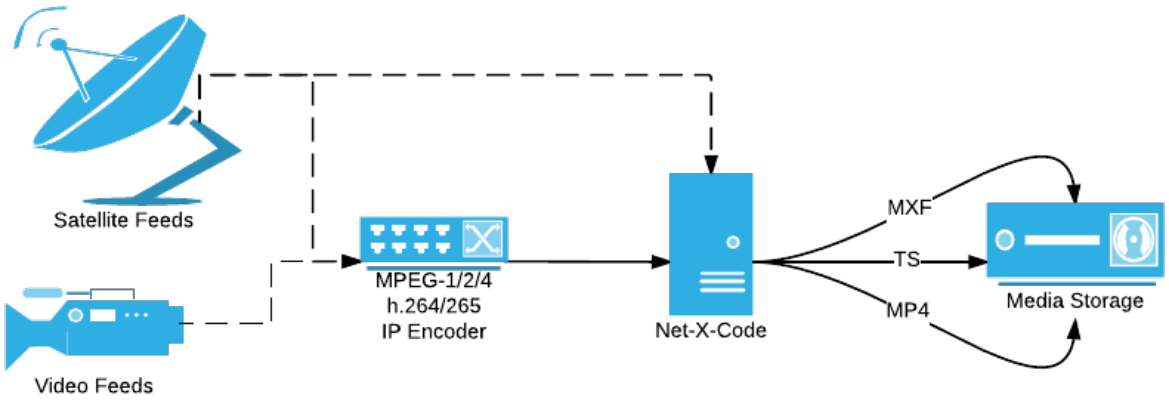
Net-X-Code is a distributed capture and conversion system. It can be run on one or more servers and be controlled from one, central interface. This section of the manual will give an overview of how the various parts of **Net-X-Code** interact to make it easier to design deployments and implement controllers using the API described in the next section.

Net-X-Code is made up of a number of servers, programs, and plugins:

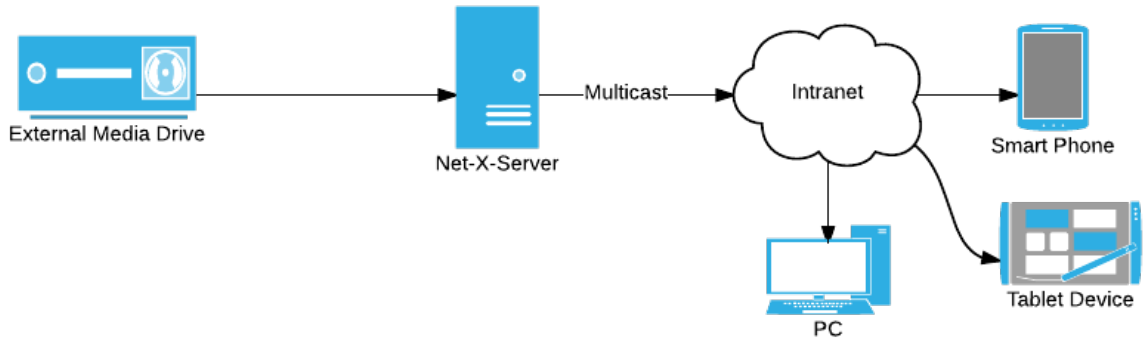
- **Net-X-Cmd**: This component provides the central connection for all the other components. It uses a Bonjour-like protocol to auto-sense components within its group in the network, and provides the HTTP/RESTful/HTML API
- **Net-X-Code**: Provides capture from network IP video sources to TS, MP4, fMP4, ISM, MXF, etc files. A Net-X-Code server can capture 1 or more groups of up to 10 streams per group. Files can also include DASH, HLS and Smooth Streaming sidebar files. These recorded files can be stored locally or anywhere else on the network
- **Net-X-Server**: Can take recording or pre-recorded network IP video streams from disk anywhere in the network, and re-stream them via RTP or UDP back out to the network
- **Net-X-Copy**: Is both a real time video translator, real time clipping engine and partial file restore system. Any recording stream can be used as a source while it is still recording, or near line/tape backup files can be restored, only accessing the part of the file required for the restore
- **Net-X-Player**: A HTML5 based web player that can play timecoded, frame accurate files, including RTIN real time files during record. Net-X-Player can also send clipping commands directly to Net-X-Base.
- **videoQC**: Video preview is available from on disk, live recording and network video sources. videoQC provides video waveform, vectorscope, histogram and metadata displays, along with clipping and conversions.
- **MediaReactor Workstation**: This series of plug ins allow professional editing and finishing system to access all the file types created by the Net-X-Code system and access the real time recording files, while they are still growing. MRWS is built into software like Assimilate and NuCoda, and available as an option for other systems like Adobe, Avid, Autodesk, Quantel, Final Cut, and even QuickTime compatible apps.

Typical Applications

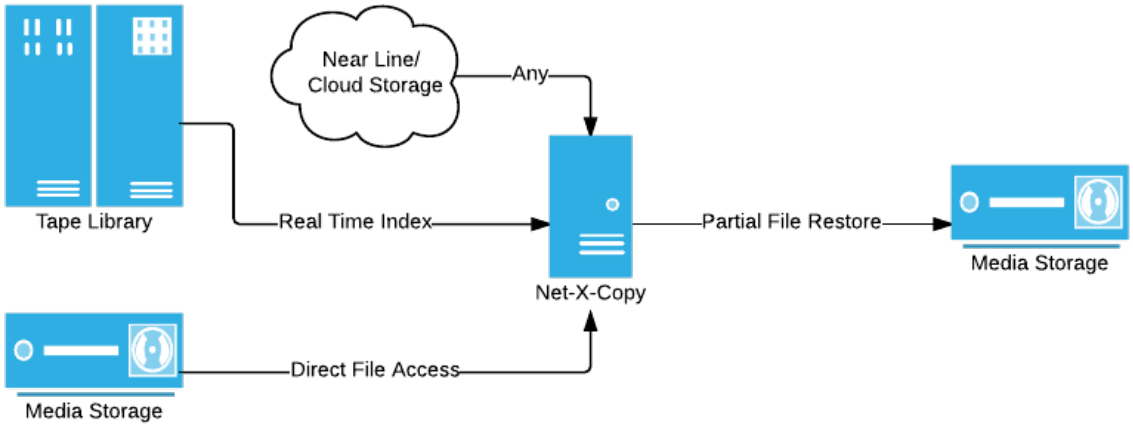
Net-X-Code IP Video Capture



Net-X-Code IP Video Streaming



Net-X-Code Partial File Restore, Conversion and Proxy



Command And Return Structure/Format

Commands are sent as requests to **Net-X-Base**'s integrated server. By default, this server exists at port 1080 on the server's external IP. The beginning of the command will always be **netx?request=get&** or **netx?request=set&** followed by the request. Please note that as soon as the command is validated, a positive response is returned. This keeps the server running quickly for status updates and multiple commands. For commands that may take some time, the result of the command should be checked and waited for by monitoring its status. For instance, once a create group command has been sent, requesting the status for that group's key name will let the caller know when it is ready for more commands. To get the initial status of the system, the recommended order of events is:

- Request client count
- Request client address(s) from 0 - client count
- Request group count from client
- Request info (channel count is fixed at 3)

Example: Determining the number of clients.

A basic request for client count would look like this:

http://127.0.0.1:1080/netx?request=get&client=count

The response will be an XML buffer like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<client count="1"/>
```

Command Elements

The commands are sent via HTTP/REST as a HTTP request string. The commands must start with one of two strings

- netx? - Command will return an XML response
- netxjson? - Command will return a JSON response

Depending on whether it is a command or a request for information, this is followed by "request=get" or "request=set" without quotes:

Basic command start

http://127.0.0.1:1080/netx?request=get
http://127.0.0.1:1080/netx?request=set

Command Parameters

The parameters may be applied to the command in any order. A command may not use any parameter more then once in a single command, nor may channel sets be combined with a general group command (more on this in the group command section). The available command parameters are:

- **client**=<ip of client, count, address> Used to specify the client server machine the request or command it is intended for. For requests, if 'count' is specified it will return the number of clients. Also for requests, if 'address' is specified, then it will return the client address.
- **group**=<#, KEY, all, count> Used to specify the group on the machine the request or command will act on. The group parameters can have four forms:
  - **#** the group number, as returned in the list
  - **KEY** the group name as set when created, or later, by the caller

    - **all** specify that all channels should be returned or acted upon
    - **count** to request the total number of groups
- **index=#** specifies the index number for the command or request.
- **channel=<#, all>** will specify the channel within the group the command or request requires. If it is set to 'all' then it will return a request for all channels in that group.\
- **value=<#>** set a value (numeric) for a command
- **command=<cmd>** is used when sending a command. More information on this is available below. The possible commands are:
    - **start** – start a capture
    - **stop** – stop a capture
    - **restart** – restart a capture
    - **add** – add a new group
    - **remove** – remove an existing group
    - **delete** – delete files on the disk. List files with &file=<file path name> pairs.

Command - get

To get the status of all the servers, groups and channels, the following order is recommended:

1. Request client count
2. Request client address(es) from 0 client count
3. Request group count from client
4. Request info (channel count is fixed at 3)

Client Count

Request the client count client=count

http://127.0.0.1:1080/netx?request=get&client=count

```
<?xml version="1.0" encoding="ISO88591"?>
<client count="1"/>
```

Client IP

Request the first client IP address client=ip&index=#

http://127.0.0.1:1080/netx?request=get&client=address&index=0

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" version="v#.#.#.###"/>
```

Number Of Groups

Request the number of groups client=ip&group=count

http://127.0.0.1:1080/netx?request=get&client=192.168.100.176&group=count

```
<client address="192.168.100.176"  version="v#.#.#.###" groups="5" autostart=”1”>
        <group count="5"/>
</client>
```

Attributes/Request Info

Request attributes client=ip&group=#&channel=#&value=<*setting*> where 'setting' can be:

- **name** - the target file name
- **capturefile** – the full name and path of the actual capture file
- **address** - the address for the capture
- **port** - the port for the capture
- **directory** - the target directory
- **rectc** - type or record timecode capture
- **type** - the target file type (.mp4, .fmp4, .mov, .ism, .mxf, .ts)
- **state** - the current state
- **ms** - the last millisecond count
- **bytes** - the last byte count
- **uuid** - the channels uuid
- **metadata** – the metadata string from the stream
- **frames** – the number of video frames captured
- **tc** - the last gop start captured timecode
- **timecode** – the last captured timecode
- **ub** – the current captured userbits
- **firsttc** – the first timecode captured for a stream
- **firstub** – the first userbits captured for a stream
- **tctype** - channels timecode type
- 1 = 24.0 fps
- 2 = 30 fps (non drop frame 30)
- 4 = 29.97 fps (drop frame 30)
- 8 = 25.0 fps
- 16 = 50.0 fps
- 32 = 59.94 fps (drop frame 60)
- 64 = 60.0 fps (non drop frame 60
- **starttc** - capture start time
- **endtc** - capture end time
- **all (or "-1")** - return all of the above
- **previewenabled** – retrieve the current state of preview frames
- **preview** – retrieve a jpeg preview embedded in the xml response of the last known valid I-Frame
- **mpreview** – retrieve a jpeg preview of the last known valid I-Frame as raw image info
- **property** – retrieve the stream video/audio properties
- **status** – retrieve a list of status messages

Stream States

The possible stream states are:

NetXState_Idle = 0
"Idle"
NetXState_Active = 1
"Active"
NetXState_SourceConnectionPending = 2
"Connection Pending"
NetXState_SourceClosing = 3
"Source Closing"
NetXState_SourceDisconnected = 4
"Source Disconnected"
NetXState_SourceCapturePending = 5
"Source Capture Pending"
NetXState_SourcePending = 6
"Waiting For Source"
NetXState_Error = 7

"Error"
NetXState_NotFound = 8
"Not Running"
NetXState_Unknown = -2
"Unknown"

Info about all groups and/or all channels can be returned at once. Setting "all" or a specific value for any of the parameters group="", channel="", value="" will return different results. Here are some examples:

Requesting a single parameter from one group and one channel.

