Net-X-Code version 7

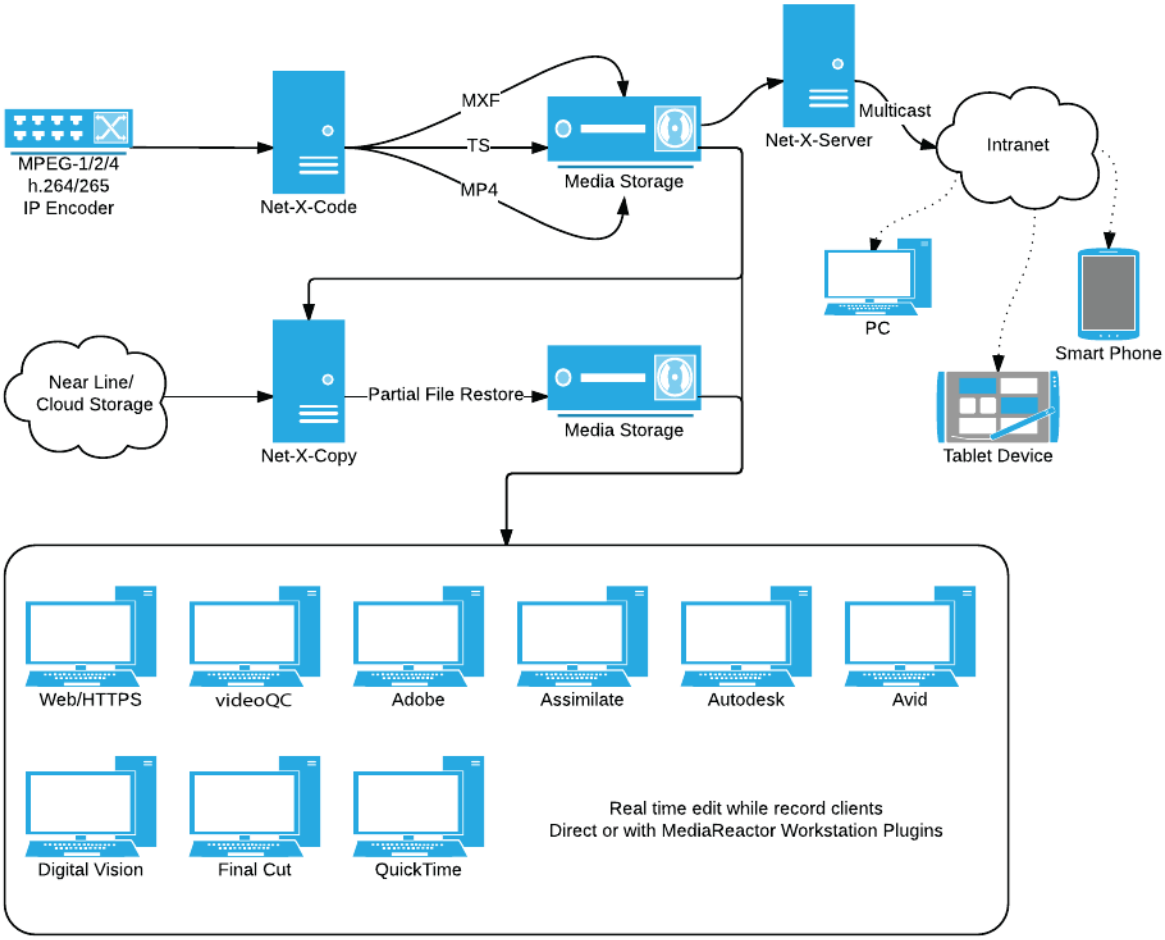
API Guide

March 10, 2019

Table of Contents

Introduction

Net-X-Code is a series of interconnected applications (Net-X-Code, Net-X-Cmd, Net-X-Copy, Net-X-Streamer, videoQC and MediaReactor Workstation plugins) for IP based video capture, streaming, and conversion. 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 sidecar 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 SMPTE 2110, SMPTE 2022=6, 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

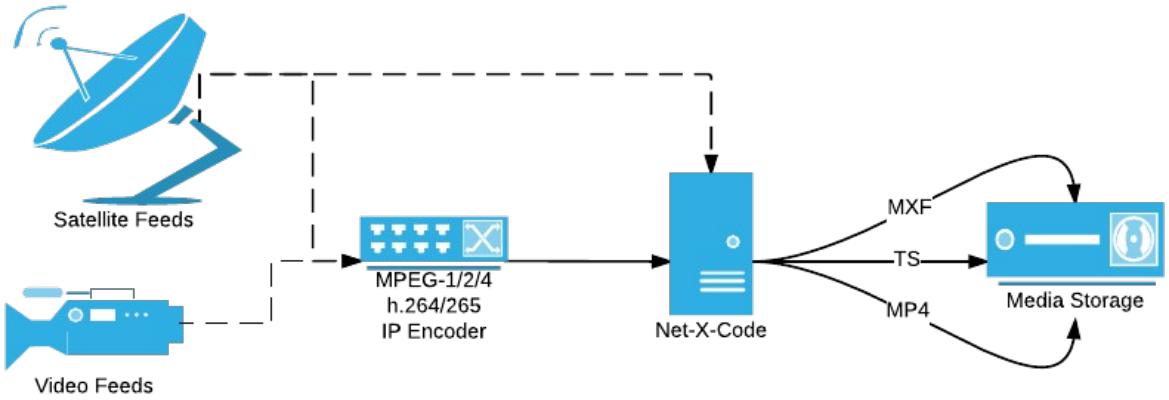
Net-X-Player: An HTML5 based web player that can play time code based, frame accurate files, including RTIN real time files during record. Net-X-Player can also send clipping commands directly to Net-X-Base.

videoQC: Video preview is available from on disk, live recording and network video sources. videoQC also provides video waveform, vector scope, histogram and metadata displays, along with clipping and conversions.

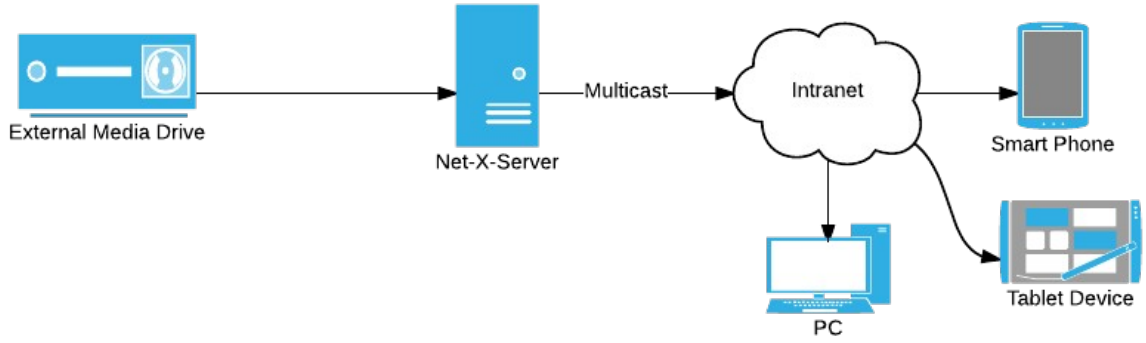
MediaReactor Workstation: This series of plugins allow professional editing and finishing system to access all the file types created by the Net-X-Code system and access the real time recording files, while they are still growing. MRWS is built into software like Assimilate and NuCoda, and available as an option for other systems like Adobe, Avid, Autodesk, Quantel, Final Cut, and even QuickTime compatible apps.

Typical Applications

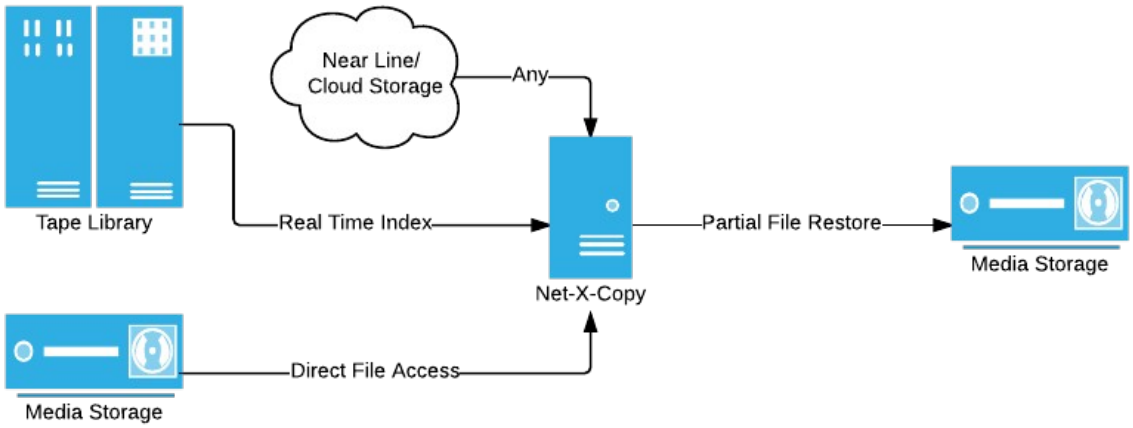
Net-X-Code IP Video Capture



Net-X-Code IP Video Streaming



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



Changes In Version 7 API:

- All XML returns use “UTF-8”. Previous APIs used “ISO 8859-1”
- All returns include a <result> key which has a 0 for success or any other number for failure
- All returns include a <resultstring> with a short result messages
- All multi element returns (like lists of <clip> returns) include an “index=#” attribute
- ‘message’ and other string return attributes are now in <resultstring>
- JSON is now officially supported (with ‘/netxjson?’) for most commands (see exceptions in this manual)

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, or a JSON buffer if using ‘netxjson’:

<?xml version="1.0" encoding="ISO88591"?>
<client count="1">
<result>0</result>
<resultstring>returning client count</resultstring>
</client>
	{
	"count": "1",
	"result": "0",
	"resultstring": "returning client count"
	}

NOTE: for the rest of the returns in this manual, we will not include the ‘<?xml version="1.0" encoding="ISO88591"?>’ start, but it will always be returned if returning XML.

Command Elements

The commands are sent via HTTP/REST as an 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 than 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 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 of 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.

Return Elements

All XML returns from the NetXCode rest API will always start with <?xml version="1.0" encoding="ISO88591"?> followed by at least one <key>. That first <key> will always include two keys within it:

<result>0/result> - which will contain a 1 (true) for success, and any other number for failure. The most common will be 0 (false) for a failure. If there are extended error numbers available (like errno or other values) then a failure may in a negative or positive number.

<resultstring>command=success</resultstring> - a short message describing the results. It will usually be fairly general, but may include specifics, especially if the result is not 0 or 1.

Common return keys include:

- active, testfunc, request,
- netxbase, copy, client, clients, errors, count,
- group, getdiscontinuities, getcopyinout, copyinoutinfo,
- getfirstandlasttimecode

<something>
<result>0/result>
<resultstring>returning client count</resultstring>
</something>

Command – general

Active

http://127.0.0.1:1080/netxjson?active=1

<?xml version="1.0" encoding="ISO88591"?>
<active>
  <result>0</result>
  <resultstring>active=true</resultstring>
</active>

```
{
  "result": "0",
  "resultstring": "active=true"
}
```

Is the REST API available. May not return is the service is not running, otherwise it should return a result of 1 and a resultstring of active=true.

TestFunc

http://127.0.0.1:1080/netx?testfunc=createpath&pathtocreate

<?xml version="1.0" encoding="ISO88591"?>
<testfunc>
  <result>0</result>
  <resultstring>unknown test function</resultstring>
</testfunc>

```
{
  "result": "0",
  "resultstring": "unknown test function"
}
```

TestFunc contains test or utility functions. Createpath can be used to make a multi level path on the API server. This, and other TestFunc are reserved for internal use. Do not use, as it can cause the API to crash.

netxbase=version

http://127.0.0.1:1080/netxjson?netxbase=version

<?xml version="1.0" encoding="ISO88591"?>
<netxbase version="v7.0.0.1" address="192.168.50.100">
  <result>0</result>
  <resultstring>nexbase version returned</resultstring>
</netxbase>

```
{
  "address": "192.168.50.100",
  "result": "0",
  "resultstring": "nexbase version returned",
  "version": "v7.1.0.3"
}
```

This request returns the version and the IP address of the NetXBase REST API.

Return – bad/malformed request

If the system is shutting down, then all commands will return the following

```
<request netxbaseip="192.168.50.100" type="" message="failure">
  <result>0</result>
  <resultstring>failure</resultstring>
</request>
```

Return – shutdown

If the system is shutting down, then all commands will return the following

```
<request message="shut down in progress">
  <result>0</result>
  <resultstring>ignored – shut down in progress</resultstring>
</request>
```

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

<table>
<tr>
<td>

```
<client count="1">
  <result>0</result>
  <resultstring>returning client
count</resultstring>
</client>
```

</td>
<td>

```
{
  "count": "1",
  "result": "0",
  "resultstring": "returning client count"
}
```

</td>
</tr>
</table>

Client IP

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

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

<client netxbaseip="192.168.50.100"
address="192.168.100.229" version="v7.1.0.1">
  <result>0</result>
  <resultstring>returning client</resultstring>
</client>

{
  "address": "192.168.100.229",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "returning client",
  "version": "v7.1.0.3"
}

Number Of Groups

Request the number of groups client=ip&group=count. The client can be its numerical value (0..n), or its IP address.

http://127.0.0.1:1080/netx?request=get&client=0&group=count

<client netxbaseip="192.168.50.100" address="1
92.168.100.229" version="v7.1.0.1" groups="3" a
utostart="0">
  <group count="3"/>
  <result>0</result>
  <resultstring>group returned</resultstring>
</client>

{
  "address": "192.168.100.229",
  "autostart": "0",
  "group": "",
  "groups": "3",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "group returned",
  "version": "v7.1.0.3"
}

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 time code 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 time code
- timecode – the last captured time code
- ub – the current captured userbits
- firsttc – the first time code captured for a stream
- firstub – the first userbits captured for a stream
- tctype - channel's time code 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/netxjson?request=get&client=0&group=0&channel=0&value=name

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.3" groups="3" autostart="0">
  <group index="0" key="newgroupname" version="v7.1.0.3" autostart="1">
    <channel index="0">
      <name>Channel1</name>
    </channel>
  </group>
  <result>0</result>
  <resultstring>group returned</resultstring>
</client>

{
  "address": "192.168.100.229",
  "autostart": "0",
  "group": {
    "autostart": "1",
    "channel": {
      "index": "0",
      "name": "Channel1"
    },
    "index": "0",
    "key": "newgroupname",
    "version": "v5.1.0.13"
  },
  "groups": "3",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "group returned",
  "version": "v7.1.0.3"
}

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

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.3"

{
  "address": "192.168.100.229",

```
groups="3" autostart="0">
  <group index="0" key="DemoCapture"
version="v5.1.0.13" autostart="1">
    <channel index="0">
      <timecode
string="00:00:00;00">0</timecode>
    </channel>
  </group>
  <result>0</result>
  <resultstring>group returned</resultstring>
</client>
```

```
  "autostart": "0",
  "group0": {
    "autostart": "1",
    "channel0": {
      "index": "0",
      "timecode": {
        "string": "00:00:00;00",
        "value": "0"
      }
    },
    "index": "0",
    "key": "DemoCapture",
    "version": "v5.1.0.13"
  },
  "groups": "3",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "group returned",
  "version": "v7.1.0.3"
}
```

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

```
<client netxbaseip="192.168.50.100"
address="192.168.100.229" version="v7.1.0.3"
groups="3" autostart="0">
  <group index="0" key="DemoCapture"
version="v5.1.0.13" autostart="1">
    <channel index="0">
      <name>HBR-Live</name>
      <capturefile/>
      <enabled>1</enabled>
      <address>239.254.30.30</address>
      <uuid/>
      <port>5004</port>
      <protocol>RTP</protocol>
      <rectc>1</rectc>
      <directory>D:\drastic.trunk\applications\web\
netx\video\live</directory>
      <type>.mxf-op1a</type>
      <capturecc>-1</capturecc>
```

```
{
  "address": "192.168.100.229",
  "autostart": "0",
  "group0": {
    "autostart": "1",
    "channel0": {
      "address": "239.254.30.30",
      "bcdtc": "",
      "bcdub": "",
      "bytes": "0",
      "capturecc": "-1",
      "capturefile": "",
      "directory": "D:\\drastic.trunk\\applications\\
web\\netx\\video\\live",
      "enabled": "1",
      "endtc": "-1",
      "firsttc": {
        "string": "00:00:00;00",
```

```
<streamaddress>239.254.30.30</streamaddress>
      <streamport>5004</streamport>
      <streamfile>na</streamfile>
      <streamenabled>1</streamenabled>
      <streamduration>0</streamduration>
      <previewenabled>0</previewenabled>
      <mp4proxymode>0</mp4proxymode>
      <mp4proxyfcc>1635148593</mp4proxyfcc>
<mp4proxydatarate>2000000</mp4proxydatarate>
<mp4proxyscaledown>2</mp4proxyscaledown>
      <ms>0</ms>
      <bytes>0</bytes>
      <state desc="Not Running">8</state>
      <metadata/>
      <tc string="00:00:00;00">0</tc>
      <metatcpid>4294967294</metatcpid>
      <bcdtc/>
      <timecode string="00:00:00;00">0</timecode>
      <firsttc string="00:00:00;00">0</firsttc>
      <frames string="00:00:00;00">0</frames>
      <firstub>0</firstub>
      <tctype string="29.97">4</tctype>
      <bcdub/>
      <userbits>0</userbits>
      <starttc>-1</starttc>
      <endtc>-1</endtc>
      <vitctimecode>3452816845</vitctimecode>
      <vitcub>3452816845</vitcub>
    </channel>
  </group>
  <result>0</result>
  <resultstring>group returned</resultstring>
</client>
```

```
    "value": "0"
  },
  "firstub": "0",
  "frames": {
    "string": "00:00:00;00",
    "value": "0"
  },
  "index": "0",
  "metadata": "",
  "metatcpid": "4294967294",
  "mp4proxydatarate": "2000000",
  "mp4proxyfcc": "1635148593",
  "mp4proxymode": "0",
  "mp4proxyscaledown": "2",
  "ms": "0",
  "name": "HBR-Live",
  "port": "5004",
  "previewenabled": "0",
  "protocol": "RTP",
  "rectc": "1",
  "starttc": "-1",
  "state": {
    "desc": "Not Running",
    "value": "8"
  },
  "streamaddress": "239.254.30.30",
  "streamduration": "0",
  "streamenabled": "1",
  "streamfile": "na",
  "streamport": "5004",
  "tc": {
    "string": "00:00:00;00",
    "value": "0"
  },
  "tctype": {
    "string": "29.97",
    "value": "4"
  },
  "timecode": {
    "string": "00:00:00;00",
    "value": "0"
  },
```