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=1&value=name

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" groups="5" autostart=”1”>
        <group index="0" key=”GroupName” autostart=”1”>
                <channel index="1">
                        <name>ts_channel2</name>
                </channel>
        </group>
</client>
```

Or get everything about a single group and a single channel.

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=GroupName&channel=1&value=all

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" groups="5" autostart=”1”>
<group index="0" key=”GroupName” autostart=”1”>
        <channel index="1">
                <name>ts_channel2</name>
                <capturefile>E:/Record/NetXCode/channel2.ts</capturefile>
                <address>239.255.40.40</address>
                <port>1234</port>
                <directory>E:/Record/NetXCode</directory>
                <type>.ts</type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>8</state>
        </channel>
</group>
</client>
```

Or all settings for all groups and all channels

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=all&channel=all&value=all

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" groups="5" autostart=”1”>
        <group index="0" key=”GroupName” autostart=”1”>
                <channel index="0">
                        <name>ts_channel1</name>
                        <capturefile>e:/Record/NetXCode/ts_channel1.ts</capturefile>
                        <address>239.255.40.40</address>
                        <port>1233</port>
                        <directory>E:/Record/NetXCode</directory>
```

```
                <type>.ts</type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>8</state>
        </channel>
        <channel index="1">
                <name>ts_channel2</name>
                <capturefile>e:/Record/NetXCode/ts_channel2.ts</capturefile>
                <address>239.255.40.40</address>
                <port>1234</port>
                <directory>E:/Record/NetXCode</directory>
                <type>.ts</type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>8</state>
        </channel>
        <channel index="2">
                <name></name>
                <capturefile></capturefile>
                <address>239.255.40.40</address>
                <port>1235</port>
                <directory>E:/Record/NetXCode</directory>
                <type>.mxf</type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>8</state>
        </channel>
</group>
<group index="1" key=”GroupName2” autostart=”1”>
        <channel index="0">
                <name>Channel 1</name>
                <capturefile></capturefile>
                <address>239.255.40.41</address>
                <port>1234</port>
                <directory></directory>
                <type>.mp4</type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>8</state>
        </channel>
        <channel index="1">
                <name>Channel 2</name>
                <capturefile></capturefile>
                <address>239.255.40.41</address>
                <port>1235</port>
                <directory></directory>
                <type>.mxf</type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>8</state>
        </channel>
        <channel index="2">
                <name>Channel 3</name>
                <capturefile></capturefile>
                <address>239.255.40.41</address>
                <port>1236</port>
                <directory></directory>
                <type>.mov</type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>8</state>
```

```
        </channel>
</group>
<group index="2" key=”GroupName3” autostart=”1”>
        <channel index="0">
                <name></name>
                <capturefile></capturefile>
                <address></address>
                <port>0</port>
                <directory></directory>
                <type></type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>0</state>
        </channel>
        <channel index="1">
                <name></name>
                <capturefile></capturefile>
                <address></address>
                <port>0</port>
                <directory></directory>
                <type></type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>0</state>
        </channel>
        <channel index="2">
                <name></name>
                <capturefile></capturefile>
                <address></address>
                <port>0</port>
                <directory></directory>
                <type></type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>0</state>
        </channel>
</group>
<group index="3" key=”GroupName4” autostart=”1”>
        <channel index="0">
                <name></name>
                <capturefile></capturefile>
                <address></address>
                <port>0</port>
                <directory></directory>
                <type></type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>0</state>
        </channel>
        <channel index="1">
                <name></name>
                <capturefile></capturefile>
                <address></address>
                <port>0</port>
                <directory></directory>
                <type></type>
                <ms>0</ms>
                <bytes>0</bytes>
                <state>0</state>
        </channel>
        <channel index="2">
```

```
                        <name></name>
                        <capturefile></capturefile>
                        <address></address>
                        <port>0</port>
                        <directory></directory>
                        <type></type>
                        <ms>0</ms>
                        <bytes>0</bytes>
                        <state>0</state>
                </channel>
        </group>
        <group index="4" key=”GroupName5” autostart=”1”>
                <channel index="0">
                        <name></name>
                        <capturefile></capturefile>
                        <address></address>
                        <port>0</port>
                        <directory></directory>
                        <type></type>
                        <ms>0</ms>
                        <bytes>0</bytes>
                        <state>0</state>
                </channel>
                <channel index="1">
                        <name></name>
                        <capturefile></capturefile>
                        <address></address>
                        <port>0</port>
                        <directory></directory>
                        <type></type>
                        <ms>0</ms>
                        <bytes>0</bytes>
                        <state>0</state>
                </channel>
                <channel index="2">
                        <name></name>
                        <capturefile></capturefile>
                        <address></address>
                        <port>0</port>
                        <directory></directory>
                        <type></type>
                        <ms>0</ms>
                        <bytes>0</bytes>
                        <state>0</state>
                </channel>
        </group>
</client>
```

Or a single setting from a single group and all channels

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=1&channel=all&value=address

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<client address="192.168.100.176" groups="2" autostart=”1”>
   <group index="1" key="GroupName" autostart=”1”>
      <channel index="0">
         <address>239.255.42.42</address>
      </channel>
      <channel index="1">
         <address>239.255.42.42</address>
```

```
      </channel>
      <channel index="2">
         <address>239.255.42.42</address>
      </channel>
   </group>
</client>
```

previewenable

Determine if preview jpeg capture is currently enabled

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=previewenable

preview

Getting a preview jpeg and embedded in the xml response

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=preview

The data returned is base 64 encoded jpeg data located in the following xml structure:

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" version=v4.2.0.274 groups="1">
   <group index="0" key="GroupName" autostart=”1”>
      <channel index="0">
         <pts>432400</pts>
          <preview timestamp=”6521600”>(base 64 jpeg data)
          </preview>
      </channel>
   </group>
</request>
```

pts

Getting the current pts values for audio, video and time code

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=pts

The data returned is base 64 encoded jpeg data located in the following xml structure:

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" version=v4.2.0.274 groups="1" autostart=”1”>
   <group index="0" key="GroupName" autostart=”1”>
      <channel index="0">
         <video_pts>432523</video_pts>
         <audio_pts>432590</audio_pts>
         <tc_pts>432400</tc_pts>
         <days>25</days>
         <months>2</months>
         <years>2015</years>
      </channel>
```

```
   </group>
</request>
```

datarate

Getting the last known stream datarate. By default, all rate types are returned.

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=datarate

The value returned will contains rates for video/audio & total. Specific values can be retrieved by appending the desired type to the call:

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=datarate&total=

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=datarate&video=

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" version=v4.2.0.274 groups="1" autostart=”1”>
   <group index="0" key="GroupName" autostart=”1”>
      <channel index="0">
         <datarate_total timestamp=”7742166”>9.34 Mbps</datarate_total>
         <datarate_video timestamp=”7742166”>9.03 Mbps</datarate_video>
         <datarate_audio timestamp=”7742166”>0.31 Mbps</datarate_audio>
      </channel>
   </group>
</request>
```

property

Getting a stream property

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=property&video_width=

The value returned in this case will be the width in pixels of the captured stream (see response below). An XML attribute is provided with each property to denote if there has been a change for some reason. Properties will not be available until the stream starts capturing. All properties can be retrieved at once by calling:

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=property&all=

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" version=v4.2.0.274 groups="1" autostart=”1”>
   <group index="0" key="GroupName" autostart=”1”>
      <channel index="0">
         <video_width changed=”0”>1280</video_width>
      </channel>
   </group>
</request>
```

status

Getting a status list

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=status

The xml returned will contain a list of status messages for the given channel(s). Provided is a status count and an index for each status message. The character value of each message specifies the severity level. All messages will be remove from the internal list once they have been retrieved. Any non-critical messages will be removed after 5 minutes from its arrival.

0 = Information, 1 = Warning, 2 = Critical

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" version=v4.2.0.274 groups="1" autostart=”1”>
   <group index="1" key="GroupName" autostart=”1”>
      <channel index="0">
         <status count=1”>
            <status0 index=”0” timestamp=”6516120” desc=”meta tc pid changed”>2</status0>
         </status>
      </channel>
   </group>
</request>
```

streamstate

Get the running state of a capturing stream

http://192.168.100.229:1080/netx?
request=get&client=192.168.100.229&group=0&channel=0&value=streamstate

This command retrieves the running information on a given stream. This includes information derived from the incoming stream like video width/height, audio frequency, packets captured, pids and the UUID of the stream. This command is slightly different then the other gets, as it requires two calls to get the information. The first sets up the request from Net-X-Base to Net-X-Code, and the second actually gets the data. The first call will return:

```
<client address="192.168.100.176" version="v5.0.0.39" groups="1" autostart=”1”>
  <group index="0" key="Default" version="v5.0.0.39" autostart=”1”>
    <channel index="0">
      <streamstate>requested</streamstate>
    </channel>
  </group>
</client>
```

The second will then return with the information:

```
<client address="192.168.100.176" version="v5.0.0.39" groups="1" autostart=”1”>
  <group index="0" key="Default" version="v5.0.0.39" autostart=”1”>
    <channel index="0">
      <streamstate>
        <netxcode-stream id="0">
          <video_width>1280</video_width>
          <video_height>720</video_height>
          <video_bits>24</video_bits>
          <video_rate>60000</video_rate>
```

```
        <video_scale>1001</video_scale>
        <audio_frequency>48000</audio_frequency>
        <audio_bits>16</audio_bits>
        <uuid>00000000000000000000000000000000</uuid>
        <framecount>423686</framecount>
        <length>557570776</length>
        <packets_total>2965789</packets_total>
        <packets_video>2282451</packets_video>
        <packets_audio>64638</packets_audio>
        <packets_tc>10275</packets_tc>
        <packets_metadata>0</packets_metadata>
        <packets_cc>0</packets_cc>
        <pid_video>2001</pid_video>
        <pid_audio>3001</pid_audio>
        <pid_audio1>4294967294</pid_audio1>
        <pid_audio2>4294967294</pid_audio2>
        <pid_audio3>4294967294</pid_audio3>
        <pid_dolby>4294967294</pid_dolby>
        <pid_tc>501</pid_tc>
        <pid_cc>4294967294</pid_cc>
      </netxcode-stream>
    </streamstate>
  </channel>
 </group>
</client>
```

state

Return the NetXBase status including last command, time of last command and times of the last communication with a NetXCmd.

http://127.0.0.1:1080/netx?request=get&netxbase=state

xmldisable

Disable XML side car generation for captured files

This command gets the state of XML side car generation

http://192.168.100.229:1080/netx?
request=get&client=192.168.100.229&group=0&channel=0&value=xmldisable

If set to 0, no side car XML files will be generated. If set to one, standard Drastic XML side car files will be generated.

autostart

Autostart is not a command, like the others in this list. It is included in all returns as an attribute of the <client> and <group> tags. Autostart, if enabled, causes any stream that is lost due to NetXCmd or a NetXCode stopping to be automatically restarted when they are restarted. There are two levels of autostart:

1. <client> autostart must be enabled for any kind of autostart to occur on a client (NetXCmd) server. If this is disabled, the next level is ignored.
2. <group> if the client autostart is enabled, then the NetXCmd will look at the group autostarts to determine if a group should be restarted when NetXCmd or NetXCode gets closed. This allows the caller to setup a server that only restarts some channels automatically.

getDiscontinuities

Get the xml containing the info on discontinuities encountered during a capture.

You must make this request in 2 parts, first send a set request with the filename of the source, ex:

http://192.168.100.234:1080/netx?
request=set&client=192.168.100.234&command=getDiscontinuities&source=C:\Users\Ryan\Videos\record0\Channel0.mp4

Then send the get request with the same source, ex:

http://192.168.100.234:1080/netx?
request=get&client=192.168.100.234&command=getDiscontinuities&source=C:\Users\Ryan\Videos\record0\Channel0.mp4

getCopyInOut

Get the byte ranges necessary to clip a file in a given timecode range.

You must make this request in 2 parts, first send a set request with the filename of the source, an in timecode, and an out timecode, ex:

http://192.168.100.234:1080/netx?
request=set&client=192.168.100.234&command=getCopyInOut&source=C:\Users\Ryan\Videos\record0\Channel0.mp4&in=18:58:23:51&out=18:58:43:51

Then send the get request with the same source, in, and out, ex:

http://192.168.100.234:1080/netx?
request=get&client=192.168.100.234&command=getCopyInOut&source=C:\Users\Ryan\Videos\record0\Channel0.mp4&in=18:58:23:51&out=18:58:43:51

The returned xml will contain the file you need to get off tape (originalFile) as well as a target filename (tempFile). You'll need to bring the byte ranges (in,out) of the originalFiles off the tape and name according to their tempFile. Put all the temp files in the same folder (<tempFolder>). Then make the copy request as normal with an additional parameter: &tempfolder=<tempFolder>.

You may get multiple files back in the xml, this means the audio and video are in separate files, ex:

```
<getCopyInOut>
   <copyInOutInfo>
      <originalFile>
         C:\Users\Ryan\Videos\record0\Channel0\media\Channel0_v00.mxf
      </originalFile>
      <tempFile>Channel0_v00_87706372_191637286.mxf</tempFile>
      <in>87706372</in>
      <out>191637286</out>
   </copyInOutInfo>
   <copyInOutInfo>
      <originalFile>
         C:\Users\Ryan\Videos\record0\Channel0\media\Channel0_a00.mxf
      </originalFile>
      <tempFile>Channel0_a00_2526245_5519636.mxf</tempFile>
      <in>2526245</in>
      <out>5519636</out>
   </copyInOutInfo>
   .
```

.
.
</getCopyInOut>

getFirstAndLastTimecode

Get the first and last timecodes from a file.

You must make this request in 2 parts, first send a set request with the filename of the source, ex:

http://192.168.100.234:1080/netx?
request=set&client=192.168.100.234&command=getFirstAndLastTimecode&source=C:/Users/Ryan/Videos/record0/Channel0.rtin

Then send the get request with the same source, ex:

http://192.168.100.234:1080/netx?
request=get&client=192.168.100.234&command=getFirstAndLastTimecode&source=C:/Users/Ryan/Videos/record0/Channel0.rtin

Command – Copy, Convert and PFR

***UPDATE**: Copy status and purge calls now require the group id assigned by the system. It starts at -2 and counts down into negative numbers. -1 will apply the command to all copy groups for the given client.*

COMMANDS: Require client address only. The Net-X-Copy application will only run as a single instance on a given machine. Group id and channel id will be ignored. The command will always equal copy, regardless of whether the desired effect is copy or convert.

PARAMETERS: source, target, profile, type, tcin, tcout, absin, absout, tcoffset, tc, copylimit, copy, dest, width, height, uuid, kilobitrate, h26xprofile, h26xlevel, gopsize, encodemode, tempfile, ccfile, afile, vfile

TRIGGERS: clear, abort, purge

set – initiate copy/convert/pfr

Can be used to initiate a copy or conversion, if the profile parameter is not specified, then a simple file copy will be performed. Additionally, the set command can be used to clean up old or stale copy requests, or cancel/clear active and/or pending copy requests.

A typical copy/conversion initial call would be as follows:

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&command=copy&profile=copy&source=C:/sourcepath/sourcefile.mp4&target= C:/targetpath/targetfile.mp4

There is also a multi source/target version of the copy. The profile, in and out must be the same, but multiple source and associated targets can be sent as one command.

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.101&command=copy&source=E:/Record/netx/copy/source/file001.mov&target=E:/Record/netx/copy/target/file001.mov&source=E:/Record/netx/copy/source/file002.mov&target=E:/Record/netx/copy/target/file002.mov&profile=copy

To create a RTIN index of a file, use the index command. If the target directory and file name are not

specified, then the RTIN will use the same base directory and file name of the source file, with the rtin extension. Please note, when restoring the index file, the source file must be in its original location or in the same directory as the index file, so that it can be found.

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&command=copy&profile=index&source=C:/sourcepath/sourcefile.mxf&target= C:/indexfilepath/sourcefile_index.rtin

To run the copy commands on a particular server, the client is normally specified as an IP address with 'client=192.168.100.176'. If you are running multiple servers, NetXBase can automatically select the next available server. To use this round robin method, specify 'client=any' in the command.

http://127.0.0.1:1080/netx?
request=set&client=any&command=wrap&source=\mnt\server\media\source.mxf&target= \home\user\Videos\edit.mxf&tcin=01:01:30:00&tcout=01:02:00:00

To create a picon (JPEG picture icon) of the frame at timecode 01:01:30:00 at 10% of the original file, use a command similar to this:

http://127.0.0.1:1080/netx?
request=set&client=any&command=copy&pisrc=\mnt\server\media\source.mxf&pidst= \home\user\Videos\edit_01013000.picon.jpg&piSize=10&piframe=01:01:30:00

If the profile is not specified and the tcin or tcout parameters are, a copy will take place instead of a conversion. Specifying the profile will initiate a conversion, if the tcin and tcout parameters are not specified then the entire file will be converted.

Parameters:

command=copy
source=<sourcefile>
target=<targetfile>
ack=<ackfile>
tcin=start as a timecode reference
tcout=end as a timecode reference (exclusive)
absin=first frame number in absolute frames from 0
absout=last frame number in absolute frames from 0 (exclusive)
tcoffset=amount to offset the timecode of the output clip. Does not affect the tcin/tcout.
copy=make a copy of the file section we need, instead of reading directly
dest=folder or folder and file name for the temp file when using copy
type=
- mxf = Normal OP1a MXF
- mxf-as02 = AS-02 MXF
- mxf-p2 = Panasonic P2 MXF
- mxf-sonyhd = Sony style OP1a
- mxf-xdcam = True Sony XDCam

profile=
- copy = copy any file completely
- index = make an rtindex file for a source file
- wrap = re-wrap a section of a file to another file of the same type and codec
- mp3-128kbps = Audio only MPEG Layer 3 audio
- mov-YCbCr8Bit = QuickTime MOV uncompress 8 bit YcbCr
- mov-dvcprohd = QuickTime MOV dv-100
- mp4-h264 = QuickTime MOV h.264/AVC1
- fmp4-h264 = DASH fragmented MP4
- ism = Microsoft Smooth Streaming
- mxf-xdcam-720p = MXF OP1a XDCam
- mxf-dvcprohd-720p = MXF OP1a DVCPro HD

mxf-xdcam-1080i = MXF OP1a XDCam
mxf-dvcprohd-1080i = MXF OP1a DVCPro HD
mov-prores422 = QuickTime MOV ProRes 4:2:2
mov-proreshq = QuickTime MOV ProRes HQ
mxf-OP1a-MPEG = XDCam like with up to 32 audio channels
mxf-OP1a-HDF = HDF XDCam Format
mxf-as-11-hd-dpp = AS-11 DPP HD 1080i50
mxf-as-11-sd-dpp = AS-11 DPP SD IMX 50
hd1080-5000kbs = MP4 - HD 1080 with a target bitrate of 5 mbs
hd720-2500kbs = MP4 - HD 720p with a target bitrate of 2.5 mbs
hd360-1250kbs = MP4 - HD 360p with a target bitrate of 1.25 mbs
h264-7500kbs = MP4 - Any resolution with a target bitrate of 7.5 mbs
Proxy h264-5000kbs = MP4 - Any resolution with a target bitrate of 5 mbs
LBR h264-10000kbs = MP4 - Any resolution with a target bitrate of 10 mbs
mxf-OP1a-JPEG2K = MXF OP1a JPEG-2000 4:2:2
mxf-AS-02-h264-10 = MXF AS-02 h.264 50 mbs
DASH MP4 Mutibitrate = MP4 - DASH MP4 Mutibitrate
HLS TS Mutibitrate = TS - HLS Mutibitrate
TS TR-01 JPEG-2000 = TR-01 Transports Stream JPEG-2000 150 mbs

Width=Sets the width of the picon or video output, when not a fixed size codec (like DVHD)
Height=Sets the height of the video output, when not a fixed size codec (like DVHD). No effect on picons.
piSrc=Source file to be used to create a picon JPEG image
piDst=Destination file for the JPEG picon image
piSize=Percentage size for the JPEG, integer 1..100% (eg. 1920x1080 @ 17 would be 320x180)
piFrame=Either the absolute frame value (integer) or a timecode string of the frame to use
piSkip=Number of frames to skip between picons. If set, as series of picons will be produced at 'skip' intervals

Set can also be used to cancel and or clear processes and information.

To set the max number of copy operations for the client:
http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&command=copy&copylimit=5

To clear what is in the current working information, call:
http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&command=copy&clear=

To abort the current process, call:
http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&command=copy&abort=

To clean out the copy/convert history for all items, call:
http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=-1&command=copy&purge=

To clean out the copy/convert history for a single item, call as you would for the purge trigger and specify the source, target, and profile.

The triggers for clear, purge and abort can all be used simultaneously for varied effects:

abort + clear will abort the current process and clear out any pending copy/conversions.

copylimit

Set the maximum number of simultaneous copies allowed on one server.

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=1&command=copy&copylimit=5

get - status/completion

Is used to obtain information about current or past copy requests. There are two queues of information; the active copies and the completed copies. To get the completed copies, add 'complete=1' to the request. If the source and target parameters are not set, then information about the current copy request is returned. If they are specified, then the system will try to locate information about the specific copy/conversion from memory. (NOTE: if a purge request was made then the specified item may no longer exist in memory)

A simple get to obtain information about the current process would be as follows (note that not specifying a group is the same as specifying the -1 all group):

http://127.0.0.1:1080/netx?request=get&client=192.168.100.176&command=copy
http://127.0.0.1:1080/netx?request=get&client=192.168.100.176&group=-1&command=copy

Alternately, a copy request may be found by specifying the complete target of the copy request and setting the group to 'find':

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&command=copy&group=find&target=/mnt/targetdir/targetfile.mxf

To get the status of a specific active copy, specify the running copy as the group value. For copies, group=-1 means all copies. Group=-2 or less specifies a specific copy. The maximum total copies on a server is dependent on how many have been sent and what the copy limit has been set to.

http://127.0.0.1:1080/netx?request=get&client=192.168.100.176&group=-2&command=copy

A get with the id, source, target and profile specified will get what is known for that particular copy/conversion.

To get a list of the completed copy request, add the complete=1 flag

http://127.0.0.1:1080/netx?request=get&client=192.168.100.176&command=copy&complete=1

By default, calls to the copy complete do not purge the complete status. This is so that multiple callers can all get the complete for a single file. To remove the complete, a 'purge' must be added.

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&command=copy&complete=1&purge=1

To get a list of the active copy requests, the command would be:

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&command=copy

If the client is unknown, it can be found by using client=any with the target specified by the copy request.

http://127.0.0.1:1080/netx?
request=get&client=any&target=###

Will not remove or purge copies that have errors or are marked as complete.

Parameters:
target=<targetfile>

Because copy slots are re-used, it is possible to have more the one complete for the same group. If you are asking for the group specifically, you will need to ask for each complete until there are no more. If you are asking for the -1 all group, in the return the multiple completes for a single channel will all be there. Once a complete (or progress) has been requested, it is removed from our list and is no longer available

Typical Proxy, Convert and PFR Session

This section has some typical proxy, convert and partial file restore workflows. The first scenario is a complete list. The other scenario's assume the first few steps in the first scenario have already been accomplished.

Scenario 1 – full access

- The file (MXF, MOV, AVI, CINE, etc) arrives at ingest
- A "command=copy&profile=index" command is sent to index to original filename
- A "command=copy&profile=mp4-h264" command is sent to make a proxy file with timecode, multitrack audio, closed captions, metadata and proxy index filename
- Two "command=copy&pisrc&pidst" commands are sent to create JPEG images for the source and proxy files
- *At this point, the proxy and main index can be stored in a real or near real time storage, and the main file may be moved to long term storage, tape, cloud (google/s3) or other offline storage*
- The user used the HTML5 player's time code (or other timecode source) to set one or more In and Out points on the file that needs to be restored.
- A "command=copy&profile=wrap" is sent to access the bytes of the original file and create a new file of the same type, without any recompression of audio or video, at the target location

Scenario 2 – tape restore

- *This assumes the basic processing in Scenario 1 has been done*
- A "command=getcopyinout" is sent with the absolute or timecode based in and out points, and the index of source file it will come from
- This returns a series of one or more file names with start and end byte locations
- *At this point, the controller restores those byte areas of the files to the name specified by the return*
- Once the areas are restored, a "command=copy&profile=wrap" is sent along with the temp folder to create the new output filename

Scenario 3 – cloud restore

- *This assumes the basic processing in Scenario 1 has been done*
- If the index file is stored on cloud, it can be restored locally first, or read directly from the cloud (https, ftps, aws)
- If the main file is in Glacier, then a command will be sent to restore the section needed to S3 before the restore is done
- Once there is access to the file, or file part, the "command=copy&profile=wrap" can be called normally
- If the resource is in available cloud storage (e.g. not Glacier), then partial file restores may be done from the original file without indexing it first. This will cause more data to be read, but only the headers and tables necessary to find the audio/video/data the restoration needs

Scenario 4 – in line conversions

- *This assumes the basic processing in Scenario 1 has been done*
- For any restore scenario, the file restored can be a byte accurate re-wrap of the original into a new container, or it can be translated in process (on the fly) to any supported standard format. These formats include MXF Op1a, Op-Atom, P2, IMX, D11, IMF, MOV, Uncompressed and many other containers, with codecs including JPEG-2000, XDCam, MPEG-2, h.264, HEVC, AVCi 100/200, XAVC-S. XAVC, Long-G, TR-01, DV and many others

- The commands can also be used with or without index files to convert all our part of local clips to any of these formats

Command - set

COMMANDS: Require client address and group id or group keyname, as well as a command (start, stop, restart, add, remove). These commands (start, stop, restart, add and remove) cannot be combined with the SETTINGS commands listed below. To have a command applied to all available groups, use the 'group=all'.

(Note that result values returned in xml structure indicate that the command was identified and dispatched by the HTTP server, they do not reflect the success or failure of the command result)

start

Start a group recording

http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0&command=start