	"type": ".mxf-op1a", "userbits": "0", "uuid": "", "vitctimecode": "3452816845", "vitcub": "3452816845" }, "index": "0", "key": "DemoCapture", "version": "v5.1.0.13" }, "groups": "3", "netxbaseip": "192.168.50.100", "result": "0", "resultstring": "group returned", "version": "v7.1.0.3" }

Or all settings for all groups and all channels

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

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.1" groups="3" autostart="0">
<group index="0" key="DemoCapture" version="0" autostart="1">
<channel index="0">
<name>HBR-Live</name>
<capturefile/>
<enabled>1</enabled>
<address>239.254.30.30</address>
<uuid/>
<port>5004</port>
<protocol>RTP</protocol>
<rectc>1</rectc>
<directory>D:\drastic.trunk\applications\web\netx\video\live</directory>
<type>.mxf-op1a</type>
<capturecc>-1</capturecc>
<streamaddress>na</streamaddress>
<streamport>0</streamport>
<streamfile>na</streamfile>
<streamenabled>0</streamenabled>
<streamduration>0</streamduration>
<previewenabled>1</previewenabled>
	{
	"address": "192.168.100.229",
	"autostart": "0",
	"group0": {
	"autostart": "1",
	"channel0": {
	"address": "239.254.30.30",
	"bcdtc": "",
	"bcdub": "",
	"bytes": "0",
	"capturecc": "-1",
	"capturefile": "",
	"directory": "D:\\drastic.trunk\\applications\\web\\netx\\video\\live",
	"enabled": "1",
	"endtc": "-1",
	"firsttc": {
	"string": "00:00:00;00",
	"value": "0"
	},

```
    <mp4proxymode>0</mp4proxymode>
    <mp4proxyfcc>1635148593</mp4proxyfcc>

<mp4proxydatarate>2000000</mp4proxydatarate
>

<mp4proxyscaledown>2</mp4proxyscaledown>
    <ms>0</ms>
    <bytes>0</bytes>
    <state desc="Not Running">8</state>
    <metadata/>
    <tc string="00:00:00;00">0</tc>
    <metatcpid>4294967294</metatcpid>
    <bcdtc/>
    <timecode
string="00:00:00;00">0</timecode>
    <firsttc string="00:00:00;00">0</firsttc>
    <frames string="00:00:00;00">0</frames>
    <firstub>0</firstub>
    <tctype string="29.97">4</tctype>
    <bcdub/>
    <userbits>0</userbits>
    <starttc>-1</starttc>
    <endtc>-1</endtc>
    <vitctimecode>3452816845</vitctimecode>
    <vitcub>3452816845</vitcub>
  </channel>
  <channel index="1">
                              . . .
  </channel>
 </group>
 <group index="1" key="DemoStream"
version="v5.1.0.13" autostart="1">
  <channel index="0">
    <name>Channel0</name>
    <capturefile/>
    <enabled>0</enabled>
    <address>239.0.0.0</address>
    <uuid/>
    <port>5004</port>
    <protocol>RTP</protocol>
    <rectc>0</rectc>
    <directory>
      D:\drastic.trunk\applications\web\netx\video\
streamTS
    </directory>
    <type>.ts</type>
    <capturecc>-1</capturecc>

<streamaddress>239.254.30.30</streamaddress
>
```

```
    "firstub": "0",
    "frames": {
      "string": "00:00:00;00",
      "value": "0"
    },
    "index": "0",
    "metadata": "",
    "metatcpid": "4294967294",
    "mp4proxydatarate": "2000000",
    "mp4proxyfcc": "1635148593",
    "mp4proxymode": "0",
    "mp4proxyscaledown": "2",
    "ms": "0",
    "name": "HBR-Live",
    "port": "5004",
    "previewenabled": "0",
    "protocol": "RTP",
    "rectc": "1",
    "starttc": "-1",
    "state": {
      "desc": "Not Running",
      "value": "8"
    },
    "streamaddress": "239.254.30.30",
    "streamduration": "0",
    "streamenabled": "1",
    "streamfile": "na",
    "streamport": "5004",
    "tc": {
      "string": "00:00:00;00",
      "value": "0"
    },
    "tctype": {
      "string": "29.97",
      "value": "4"
    },
    "timecode": {
      "string": "00:00:00;00",
      "value": "0"
    },
    "type": ".mxf-op1a",
    "userbits": "0",
```

```
    <streamport>5004</streamport>
    <streamfile>na</streamfile>
    <streamenabled>1</streamenabled>
    <streamduration>0</streamduration>
    <previewenabled>0</previewenabled>
    <mp4proxymode>0</mp4proxymode>
    <mp4proxyfcc>1635148593</mp4proxyfcc>

<mp4proxydatarate>2000000</mp4proxydatarate
>

<mp4proxyscaledown>2</mp4proxyscaledown>
    <ms>0</ms>
    <bytes>0</bytes>
    <state desc="Not Running">8</state>
    <metadata/>
    <tc string="00:00:00;00">0</tc>
    <metatcpid>4294967294</metatcpid>
    <bcdtc/>
    <timecode
string="00:00:00;00">0</timecode>
    <firsttc string="00:00:00;00">0</firsttc>
    <frames string="00:00:00;00">0</frames>
    <firstub>0</firstub>
    <tctype string="29.97">4</tctype>
    <bcdub/>
    <userbits>0</userbits>
    <starttc>-1</starttc>
    <endtc>-1</endtc>
    <vitctimecode>3452816845</vitctimecode>
    <vitcub>3452816845</vitcub>
  </channel>
                              . . .
 </group>
 <group index="2" key="DemoRestream"
version="0" autostart="1">
                              . . .
 </group>
 <result>0</result>
 <resultstring>group returned</resultstring>
</client>
```

```
    "uuid": "",
    "vitctimecode": "3452816845",
    "vitcub": "3452816845"
  },
  "channel1": {
                    . . .
  },
  "channel2": {
                    . . .
  },
  "channel3": {
                    . . .
  },
  "channel4": {
                    . . .
  },
  "channel5": {
                    . . .
  },
  "channel6": {
                    . . .
  },
  "channel7": {
                    . . .
  },
  "channel8": {
                    . . .
  },
  "channel9": {
                    . . .
  },
  "index": "0",
  "key": "DemoCapture",
  "version": "v5.1.0.13"
},
"group1": {
  "autostart": "1",
                    . . .
  "index": "1",
  "key": "DemoStream",
  "version": "v5.1.0.13"
},
```

	"group2": {
	. . .
	},
	"groups": "3",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "group returned",
	"version": "v7.1.0.3"
	}

Or a single setting from a single group and all channels

http://127.0.0.1:1080/netxjson?request=get&client=0&group=1&channel=all&value=address

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.1" groups="3" autostart="0">
<group index="1" key="DemoStream" version="v5.1.0.13" autostart="1">
<channel index="0">
<address>239.0.0.0</address>
</channel>
<channel index="1">
<address>239.0.0.0</address>
</channel>
<channel index="2">
<address>239.0.0.0</address>
</channel>
<channel index="3">
<address>239.0.0.0</address>
</channel>
<channel index="4">
<address>239.0.0.0</address>
</channel>
<channel index="5">
<address>239.0.0.0</address>
</channel>
<channel index="6">
<address>239.0.0.0</address>
</channel>
<channel index="7">
<address>239.0.0.0</address>
</channel>
<channel index="8">
<address>239.0.0.0</address>
</channel>
<channel index="9">
<address>239.0.0.0</address>
	{
	"address": "192.168.100.229",
	"autostart": "0",
	"group1": {
	"autostart": "1",
	"channel0": {
	"address": "239.0.0.0",
	"index": "0"
	},
	"channel1": {
	"address": "239.0.0.0",
	"index": "1"
	},
	"channel2": {
	"address": "239.0.0.0",
	"index": "2"
	},
	"channel3": {
	"address": "239.0.0.0",
	"index": "3"
	},
	"channel4": {
	"address": "239.0.0.0",
	"index": "4"
	},
	"channel5": {
	"address": "239.0.0.0",
	"index": "5"

<table>
<tr>
<td>

```
  </channel>
 </group>
 <result>0</result>
 <resultstring>group returned</resultstring>
</client>
```

</td>
<td>

```
  },
  "channel6": {
    "address": "239.0.0.0",
    "index": "6"
  },
  "channel7": {
    "address": "239.0.0.0",
    "index": "7"
  },
  "channel8": {
    "address": "239.0.0.0",
    "index": "8"
  },
  "channel9": {
    "address": "239.0.0.0",
    "index": "9"
  },
  "index": "1",
  "key": "DemoStream",
  "version": "v5.1.0.13"
 },
 "groups": "3",
 "netxbaseip": "192.168.50.100",
 "result": "0",
 "resultstring": "group returned",
 "version": "v7.1.0.3"
}
```

</td>
</tr>
</table>

previewenable

Determine if preview JPEG capture is currently enabled

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

<table>
<tr>
<td>

```
<client netxbaseip="192.168.50.100"
address="192.168.100.229" version="v7.1.0.1"
groups="3" autostart="0">
 <group index="0" key="DemoCapture"
version="0" autostart="1">
  <channel index="0">
   <unknown_request/>
  </channel>
```

</td>
<td>

```
{
 "address": "192.168.100.229",
 "autostart": "0",
 "group0": {
  "autostart": "1",
  "channel0": {
   "index": "0",
```

</td>
</tr>
</table>

</group> <result>0</result> <resultstring>group returned</resultstring> </client>	"unknown_request": "" }, "index": "0", "key": "DemoCapture", "version": "v5.1.0.13" }, "groups": "3", "netxbaseip": "192.168.50.100", "result": "0", "resultstring": "group returned", "version": "v7.1.0.3" }

preview

Getting a preview JPEG embedded in the XML response

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

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.1" groups="3" autostart="0"> <group index="0" key="DemoCapture" version="0" autostart="1"> <channel index="0"> <preview timestamp="95479532"> /9j/4AAQSkZJRgABAQEAYABgAAD//gAW9k= . . . </preview> </channel> </group> <result>0</result> <resultstring>group returned</resultstring> </client>	

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

pts

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

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

<client netxbaseip="192.168.50.100"	{

```
address="192.168.100.229" version="v7.1.0.1"
groups="3" autostart="0">
  <group index="0" key="DemoCapture"
version="0" autostart="1">
    <channel index="0">
      <video_pts>23470552</video_pts>
      <audio_pts>23385674</audio_pts>
      <tc_pts>0</tc_pts>
      <days>0</days>
      <months>23475929</months>
      <years>24</years>
    </channel>
  </group>
  <result>0</result>
  <resultstring>group returned</resultstring>
</client>
```

```
  "address": "192.168.100.229",
  "autostart": "0",
  "group0": {
    "autostart": "1",
    "channel0": {
      "audio_pts": "0",
      "days": "0",
      "index": "0",
      "months": "6400221249",
      "tc_pts": "0",
      "video_pts": "18174762",
      "years": "5"
    },
    "index": "0",
    "key": "DemoCapture",
    "version": "v5.1.0.13"
  },
  "groups": "3",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "group returned",
  "version": "v7.1.0.3"
}
```


datarate

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

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


```
<client netxbaseip="192.168.50.100"
address="192.168.100.229" version="v7.1.0.1"
groups="3" autostart="0">
  <group index="0" key="DemoCapture"
version="0" autostart="1">
    <channel index="0">
      <datarate_total timestamp="3452816845"/>
      <datarate_video timestamp="3452816845"/>
      <datarate_audio timestamp="3452816845"/>
    </channel>
  </group>
  <result>0</result>
```

```
{
  "address": "192.168.100.229",
  "autostart": "0",
  "group0": {
    "autostart": "1",
    "channel0": {
      "datarate_audio": "",
      "datarate_total": "",
      "datarate_video": "",
      "index": "0"
```

<resultstring>group returned</resultstring> </client>	}, "index": "0", "key": "DemoCapture", "version": "v5.1.0.13" }, "groups": "3", "netxbaseip": "192.168.50.100", "result": "0", "resultstring": "group returned", "version": "v7.1.0.3" }

The value returned will contain 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=0&group=0&channel=0&value=datarate&total=

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.1" groups="3" autostart="0">
<group index="0" key="DemoCapture" version="0" autostart="1">
<channel index="0">
<datarate_total timestamp="3452816845"/>
</channel>
</group>
<result>0</result>
<resultstring>group returned</resultstring>
</client>
	{
	"address": "192.168.100.229",
	"autostart": "0",
	"group0": {
	"autostart": "1",
	"channel0": {
	"datarate_total": "",
	"index": "0"
	},
	"index": "0",
	"key": "DemoCapture",
	"version": "v5.1.0.13"
	},
	"groups": "3",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "group returned",
	"version": "v7.1.0.3"
	}

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

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.1" groups="3" autostart="0">	{ "address": "192.168.100.229", "autostart": "0",

<group index="0" key="DemoCapture" version="0" autostart="1">
<channel index="0">
<datarate_video timestamp="3452816845"/>
</channel>
</group>
<result>0</result>
<resultstring>group returned</resultstring>
</client>
	"group0": {
	"autostart": "1",
	"channel0": {
	"datarate_video": "",
	"index": "0"
	},
	"index": "0",
	"key": "DemoCapture",
	"version": "v5.1.0.13"
	},
	"groups": "3",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "group returned",
	"version": "v7.1.0.3"
	}

property

Getting a stream property

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

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.1" groups="3" autostart="0">
<group index="0" key="DemoCapture" version="0" autostart="1">
<channel index="0">
<video_width changed="0">1280</video_width>
</channel>
</group>
<result>0</result>
<resultstring>group returned</resultstring>
</client>
	{
	"address": "192.168.100.229",
	"autostart": "0",
	"group0": {
	"autostart": "1",
	"channel0": {
	"index": "0",
	"video_width": {
	"changed": "0",
	"value": "1280"
	}
	},
	"index": "0",
	"key": "DemoCapture",
	"version": "v5.1.0.13"
	},
	"groups": "3",
	"netxbaseip": "192.168.50.100",

	"result": "0", "resultstring": "group returned", "version": "v7.1.0.3" }

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=0&group=0&channel=0&value=property&all=

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.1" groups="3" autostart="0">
<group index="0" key="DemoCapture" version="0" autostart="1">
<channel index="0">
<previewenabled>1</previewenabled>
<video_channel changed="0">1</video_channel>
<video_codec changed="0">0</video_codec>
<video_bitdepth changed="0">24</video_bitdepth>
<video_rate changed="0">59.94</video_rate>
<video_width changed="0">1280</video_width>
<video_height changed="0">720</video_height>
<audio_channel changed="0">2</audio_channel>
<audio_bitdepth changed="0">24</audio_bitdepth>
<audio_rate changed="0">48000</audio_rate>
</channel>
</group>
<result>0</result>
<resultstring>group returned</resultstring>
</client>


```
{
 "address": "192.168.100.229",
 "autostart": "0",
 "group0": {
  "autostart": "1",
  "channel0": {
   "audio_bitdepth": {
    "changed": "0",
    "value": "16"
   },
   "audio_channel": {
    "changed": "0",
    "value": "0"
   },
   "audio_rate": {
    "changed": "0",
    "value": "48000"
   },
   "index": "0",
   "previewenabled": "1",
   "video_bitdepth": {
    "changed": "0",
    "value": "24"
   },
   "video_channel": {
    "changed": "0",
    "value": "1"
   },
   "video_codec": {
    "changed": "1",
    "value": "1635148593"
```

	<pre>{
	  "address": "192.168.100.229",
	  "autostart": "0",
	  "group0": {
	  "autostart": "1",
	  "channel0": {
	  "index": "0",
	  "status": {
	  "count": "292",
	  },
	  "video_height": {
	  "changed": "0",
	  "value": "720"
	  },
	  "video_rate": {
	  "changed": "0",
	  "value": "59.94"
	  },
	  "video_width": {
	  "changed": "0",
	  "value": "1280"
	  }
	  },
	  "index": "0",
	  "key": "DemoCapture",
	  "version": "v5.1.0.13"
	  },
	  "groups": "3",
	  "netxbaseip": "192.168.50.100",
	  "result": "0",
	  "resultstring": "group returned",
	  "version": "v7.1.0.3"
	}</pre>

status

Getting a status list

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

<pre><client netxbaseip="192.168.50.100"
address="192.168.100.229" version="v7.1.0.1"
groups="3" autostart="0">
  <group index="0" key="DemoCapture"
version="0" autostart="1">
  <channel index="0">
  <status count="1">
  <status0 index="0" timestamp="95468907"
desc="meta tc pid changed:old">2</status0>
  </status>
  </channel>
  </group></pre>
	<pre>{
	  "address": "192.168.100.229",
	  "autostart": "0",
	  "group0": {
	  "autostart": "1",
	  "channel0": {
	  "index": "0",
	  "status": {
	  "count": "292",
	  }</pre>

<result>0</result> <resultstring>group returned</resultstring> </client>	} }, "index": "0", "key": "DemoCapture", "version": "v5.1.0.13" }, "groups": "3", "netxbaseip": "192.168.50.100", "result": "0", "resultstring": "group returned", "version": "v7.1.0.3" }

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 removed from the internal list once they have been retrieved. Any non-critical messages will be removed after 5 minutes.