```
<?xml version="1.0" encoding="ISO88591"?>
<request type="command">
        <key>group_key_name</key>
        <start success="true">success</start>
        <result>1</result>
</request>
```

startstream

Start a group/channel streaming

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=0&command=startstream

```
<?xml version="1.0" encoding="ISO88591"?>
<request type="command">
        <key>group_key_name</key>
        <start success="true">success</start>
        <result>1</result>
</request>
```

stop

Stop a group record.

http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0&command=stop

suspend

Suspend a group so it will not auto start when NetXCmd starts.

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&command=suspend

restart

Restart a group and make all settings current.

You may supply the index of the group:

http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0&command=restart

Or the keyname of a group:

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=ab1&command=restart

Or -1 or all to restart all groups:

http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=all&command=restart
http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=-1&command=restart

Or -2 to restart all the groups as well as the client:

http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=-2command=restart

xmldisable

Disable XML side car generation for captured files

This command gets the state of XML side car generation

http://192.168.100.229:1080/netx?
request=set&client=192.168.100.229&group=0&channel=0&xmldisable=1

If set to 0, no side car XML files will be generated. If set to one, standard Drastic XML side car files will be generated.

autostart

Autostart, if enabled, causes any stream that is lost due to NetXCmd or a NetXCode stopping to be automatically restarted when they are restarted. There are two levels of autostart:

1. <client> autostart must be enabled for any kind of autostart to occur on a client (NetXCmd) server. If this is disabled, the next level is ignored.
2. <group> if the client autostart is enabled, then the NetXCmd will look at the group autostarts to determine if a group should be restarted when NetXCmd or NetXCode gets closed. This allows the caller to setup a server that only restarts some channels automatically

To set the client/NetXCmd autostart, use

http://127.0.0.1:1080/netx?request=set&client=192.168.100.126&autostart=0

To set the group within a client, use

http://127.0.0.1:1080/netx?request=set&client=192.168.100.126&group=0&autostart=1

note: If setting autostart during group creation, the &parameter=value set needs to be included only once per group, it is not on a per channel basis. If not specified, the default value for the group is 1 (true). For each

NetXCmd instance that runs, the default value will also be true until it is changed by the user

add

Add a new group (group # not used for this action). Like all commands, this will return as soon as the command has been validated, but likely before the group has actually been created. A request for the channel using the key= name can determine when the channel has actually been allocated. There are 3 ways to add a group;

- You must supply a unique key name and it must contain at least one alpha character that isn't -.
- 0, 5, 243, and -1 are all invalid key names.
- aa, 365b, and -5abc are all valid.

a. add an empty group:

http://127.0.0.1:1080/netx?request=set&client=127.0.0.1&command=add&key=groupname

b. setting all channels with defined default values:

http://127.0.0.1:1080/netx?
request=set&client=127.0.0.1&command=add&key=groupname&channel=-1&address=239.255.40.40&port=1234&type=.mov&protocol=UDP

c. setting one or all channels with specific values:

http://127.0.0.1:1080/netx?
request=set&client=127.0.0.1&command=add&key=groupname&channel=0&address=239.255.40.40&name=chan1&port=1111&type=.mp4&protocol=RTP&channel=1&name=chan2&port=1222&address=239.255.41.41&type=.ts&protocol=UDP&channel=2&name=chan2&port=1333&address=239.255.42.42&type=.mov&protocol=RTP

NOTICE: When adding groups with channel settings, the channel specified must exist before its settings. &channel=0&name=name will work, &name=name&channel=0 will not. However, the order of the channels does not matter.

remove

Remove a group. Please note, this will only work on one channel at a time. The 'group=all' is not supported for this command.

http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0&command=remove

starttc/endtc

Set the desired start/end time for the given channel(s). Any channel(s) must be inactive for the setting to take effect. Once the tc values have been set and the given channel(s) have started, then the tc values will no longer be valid for subsequent connections and must then be set again. The format for the tc value is simply 24 hour time of day with an option date part. The two forms are:

##:##:##:## (e.g. 17:00:00:00 for 5pm)
##:##:##:##-dd-mm-yyyy (e.g. 07:00:00:00-02-01-2017 for 7AM 2nd of January, 2017)
Note: all digits must be filled in, including leading zeros. 8 digits for timecode, 2 for day, 2 for month and 4 for year.

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=0&starttc=11:11:11:11&endtc=22:22:

22:22

Delete

Delete one or more files on the file system

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.101&command=delete&file=E:/Record/netx/copy/source/temp.txt
http://127.0.0.1:1080/netx?
request=set&client=192.168.100.101&command=delete&file=E:/Record/netx/copy/source/temp1.txt&file=E:/Record/netx/copy/source/temp 2.txt

tsenable

Enable secondary transport stream capture. If capturing an MXF, MP4, fMP4, MOV or other main file type, enabling this will cause a second capture of the raw transport stream, with associated rtin and m3u8, to be captured as well. Before enabling, the tsdir and tsfile should also be set.

http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0&tsenable=1

tsdir

Set the directory where the secondary transport stream capture will be stored. The tsenable must be 1 for this to have effect.

http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0HYPERLINK "http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0&command=remove"&**tsdir=/tmp/capture**

tsfile

Set the file name for the secondary transport stream capture. The tsenable must be 1 for this to have effect.

http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0HYPERLINK "http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0&command=remove"&**tsfile=hls.ts**

SETTINGS

Require client address, group id number of key name, channel, parameter and a value. Settings can now be grouped, so multiple settings to one channel can be sent as one command. The settings commands cannot, however, be combined with the commands above (start, stop, restart, add and remove). These would have to be sent as two separate commands.

- **group** – the group or key name

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=oldgroupname&key=newgroupname

- **name** - File name

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&name=myname

- **address** - Source stream address

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&address=239.255.40.44

- **port** - Source stream port

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&port=5004

- **protocol** - Network capture protocol: udp, rtp

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&protocol=rtp

- **directory** - File directory

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&directory=/Volumes/capturedrive/current/

- **rectc -** Timecode recording method:

0 Record every frame time code (with TC PID Only)
1 Record every I frame, interpolate between (recommended) (with TC PID Only)
2 Record the first time code, interpolate forward (with TC PID Only)
3 Record time code converted from first PTS, interpolate forward
4 Record time of day as time code, interpolate forward

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&rectc=1

- **type** - File target type: .ts, .mp4, .fmp4, .mov, .ism, .mxf, .h264

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&type=.mp4

- **previewenable –** Enable/disable preview for channel(s). Can be set across all groups/channels. Can also use the "preview" parameter tag.

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&previewenable=1

- **threshold –** Stream data rate threshold. Set the data rate alarm limit.

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&threshold=6291456

The settings commands can also be combined into one command, but settings commands cannot be mixed with the group commands above. The settings must be set, then the group command sent separately. For example:

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=1&name=myname&address=239.255.40.44&port=5004&protocol=rtp&directory=/Volumes/capturedrive/current/&type=.mp4
http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0&command=restart

NOTE: if group is actively capturing from a stream, setting changes will not take effect until the group is restarted

NOTE: it is recommended to update all statuses for any client that has a command performed on it

QC Processing

NetXCode supports up to 100 audio/video QC sessions via set commands. These commands will take the IDs between -200 and -299.

Watch Folders

Up to 100 watch folders can be setup to trigger other NetXCode processes. These commands will take the IDs between -300 and -400.

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.126&group=0&channel=0&watchstart=&watchfolder=E:/watch/source&watchprofile=analize&watchfolder=E:/watch/target

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.126&group=0&channel=0&watchid=-300&watchcommand=start

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.126&group=0&channel=0&watchid=-300&watchcommand=stop

Command - callstate

REQUIREMENTS: The client address and group id or group keyname as well as the command. If the command pertains to a channel, the channel index, parameter and value are required.

The "callstate" request is used to determine if a previous set command or set channel value has been processed by the system. There is a timeout involved with this functionality for performance concerns. The structure of the "callstate" command is the same as the set request for channel parameters, and very similar to the set request for a command. A typical cycle will look like this:

- **issue command** - Start a group recording
http://127.0.0.1:1080/netx?request=set&client=192.168.100.176&group=0&command=start

The response from the http server states that the request was received and issued to the system.