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>
  <result>0</result>
  <resultstring>group returned</resultstring>
</request>
```

streamstate

Get the running state of a capturing stream

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

<client netxbaseip="192.168.50.100" address="192.168.100.229" version="v7.1.0.1" groups="3" autostart="0">
<group index="0" key="DemoCapture" version="0" autostart="1">
<channel index="0">
	{
	"address": "192.168.100.229",
	"autostart": "0",
	"group0": {
	"autostart": "1",

<streamstate>requested</streamstate>
</channel>
</group>
<result>0</result>
<resultstring>success</resultstring>
</client>
	"channel0": {
	"index": "0",
	"streamstate": ""
	},
	"index": "0",
	"key": "DemoCapture",
	"version": "v5.1.0.13"
	},
	"groups": "3",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "group returned",
	"version": "v7.1.0.3"
	}

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 than 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>
  <result>0</result>
  <resultstring>success</resultstring>
</client>
```

The second will then return with the information:

```
<client address="192.168.100.176" version="v5.0.0.39" groups="1" autostart=”1”>
  <result>0</result>
  <resultstring>success</resultstring>
  <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 Net-X-Base status including last command, time of last command and times of the last communication with a Net-X-Cmd.

NOTE: Possible JSON issue – please check

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


```
<netxbase>
  <result>0</result>
  <resultstring>output state
returned</resultstring>
  <request-ms>96225028</request-ms>
  <client-info>
    <state>0</state>
    <ms>96224882</ms>
    <data>
group=0&channel=0&stats=&metatcpid=501&uui
d=55E869401C2FE911947D15250A4E0C00&tc=
4021154&timecode=4021159&tctype=32&bcdub=
40608010&userbits=276848704&lastcapturems=
299122&lastcapturebytes=2964720304&vitctimec
ode=2942239&vitcub=276848704&firsttc=401181
4&firstub=276848704&frames=17905&starttc=-
1&endtc=-1
    </data>
```

```
{
  "client-command": {
    "data":
"address=192.168.100.229&group=0&channel=0
&param=preview&value=1",
    "ms": "165793269",
    "state": "0"
  },
  "client-handler": {
    "ms": "166267733",
    "state": "0"
  },
  "client-info": {
    "data":
"group=0&channel=3&pts=&video_pts=63784083
```

<table>
<tr>
<td>
<pre>
  <group-info>
    <state>0</state>
    <ms>96224882</ms>
    <channel-info>
      <state>0</state>
      <ms>96224882</ms>
    </channel-info>
  </group-info>
</client-info>
<client-command>
  <state>0</state>
  <ms>0</ms>
  <data/>
</client-command>
<start-command>
  <state>0</state>
  <ms>0</ms>
</start-command>
<stop-command>
  <state>0</state>
  <ms>0</ms>
</stop-command>
<client-read>
  <state>0</state>
  <ms>96224881</ms>
</client-read>
<client-handler>
  <state>0</state>
  <ms>96224881</ms>
</client-handler>
<client-command>
  <state>0</state>
  <ms>96100291</ms>
</client-command>
</netxbase>
</pre>
</td>
<td>
<pre>
66&audio_pts_0=6378434904&audio_pts_1=0&audio_pts_2=0&audio_pts_3=0&audio_pts_4=0&audio_pts_5=0&audio_pts_6=0&audio_pts_7=0&tc_pts=6378468426&days=5&months=4",
  "group-info": {
    "channel-info": {
      "ms": "166267736",
      "state": "0"
    },
    "ms": "166267736",
    "state": "0"
  },
  "ms": "166267736",
  "state": "0"
 },
 "client-read": {
  "ms": "166267733",
  "state": "0"
 },
 "request-ms": "166267973",
 "result": "0",
 "resultstring": "output state returned",
 "start-command": {
  "ms": "0",
  "state": "0"
 },
 "stop-command": {
  "ms": "0",
  "state": "0"
 }
}
</pre>
</td>
</tr>
</table>

xmldisable

Disable XML sidecar generation for captured files

This command gets the state of XML sidecar generation

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

<table>
<tr>
<td>
<pre>
<client netxbaseip="192.168.50.100"
</pre>
</td>
<td>
<pre>
{
</pre>
</td>
</tr>
</table>

address="192.168.100.229" version="v7.1.0.1" groups="3" autostart="0">
<group index="0" key="DemoCapture" version="0" autostart="1">
<channel index="0">
<xmldisable>0</xmldisable>
</channel>
</group>
<result>0</result>
<resultstring>group returned</resultstring>
</client>
	"address": "192.168.100.229",
	"autostart": "0",
	"group0": {
	"autostart": "1",
	"channel0": {
	"index": "0",
	"xmldisable": "0"
	},
	"index": "0",
	"key": "DemoCapture",
	"version": "v5.1.0.13"
	},
	"groups": "3",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "group returned",
	"version": "v7.1.0.3"
	}

If set to 0, no sidecar XML files will be generated. If set to one, standard Drastic XML sidecar 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 Net-X-Cmd or Net-X-Code 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 (Net-X-Cmd) server. If this is disabled, the next level is ignored.
2. <group> if the client autostart is enabled, then the Net-X-Cmd will look at the group autostarts to determine if a group should be restarted when Net-X-Cmd or Net-X-Code gets closed. This allows the caller to set up 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://127.0.0.1:1080/netx?request=set&client=0&command=getDiscontinuities&source=M:\t3media\Streams\espn\17-06-13_Discontinuities\Channel0.mp4

<request netxbaseip="192.168.50.100" type="command=getDiscontinuities">
  <getDiscontinuities success="1">success</getDiscontinuities>
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "getDiscontinuities": {
    "success": "1",
    "value": "success"
  },
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=getDiscontinuities"
}
```


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

http://127.0.0.1:1080/netx?request=get&client=0&command=getDiscontinuities&source=M:\t3media\Streams\espn\17-06-13_Discontinuities\Channel0.mp4

<getdiscontinuities>
  <result>0</result>
  <resultstring>discontinuities returned</resultstring>
  <DISCONTINUITIES>
    <Discontinuity type="video" duration="14231ms" time="11:52:02 Wednesday, June 14, 2017" frame="1210" LastITimeCode="4094240" DisconTimeCode="4094250" timecodetype="d">18:58:25;50</Discontinuity>
    <Discontinuity type="audio" duration="14308ms" time="11:52:02 Wednesday, June 14, 2017" sample="968967" LastITimeCode="4094240" DisconTimeCode="4094250" timecodetype="d">18:58:25;50</Discontinuity>
    <Discontinuity type="timecode" time="11:52:02 Wednesday, June 14, 2017" frame="1212" LastITimeCode="4094240" DisconTimeCode="4094252" timecodetype="d" comment="CurrentTimeCode:

NOTE: Currently JSON returns are not supported for this command

4095050">18:58:25;52</Discontinuity> <Discontinuity type="video" duration="6641ms" time="11:52:25 Wednesday, June 14, 2017" frame="2241" LastITimeCode="4096070" DisconTimeCode="4096078" timecodetype="d">18:58:56;18</Discontinuity> </DISCONTINUITIES> </getdiscontinuities>	

getCopyInOut

Get the byte ranges necessary to clip a file in a given time code range. This can be made via the REST API or the command line.

For the REST API, you must make this request in 2 parts; first send a set request with the filename of the source, an in time code, an out time code, temp folder, and optionally a byte alignment. If not supplied the alignment will be 1 byte.

Request ex:

```
http://127.0.0.1:1080/netx?request=set&client=0&command=getCopyInOut&source=M:\t3media\Streams\NBA\NBA_Reference_Test_Files\amberfin_new_0000750142.rtin&in=22:29:28;14&out=22:29:38;14&alignment=4096
```

<request netxbaseip="192.168.50.100" type="command=getCopyInOut">
<getCopyInOut success="1">success</getCopyInOut>
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"getCopyInOut": {
	"success": "1",
	"value": "success"
	},
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=getCopyInOut"
	}

And you should get a response saying the command was understood, and is processing.

Then send the exact same command, except replace the ‘set’ with a ‘get’ to get the response to the previous request.

Get request response ex:

```
http://127.0.0.1:1080/netx?request=get&client=0&command=getCopyInOut&source=M:\t3media\Streams\NBA\NBA_Reference_Test_Files\amberfin_new_0000750142.rtin&in=22:29:28;14&out=22:29:38;14&alignment=4096
```

<getcopyinout>
  <result>0</result>
  <resultstring>returning copy in out</resultstring>
  <copyinoutinfo>
    <originalFile>
\\192.168.101.200\media\t3media\Streams\NBA\
NBA_Reference_Test_Files\
amberfin_new_0000750142.mxf
    </originalFile>

<tempFile>amberfin_new_0000750142_1478942
72_298274816.mxf</tempFile>
    <in>147894272</in>
    <out>298274816</out>
  </copyinoutinfo>
</getcopyinout>


```
{
  "copyinoutinfo": {
    "in": "147894272",
    "originalFile": "\\\\192.168.101.200\\media\\t3media\\
Streams\\NBA\\NBA_Reference_Test_Files\\
amberfin_new_0000750142.mxf",
    "out": "298274816",
    "tempFile":
"amberfin_new_0000750142_147894272_298274816.
mxf"
  },
  "result": "0",
  "resultstring": "returning copy in out"
}
```


And you will get back an XML with the source filename, temp name to copy the segment to, and the uint64 start and end point to copy from.

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>. See “Command – Copy, Convert and PFR” below for info on copy requests.

Restore using the temp files ex:

http://127.0.0.1:1080/netx?request=set&client=&command=copy&profile=wrap&source=M:\t3media\Streams\
NBA\NBA_Reference_Test_Files\
amberfin_new_0000750142.rtin&in=22:29:28;14&out=22:29:38;14&alignment=4096&target=m:\restorepoint\
partial\amberfin_clip.mxf&tempfolder=D:\Record

<request netxbaseip="192.168.50.100"
type="command=copy" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
}
```


You may get multiple files back in the XML, this means the audio and video are in separate files. This will happen if the source file is a Panasonic P2, AS-02, DCP or other multi part file. In this case, copy

the data from each source file to its temp file in the temp folder, then pass the temp folder to the translate as above.

Return 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>
```

Tape PFR using the command line

The getCopyInOut command for partial file restore from binary object are also available via NetXCopy's command line. To get the in/out positions in the source file, send a getCopyInOut command to NetXCopy

NetXCopy -p getCopyInOut -s M:\t3media\Streams\NBA\NBA_Reference_Test_Files\ amberfin_new_0000750142.rtin -in 22:29:28;14 -out 22:29:38;14 -alignment 4096

This will return the source file(s) and the start and end position for the copy

source=M:\t3media\Streams\NBA\NBA_Reference_Test_Files\ amberfin_new_0000750142.rtin&in=22:29:28;14&out=22:29:38;14&alignment=4096&getCopyInOut=& originalFile=\\192.168.101.200\media\t3media\Streams\NBA\NBA_Reference_Test_Files\ amberfin_new_0000750142.mxf&tempFile=amberfin_new_0000750142_147894272_298274816.mxf &u64in=147894272&u64out=298274816

Once you have copied the area from the 'originalFIle=' from 'u64in=' to 'u64out=' into a temporary directory, you can then call the partial file restore as you normally would, but with a 'tempfolder=' set pointing to where the binary object was restored to:

NetXCopy -p wrap -s M:\t3media\Streams\NBA\NBA_Reference_Test_Files\ amberfin_new_0000750142.rtin -in 22:29:28;14 -out 22:29:38;14 -alignment 4096 -t m:

restorepoint\partial\amberfin_clip.mxf -tempfolder d:\record

getFirstAndLastTimecode

Get the first and last time code locations from a file.

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

http://127.0.0.1:1080/netx?request=set&client=0&command=getFirstAndLastTimecode&source=M:\t3media\Streams\NBA\NBA_Reference_Test_Files\amberfin_0000683858.rtin

<request netxbaseip="192.168.50.100" type="command=getFirstAndLastTimecode">
  <getFirstAndLastTimecode success="1">success</getFirstAndLastTimecode>
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "getFirstAndLastTimecode": {
    "success": "1",
    "value": "success"
  },
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=getFirstAndLastTimecode"
}
```


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

http://127.0.0.1:1080/netx?request=get&client=0&command=getFirstAndLastTimecode&source=M:\t3media\Streams\NBA\NBA_Reference_Test_Files\amberfin_0000683858.rtin

<getfirstandlasttimecode>
  <source>
    M:\t3media\Streams\NBA\NBA_Reference_Test_Files\amberfin_0000683858.rtin
  </source>
  <tcIn>19:45:01;01</tcIn>
  <tcOut>22:05:16;11</tcOut>
  <result>0</result>
  <resultstring>returning first and last timecode</resultstring>
</getfirstandlasttimecode>


```
{
  "result": "0",
  "resultstring": "returning first and last timecode",
  "source": "M:\\t3media\\Streams\\NBA\\NBA_Reference_Test_Files\\amberfin_0000683858.rtin",
  "tcIn": "19:45:01;01",
  "tcOut": "22:05:16;11"
}
```

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. The major types of copy include:

- wrap – re wrap the audio/video essence without recompressing
- index – make an RTIN index of a file for later partial file restore
- getCopyInOut – find a byte range to restore from tape for a PFR
- a series of conversion types (mxf-xdcam, mov-proreshq, etc). These will re compress the created clip to a certain video/audio compression set in a particular container. For a complete list, please see the profile list below
- When converting, there is also a type parameter. This lets you override the original type, in the case of a wrap, or the set type, in the case of a converting profile, to some standard containers. Please note, not all containers support all codecs

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

http://127.0.0.1:1080/netx?request=set&client=0&command=copy&profile=copy&source=D:/record/sourcefile.mp4&target= D:/record/targetfile.mp4

<request netxbaseip="192.168.50.100" type="command=copy" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
}
```

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=0&command=copy&source=d:/Record/file001.mov&target=D:/Record/target001.mov&source=D:/Record/file002.mov&target=D:/Record/target002.mov&profile=copy

<request netxbaseip="192.168.50.100" type="command=copy" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
}
```


To create an 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=0&command=copy&profile=index&source=D:\record\amberfin_0000572516.mxf&target=D:\record\amberfin_0000572516.mxf.rtin

<request netxbaseip="192.168.50.100" type="command=copy" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
}
```


To run the copy commands on a particular server, the client is normally specified as an IP address with 'client=0'. If you are running multiple servers, Net-X-Base 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=copy&profile=wrap&source=/mnt/server/media/source.mxf&target=/home/user/Videos/edit.mxf&tcin=01:01:30:00&tcout=01:02:00:00

<request netxbaseip="192.168.50.100" type="command=copy" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
```

	}

To create a picon (JPEG picture icon) of the frame at time code 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=D:\record\amberfin_0000572516.mxf&pidst=D:\recored\edit_01013000.picon.jpg&piSize=10&piframe=20:45:30:00

<request netxbaseip="192.168.50.100"
type="command=copy" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>
	{
	  "message": "success",
	  "netxbaseip": "192.168.50.100",
	  "result": "0",
	  "resultstring": "success",
	  "type": "command=copy"
	}

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.

set/get – cardinfo (camera card clip lists)

To get the clips from a camera card, send a set to start the process.

http://127.0.0.1:1080/netxjson?request=get&client=any&command=copy&profile=cardinfo&source=M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\PX270MP_LasVegas\

<request netxbaseip="192.168.50.100"
type="command=copy">
  <cardinfo success="1">0</cardinfo>
  <result>0</result>
  <resultstring>success</resultstring>
</request>
	{
	  "cardinfo": {
	  "success": "1",
	  "value": "0"
	  },
	  "netxbaseip": "192.168.50.100",
	  "result": "0",
	  "resultstring": "success",
	  "type": "command=copy"
	}

You will probably receive a few of these ‘waiting’ returns until the cardinfo is ready. At that point, you will get back the actual cardinfo:

http://127.0.0.1:1080/netxjson?request=get&client=any&command=copy&profile=cardinfo&source=M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\PX270MP_LasVegas\

<request netxbaseip="192.168.50.100"
type="command=copy">
  <cardinfo success="1">0</cardinfo>
  <result>0</result>
  <resultstring>success</resultstring>
</request>

<copy>
  <address>127.0.0.1</address>
  <netxbaseip>192.168.50.100</netxbaseip>
  <result>0</result>
  <resultstring>no data from previous metadata call found yet</resultstring>
</copy>