```
<?xml version="1.0" encoding="ISO88591"?>
<request type="command">
        <key>group_key_name</key>
        <start success="true">success</start>
        <result>1</result>
</request>
```

- **callstate (command)** - Query the system to see if it has been processed
http://127.0.0.1:1080/netx?request=callstate&client=192.168.100.176&group=0&start=

The reponse from the http server shows a result code of -1 if the command does not exist in memory or if it has timed out. Otherwise the result will either be 0 if it has not been processed yet, or 1 if it has.

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.1678.100.176"groups=”2” autostart=”1”>
   <group index="0" autostart=”1”>
      <param>start</param>
      <result>1</result>
      <code>-1</code> <-- recorded error
   </group>
</client>
```

- **issue set parameter value** - Set name of a channel

http://127.0.0.1:1080/netx?
request=set&client=192.168.100.176&group=0&channel=0&name=Channel1

```
<?xml version="1.0" encoding="ISO88591"?>
<request type="channel">
        <setting success="1">success</setting>
        <result>1</result>
</request>
```

- **callstate (channel param)** - Query the system to see if it has been processed

http://127.0.0.1:1080/netx?
request=callstate&client=192.168.100.176&group=0&channel=0&name=Channel1

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.1678.100.176" groups=”1” autostart=”1”>
   <group index="0" autostart=”1”>
      <channel index="0">
         <param>name</param>
         <value>Channel1</value>
         <result>1</result>
         <code>-1</code> <-- recorded error
   </group>
</client>
```

NOTE: Once a command or set parameter request has been issued, it is recorded in memory, it will become stale after 10 seconds if not set to 1 by the system indicating that it has been handled. Once it has been set to 1 however, it will live for an additional 10 seconds before being discarded. The exception is if the system detects that a request has failed, then the expire time will be set to 2 minutes from when the error was detected.

At any time the system can be queried for errors by making the following call:

http://127.0.0.1:1080/netx?request=callstate&client=192.168.100.176&errors=

(The client need not be specified for this)
http://127.0.0.1:1080/netx?request=callstate&errors=

The response will either be a statement that there are not any errors, or a list of requests that failed with an error count. Keep in mind that if an error was detected and it occurred longer than 2 minutes prior to the error check, it will not be reported.

Response with errors:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<errors count="2">
   <error index="0">
      <client address="192.168.100.109">
         <group index="0">
            <channel index="0">
               <param>name</param>
               <value>Channel1</value>
               <result>1</result>
               <code>-1</code> <-- recorded error
               <description>unable to set channel parameter</description
            </channel>
         </group>
      </client>
   </error>
```

```
   <error index="1">
      <client address="192.168.100.109">
         <group index="0">
            <channel index="0">
               <param>address</param>
               <value>239.40.40.41</value>
               <result>1</result>
               <code>-1</code> <-- recorded error
               <description>unable to set channel parameter</description>
            </channel>
         </group>
      </client>
   </error>
</errors>
```

Response without errors:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<errors count="0" />
```

NOTE: If an error occurs or if the command has not been processed and it expires, it will be written to a log file called "NetXCmdErrs.log" in the netxlog folder found in the users home directory.

Command – close/restart

To close or restart a server on a client, use close or restart command with that client's IP address:

http://127.0.0.1:1080/netx?request=close&client=192.168.100.176

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<request close="shutting down NetXBase server">
  <result>1</result>
</request>
```

http://127.0.0.1:1080/netx?request=restart&client=192.168.100.176

Response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<request restart="restarting NetXBase server">
  <result>1</result>
</request>
```

Command – Thumbnail/JPEG/Picons

NetXBase also has the ability to create thumbnail/picon images for frames of video within a recorded or recording file, as well as return images for the current capture as a confidence monitor. The three main types of generation are: live confidence monitor, image on disk creation, image return as mime/jpeg for HTML. This feature is enabled or disabled with the 'previewenable' command.

Live Confidence Monitoring

For any recording stream, a request can be made for the last I frame captured as a JPEG thumbnail. This can be returned as an XML with an embedded base64 JPEG image, or as a mime/jpeg for use directly in an <IMG> tag in a web page.

preview

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=preview

Getting a preview jpeg

The data returned is base 64 encoded jpeg data located in the following xml structure:

```
<?xml version="1.0" encoding="ISO88591"?>
<client address="192.168.100.176" version=v4.2.0.274 groups="1" autostart=”1”>
   <group index="0" key="GroupName" autostart=”1”>
      <channel index="0">
         <pts>432400</pts>
          <preview timestamp=”6521600”>(base 64 jpeg data)
          </preview>
      </channel>
   </group>
</request>
```

mpreview

Getting a preview jpeg returned as a mime/jpeg suitable for an <IMG> tag.

http://127.0.0.1:1080/netx?
request=get&client=192.168.100.176&group=0&channel=0&value=mpreview

The data returned is raw jpeg, marked in the HTML header as mime/jpeg.

Create JPEGs On Disk

NetXCopy can be used to create a JPEG image from a file (live or prerecorded) at any valid frame to any available location, from the NetXCode server used. Please note, if making images from recording streams you must use the associated RTIN file and not the main file of the record. Once the record is complete, either may be used, but the RTIN will be faster.

Picon

Create a JPEG picon to a target dir using the source name

http://127.0.0.1:1080/netx?
request=picon&pisrc=E:/Record/netx/copy/target/dtl001.mov&pidst=E:/Record/netx/copy/target&pisize=10&piframe=25

Create a JPEG picon to a target name

http://127.0.0.1:1080/netx?
request=picon&pisrc=E:/Record/netx/copy/target/dtl001.mov&pidst=E:/Record/netx/copy/target/dtl001.picon.jpg&pisize=10&piframe=25

To create a series of JPEG files from a file, with a distance between frames, the piskip parameter can be used. This command will create a numbered jpeg icon every 60 frames, starting at frame 150. This is useful for

creating HTML scroll previews.

http://127.0.0.1:1080/netx?
request=picon&pisrc=\mnt\Record\netx\target\dtl001.mov&pidst=\mnt\Record\netx\proxy\/dt.jpg&width=180&piframe=150&piskip=60

Return A JPEG From A File

To get a JPEG suitable to display on a web page as an <IMG>, a call is made directly to the NetXBase. NetXBase must handle this request, as the picon has to be generated for the return from the call. As such, if you are doing a lot of generation, a separate instance of NetXBase should be set up, so that it does not interfere with the other operations NetXBase is handling for capture and clipping.

Picon

Create a JPEG picon and return it as a mime/jpeg

http://127.0.0.1:1080/netx?
request=picon&pisrc=E:/Record/netx/copy/target/dtl001.mov&pidst=&pisize=10&piframe=25
http://127.0.0.1:1080/netx?netx?
request=picon&pisrc=E:/Record/netx/copy/target/dtl001.mov&pisize=10&piframe=25
http://127.0.0.1:1080/netx**?**
netx?request=picon&pisrc=E:/Record/netx/copy/target/dtl001.mov

NetXSDI – MediaCMD

Start NetXSdi.exe
http://192.168.100.126:1080/netx?
request=set&client=192.168.100.126&group=sdi&sdistart&channel=0

Stop NetXSdi.exe
http://192.168.100.126:1080/netx?
request=set&client=192.168.100.126&group=sdi&sdistop&channel=0

Get status:
http://192.168.100.126:1080/netx?
request=set&client=192.168.100.126&group=sdi&sdistatus&channel=0

Process MediaCmd
http://192.168.100.126:1080/netx?
request=set&client=192.168.100.126&group=sdi&sdicmd&channel=0&play

Also there are:
sdipicon
sdinextclip
sdiclipinfo
sdiedlstate
sdiedlinfo
sdinextdirentry
sdimakedir
sdifileinfo
sdierrormsg
sdierrorlogsave

OLD:
http://localhost/VVWXMLMediaCmd?Play&speed=65520

REPLACE:
http://localhost:1080/netx?
*request=set&client=192.168.100.176&group=sdi&sdicmd&*channel=0&play&speed=65520

http://localhost:1080/netx?request=set&client=192.168.100.176&group=sdi&sdicmd&

SetCurChannel

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.

Play

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Play

Play at normal speed.

PlayAtSpeed

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Play&speed=lVVWSpeed

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

Returns 0 if successful, else an error code.

PlayFromTo

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Play&start=lFrom&end=lTo

Play from a frame to another frame. As with editing systems, the 'from' point is included and will be displayed but the 'to' point is NOT included and will not be displayed. This means that the last frame displayed will be lFrom – 1. The deferred flag allows PlayFromTos to be stacked so that they will play back to back. The deferred flag in the status return should be false before another deferred command is added.

Returns 0 if successful, else an error code.

LoadClip

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Pause&ClipID=szClipname

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

Returns 0 if successful, else an error code.

PlayClip

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&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

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Play&start=lFrom&end=lTo&ClipID=szClipname

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

Returns 0 if successful, else an error code.

FastForward

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Play
&speed=655200

Set the channel into its fastest possible forward motion state.

Returns 0 if successful, else an error code.

FastRewind

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Play
&speed=-655200

Set the channel into its fastest possible reverse motion state.

Returns 0 if successful, else an error code.

Pause

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Pause

Stop playback and display the current frame.

Returns 0 if successful, else an error code.

Seek

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Pause
&position=lFrame

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

Returns 0 if successful, else an error code.

SeekRelative

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Pause
&position=lFrameOffset

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

Stop

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&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

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&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

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Record
&start=lStart&end=lEnd

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

Returns 0 if successful, else an error code.

RecordStop (prepare record)

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&RecStop

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

Returns 0 if successful, else an error code.

SetRecordPresets

http://localhost:1080/netx?
r**equest=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Record&videochannels=lVidEdit&audiochannels=lAudEdit&infochannels=lInfEdit**

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

Returns 0 if successful, else an error code.

Eject

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Eject

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

Returns 0 if successful, else an error code.

Transfer

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Transfer
&channel=lTargetChannel&position=lPosition
&start=lStart&end=lEnd&videochannels=lVidEdit
&audiochannels=lAudEdit&infochannels=lInfEdit
&Flags=Invert

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

Returns 0 if successful, else an error code.

Update Status

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&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

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&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)

GetSpeed

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&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

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&VVWXMLGetStatus?

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

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

GetLastMS

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&VVWXMLGetStatus?

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

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

GetStart

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&VVWXMLGetStatus?

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

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

GetEnd

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&VVWXMLGetStatus?

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

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

GetClipName

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&VVWXMLGetStatus?

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

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

GetCurTC

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&VVWXMLGetStatus?

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

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

GetCurState

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&VVWXMLGetStatus?

Returns the current state as a string (e.g. "Play"). For Direct access, the memory must always be at least 15 bytes long (14 byte state + NULL) and is always ANSI.

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

GetNextClip

http://localhost:1080/netx?

request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&VVWXMLNextClip?

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

Returns 0 if successful, else an error code.

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

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

GetClipInfo

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&VVWXMLClipInfo?

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

Returns 0 if successful, else an error code.

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

EDLGetEdit

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&position=0&videochannels=0&audiochannels=0&infochannels=0

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

Returns 0 if successful else an Error code.

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