```
{
  "cardinfo": {
    "success": "1",
    "value": "0"
  },
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "no data from previous metadata call found yet",
  "type": "command=copy"
}
```


Once the set is sent, you can request the results by sending the same command with a get.

http://127.0.0.1:1080/netxjson?request=get&client=any&command=copy&profile=cardinfo&source=M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\PX270MP_LasVegas\

<copy>
  <address>192.168.100.229</address>
  <netxbaseip>192.168.50.100</netxbaseip>
  <result>0</result>
  <resultstring>Cardinfo returned</resultstring>
  <cardinfo source="M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\PX270MP_LasVegas\">
    <clip index="0">
      <filenamepath>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\PX270MP_LasVegas\CONTENTS\AVCLIP\009RA3LL.MXF</filenamepath>
      <clipname>009RA3LL</clipname>
    </clip>
    <clip index="1">
      <filenamepath>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\PX270MP_LasVegas\CONTENTS\AVCLIP\010UU0ZI.MXF</filenamepath>
      <clipname>010UU0ZI</clipname>
    </clip>
    <clip index="2">
      <filenamepath>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\PX270MP_LasVegas\CONTENTS\AVCLIP\011IQ7HW.MXF</filenamepath>


```
{
  "address": "192.168.100.229",
  "cardinfo": {
    "clip0": {
      "clipname": "009RA3LL",
      "filenamepath": "M:\\t3media\\Streams\\MXF\\Panasonic\\Ultra-LongGop25\\PX270MP_LasVegas\\CONTENTS\\AVCLIP\\009RA3LL.MXF",
      "index": "0"
    },
    "clip1": {
      "clipname": "010UU0ZI",
      "filenamepath": "M:\\t3media\\Streams\\MXF\\Panasonic\\Ultra-LongGop25\\PX270MP_LasVegas\\CONTENTS\\AVCLIP\\010UU0ZI.MXF",
      "index": "1"
    },
    "clip10": {
      "clipname": "020WO9TY",
      "filenamepath": "M:\\t3media\\Streams\\MXF\\Panasonic\\Ultra-LongGop25\\PX270MP_LasVegas\\CONTENTS\\AVCLIP\\020WO9TY.MXF",
```

```
    <clipname>011IQ7HW</clipname>
   </clip>
   <clip index="3">
    <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\012OQ0MQ.MXF
    </filenamepath>
    <clipname>012OQ0MQ</clipname>
   </clip>
   <clip index="4">
    <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\014MJ0XQ.MXF
    </filenamepath>
    <clipname>014MJ0XQ</clipname>
   </clip>
   <clip index="5">
    <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\015KF5QO.MXF
    </filenamepath>
    <clipname>015KF5QO</clipname>
   </clip>
   <clip index="6">
    <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\016SP1HV.MXF
    </filenamepath>
    <clipname>016SP1HV</clipname>
   </clip>
   <clip index="7">
    <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\017UE4IK.MXF
    </filenamepath>
    <clipname>017UE4IK</clipname>
   </clip>
   <clip index="8">
    <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\018WI7GJ.MXF
    </filenamepath>
    <clipname>018WI7GJ</clipname>
   </clip>
   <clip index="9">
```

```
    "index": "10"
   },
   "clip11": {
    "clipname": "021XE8GX",
    "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
021XE8GX.MXF",
    "index": "11"
   },
   "clip12": {
    "clipname": "022NP0YW",
    "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
022NP0YW.MXF",
    "index": "12"
   },
   "clip2": {
    "clipname": "011IQ7HW",
    "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
011IQ7HW.MXF",
    "index": "2"
   },
   "clip3": {
    "clipname": "012OQ0MQ",
    "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
012OQ0MQ.MXF",
    "index": "3"
   },
   "clip4": {
    "clipname": "014MJ0XQ",
    "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
014MJ0XQ.MXF",
    "index": "4"
   },
```

```
      <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\019GU1RS.MXF
      </filenamepath>
      <clipname>019GU1RS</clipname>
    </clip>
    <clip index="10">
      <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\020WO9TY.MXF
      </filenamepath>
      <clipname>020WO9TY</clipname>
    </clip>
    <clip index="11">
      <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\021XE8GX.MXF
      </filenamepath>
      <clipname>021XE8GX</clipname>
    </clip>
    <clip index="12">
      <filenamepath>
M:\t3media\Streams\MXF\Panasonic\Ultra-
LongGop25\PX270MP_LasVegas\CONTENTS\
AVCLIP\022NP0YW.MXF
      </filenamepath>
      <clipname>022NP0YW</clipname>
    </clip>
  </cardinfo>
</copy>
```

```
    "clip5": {
      "clipname": "015KF5QO",
      "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
015KF5QO.MXF",
      "index": "5"
    },
    "clip6": {
      "clipname": "016SP1HV",
      "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
016SP1HV.MXF",
      "index": "6"
    },
    "clip7": {
      "clipname": "017UE4IK",
      "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
017UE4IK.MXF",
      "index": "7"
    },
    "clip8": {
      "clipname": "018WI7GJ",
      "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
018WI7GJ.MXF",
      "index": "8"
    },
    "clip9": {
      "clipname": "019GU1RS",
      "filenamepath": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
PX270MP_LasVegas\\CONTENTS\\AVCLIP\\
019GU1RS.MXF",
      "index": "9"
    },
    "source": "M:\\t3media\\Streams\\MXF\\
Panasonic\\Ultra-LongGop25\\
```

	PX270MP_LasVegas\\"
	  },
	  "netxbaseip": "192.168.50.100",
	  "result": "0",
	  "resultstring": "Cardinfo returned"
	}

set/get – metdata (file information and metadata in XMP format)

To save the metadata for a file to a sidebar XMP filename

http://127.0.0.1:1080/netx?request=set&client=any&command=copy&profile=metadata&source=M:\t3media\Streams\MXF\Sony%20Raw\Venice%206k\C001C014_18031075_F55_X-OCN_ST_6k\C001C014_18031075.mxf

<request netxbaseip="192.168.50.100" type="command=copy"> <metadata success="1">0</metadata> <result>0</result> <resultstring>success</resultstring> </request>	***This command does not support JSON returns.***

You will probably receive a few of these ‘waiting’ returns until the media is ready. At that point, you will get back the actual cardinfo:

http://127.0.0.1:1080/netx?request=get&client=any&command=copy&profile=metadata&source=M:\t3media\Streams\MXF\Sony%20Raw\Venice%206k\C001C014_18031075_F55_X-OCN_ST_6k\C001C014_18031075.mxf

<copy> <address>127.0.0.1</address> <netxbaseip>192.168.50.100</netxbaseip> <result>0</result> <resultstring>no data from previous metadata call found yet</resultstring> </copy>	***This command does not support JSON returns.***

To return the metadata as an XML XMP via the rest API

http://127.0.0.1:1080/netx?request=get&client=any&command=copy&profile=metadata&source=M:\t3media\Streams\MXF\Sony%20Raw\Venice%206k\C001C014_18031075_F55_X-OCN_ST_6k\C001C014_18031075.mxf

See below:	***This command does not support JSON returns.***

```
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 5.1.2">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
        xmlns:aux="http://ns.adobe.com/exif/1.0/aux/">
      <aux:SerialNumber>B4TCA0016</aux:SerialNumber>
    </rdf:Description>
    <rdf:Description rdf:about=""
        xmlns:tiff="http://ns.adobe.com/tiff/1.0/">
      <tiff:Model>AJ-PX270</tiff:Model>
    </rdf:Description>
    <rdf:Description rdf:about=""
        xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:source>SHOOTING</dc:source>
    </rdf:Description>
    <rdf:Description rdf:about=""
        xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">

<xmpMM:DocumentID>060A2B340101010501010D431300000095EB7CB7526705CF008045822D3
24006</xmpMM:DocumentID>
      <xmpMM:VersionID>7.1.0.6</xmpMM:VersionID>
    </rdf:Description>
    <rdf:Description rdf:about=""
        xmlns:xmpDM="http://ns.adobe.com/xmp/1.0/DynamicMedia/"
        xmlns:stDim="http://ns.adobe.com/xap/1.0/sType/Dimensions#">
      <xmpDM:audioSampleRate>48000</xmpDM:audioSampleRate>
      <xmpDM:duration rdf:parseType="Resource">
        <xmpDM:value>495</xmpDM:value>
        <xmpDM:scale>1001/30000</xmpDM:scale>
      </xmpDM:duration>
      <xmpDM:videoFrameSize rdf:parseType="Resource">
        <stDim:unit>pixel</stDim:unit>
        <stDim:w>1920</stDim:w>
        <stDim:h>1080</stDim:h>
      </xmpDM:videoFrameSize>
      <xmpDM:startTimecode rdf:parseType="Resource">
        <xmpDM:timeFormat>2997DropTimecode</xmpDM:timeFormat>
        <xmpDM:timeValue>00:01:49;17</xmpDM:timeValue>
      </xmpDM:startTimecode>
      <xmpDM:altTimecode rdf:parseType="Resource">
        <xmpDM:timeFormat>2997DropTimecode</xmpDM:timeFormat>
        <xmpDM:timeValue>00:01:49;17</xmpDM:timeValue>
      </xmpDM:altTimecode>
      <xmpDM:audioChannelType>Other</xmpDM:audioChannelType>
      <xmpDM:audioSampleType>24Int</xmpDM:audioSampleType>
    </rdf:Description>
    <rdf:Description rdf:about=""
        xmlns:dt="http://www.drastictech.com/metadata/elements">
      <dt:tracks>
```

```
        <rdf:Bag>
          <rdf:li rdf:parseType="Resource">
            <rdf:value/>
            <dt:Track_Type>General</dt:Track_Type>
            <dt:Complete_name>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\
PX270MP_LasVegas\CONTENTS\AVCLIP\015KF5QO.MXF</dt:Complete_name>
            <dt:Format>MXF</dt:Format>
            <dt:Format_version>1.3</dt:Format_version>
            <dt:Format_profile>OP-1b</dt:Format_profile>
            <dt:Format_settings>Closed / Complete</dt:Format_settings>
            <dt:File_size>63.5 MiB</dt:File_size>
            <dt:Duration>16 s 517 ms</dt:Duration>
            <dt:Overall_bit_rate_mode>Variable</dt:Overall_bit_rate_mode>
            <dt:Overall_bit_rate>32.3 Mb/s</dt:Overall_bit_rate>
            <dt:Package_name>015KF5QO</dt:Package_name>
            <dt:Encoded_date>2014-04-05 17:48:53.000</dt:Encoded_date>
            <dt:Writing_application>Panasonic P2 3.0</dt:Writing_application>
            <dt:FileName>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\
PX270MP_LasVegas\CONTENTS\AVCLIP\015KF5QO.MXF</dt:FileName>
            <dt:NativeLocator>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\
PX270MP_LasVegas\CONTENTS\AVCLIP\015KF5QO.MXF</dt:NativeLocator>
            <dt:UniversalLocator>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\
PX270MP_LasVegas\CONTENTS\AVCLIP\015KF5QO.MXF</dt:UniversalLocator>
            <dt:Title>015KF5QO</dt:Title>
            <dt:Source>SHOOTING</dt:Source>
            <dt:FullName>015KF5QO</dt:FullName>
            <dt:Model>AJ-PX270</dt:Model>
            <dt:EditData3>AVC-LongG422</dt:EditData3>
            <dt:VersionString>7.1.0.6</dt:VersionString>
            <dt:Manufacturer>Panasonic</dt:Manufacturer>
            <dt:DeviceSerialNum>B4TCA0016</dt:DeviceSerialNum>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <rdf:value/>
            <dt:Track_Type>Video</dt:Track_Type>
            <dt:ID>2</dt:ID>
            <dt:Format>AVC</dt:Format>
            <dt:Format_Info>Advanced Video Codec</dt:Format_Info>
            <dt:Format_profile>High 4:2:2@L4</dt:Format_profile>
            <dt:Format_settings__CABAC>Yes</dt:Format_settings__CABAC>
            <dt:Format_settings__ReFrames>3 frames</dt:Format_settings__ReFrames>
            <dt:Format_settings__GOP>M=3, N=30</dt:Format_settings__GOP>
            <dt:Format_settings__wrapping_mode>Frame</dt:Format_settings__wrapping_mode>
            <dt:Codec_ID>0D01030102106001-0401020201316001</dt:Codec_ID>
            <dt:Duration>16 s 517 ms</dt:Duration>
            <dt:Bit_rate_mode>Variable</dt:Bit_rate_mode>
            <dt:Bit_rate>25.0 Mb/s</dt:Bit_rate>
            <dt:Maximum_bit_rate>37.5 Mb/s</dt:Maximum_bit_rate>
            <dt:Width>1920</dt:Width>
            <dt:Height>1080</dt:Height>
            <dt:Display_aspect_ratio>16:9</dt:Display_aspect_ratio>
```

```
      <dt:Frame_rate>29.970 (30000/1001) FPS</dt:Frame_rate>
      <dt:Standard>Component</dt:Standard>
      <dt:Color_space>YUV</dt:Color_space>
      <dt:Chroma_subsampling>4:2:2</dt:Chroma_subsampling>
      <dt:Bit_depth>10 bits</dt:Bit_depth>
      <dt:Scan_type>Interlaced</dt:Scan_type>
      <dt:Scan_type__store_method>Separated fields</dt:Scan_type__store_method>
      <dt:Scan_order>Top Field First</dt:Scan_order>
      <dt:Bits__Pixel_Frame_>0.402</dt:Bits__Pixel_Frame_>
      <dt:Stream_size>49.2 MiB (77%)</dt:Stream_size>
      <dt:Color_range>Limited</dt:Color_range>
      <dt:Color_primaries>BT.709</dt:Color_primaries>
      <dt:Transfer_characteristics>BT.709</dt:Transfer_characteristics>
      <dt:Matrix_coefficients>BT.709</dt:Matrix_coefficients>
      <dt:BitRate_Maximum_Original>37499904</dt:BitRate_Maximum_Original>
      <dt:Size>40</dt:Size>
      <dt:Planes>1</dt:Planes>
      <dt:BitCount>30</dt:BitCount>
      <dt:Compression>1635142740</dt:Compression>
      <dt:Size_Image>230400</dt:Size_Image>
      <dt:TimeCode>3285</dt:TimeCode>
      <dt:VITCTimeCode>3285</dt:VITCTimeCode>
      <dt:VersionNumber>70100006</dt:VersionNumber>
      <dt:TimeCodeType>4</dt:TimeCodeType>
      <dt:LTCTimeCodeType>4</dt:LTCTimeCodeType>
      <dt:VITCTimeCodeType>4</dt:VITCTimeCodeType>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
      <rdf:value/>
      <dt:Track_Type>Audio</dt:Track_Type>
      <dt:ID>3</dt:ID>
      <dt:Format>PCM</dt:Format>
      <dt:Format_settings__Endianness>Little</dt:Format_settings__Endianness>
      <dt:Codec_ID>0D01030102060B00</dt:Codec_ID>
      <dt:Duration>16 s 517 ms</dt:Duration>
      <dt:Bit_rate_mode>Constant</dt:Bit_rate_mode>
      <dt:Bit_rate>1 152 kb/s</dt:Bit_rate>
      <dt:Channel_s_>1 channel</dt:Channel_s_>
      <dt:Sampling_rate>48.0 kHz</dt:Sampling_rate>
      <dt:Bit_depth>24 bits</dt:Bit_depth>
      <dt:Stream_size>2.27 MiB (4%)</dt:Stream_size>
      <dt:Locked>Yes</dt:Locked>
      <dt:Format_Tag>1</dt:Format_Tag>
      <dt:Channels>4</dt:Channels>
      <dt:Samples_Per_Sec>48000</dt:Samples_Per_Sec>
      <dt:Avg_Bytes_Per_Sec>576000</dt:Avg_Bytes_Per_Sec>
      <dt:Block_Align>16</dt:Block_Align>
      <dt:Bits_Per_Sample>24</dt:Bits_Per_Sample>
      <dt:Reserved>1935963489</dt:Reserved>
      <dt:ccType>1935963489</dt:ccType>
      <dt:ccHandler>1</dt:ccHandler>
```

```
      <dt:Scale>12</dt:Scale>
      <dt:Rate>576000</dt:Rate>
      <dt:Length>792792</dt:Length>
      <dt:Suggested_Buffer_Size>288000</dt:Suggested_Buffer_Size>
      <dt:Sample_Size>12</dt:Sample_Size>
      <dt:File_Type>172</dt:File_Type>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
      <rdf:value/>
      <dt:Track_Type>Audio</dt:Track_Type>
      <dt:ID>4</dt:ID>
      <dt:Format>PCM</dt:Format>
      <dt:Codec_ID>0D01030102060B00</dt:Codec_ID>
      <dt:Duration>16 s 517 ms</dt:Duration>
      <dt:Bit_rate>1 152 kb/s</dt:Bit_rate>
      <dt:Channel_s_>1 channel</dt:Channel_s_>
      <dt:Sampling_rate>48.0 kHz</dt:Sampling_rate>
      <dt:Bit_depth>24 bits</dt:Bit_depth>
      <dt:Stream_size>2.27 MiB (4%)</dt:Stream_size>
      <dt:Locked>Yes</dt:Locked>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
      <rdf:value/>
      <dt:Track_Type>Audio</dt:Track_Type>
      <dt:ID>5</dt:ID>
      <dt:Format>PCM</dt:Format>
      <dt:Codec_ID>0D01030102060B00</dt:Codec_ID>
      <dt:Duration>16 s 517 ms</dt:Duration>
      <dt:Bit_rate>1 152 kb/s</dt:Bit_rate>
      <dt:Channel_s_>1 channel</dt:Channel_s_>
      <dt:Sampling_rate>48.0 kHz</dt:Sampling_rate>
      <dt:Bit_depth>24 bits</dt:Bit_depth>
      <dt:Stream_size>2.27 MiB (4%)</dt:Stream_size>
      <dt:Locked>Yes</dt:Locked>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
      <rdf:value/>
      <dt:Track_Type>Audio</dt:Track_Type>
      <dt:ID>7</dt:ID>
      <dt:Format>PCM</dt:Format>
      <dt:Codec_ID>0D01030102060B00</dt:Codec_ID>
      <dt:Duration>16 s 517 ms</dt:Duration>
      <dt:Bit_rate>1 152 kb/s</dt:Bit_rate>
      <dt:Channel_s_>1 channel</dt:Channel_s_>
      <dt:Sampling_rate>48.0 kHz</dt:Sampling_rate>
      <dt:Bit_depth>24 bits</dt:Bit_depth>
      <dt:Stream_size>2.27 MiB (4%)</dt:Stream_size>
      <dt:Locked>Yes</dt:Locked>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
      <rdf:value/>
```

```
            <dt:Track_Type>Other</dt:Track_Type>
            <dt:ID>1-Material</dt:ID>
            <dt:Type>Time code</dt:Type>
            <dt:Format>MXF TC</dt:Format>
            <dt:Time_code_of_first_frame>00:01:49;17</dt:Time_code_of_first_frame>
            <dt:Time_code_settings>Material Package</dt:Time_code_settings>
            <dt:Time_code__striped>Yes</dt:Time_code__striped>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <rdf:value/>
            <dt:Track_Type>Other</dt:Track_Type>
            <dt:ID>1-Source</dt:ID>
            <dt:Type>Time code</dt:Type>
            <dt:Format>MXF TC</dt:Format>
            <dt:Time_code_of_first_frame>00:01:49;17</dt:Time_code_of_first_frame>
            <dt:Time_code_settings>Source Package</dt:Time_code_settings>
            <dt:Time_code__striped>Yes</dt:Time_code__striped>
          </rdf:li>
        </rdf:Bag>
      </dt:tracks>
      <dt:CreationData>2014-04-05T17:49:10+00:00</dt:CreationData>
      <dt:EditData3>AVC-LongG422</dt:EditData3>
      <dt:FileName>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\PX270MP_LasVegas\
CONTENTS\AVCLIP\015KF5QO.MXF</dt:FileName>
      <dt:DeviceModelNum>AJ-PX270</dt:DeviceModelNum>
      <dt:DeviceSerialNum>B4TCA0016</dt:DeviceSerialNum>
      <dt:FullName>015KF5QO</dt:FullName>
      <dt:Manufacturer>Panasonic</dt:Manufacturer>
      <dt:Model>AJ-PX270</dt:Model>
      <dt:NativeLocator>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\
PX270MP_LasVegas\CONTENTS\AVCLIP\015KF5QO.MXF</dt:NativeLocator>
      <dt:Source>SHOOTING</dt:Source>
      <dt:Title>015KF5QO</dt:Title>
      <dt:UniversalLocator>M:\t3media\Streams\MXF\Panasonic\Ultra-LongGop25\
PX270MP_LasVegas\CONTENTS\AVCLIP\015KF5QO.MXF</dt:UniversalLocator>
      <dt:VersionString>7.1.0.6</dt:VersionString>
      <dt:A-Frame>0</dt:A-Frame>
      <dt:PosterFrame>0</dt:PosterFrame>
      <dt:TimeCode>3285</dt:TimeCode>
      <dt:UserBits>335807767</dt:UserBits>
      <dt:VITCTimeCode>3285</dt:VITCTimeCode>
      <dt:VITCUserBits>335807767</dt:VITCUserBits>
      <dt:VersionNumber>70100006</dt:VersionNumber>
      <dt:TimeCodeType>4</dt:TimeCodeType>
      <dt:LTCTimeCodeType>4</dt:LTCTimeCodeType>
      <dt:VITCTimeCodeType>4</dt:VITCTimeCodeType>
    </rdf:Description>
    <rdf:Description rdf:about=""
        xmlns:dtdev="http://www.drastictech.com/metadata/deviceelements">
      <dtdev:DTDeviceMetadataType>PANASONICP2</dtdev:DTDeviceMetadataType>
      <dtdev:ClipContent-
```

```
GlobalClipID>060A2B340101010501010D431300000095EB7CB7526705CF008045822D324006</
dtdev:ClipContent-GlobalClipID>
        <dtdev:ClipContent-Duration>495</dtdev:ClipContent-Duration>
        <dtdev:ClipContent-EditUnit>1001/30000</dtdev:ClipContent-EditUnit>
        <dtdev:Video-Codec>AVC-LongG422</dtdev:Video-Codec>
        <dtdev:Video-ActiveLine>1080</dtdev:Video-ActiveLine>
        <dtdev:Video-FrameRate>59.94i</dtdev:Video-FrameRate>
        <dtdev:Video-TimecodeType>Drop</dtdev:Video-TimecodeType>
        <dtdev:Video-StartTimecode>00:01:49:17</dtdev:Video-StartTimecode>
        <dtdev:Video-EndTimecode>00:02:06:03</dtdev:Video-EndTimecode>
        <dtdev:Video-StartBinaryGroup>14040517</dtdev:Video-StartBinaryGroup>
        <dtdev:Audio-Channel>4</dtdev:Audio-Channel>
        <dtdev:Audio-SamplingRate>48000</dtdev:Audio-SamplingRate>
        <dtdev:Audio-BitsPerSample>24</dtdev:Audio-BitsPerSample>
        <dtdev:ClipMetadata-UserClipName>015KF5QO</dtdev:ClipMetadata-UserClipName>
        <dtdev:ClipMetadata-DataSource>SHOOTING</dtdev:ClipMetadata-DataSource>
        <dtdev:Access-CreationDate>2014-04-05T17:48:53+00:00</dtdev:Access-CreationDate>
        <dtdev:Access-LastUpdateDate>2014-04-05T17:49:10+00:00</dtdev:Access-
LastUpdateDate>
        <dtdev:Device-Manufacturer>Panasonic</dtdev:Device-Manufacturer>
        <dtdev:Device-SerialNo.>B4TCA0016</dtdev:Device-SerialNo.>
        <dtdev:Device-ModelName>AJ-PX270</dtdev:Device-ModelName>
        <dtdev:Shoot-StartDate>2014-04-05T17:48:53+00:00</dtdev:Shoot-StartDate>
        <dtdev:Shoot-EndDate>2014-04-05T17:49:10+00:00</dtdev:Shoot-EndDate>
        <dtdev:Thumbnail-FrameOffset>0</dtdev:Thumbnail-FrameOffset>
        <dtdev:Thumbnail-ThumbnailFormat>BMP</dtdev:Thumbnail-ThumbnailFormat>
        <dtdev:Thumbnail-Width>80</dtdev:Thumbnail-Width>
        <dtdev:Thumbnail-Height>45</dtdev:Thumbnail-Height>
        <dtdev:Proxy-ProxyFormat>MOV</dtdev:Proxy-ProxyFormat>
        <dtdev:ProxyVideo-ProxyVideoCodec>AVC</dtdev:ProxyVideo-ProxyVideoCodec>
        <dtdev:ProxyVideo-ProxyVideoBitRate>6000000</dtdev:ProxyVideo-ProxyVideoBitRate>
        <dtdev:ProxyVideo-ProxyFrameRate>29.97</dtdev:ProxyVideo-ProxyFrameRate>
        <dtdev:ProxyVideo-ProxyResolution>1920x1080</dtdev:ProxyVideo-ProxyResolution>
        <dtdev:ProxyVideo-AspectRatio>16:9</dtdev:ProxyVideo-AspectRatio>
        <dtdev:ProxyVideo-OnScreenTimecode>OFF</dtdev:ProxyVideo-OnScreenTimecode>
        <dtdev:ProxyAudio-ProxyAudioCodec>AAC</dtdev:ProxyAudio-ProxyAudioCodec>
        <dtdev:ProxyAudio-ProxyAudioBitRate>64000</dtdev:ProxyAudio-ProxyAudioBitRate>
        <dtdev:ProxyAudio-ProxySamplingRate>48000</dtdev:ProxyAudio-ProxySamplingRate>
        <dtdev:ProxyAudio-OriginalChannel>1</dtdev:ProxyAudio-OriginalChannel>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
```

This command also supports writing the metadata to a disk locations with.

```
http://127.0.0.1:1080/netx?request=set&client=any&command=copy&profile=metadata&source=M:\t3media\
Streams\MXF\Sony%20Raw\Venice%206k\C001C014_18031075_F55_X-OCN_ST_6k\
C001C014_18031075.mxf&target=m:\t3media\metadata\C001C014_18031075.mxf.xmp
```

<copy> <address>192.168.100.229</address>	{ "address": "192.168.100.229",

<netxbaseip>192.168.50.100</netxbaseip> <result>0</result> <resultstring>no data from previous cardinfo call found yet</resultstring> </copy>	"netxbaseip": "192.168.50.100", "result": "0", "resultstring": "no data from previous metadata call found yet" }

NOTE: The XMP output is compatible with the XMP standard. There are two extensions included to allow for more information then is normally available in an XMP file. The first is the ‘dt: space that includes extended track and metadata information. It has each of the files ‘tracks’ set in an array of ‘li’ elements with a ‘bag’ within the document. The structure of the tracks is

```
<rdf:Description rdf:about=""
        xmlns:dt="http://www.drastictech.com/metadata/elements">

      <dt:tracks>
        <rdf:Bag>
          <rdf:li rdf:parseType="Resource">
… track info
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
… track info
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
          </rdf:li>
        </rdf:Bag>
      </dt:tracks>
```

After that there will be a number of general metadata parameters.

The second section is the ‘dtdev:’. This is a device specific area that passes back advanced parameters from the file, if available. This is commonly used for the shooting parameters of cameras from RED, Sony, Panasonic, Canon, BlackMagic and others. It will contain any available paramters set in the camera, usually including items like colourmetry, camera setting, gps coordinates, model/serial number and shoot/take/reel information.

Parameters:

command=copy – must be copy to use the copy/transcode/PFR
source=<sourcefile> - source name may be direct path, unc path or other system accessible path
target=<targetfile> - target name may be direct path, unc path or other system accessible path
ack=<ackfile> - the XML ACK file that will be created on completion with status and metadata
tcin=01:00:00:00 - start as a time code
tcout=01:01:00:00 - end as a time code reference (exclusive)
absin=00:00:10:00 - first frame number in absolute frames from 0. Also support frame number
absout=450 - last frame number in absolute frames from 0 (exclusive). Also support time code
tcoffset=01:00:00:00 - amount to offset the time code of the output clip. Does not affect the tcin/tcout.
copy - make a copy of the file section we need before translating, instead of reading directly
dest=<destfolder> - folder or folder and file name for the temp file when using copy
profile=*** - Profile to use. Current profiles include:
copy - copy the whole file
wrap - re wrap file or part of a file

index - create an RTIndex for a file
getCopyInOut - get the extents required for a pfr, or use them with a temp file
mp3-128kbps - Audio MP3 file
mov-YCbCr8Bit - QuickTime MOV 8 bit uncompressed YcbCr file
mov-dvcprohd - QuicktTime MOV DVCPro HD (1080/720)
mp4-h264 - MPEG-4 h264 AAC Audio
mxf-xdcam-720p - True XDCam MXF 8 channel audio
mxf-dvcprohd-720p - MXF DVCPro HD 720p
mxf-xdcam-1080i - True XDCam MXF 1080i 8 channel audio
mxf-dvcprohd-1080i - MXF DVCPro HD 1080i 29/25 fps
mxf-OP1a-MPEG - OpenMXF XDCam MPEG-2 16 channel audio
mxf-OP1a-h264 - MXF h.264
mxf-OP1a-HDF - MXF MPEG-2 HDF Standard
mxf-as-11-sd-pal-dpp - MXF AS-11 SD PAL DPP
mxf-as-11-sd-ntsc-dpp - MXF AS-11 SD NTSC DPP
mxf-as-11-hd-dpp - MXF DPP AS-11 AVCi HD
mov-proreshq - QuickTime MOV ProRes HQ
mov-proreslt - QuickTime MOV ProRes LT
mov-prores422 - QuickTime MOV ProRes 422
mov-prores444 - QuickTime MOV ProRes 444(4)
scaledown2000k - MP4 264 960x540, 2mbs, AAC
scaledown500k - MP4 264 480x272, 0.5mbs, AAC
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
TS-MPEG2 - MPEG-2 4:2:0/passthrough transport stream
TS-h264 - h.264 4:2:0/passthrough transport stream
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
aces - ACES image files
dnxhd-mxf-720p - DNxHD 720p 50, 59, 60
dnxhd-mxf-1080p - DNxHD 1080p 25, 29
dnxhd-mxf-1080i - DNxHD 1080i 25, 29
dnxhr-mxf-10-hq - DNxHR High Quality 10 bit
dnxhr-mxf-8-hq - DNxHR High Quality 8 bit
dnxhr-mxf-sq - DNxHR Standard Quality
dnxhr-mxf-lq - DNxHR Low Quality

type=*** - the exact file type to write, otherwise auto
mxf-op1a - standard OP1a
mxf-sonyhd - Sony XDCam compatible
mxf-as02 - AS-02 spec MXF
mxf-open - Edit while ingest MXF

mp4-fmp4 - Fragmented MP4(normal MP4 if not set)
mov - QuickTime MOV
metadata[=optional file name] - Save metadata from a file to an XML/XMP file
Width=480 - Sets the width of the picon or video output, when not a fixed size codec (like DVHD)
Height=272 - Sets the height of the video output, when not a fixed size codec (like DVHD). No effect on picons.
cc=<ccfile> - optional alternate close caption file (SCC or MCC preferred)
afile=<audiofile> - alternate audio file source (aiff or wav)
video=<videofile> - alternate video file source
stereo - sum all the audio channels into a single stereo pair
aroute=78123456 - route any audio channel from any audio channel
uuid=<uuid> - apply a specific UUID to the output filename
kilobitrate=5000 - override the default compression bit rate with this rate (mbs * 1024)
h26xprofile=41 - use this profile for h264/h265 (see below)
h26xlevel=77 - use this level for h264/h265 (see below)
gopsize=15 - number of frames in a GOP
encodemode=0 - fast (1) or quality (0) encoding
tempfolder=<tempdir> - temporary folder to hold the tape PFR binary chunk
alignment=4096 - required alignment of the PFR tape or disk source
fg - force the NetXCopy GUI to be displayed
fc - force the command line to be used

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 time code string of the frame to use
piSkip = Number of frames to skip between picons. If set, a series of picons will be produced at ‘skip’ intervals
width = set the width of the output picon. The height will be calculated from this

h26x Profiles:
10, 11, 12, 13, 20, 21, 22, 30, 31, 32, 40, 41, 42, 50, 51, 52

h26x Levels:
66 – Baseline
77 – Main
88 – Extended
100 – High
110 – High 10
122 – High 4:2:2
144 – High 4:4:4
166 – Advanced 4:4:4 Intra
188 – Advanced 4:4:4

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=0&command=copy&copylimit=5	
<request netxbaseip="192.168.50.100" type="command=copy" message="success">	{

<result>0</result> <resultstring>success</resultstring> </request>	"message": "success", "netxbaseip": "192.168.50.100", "result": "0", "resultstring": "success", "type": "command=copy" }

To clear what is in the current working information, call:

http://127.0.0.1:1080/netx?request=set&client=0&command=copy&clear=

<request netxbaseip="192.168.50.100"
type="command=copy" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

To abort the current process, call:

http://127.0.0.1:1080/netx?request=set&client=0&command=copy&abort=

<request netxbaseip="192.168.50.100"
type="command=copy" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

To clean out the copy/convert history for all items, call:

http://127.0.0.1:1080/netx?request=set&client=0&group=-1&command=copy&purge=

<request netxbaseip="192.168.50.100"
type="command=copy" message="failure">
<result>-2</result>
<resultstring>failure</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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=0&group=1&command=copy&copylimit=5

<request netxbaseip="192.168.50.100" type="command=copy" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>

{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
}

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)

http://127.0.0.1:1080/netx?request=set&client=0&group=-1&command=copy&profile=mxf-OP1a-MPEG&source=d:\record\first_file.mxf&target=d:\record\second_file.mxf

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=0&command=copy

<copy>
  <address>192.168.100.229</address>
  <netxbaseip>192.168.50.100</netxbaseip>

{
  "address": "192.168.100.229",

```
 <copyid id="-2" index=”0”>
   <source>
M:\t3media\Streams\NBA\
NBA_Reference_Test_Files\
amberfin_new_0000750142.rtin
   </source>
   <target>m:\restorepoint\partial\
amberfin_clip.mxf</target>
   <targetrtin>m:\restorepoint\partial\
amberfin_clip.rtin</targetrtin>
   <targetack>m:\restorepoint\partial\
amberfin_clip.ack</targetack>
   <profile>wrap</profile>
   <tcoffset>0</tcoffset>
   <type>copy</type>
   <progress>3306</progress>
   <startms>166595260</startms>
   <currentms>168142191</currentms>
 </copyid>
 <result>0</result>
 <resultstring>returning requested
copy</resultstring>
</copy>
```