```
<?xml version="1.0" ?>
<MediaCmd>
        <!--  Drastic MEDIACMD xml structure version 1,0 -->
        <CmdID Value="-98238205" />
        <StructSize Value="336" />
        <Channel Value="0" />
        <Cmd Value="14" UseClipID="1">GetValue</Cmd>
        <VideoChannels Value="1" />
        <AudioChannels Value="0" />
        <InfoChannels Value="0" />
        <CmdAlt Value="93" />
```

```
        <Position Value="5" TcType="non-drop-frame">00:00:00:05</Position>
        <Start Value="0" TcType="non-drop-frame">00:00:00:00</Start>
        <End Value="5" TcType="non-drop-frame">00:00:00:05</End>
        <FileName>V:\Drastic Base Media\avi_er001_720x486_YUY2.avi</FileName>
</MediaCmd>
```

Insert

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Insert&ClipID=szClipNam
e&position=lPosition&start=lStart&end=lEnd&videochannels=lVidEdit&audiochannels=lAudEdit&i
nfochannels=lInfEdit&Flags=Ripple

Internal Channels Only. Do not use yet.

Blank

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Blank
&ClipID=szClipName&position=lPosition
&start=lStart&end=lEnd&videochannels=lVidEdit
&audiochannels=lAudEdit&infochannels=lInfEdit
&Flags=Ripple

Internal Channels Only. Do not use yet.

Delete

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Delete
&ClipID=szClipName&position=lPosition
&start=lStart&end=lEnd&videochannels=lVidEdit
&audiochannels=lAudEdit&infochannels=lInfEdit
&Flags=Ripple

Internal Channels Only. Do not use yet.

Trim

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&Trim&position=lPosition&
start=lStartOffset
&end=lEndOffset&videochannels=lVidEdit
&audiochannels=lAudEdit&infochannels=lInfEdit
&Flags=Ripple

Internal Channels Only. Do not use yet.

ValueSupported

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&ValueSupported&cmdalt=
valuetype&position=lValueType

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

ValueGet

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&GetValue&cmdalt=valuetype&position=lValueType

Returns the current setting for a get/set value.

ValueSet

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&SetValue&cmdalt=valuetype&position=lSetting

Sets the get/set value to setting.

Get/SetClipMode

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=clipmode&position=0

Calls ValueXXX with gsClipMode. If equal to 1 then the channel is in Clip mode, if 0 the channel is in VTR mode.

Get/SetTCType

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=tctype&position=2

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
```

Get/SetTCSource

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?GetValue&cmdalt=tcsource

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

Get/SetAudioInput

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsAudInSelect&position=lSetting&videochannels=0&audiochannels=lAudChannels&infochannels=0

ADD FUNCTION lAudIn

Get the current audio input.

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

Get/SetAudioInputLevel

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsAudInputLevel&position=lSetting&videochannels=0&audiochannels=lAudChannels&infochannels=0

Get the current audio input level. This requires capture hardware the supports input level setting.

Get/SetAudioOutput

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsAudOutSelect&position=lSetting&videochannels=0&audiochannels=lAudChannels&infochannels=0

Get the current audio Output – See Get/SetAudioInput

Get/SetAudioOutputLevel

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsAudOutputLevel&position=lSetting&videochannels=0&audiochannels=lAudChannels&infochannels=0

Get the current audio output level.

Get/AudioPeakRMS

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?GetValue&cmdalt=gsAudWavePeakRMS

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

Get/SetVideoInput

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsVidInSelect&position=lSetting

Get the current video input.

```
//! Standard NTSC or PAL composite video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPOSITE                  0x001
//! SVHS/S-Video four wire NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_SVIDEO                     0x002
//! Secondary NTSC or PAL video (often monitor selection) (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPOSITE_2                    0x004
//! BetaCam level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV              0x010
//! Panasonic M2 level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV_M2           0x020
//! SMPTE standard level YCrCb NTSC or PAL video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_YUV_SMPTE        0x040
//! RGB at video standard rate (cmdGetSetValue::gsVidInSelect cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_COMPONENT_RGB              0x080
//! D1 Serial Digital or HDSDI video (cmdGetSetValue::gsVidInSelect cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_D1_SERIAL                  0x100
//! D1 Serial Parallel video (cmdGetSetValue::gsVidInSelect cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_D1_PARALLEL                0x200
//! SDTI/SDI including high speed transfer video (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_SDTI                       0x400
//! No video available or no configurable settings (cmdGetSetValue::gsVidInSelect
cmdGetSetValue::gsVidOutSelect)
#define GS_VIDSELECT_NONE                       0
```

Get/SetVideoInputSetup

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsVidInSetup&position=lSetting

Get the current video input's 'Setup' TBC setting. This requires capture hardware that has a built in time base corrector.

Get/SetVideoInputVideo

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsVidInVideo&position=lSetting

Get the current video input's 'Video' TBC setting. This requires capture hardware that has a built in time base corrector.

Get/SetVideoInputHue

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsVidInVideo&position=lSetting

Get the current video input's 'Hue' TBC setting. This requires capture hardware that has a built in time base corrector.

Get/SetVideoInputChroma

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsVidInChroma&position=lSetting

Get the current video input's 'Chroma' TBC setting. This requires capture hardware that has a built in time base corrector.

Get/SetVideoTBCSetup

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsVidSetup&position=lSetting

Get the current global TBC's 'Setup' setting. This requires capture hardware that has a built in time base corrector.

Get/SetVideoTBCVideo

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsVidVideo&position=lSetting

Get the current global TBC's 'Video' setting. This requires capture hardware that has a built in time base corrector.

Get/SetVideoTBCHue

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsVidHue&position=lSetting

Get the current global TBC's 'Hue' setting. This requires capture hardware that has a built in time base corrector.

Get/SetVideoTBCChroma

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&channel=0&XMLMediaCmd?SetValue&cmdalt=gsVidChroma&position=lSetting

Get the current global TBC's 'Chroma' setting. This requires capture hardware that has a built in time base corrector.