```
 "copyid0": {
   "currentms": "168126365",
   "id": "-2",
   “index”: “0”,
   "profile": "wrap",
   "progress": "3273",
   "source": "M:\\t3media\\Streams\\NBA\\
NBA_Reference_Test_Files\\
amberfin_new_0000750142.rtin",
   "startms": "166595260",
   "target": "m:\\restorepoint\\partial\\
amberfin_clip.mxf",
   "targetack": "m:\\restorepoint\\partial\\
amberfin_clip.ack",
   "targetrtin": "m:\\restorepoint\\partial\\
amberfin_clip.rtin",
   "tcoffset": "0",
   "type": "copy"
 },
 "netxbaseip": "192.168.50.100",
 "result": "0",
 "resultstring": "returning requested copy"
}
```

http://127.0.0.1:1080/netx?request=get&client=0&group=-1&command=copy

```
<copy>
 <address>192.168.100.229</address>
 <netxbaseip>192.168.50.100</netxbaseip>
 <copyid id="-2" index=”1”>
   <source>
M:\t3media\Streams\NBA\
NBA_Reference_Test_Files\
amberfin_new_0000750142.rtin
   </source>
   <target>m:\restorepoint\partial\
amberfin_clip.mxf</target>
   <targetrtin>m:\restorepoint\partial\
amberfin_clip.rtin</targetrtin>
   <targetack>m:\restorepoint\partial\
amberfin_clip.ack</targetack>
```

```
{
 "address": "192.168.100.229",
 "copyid1": {
   "currentms": "168179870",
   "id": "-2",
   “index”: “1”,
   "profile": "wrap",
   "progress": "3379",
   "source": "M:\\t3media\\Streams\\NBA\\
NBA_Reference_Test_Files\\
amberfin_new_0000750142.rtin",
   "startms": "166595260",
   "target": "m:\\restorepoint\\partial\\
amberfin_clip.mxf",
   "targetack": "m:\\restorepoint\\partial\\
```

<table>
<tr>
<td>

```
   <profile>wrap</profile>
   <tcoffset>0</tcoffset>
   <type>copy</type>
   <progress>3443</progress>
   <startms>166595260</startms>
   <currentms>168210287</currentms>
  </copyid>
  <result>0</result>
  <resultstring>returning requested
copy</resultstring>
</copy>
```

</td>
<td>

```
amberfin_clip.ack",
   "targetrtin": "m:\\restorepoint\\partial\\
amberfin_clip.rtin",
   "tcoffset": "0",
   "type": "copy"
  },
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "returning requested copy"
}
```

</td>
</tr>
</table>

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=0&command=copy&group=find&target=d:\record\second_file
```

<table>
<tr>
<td>

```
<copy>
  <address>192.168.100.229</address>
  <netxbaseip>192.168.50.100</netxbaseip>
  <copyid id="-1" index=”0”>
   <source>d:\record\first_file.mxf</source>
   <target/>
   <targetrtin>d:\record\first_file .rtin</targetrtin>
   <targetack>d:\record\first_file .ack</targetack>
   <profile>mxf-OP1a-MPEG</profile>
   <tcoffset/>
   <progress>535</progress>
   <startms>100345709</startms>
   <currentms>-1</currentms>
   <type/>
   <message>Copy Pending</message>
  </copyid>
  <result>0</result>
  <resultstring>returning requested
copy</resultstring>
</copy>
```

</td>
<td>

```
{
  "address": "192.168.100.229",
  "copyid0": {
   "currentms": "",
   "id": "-1",
   “index”: “0”,
   "message": "Unable to open source file for
transcode",
   "profile": "mxf-OP1a-MPEG",
   "progress": "-1",
   "source": "d:\\record\\first_file.mxf",
   "startms": "",
   "target": "d:\\record\\second_file.mxf",
   "targetack": "d:\\record\\second_file.ack",
   "targetrtin": "d:\\record\\second_file.rtin",
   "tcoffset": "0",
   "type": "copy"
  },
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "returning requested copy"
}
```

</td>
</tr>
</table>

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=0&group=-1&command=copy

```
<copy>
  <address>192.168.100.229</address>
  <netxbaseip>192.168.50.100</netxbaseip>
  <copyid id="-1" index=”1”>
    <source>d:\record\first_file.mxf</source>
    <target/>
    <targetrtin>.rtin</targetrtin>
    <targetack>.ack</targetack>
    <profile>mxf-OP1a-MPEG</profile>
    <tcoffset/>
    <type/>
    <progress>0</progress>
    <startms>100345709</startms>
    <currentms>-1</currentms>
    <message>Copy Pending</message>
  </copyid>
  <copyid id="-2" index=”1”>
    <source>d:\record\first_file.mxf</source>
    <target>d:\record\second_file.mxf</target>

<targetrtin>d:\record\second_file.rtin</targetrtin>

<targetack>d:\record\second_file.ack</targetack>
    <profile>mxf-OP1a-MPEG</profile>
    <tcoffset/>
    <type/>
    <progress>6783</progress>
    <startms>100346586</startms>
    <currentms>100460047</currentms>
  </copyid>
  <result>0</result>
  <resultstring>returning requested
copy</resultstring>
</copy>
```

```
{
  "address": "192.168.100.229",
  "copyid1": {
    "currentms": "168547725",
    "id": "-2",
    “index”: “1”,
    "profile": "wrap",
    "progress": "4153",
    "source": "M:\\t3media\\Streams\\NBA\\
NBA_Reference_Test_Files\\
amberfin_new_0000750142.rtin",
    "startms": "166595260",
    "target": "m:\\restorepoint\\partial\\
amberfin_clip.mxf",
    "targetack": "m:\\restorepoint\\partial\\
amberfin_clip.ack",
    "targetrtin": "m:\\restorepoint\\partial\\
amberfin_clip.rtin",
    "tcoffset": "0",
    "type": "copy"
  },
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "returning requested 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 requests, add the complete=1 flag

http://127.0.0.1:1080/netx?request=get&client=0&command=copy&complete=1

```
<copy>
  <address>192.168.100.229</address>
```

```
{
  "address": "192.168.100.229",
```

<netxbaseip>192.168.50.100</netxbaseip> <result>0</result> <resultstring>returning requested copy</resultstring> </copy>	"netxbaseip": "192.168.50.100", "result": "0", "resultstring": "returning requested copy" }

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=0&command=copy&complete=1&purge=1

<copy>
<address>192.168.100.229</address>
<netxbaseip>192.168.50.100</netxbaseip>
<copyid id="-2" index="0">
<source>d:\record\first_file.mxf</source>
<target>d:\record\second_file.mxf</target>
 	
<targetrtin>d:\record\second_file.rtin</targetrtin>
 	
<targetack>d:\record\second_file.ack</targetack>
<profile>mxf-OP1a-MPEG</profile>
<tcoffset/>
<type/>
<progress>-1</progress>
<startms/>
<currentms/>
<message>Copy successful</message>
</copyid>
<result>0</result>
<resultstring>returning requested copy</resultstring>
</copy>
	{
	"address": "192.168.100.229",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "returning requested copy"
	}

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

http://127.0.0.1:1080/netx?request=get&client=0&command=copy

<copy>
<address>192.168.100.229</address>
<netxbaseip>192.168.50.100</netxbaseip>
<copyid id="-1" index="0">
<source>d:\record\first_file.mxf</source>
<target/>
<targetrtin>.rtin</targetrtin>
<targetack>.ack</targetack>
	{
	"address": "192.168.100.229",
	"copyid0": {
	"currentms": "168637148",
	"id": "-2",
	“index”: “0”,
	"profile": "wrap",

<table>
<tr>
<td>

```
    <profile>mxf-OP1a-MPEG</profile>
    <tcoffset/>
    <type/>
    <progress>0</progress>
    <startms>100345709</startms>
    <currentms>-1</currentms>
    <message>Copy Pending</message>
  </copyid>
  <copyid id="-2">
    <source>d:\record\first_file.mxf</source>
    <target>d:\record\second_file.mxf</target>

<targetrtin>d:\record\second_file.rtin</targetrtin>

<targetack>d:\record\second_file.ack</targetack>
    <profile>mxf-OP1a-MPEG</profile>
    <tcoffset/>
    <type/>
    <progress>-1</progress>
    <startms/>
    <currentms/>
    <message>Copy successful</message>
  </copyid>
  <result>0</result>
  <resultstring>returning requested
copy</resultstring>
</copy>
```

</td>
<td>

```
    "progress": "4346",
    "source": "M:\\t3media\\Streams\\NBA\\
NBA_Reference_Test_Files\\
amberfin_new_0000750142.rtin",
    "startms": "166595260",
    "target": "m:\\restorepoint\\partial\\
amberfin_clip.mxf",
    "targetack": "m:\\restorepoint\\partial\\
amberfin_clip.ack",
    "targetrtin": "m:\\restorepoint\\partial\\
amberfin_clip.rtin",
    "tcoffset": "0",
    "type": "copy"
  },
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "returning requested copy"
}
```

</td>
</tr>
</table>

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&group=find&command=copy&target=d:\record\second_file

<table>
<tr>
<td>

```
<copy>
  <address>192.168.100.229</address>
  <netxbaseip>192.168.50.100</netxbaseip>
  <copyid id="-1" index=”0”>
    <source>d:\record\first_file.mxf</source>
    <target/>
    <targetrtin>d:\record\first_file.rtin</targetrtin>
    <targetack>d:\record\first_file.ack</targetack>
    <profile>mxf-OP1a-MPEG</profile>
    <tcoffset/>
    <progress>0</progress>
    <startms>100345709</startms>
    <currentms>-1</currentms>
    <type/>
```

</td>
<td>

```
{
  "address": "192.168.100.229",
  "copyid0": {
    "id": "-1",
    “index”: “0”,
    "message": "copy not found",
    "profile": "",
    "source": "",
    "target": "d:\\record\\second_file"
  },
  "netxbaseip": "192.168.50.100",
  "result": "-1",
```

</td>
</tr>
</table>

<message>Copy Pending</message> </copyid> <result>0</result> <resultstring>returning requested copy</resultstring> </copy>	"resultstring": "message" }

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 than 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 subsequent scenarios 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 time code, 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 uses the HTML5 player’s time code (or other time code 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 time code 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 or part of local clips to any of these formats

PFR File Best Practices

Different workflows require different ways of saving and restoring files, but there are some general rules that can make it easier, especially when working with tape or other non sequential storage systems. Media files can be roughly broken down into a few categories:

- Self contained, single files (MXF OP1a, MOV, AVI)
- Multiple stream files (AVI+WAV, Avid OPAtom, MOV QT Reference)
- Multiple files per stream (P2 MXF, XDCam MP4, Canon C300/700 MXF)
- Sequences (DPX, TGA, TIFF)

If you are restoring from reasonable speed, random access devices, then all these types can be simply stored and retrieved as is. If there are sequential access, speed or cost issues, then it makes more sense to make each of these as easily accessible as possible before storing them.

Self Contained

These can be indexed and stored directly.

Multiple Stream Files

These can also be indexed and stored directly, as the RTIN can point at on file per stream. NetXCode

automatically finds and joins the parts of the streams if it is a supported file type like Avid OPAtom, MOV reference, or video file with rationally named audio. If they are stored this way, there will be one chunk from each file that needs to be restored to make the output file, as the media is in separate files. This is handled in the getinoutbytes return as a series of temp names and start/end byte ranges. If you prefer a single chunk, please follow the guidelines in the Multiple Files Per Stream section.

Multiple Files Per Stream

The RTIN cannot describe streams that have multiple parts per stream. These are normally broken up to get around older disk format restrictions, often at 2 or 4 gigabytes. To deal with these, they should be pre processed (wrapped) to a self contained file like MXF OP1a before they are stored to tape. Using NetXCode to wrap them will cause the original audio/video to be copied to the new MXF without recompression and will automatically generate the RTIN as it is creating the new file. A proxy file can also optionally be created while rewrapping the file. This MXF now becomes the file you would restore from and that should be stored to tape.
NetXCode fully supports automatically joining the parts of most broadcast and post production files for playback and wrapping, including

- Panasonic P2
- IMF
- AS-02
- DCP
- Canon C300/700
- Sony XDCam MXF
- Sony XDCam MP4
- Avid OP-Atom
- Grass Valley K2 Server Format
- Multi card MXF capture

Avid OP-Atom – Special Case

NetXCode supports automatically joining Avid OP-Atom files and creating Avid OP-Atom files that can be directly dropped into the Avid bin for pickup on the next database refresh. If you are using this workflow, then then the Multiple Stream Files method works best. If you are importing files into Avid then you will not be able to use OP-Atom, as Avid cannot import even its own. It will want to see the same essence, but in an OP1a container. In this case, you should re-wrap the OP-Atom to OP1a before storing them. It is also possible to restore OP-Atom to OP1a using the “type mxf-op1a” when doing the restore, but if they are stored as OP-Atom you will still need to restore a chunk from each stream to PFR them.

Sequences

As sequences can be restored ‘per file’, they do not need to be indexed.

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=0&group=0&command=start

<request netxbaseip="192.168.50.100" type="command=start">
  <start success="1">success</start>
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "start": {
    "success": "1",
    "value": "success"
  },
  "type": "command=start"
}
```


startstream

Start a group/channel streaming

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

<request netxbaseip="192.168.50.100" type="command=start">
  <start success="1">success</start>
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "command": {
    "success": "0",
    "value": "invalid"
  },
  "netxbaseip": "192.168.50.100",
  "result": "-1",
  "resultstring": "command=invalid",
  "type": "command=startstream"
}
```

stop

Stop a group record.

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

<request netxbaseip="192.168.50.100"
type="command=start">
  <start success="1">success</start>
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "stop": {
    "success": "1",
    "value": "success"
  },
  "type": "command=stop"
}
```


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=0&group=0&command=restart

<request netxbaseip="192.168.50.100"
type="command=start">
  <start success="1">success</start>
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "netxbaseip": "192.168.50.100",
  "restart": {
    "success": "1",
    "value": "success"
  },
  "result": "0",
  "resultstring": "success",
  "type": "command=restart"
}
```


Or the keyname of a group:

http://127.0.0.1:1080/netx?request=set&client=0&group=ab1&command=restart

Or -1 or all to restart all groups:

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

http://127.0.0.1:1080/netx?request=set&client=0&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=0&group=-2&command=restart

<request netxbaseip="192.168.50.100" type="command=restart">
  <restart success="1">success</restart>
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "netxbaseip": "192.168.50.100",
  "restart": {
    "success": "1",
    "value": "success"
  },
  "result": "0",
  "resultstring": "success",
  "type": "command=restart"
}
```


xmldisable

Disable XML sidecar generation for captured files

This command gets the state of XML sidecar generation

http://127.0.0.1:1080/netxjson?request=set&client=0&group=0&channel=0&xmldisable=1

<request netxbaseip="192.168.50.100" type="channel">
  <setting success="1">success</setting>
  <result>0</result>
  <resultstring>success</resultstring>
</request>


```
{
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "setting": {
    "success": "1",
    "value": "success"
  },
  "type": "channel"
}
```

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

autostart

Autostart, if enabled, causes any stream that is lost due to Net-X-Cmd or a Net-X-Code 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 (Net-X-Cmd) server. If this is disabled, the next level is ignored.
2. <group> if the client autostart is enabled, then the Net-X-Cmd will look at the group autostarts to determine if a group should be restarted when Net-X-Cmd or Net-X-Code gets closed. This allows the caller to set up 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=0&autostart=0

```
<request netxbaseip="192.168.50.100"
type="channel">
  <setting success="1">success</setting>
  <result>0</result>
  <resultstring>success</resultstring>
</request>
```

```
{
  "autostart": {
    "success": "1",
    "value": "success"
  },
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": ""
}
```

To set the group within a client, use

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

```
<request netxbaseip="192.168.50.100"
type="channel">
  <setting success="1">success</setting>
  <result>0</result>
  <resultstring>success</resultstring>
</request>
```

```
{
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "setting": {
    "success": "1",
    "value": "success"
  },
  "type": "channel"
```

	}

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 Net-X-Cmd 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=0&command=add&key=groupname

<request netxbaseip="192.168.50.100"
type="channel">
<setting success="1">success</setting>
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"add": {
	"success": "1",
	"value": "success"
	},
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=add"
	}

b. setting all channels with defined default values:

http://127.0.0.1:1080/netxjson?request=set&client=0&command=add&key=groupname&channel=-1&address=239.255.40.40&port=1234&type=.mov&protocol=UDP

<request netxbaseip="192.168.50.100" type="channel"> <setting success="1">success</setting> <result>0</result>	{ "message": "success", "netxbaseip": "192.168.50.100", "result": "0",

<resultstring>success</resultstring> </request>	"resultstring": "success", "type": "command=copy" }

c. setting one or all channels with specific values:

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

<request netxbaseip="192.168.50.100"
type="channel">
<setting success="1">success</setting>
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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=0&group=0&command=remove

<request netxbaseip="192.168.50.100"
type="command=remove">
<remove success="1">success</remove>
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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 optional 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 time code, 2 for day, 2 for month and 4 for year.