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/netx?**request=set&client=192.168.100.176&group=sdi&sdicmd&<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 a 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 = "netx?request=set&client=192.168.100.176&group=sdi&sdicmd&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 = "netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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 = "netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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”;
}
```

sdicmd main commands

The first parameter of the **http://localhost:1080/netx?request=set&client=192.168.100.176&group=sdi&sdicmd&** (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

netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&<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

sdicmd Examples

netx?request=set&client=192.168.100.176&group=sdi&sdicmd&play
– Normal play

netx?request=set&client=192.168.100.176&group=sdi&sdicmd&play&speed=32760
– Play at 50% forward speed

netx?request=set&client=192.168.100.176&group=sdi&sdicmd&play&speed=-65520
– Play at 100% reverse play speed

netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&play&start=1:00&end=4:00&loop
– Play from one second to four seconds in a loop

netx?request=set&client=192.168.100.176&group=sdi&sdicmd&pause
– Pause the channel

netx?request=set&client=192.168.100.176&group=sdi&sdicmd&stop
– Stop (e to e passthrough) the channel

netx?request=set&client=192.168.100.176&group=sdi&sdicmd&pause&position=1:00:00
– Seek to one minute

netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&GetValue&cmdalt=1000000&clipid=Kroatien&position=4294967295
– Return the size of the picon file in the Position elements
– result: dwPosition = 7900

http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdicmd&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:1080/netx?request=set&client=192.168.100.176&group=sdi&sdigetstatus
– Returns an XML package including state, speed, position start and end points.

http://localhost:1080/netx?request=set&client=192.168.100.176&group=sdi&sdinextclip
– Returns an XML package with all the clip information. Used to retrieve the clip bin information

http://localhost:1080/netx?request=set&client=192.168.100.176&group=sdi&sdiclipinfo
– Returns an XML package with all the clip information. Used to retrieve information on a specific clip

http://localhost:1080/netx?request=set&client=192.168.100.176&group=sdi&sdiedlstate
– Used in conjunction with sdiedlinfo to retrieve the time code space edits. The command will always be sdiedlstate?position=#&videochannels=#&audiochannels=#&infochannels=#.

http://localhost:1080/netx?request=set&client=192.168.100.176&group=sdi&sdiedlinfo
– Used in conjunction with sdiedlstate to retrieve the time code space edits.

Here is a basic EDL retrieval session:

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

Take the MEDIACMD struct returned from sdiedlstate 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 sdiedlstate call and (other than first call) sdiedlstate should be called immediately before sdiedlinfo .

http://localhost:1080/netx?request=set&client=192.168.100.176&group=sdi&sdinextdirentry
– 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:1080/netx?request=set&client=192.168.100.176&group=sdi&sdinextdirentry&
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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdinextdirentry&\&\Record
Returns: <locator>\OfflineMedia</locator>
http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdinextdirentry&\&\OfflineMedia
Returns: <locator>\LocalMedia</locator>
http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdinextdirentry&\&\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:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdinextdirentry&\Record
Returns: <locator>\Record\..</locator>
http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdinextdirentry&\Record\&\Record\..
Returns: <locator>\Record\Test.wav</locator>
http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdinextdirentry&\Record\&\Record\Test.wav
Returns: <locator>\Record\Test.avi</locator>
http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdinextdirentry&\Record\&\Record\Test.avi
Returns: <locator>END OF LIST</locator>

http://localhost:1080/netx?request=set&client=192.168.100.176&group=sdi&sdifileinfo
– Used to retrieve information on a specific file.

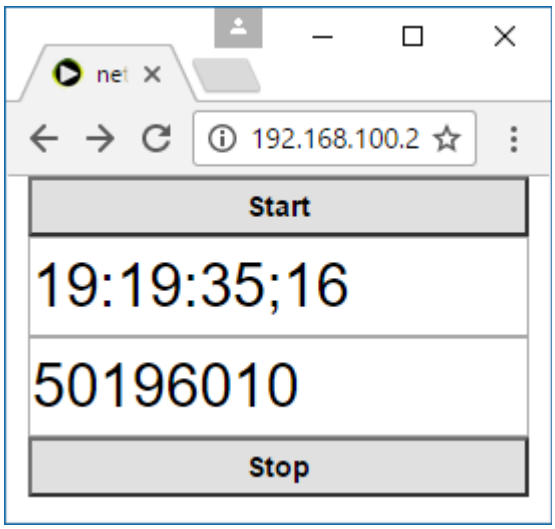
http://localhost:1080/netx?
request=set&client=192.168.100.176&group=sdi&sdigeterrormsg&#
– Used to return one error message from the current list. The first call will not include an error number (just sdigeterrormsg). This will return an ErrorNumber to use to get the next message (sdigeterrormsg&202 for instance), as will each subsequent call. When all the error messages have been returned, it will return an ErrorNumber of -1.

NetXTimeCode

The NetXTimeCode server is a server that captures an UDP or RTP stream of timecode metadata and converts it to a AJAX compatible HTTP server. The server records the latest timecode and userbits, which can then be viewed within the default HTML page, or requested via AJAX requests. The base command for NetXTimeCode is "/netxtc?". The base address is your machine's IP at port 1080. Eg.

http://127.0.0.1:1080

Which should bring up the default page and display a running timecode.



Set A Variable

Set one of the get/set variables described below

http://127.0.0.1:1080/netxtc?request=set&address=238.255.99.99

Get A Variable

Get one of the get/set variables described below

http://127.0.0.1:1080/netxtc?request=set&address=

Get/Set Variables

- All – used to return all the variables
- name – file name, not used
- port – incoming stream port
- protocol – RTP or UDP
- type – file type, not used
- state – current state
- desc – stream description
- tc – timecode
- ub - userbits
- frames – timecode in frames
- ms - milliseconds

Start Capturing Timecode

Start capturing timecode from the timecode stream source

http://127.0.0.1:1080/netxtc?request=command&start=

Stop Capturing Timecode

Stop capturing timecode from the timecode stream source

http://127.0.0.1:1080/netxtc?request=command&stop=

Net-X-Copy Command Line

NetXCopy -s <sourcefile> -t <targetfile> [-a <ackfile>] -p <profile> [-in <00:01:00:00> -out <00:02:00:00> -fg]

-s <sourcefile> The source file name and path
-t <targetfile> The target file name and path
-a <ackfile> The ACK file name and path. This is the XML acknowledgment file made after a copy
-p <profile> One of the following profile settings

- copy – copy the whole file
- wrap – re wrap file or part of a file
- index – create an RTIndex for a file
- mp3-128kbps - Audio only MPEG Layer 3 audio
- mov-YCbCr8Bit – QuickTime MOV 8 bit uncompressed YcbCr file
- mov-dvcprohd - QuickTime MOV DVHD 100
- mp4-h264 - MP4 with h264, original size
- mxf-xdcam-720p - MXF Sony XDCam 720p
- mxf-dvcprohd-720p - MXF DVHD 720p
- mxf-xdcam-1080i - MXF Sony XDCam 1080i
- mxf-dvcprohd-1080i - MXF DVHD 1080i
- mxf-OP1a-MPEG - MXF OP1a MPEG (XDCam like, 16 audio channels)
- mxf-op1a-h264 - MXF h.264
- mxf-OP1a-HDF - MXF MPEG HDF
- mxf-as-11-sd-dpp - MXF DPP SD IMX
- mxf-as-11-hd-dpp - MXF DPP HD AVCi
- mov-proreshq - QuickTime MOV ProRes HQ
- mov-prores422 - QuickTime MOV ProRes 422
- mxf-as-11-sd-pal-dpp - MXF AS-11 SD PAL DPP
- mxf-as-11-sd-ntsc-dpp - MXF AS-11 SD NTSC DPP
- scaledown2000k - MP4 264 960x540, 2mbs
- scaledown500k - MP4 264 480x272, 0.5mbs
- hd1080-5000kbs - MP4 - HD 1080 with a target bitrate of 5 mbs
- hd720-2500kbs - MP4 - HD 720p with a target bitrate of 2.5 mbs
- hd360-1250kbs - MP4 - HD 360p with a target bitrate of 1.25 mbs
- h264-7500kbs - MP4 - Any resolution with a target bitrate of 7.5 mbs
- Proxy-h264-5000kbs - MP4 high quality proxy for web
- LBR-h264-10000kbs - Low bit rate, high quality local MP4
- mxf-OP1a-JPEG2K - Samma style JPEG2000 YCbCr
- mxf-AS-02-h264-10 - 10 bit 50 Mbs h.264 in AS-02 MXF
- DASH-MP4-Mutibitrate - Multi bitrate MP4s with DASH files
- HLS-TS-Mutibitrate - Multi bitrate TS streams with M3U8 files
- TS-TR-01-JPEG-2000 - TR-01 JPEG-2000 transport stream
- mxf-OP1a_HBR_50 - MXF OP1a h264 4:2:2 10 bit
- OP1a_HBR_50 - OP1a MXF h264 4:2:2 10 bit
- mp4-XAVC-S_4_2_0 - MP4 Sony XAVC-S 4:2:0
- mp4-XAVC-S_4_2_2 - MP4 Sony XAVC-S 4:2:2

-type mxf-op1a -- the exact file type to write, otherwise auto

- mxf-op1a – standard OP1a
- mxf-sonyhd – Sony XDCam compatible
- mxf-as02 – AS-02 spec MXF
- mp4-fmp4 – Fragmented MP4 (normal MP4 if not set)

-in <00:01:00:00> the starting point for the output file in time code or absolute position
-out <00:02:00:00> the ending point for the output file in time code or absolute position
-absin <200> the absolute (zero based) start time for the output file (overrides -in)
-absout <400> the absolute (zero based) end time, exclusive, for the output file (overrides -out)
-tcoffset <100> an offset to apply to all timecode operations
-tc <100> timecode
-width <width> the width of the video output, overrides when not fixed by codec
-height <height> the height of the video output, overrides when not fixed by codec
-copy make a copy of the file section we need, instead of reading directly
-dest folder or folder and file name for the temp file when using copy

-cc <mcc/scc file> -- replacement closed caption file>
-afile <path-audio-file> -- replacement source audio track<s>
-v <path-video-file> -- replacement source video track
-uuid <uuid string> - override the UUID of the file with this one
-kilobitrate <kbit rate> - override the kilo bit rate
-h26xprofile <baseline/main/high/high10/high422/high444> override the profile type
-h26xlevel <51> override the level
-encodemode <0/1> 0 normal, 1 fastest
-gopsize <15> size of encoded gop
-tempfolder – Temporary folder to store partial file
-fg force the GUI on
-fc force command line

For JPEG picons
-pisrc <file-to-picon> -- source for the
-pidst <where-to-make-picon> -- target folder and name
-pisize <size> -- size of picon, 100%-10%
-width <width> the width of the picon output, overrides piSize
-piframe <frame-offset-into-file> -- frame to use to make the picon
-piskip <Number of frames to skip between picons> If set, as series of picons will be produced at 'skip' intervals

NOTE: the parameters in [square brackets] are optional.

Discontinuity Sources In NetXCode

NetXCode tracks and saves any issues it encounters during capture as discontinuities. These are saved in the ACK output file with their time, timecode, absolute location and source. The possible sources are:

- PID continuity counter discontinuity
  - video
  - audio
  - dolby
  - timecode
  - metadata
  - closed caption
- Timecode dup/missing discontinuity
- UUID in metadata change
- No packets for over 100 millisecond discontinuity

Most of these are 'true' discontinuities, where there is an error in the transport layer that denotes a problem up stream. The one exception is the timecode discontinuity. Because the LBR/ABR are muxed, a missing video/audio frames may not show up as a discontinuity, as they occurred pre mux. The only indication (other then broken video/audio decodes) is that the timecode jumps, so it is important to treat this as a full discontinuity.

Error Returns

#define netxErrorNone 0
No error. Normal response.

#define netxErrorOk 1
When response is a boolean, this is equivalent to TRUE.

#define netxErrorUnknown -1
An undefined error.

#define netxErrorBadChannelParam -2
One of the parameters in the channel command is bad.

#define netxErrorClientMissing -3
A command was sent to a client that does not exists.

#define netxErrorStartFailed -4
A start command on a group failed.

#define netxErrorStopFailed -5
A stop command on a group failed.

#define netxErrorRemoveFailed -6
Unable to remove a group or a channel.

#define netxErrorBadGroupParam -7
One of the parameters in the group command is bad.

Error Returns – NetXCopy

0 – Success
1 – unable to create file (EPERM)
2 – no such file or directory (ENOENT).
5 – read or write error (EIO)
12 – unable to allocate memory
86 – unable to complete copy or conversion.
255 – unable to create index file or unable to index source.
256 – unable to acquire information from source file
257 – unable to set conversion center information
258 – unable to find TS associated with RTIN
259 – unable to retrieve frame information from RTIN
260 – start position in file greater the end position
261 – unable to set target file information
261 – DMF get bytes error

ACK/ACKC/ACKR File Format

The ACK(R)/Acknowledge file is created after a file is converted, clipped or captured with Net-X-Code/Net-X-Copy. This file will be created when the file it is associated with is complete and closed. It is an XML formatted file that includes information about the output file, its source and it only created once the output file is completely ready. Here is a sample file:

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<ACKNOWLEDGE>
  <DETAILS>
  <FILE NAME="NetXCode Capture" DSTNAME="47448317-983f-4b35-9ba5-
52265522a9ba_HBR.mxf">
    <RESULT>0</RESULT>
    <COMMENT>
Packets -> Total=546095529 [missing=2], Video=266125460 [dis=0],
Audio={29866438,29866438,29866438,29866439,0,0} [dis=0], Dolby=0 [dis=0], Tc=447550
[dis=0], MetaData=23087 [dis=0], CC=1790200 [dis=0] </COMMENT>
    <SRCPATH>NetXCode Capture</SRCPATH>
    <DSTPATH>/bass3fs/share/Ingest/recordings/2017/06/20/47448317-983f-4b35-9ba5-
52265522a9ba_HBR/47448317-983f-4b35-9ba5-52265522a9ba_HBR/47448317-983f-4b35-
9ba5-52265522a9ba_HBR.mxf</DSTPATH>
    <RTINPATH>/bass3fs/share/Ingest/recordings/2017/06/20/47448317-983f-4b35-9ba5-
52265522a9ba_HBR/47448317-983f-4b35-9ba5-
52265522a9ba_HBR/extra/drastictech.com/47448317-983f-4b35-9ba5-
52265522a9ba_HBR.rtin</RTINPATH>
    <MCCPATH>/bass3fs/share/Ingest/recordings/2017/06/20/47448317-983f-4b35-9ba5-
52265522a9ba_HBR/47448317-983f-4b35-9ba5-
52265522a9ba_HBR/extra/drastictech.com/47448317-983f-4b35-9ba5-
52265522a9ba_HBR_vanc.mcc</MCCPATH>
    <ACKPATH>/bass3fs/share/Ingest/recordings/2017/06/20/47448317-983f-4b35-9ba5-
52265522a9ba_HBR/47448317-983f-4b35-9ba5-
52265522a9ba_HBR/extra/drastictech.com/47448317-983f-4b35-9ba5-
52265522a9ba_HBR.ack</ACKPATH>
    <TIME>Wed Jun 21 00:05:01 2017
    </TIME>
    <INFO>
      <General>
        <Format>MXF AS-02</Format>
        <Duration>871908</Duration>
      </General>
      <Video>
        <Format>h.264 AVC1</Format>
        <FourCC>0x61766331</FourCC>
        <Width>1280</Width>
        <Height>720</Height>
        <BitCount>24</BitCount>
        <Rate>60000</Rate>
        <Scale>1001</Scale>
        <Length>871908</Length>
      </Video>
      <Audio>
        <Format>AES-3</Format>
        <Channels>8</Channels>
        <Frequency>48000</Frequency>
        <Bits>24</Bits>
        <Length>698223926</Length>
      </Audio>
      <Metadata>
        <Timecode>23:55:04;46 d</Timecode>
        <UserBits>10702600</UserBits>
        <VITCTimecode>23:55:04;46 d</VITCTimecode>
```

```
        <VITCUserBits>10702600</VITCUserBits>
        <UMID>828392c0-8539-4f81-bab4-b243d2771232</UMID>
        <PosterFrame>0</PosterFrame>
        <A-Frame>0</A-Frame>
       </Metadata>
      </INFO>
      <DISCONTINUITIES>
        <Discontinuity type="timecode" time="20:00:05 Tuesday, June 20, 2017" frame="17866"
LastITimeCode="318" DisconTimeCode="332" timecodetype="d" comment="CurrentTimeCode:
327">00:00:05;32</Discontinuity>
        <Discontinuity type="timecode" time="21:24:11 Tuesday, June 20, 2017"
frame="320312" LastITimeCode="302757" DisconTimeCode="302771" timecodetype="d"
comment="CurrentTimeCode: 302770">01:24:11;15</Discontinuity>
        <Discontinuity type="timecode" time="21:24:11 Tuesday, June 20, 2017"
frame="320327" LastITimeCode="302770" DisconTimeCode="302784" timecodetype="d"
comment="CurrentTimeCode: 302787">01:24:11;28</Discontinuity>
      </DISCONTINUITIES>
    </FILE>
    </DETAILS>
</ACKNOWLEDGE>
```