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

<request netxbaseip="192.168.50.100" type="channel">
<setting success="1">success</setting>
<result>0</result>
<resultstring>success</resultstring>
</request>


```
{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
}
```


delete

Delete one or more files on the file system

http://127.0.0.1:1080/netx?request=set&client=0&command=delete&file=E:/Record/netx/copy/source/temp.txt

http://127.0.0.1:1080/netx?request=set&client=9&command=delete&file=E:/Record/netx/copy/source/temp1.txt&file=E:/Record/netx/copy/source/temp 2.txt

<request netxbaseip="192.168.50.100" type="command=delete" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>


```
{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
}
```


proxymode

Set the proxy creation mode for all created files from stream source, sdi recording and clipping

http://127.0.0.1:1080/netx?request=set&client=0&group=0&channel=0&proxymode=1&proxyfiletype=197&proxycompressiontype=163514

8593&proxydatarate=5000

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

- 0 disables proxy generation
- 1 creates proxies when a file is written to
- 2 creates a proxy when a file is read from
- 3 always creates a proxy

proxyfiletype

Set the proxy file type

http://127.0.0.1:1080/netx?request=set&client=0&channel=0&group=1&proxymode=1&proxyfiletype=197&proxycompressiontype=1635148593&proxydatarate=5000

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

- 197 Standard MP4
- 210 Multirate HLS (Apple live video)
- 211 Multirate DASH (Fragmented MP4 live video)
- 201 MXF AS-02

proxycompression

Set the proxy compression. Currently must be h.264 = 1635148593

http://127.0.0.1:1080/netx?request=set&client=0&channel=0&group=0&proxymode=1&proxyfiletype=197&proxycompressiontype=1635148593&proxydatarate=5000

<request netxbaseip="192.168.50.100" type="channel" message="success">	{ "message": "success",

<result>0</result> <resultstring>success</resultstring> </request>	"netxbaseip": "192.168.50.100", "result": "0", "resultstring": "success", "type": "command=copy" }

proxydatarate

Set the proxy datarate in kilo bits per second (eg. 5000 = 5 megabits)

http://127.0.0.1:1080/netx?request=set&client=0&channel=0&group=0&proxymode=1&proxyfiletype=197&proxycompressiontype=1635148593&proxydatarate=5000

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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=0&group=0&channel=0&tsenable=1

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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=0&group=0&channel=0&tsdir=/tmp/capture

<request netxbaseip="192.168.50.100"
type="channel" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>

{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
}

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=0&group=0&channel=0&tsfile=hls.ts

<request netxbaseip="192.168.50.100"
type="channel" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>

{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"
}

SETTINGS

Requires 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=0&group=0&key=newgroupname

<request netxbaseip="192.168.50.100"
type="channel" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>

{
  "message": "success",
  "netxbaseip": "192.168.50.100",
  "result": "0",
  "resultstring": "success",
  "type": "command=copy"

	}

- name - File name

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

<request netxbaseip="192.168.50.100"
type="channel" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>
	{
	  "message": "success",
	  "netxbaseip": "192.168.50.100",
	  "result": "0”,
	  "resultstring": "success",
	  "type": "command=copy"
	}

- address - Source stream address

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

<request netxbaseip="192.168.50.100"
type="channel" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>
	{
	  "message": "success",
	  "netxbaseip": "192.168.50.100",
	  "result": "0",
	  "resultstring": "success",
	  "type": "command=copy"
	}

- port - Source stream port

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

<request netxbaseip="192.168.50.100"
type="channel" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>
	{
	  "message": "success",
	  "netxbaseip": "192.168.50.100",
	  "result": "0",
	  "resultstring": "success",
	  "type": "command=copy"
	}

- protocol - Network capture protocol: udp, rtp

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

<request netxbaseip="192.168.50.100"
type="channel" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>
	{
	  "message": "success",
	  "netxbaseip": "192.168.50.100",
	  "result": "0",
	  "resultstring": "success",
	  "type": "command=copy"

	}

- directory - File directory

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

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

- rectc – Time code 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=0&group=0&channel=1&rectc=1

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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

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

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

- 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=0&group=0&channel=1&previewenable=1

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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

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

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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=0&group=0&channel=1&name=myname&address=239.255.40.44&port=5004&protocol=rtp&directory=/Volumes/capturedrive/current/&type=.mp4

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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

<request netxbaseip="192.168.50.100" type="channel" message="success">
<result>0</result>
<resultstring>success</resultstring>
</request>
	{
	"message": "success",
	"netxbaseip": "192.168.50.100",
	"result": "0",
	"resultstring": "success",
	"type": "command=copy"
	}

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

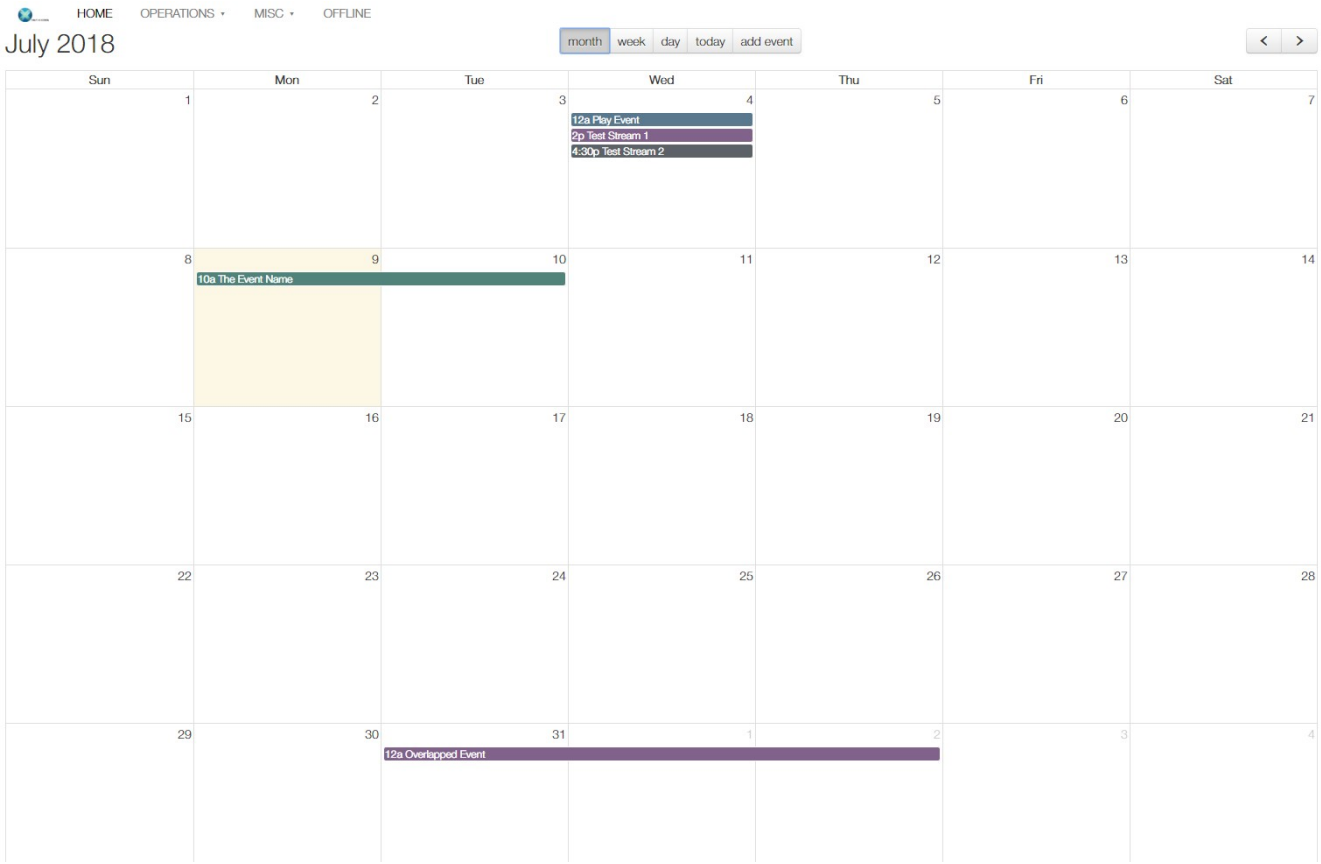
NOTE: it is recommended to update all statuses for any client that has had a command performed on it.

Scheduling

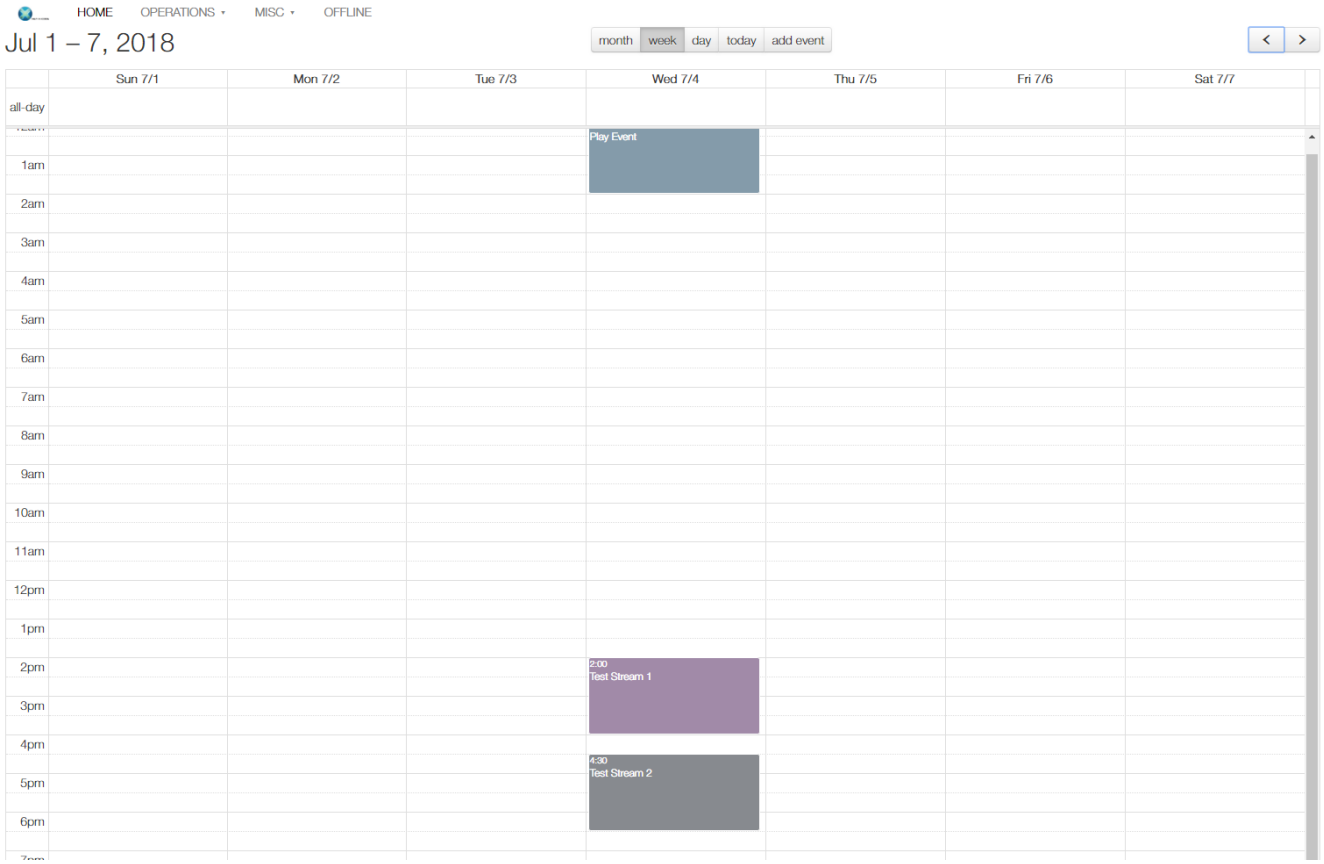
Net-X-Code includes a scheduling engine that allows you to set up events in the future that will be automatically triggered. These events may be set up in the HTML calendar provided with Net-X-Base, or can be set up/retrieved via the REST api specified here.

The calendar provides Month, Week, and Day Views. Clicking on the Month, Week, or Day tabs lets you switch between the views.

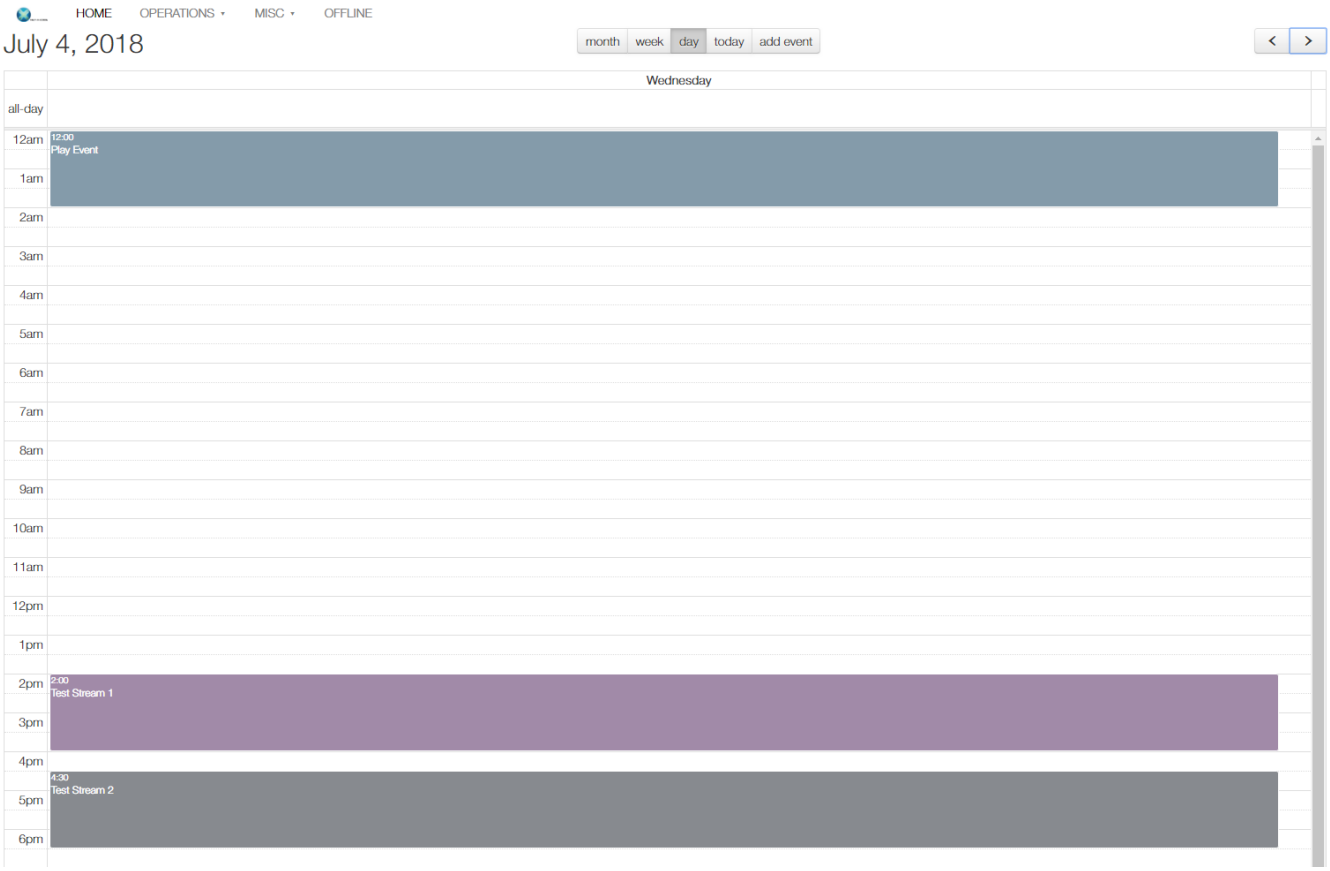
Month View


HOME
OPERATIONS
MISC
OFFLINE
July 2018
month
week
day
today
add event
Sun
Mon
Tue
Wed
Thu
Fri
Sat
12a Play Event
2p Test Stream 1
4:30p Test Stream 2
10a The Event Name
12a Overlapped Event

Week View


HOME
OPERATIONS
MISC
OFFLINE
Jul 1 – 7, 2018
month
week
day
today
add event
Sun 7/1
Mon 7/2
Tue 7/3
Wed 7/4
Thu 7/5
Fri 7/6
Sat 7/7
all-day
Play Event
1am
2am
3am
4am
5am
6am
7am
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
2:00
Test Stream 1
4:30
Test Stream 2