ACKR file sections

- **FILE** Name includes the source file and the destination file name
- **RESULT** one of the results, either Net-X-Code or Net-X-Copy above
- **SRCPATH** full source path
- **DSTPATH** full destination path
- **RTINPATH** output rtindex path
- **ACKPATH** output ack path
- **TIME** the time the operation completed
- **DISCONTINUITIES** any time gaps in the captured. Types included
  - "video"
  - "audio"
  - "dolby"
  - "timecode"
  - "metadata"
  - "closedcaptions"
- **INFO** This section includes information about the output file
  - **General**
    - **Format** the main format of the output file (MXF/MOV/MP4/etc)
    - **Duration** the total number of frames in a file
  - **Video**
    - **Format** the video format (MPEG-2/XAVC/etc)
    - **FourCC** the four character code as a hex value
    - **Width** the video width
    - **Height** the video height
    - **BitCount** the video bit depth
    - **Rate** the video rate (rate/scale = fps)
    - **Scale** the video scale (rate/scale = fps)
    - **Length** the length of the video stream in frames
  - **Audio**
    - **Format** the audio format (AAC/AC-3/etc)
    - **Channels** total number of channels containing audio
    - **Frequency** the audio frequency
    - **Bits** the bit size of an audio sample
    - **Length** of the audio stream in audio samples
  - **MetaData**
    - Any available metadata per the Drastic metadata XML spec

Discontinuity Handling in NetXCopy

NetXCopy will skip over short discontinuities and grab the next available frame. This will cause a little hiccup in the output file.

For larger discontinuities the copy will fail and report the timecode of the frame it was trying to read in the comment section of the ack.

Configuration

The configuration files/settings are stored in different places for different operating systems:

Windows:
Registry
\HKEY_CURRENT_USER\Software\Drastic\NetXBase
\HKEY_CURRENT_USER\Software\Drastic\NetXCmd
\HKEY_CURRENT_USER\Software\Drastic\NetXCmd\Groups
\HKEY_CURRENT_USER\Software\Drastic\NetXCopy
\HKEY_CURRENT_USER\Software\Drastic\NetXTimecode

Linux:
~/.config/Drastic/NetXBase.conf
~/.config/Drastic/NetXCmd.conf
~/.config/Drastic/NetXCopy.conf
~/.config/Drastic/NetXTimecode.conf

OS-X:
${HOME}/Library/Preferences/com.drastic.NetXBase.plist
${HOME}/Library/Preferences/com.drastic.NetXCmd.plist
${HOME}/Library/Preferences/com.drastic.NetXCopy.plist
${HOME}/Library/Preferences/com.drastic.NetXTimecode.plist

NetXBase (**NetXCode**)
Geometry – string – position of UI
forcegui – dword – optional, force the gui on (1) or off (0)
commandip – string – ip address of the interface for NetXCmd/Base to communicate on
multicast – string – the multicast address for NetXBase and NetXCmd to use to find each other (default 230.7.7.7)

NetXCmd
Geometry – string – position of UI
forcegui – dword – optional, force the gui on (1) or off (0)
commandip – string – ip address of the interface for NetXCmd/Base to communicate on
videoip – string – ip address of the interface for NetXCode to capture network video streams on
autostart – is true, start capture of any enabled groups as soon as NetXCmd starts up
tod – bool – use time of day
fileCollisionAction – int – what to do if the file already exists
- 1 – fail the capture
- 2 – rename the new file to continue the capture
- 3 – append if possible, otherwise rename

\Groups\
- NetXCode### ← each group of channels
  - autostart – is true, start capture of this group as soon as NetXCmd starts up
  - keyname – the name that can be used to identify the group for commands
  - name – the friendly name of the group
  - version – the version of the software that created this group
  - Channel### - each channel
    - address – string – ip address of rtp/udp sender
    - directory – string – where to record the files
    - enabled – whether this channel is enabled or not
    - mp4proxydatarate – target data rate when converting in line to mp4
    - mp4proxyfcc – the mp4 codec to use for the proxy
    - mp4proxymode – if 1 then use the proxy/down convert
    - mp4proxyscaleddown – the factor to scale the proxy by (2=1/2, 3=1/3, 4=1/4)
    - name – string – base name of the file to record to
    - port – string - RTP/UDP port
    - protocol – string – RTP or UDP
    - type – string – type of file to record: ts, mxf, mxf-as02, mp4, mov

NetXCopy
geometry – string – position of UI
forcegui – dword – optional, force the gui on (1) or off (0)
startnetx – dword – start netx

NetXTimecode
address – group address of the timecode stream
port – port for the timecode stream
protocol – currently UDP or RTP

fixedpreview – for testing
name – for testing – file source
type – for testing

System Setup

Net-X-Code makes use of a number of TCP and UDP ports for discovery, connection and capture. All of these connections must be allowed to pass through any firewall or other network protection for Net-X-Code to work. The main ports NetXCode uses include:

80/443 – TCP - Apache server for standard HTML and NetXPlayer
20/21 – TCP - Optional vsFtp for file access

7630 – TCP – NetXCmd/NetXCode server command port
57500-57XXX – UDP = Communications server port
58500 – UDP – NetXBase multicast server port
58500 – TCP – NetXCmd→NetXBase communication port (outgoing)
59000->50### – TCP – NetXBase<-NetXCmd communcations port, where ### is the max number of channels (incoming)

By default, NetXBase uses this multicast address to join all of its components. Please note, this can be changed in the configuration to allow multiple NetXCode groups to exist on the same network.

230.7.7.7 – NetXCode system discovery multicast address

The individual streams being captured or transmitted also use multi or unicast addresses, along with UDP or TCP ports. These are user configured, but by default, RTP and UDP traffic often use port 5004 (default for RTP) or port 1234 (the experimental port).

The basic connection process for the whole system is:

- NetXBase hosts multicast at 230.7.7.7:58500 UDP for NetXCmds to connect to
- NetXCmd joins 230.7.7.7:(57500 +offset) and sends a message to NetXBase
- NetXBase responds back through multicast
- NetXCmd receives the message and gets the NetXBase IP from the messages
- NetXCmd make a TCP connect from NetXCmd <ip>:58500 to NetXBase <baseip>:59000-50###
- NetXCode is spawned, it connects back to the local NetXCmd on port 7630

Linux (Centos/RedHat) - Network

If you are in a protected network, the simplest to get everything working is to disable the iptables firewall (Note: you'll need to be root for all of this)

/etc/init.d/iptables stop

This is also a good way to test that NetXCode is working properly on your system. For most system, iptables will be required, so individual exceptions should be added for each port/type on each adapter. Iptables can then be turned on and off to check that all the correct ports are being allowed through. For the multicast parts rp_filter setting must be changed to be more permissive:

Open /etc/sysctl.conf
Change the line:
net.ipv4.conf.default.rp_filter = 1
to:
net.ipv4.conf.default.rp_filter = 2

This setting may be set for individual ethernet devices so make sure to change net.ipv4.conf.eth0.rp_filter as well.

Windows - Network

If you run each of the components from the desktop, the Windows OS/firewall will ask if you want to add an exception for each of the applications (netxbase, netxcmd, netxcode, netxcopy) as they start up and connect to the network. Once you have allowed these exceptions, they can be run remotely/headless, and will work properly.

OS-X - Network

To disable to firewall for NetXCode, for each application:

1. Open System Preferences > Security & Privacy > Firewall > Firewall Options.
2. Click Add.
3. Choose an application from the Applications folder and click Add.
4. Ensure that the option next to the application is set to Allow incoming connections.
5. Click OK.

Linux – Required Packages

SDL

yum install SDL

libstatgrab

yum install libstatgrab-devel

libuuid

yum install e2fsprogs-devel

libgstreamer

yum install gstreamer gst-plugins-base

wxGTK – only for legacy apps

yum install wxGTK

Linux – SysLog Output

To set up syslog output, add the following lines to /etc/rsyslog.conf:

```
#route all dt messages to custom log
:app-name, isequal, "dtlog"                    /var/log/dtlog.log
```

and then create that file

```
sudo touch /var/log/dtlog.log
```