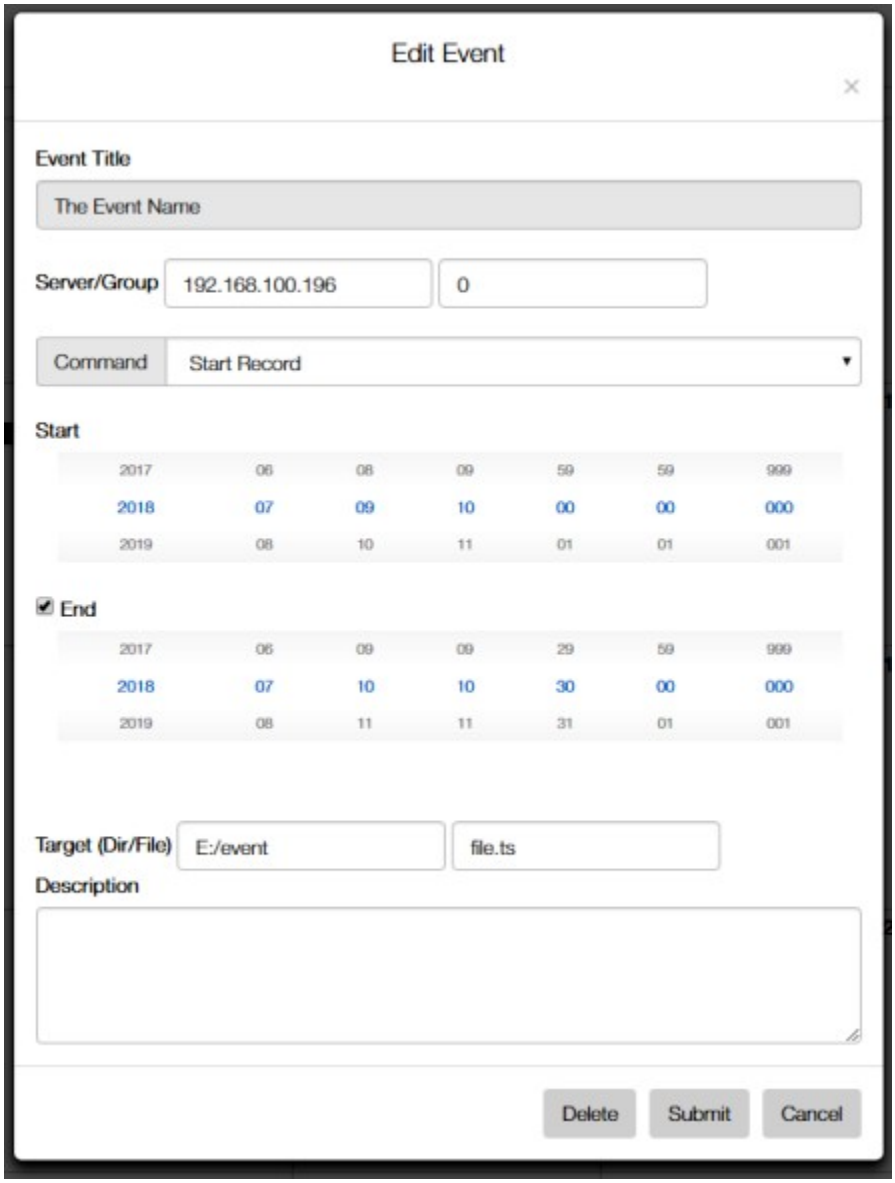
Day View



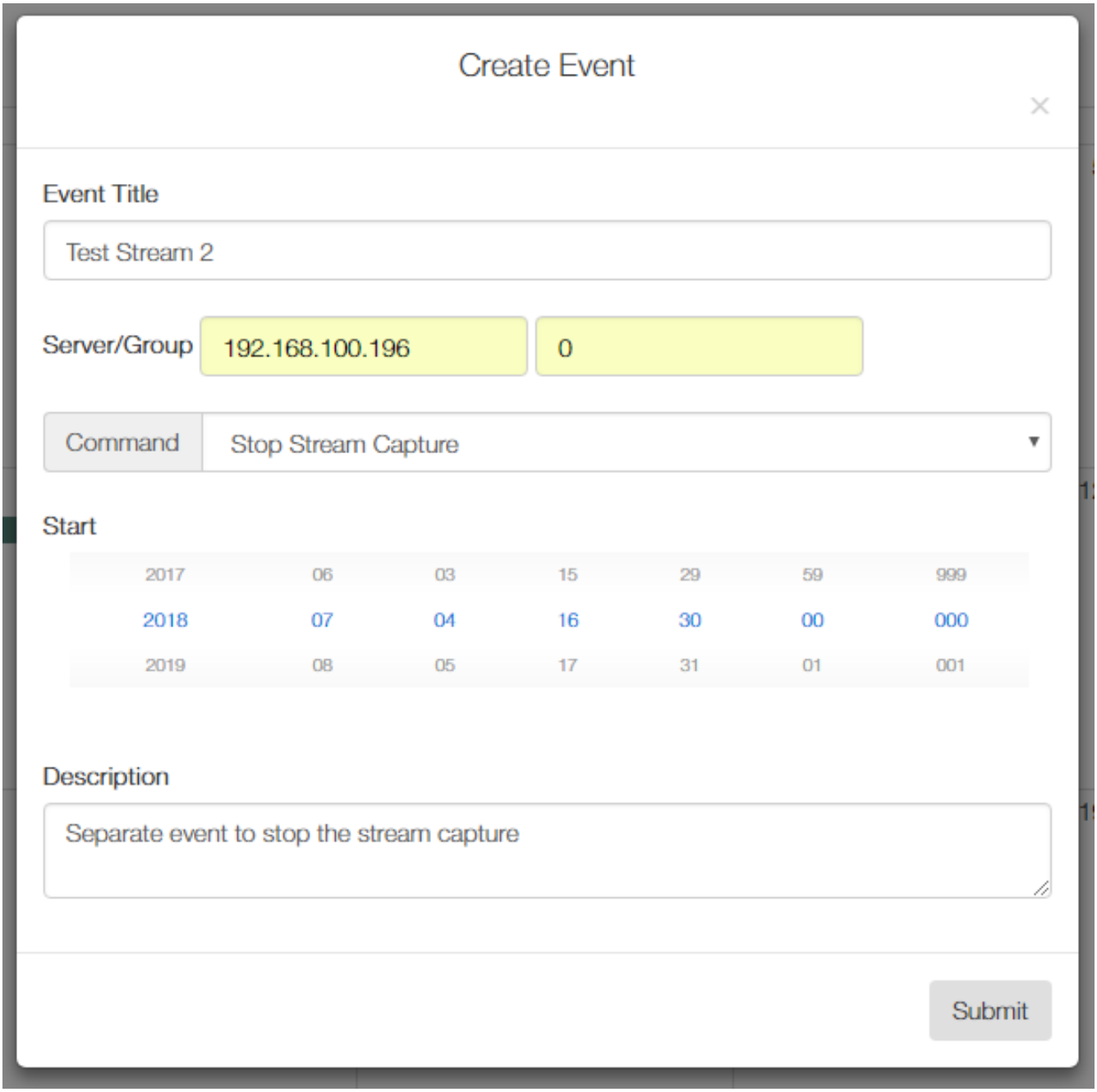
Pressing the Today tab returns the calendar to the present date.

Create Event

Pressing the Create Event tab opens the event creation dialog. Specific events can include a start and end time, or a start and duration.

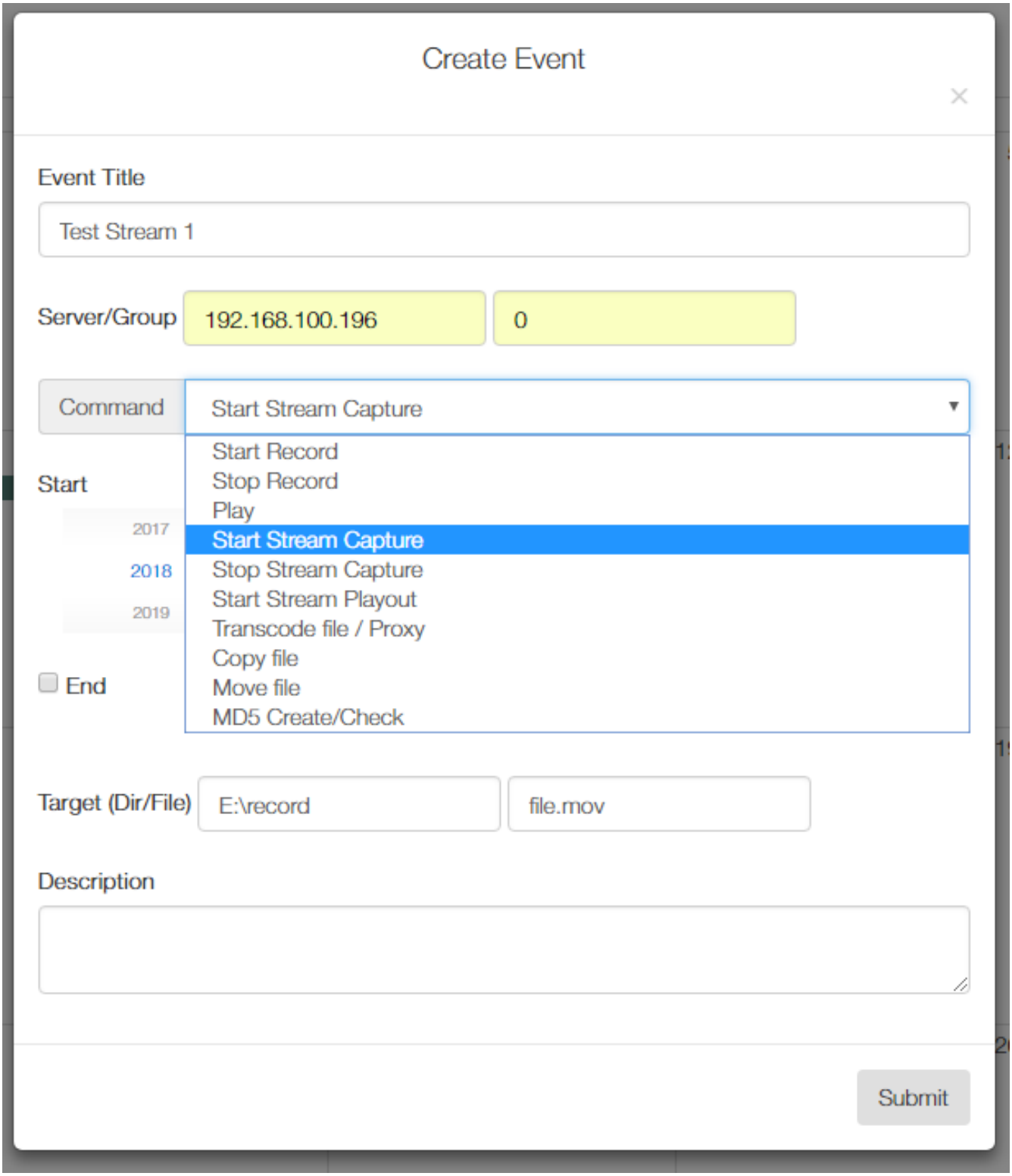


Or, the event can specify a start, and a separate event can be entered to end the process (such as a capture).



The event title can be entered, and the server and group specified.

There is a Command pulldown menu to set which command is used in the event.



The supported event types include:

- 0 - Start Record (SDI)
- 1 - Stop Record (SDI)
- 2 - Play (SDI)
- 3 - Start Stream Capture (IP)
- 4- Stop Stream Capture (IP)
- 5 - Start Stream Playout (IP)
- 6 - Transcode File/Proxy (File)
- 7- Copy (File)
- 8 - Move (File)
- 9 - MD5 Create/Check (File)

A scheduled event can contain the following parts:

- Server IP Address (server)
- Group Number or IP (group)
- Start Time/Date (start)
- End Time/Date (end)
- Type (type)
- Description (desc)
- Extended values depending on event (json)

The JSON format can include the following types

- Source directory (srcdir)
- Source file (srcfile)
- Target directory (trgdir)
- Target file (trgfile)
- Play mode (mode)
- Conversion profile (profile)
- MD5 directory (MD5Dir)
- MD4 file (MD5File)

Generate a new

http://127.0.0.1:1080/netxevent?request=add&title=<thetitle>&server=<serverip>&group=#&start=YYYY-MM-DDTHH-mm-ss&end=YYYY-MM-DDTHH-mm-ss&type=<type>&desc=<description>&json=<extended parameters>

For example, to add an event
http://127.0.0.1:1080/netxevent?
request=add&title=NewEVENT&server=192.168.50.100&group=0&start=2019-02-14T12-30-00&end=2019-02-14T14-30-00&type=0&desc=ATestEventToREcord

```
{
    "result": 1,
    "resultstring": "event added"
}
```

To update an existing event, use the same command but append the event's id number at the end (&id=#)

http://127.0.0.1:1080/netxevent?request=add&title=<thetitle>&server=<serverip>&group=#&start=YYYY-MM-DDTHH-mm-ss&end=YYYY-MM-DDTHH-mm-ss&type=<type>&desc=<description>&json=<extended parameters>&id=<#>

```
{
    "result": 1,
    "resultstring": "event updated"
}
```

Get a list of events

http://127.0.0.1:1080/netxevent?request=get&start=YYYY-MM-DDTHH-mm-ss&end=YYYY-MM-DDTHH-mm-ss

will return a JSON formatted string like this


```
{
   "events": [
      {
         "desc": "First event",
         "group": "0",
         "id": 1,
         "jsonInfo": "{\"params\":[{\"name\":\"srcdir\",\"value\":\"/mnt/record/\"},
{\"name\":\"srcfile\",\"value\":\"Test.mxf\"}]}",
         "server": "0",
         "start": "2018-06-29T10:00:00.000",
         "title": "Test",
         "type": "0"
      },
      {
         "desc": "New stream tomorrow",
         "group": "0",
         "id": 2,
         "jsonInfo": "{\"params\":[]}",
         "server": "0",
         "start": "2018-06-30T10:00:00.000",
         "title": "Tomorrow",
         "type": "3"
      }
   ]
}
```

QC Processing

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

To compare two files for basic video frame rate, audio channels, length of media, timecode, closed captioning and metadata. This would be used to quickly check a proxy file generated from a source file, where the audio/video compression and size may be different, but the basic parameters should be the same.

```
http://127.0.0.1:1080/netx?request=set&client=0&group=0&channel=0&compare&file1=d:\record\
first_file.mxf&file2=d:\record\first_file.mxf
```

```
<request netxbaseip="192.168.50.100" type="channel" message="success">
  <result>0</result>
  <resultstring>success</resultstring>
</request>
```

The compare will run, and then you can get the results of the comparison by calling
http://

To analyze the audio and video in a file (single ended/no reference)
http://127.0.0.1:1080/netx?request=set&client=0&group=0&channel=0&analyze&sourcefile=/mnt/storage/source.mxf&qcfile=/mnt/storage/source.mxf.qc.db

To validate a file against a profile
http://127.0.0.1:1080/netx?request=set&client=0&group=0&channel=0&validate&sourcefile=/mnt/storage/source.mxf&profile=XDCam

To compare a file against another file for basic parameters, for a proxy from a high res for example:
http://127.0.0.1:1080/netx?request=set&client=0&group=0&channel=0&compare&sourcefile=/mnt/storage/source.mxf&targetfile=/mnt/proxy/source.mxf.mp4

Watch Folders

Up to 100 watch folders can be set up to trigger other Net-X-Code processes. These commands will take the IDs between -300 and -400. The primary watchprofile types are

- analyze – do a full reference analysis of two files (psnr, ssim, etc)
- qcfile – check a file against a specific profile, or another file

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

```
<request netxbaseip="192.168.50.100" type="channel">
  <setting success="1">success</setting>
  <result>0</result>
```

```
  <resultstring>success</resultstring>
</request>
```

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

```
<request netxbaseip="192.168.50.100" type="channel">
  <setting success="1">success</setting>
  <result>0</result>
  <resultstring>success</resultstring>
</request>
```

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

```
<request netxbaseip="192.168.50.100" type="channel">
  <setting success="1">success</setting>
  <result>0</result>
  <resultstring>success</resultstring>
</request>
```

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=0&group=0&command=start
```

```
<request netxbaseip="192.168.50.100" type="command=start">
  <start success="1">success</start>
  <result>0</result>
  <resultstring>success</resultstring>
</request>
```

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

- callstate (command) - Query the system to see if it has been processed

```
http://127.0.0.1:1080/netx?request=callstate&client=&group=0&start=
```

The response 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>0</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=0&group=0&channel=0&name=Channel1
```

```
<request netxbaseip="192.168.50.100" type="channel">
  <setting success="1">success</setting>
  <result>0</result>
  <resultstring>success</resultstring>
</request>
```

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

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

```
<client address="192.168.100.229">
  <group index="0">
    <channel index="0">
      <param>name</param>
      <value>Channel1</value>
      <result>-1</result>
      <code>-1</code>
      <description>unspecified error</description>
    </channel>
  </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=0&group=0&errors=all
```

(The client need not be specified for this)

```
http://127.0.0.1:1080/netx?request=callstate&errors=all
```

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>0</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>0</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 user's 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=0

<request message="shutting down NetXBase server">
  <result>0</result>
  <resultstring>close - shutting down netxbase

{
  "message": "shutting down NetXBase server",
  "result": "0",

server</resultstring> </request>	"resultstring": "close - shutting down netxbase server" }

http://127.0.0.1:1080/netx?request=restart&client=0

<request message="restarting NetXBase server">
<result>0</result>
<resultstring>restart - restarting netxbase server</resultstring>
</request>
	{
	"message": "restarting NetXBase server",
	"result": "0",
	"resultstring": "restart - restarting netxbase server"
	}

Command – Thumbnail/JPEG/Picons

Net-X-Base 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=0&group=0&channel=0&value=preview

<?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)
	{
	"address": "192.168.100.229",
	"autostart": "0",
	"group0": {
	"autostart": "1",
	"channel0": {
	"index": "0",

</preview>
</channel>
</group>
</request>
	"preview": {
	"timestamp": "169586491",
	"value": "/9j/ . . . ="
	}
	},
	"index": "0",
	"key": "DemoCapture",
	"version": "v5.1.0.13"
	},
	"groups": "4",
	"netxbaseip": "192.168.50.100",
	"result": "1",
	"resultstring": "group returned",
	"version": "v7.1.0.3"
	}

The data returned is base 64 encoded JPEG data located in the following XML/JSON structure (in the example above the … is the data truncated).

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=0&group=0&channel=0&value=mpreview
```

Returns an actual JPEG image

The data returned is raw JPEG, marked in the HTML header as mime/jpeg. Note, this command does not support ‘netxjson’.

Create JPEGs On Disk

Net-X-Copy can be used to create a JPEG image from a file (live or prerecorded) at any valid frame to any available location, from the Net-X-Code 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=d:/Record/second_file.rtin&pidst=d:/Record/
second_file.picon.jpg&pisize=10&piframe=25
```

Returns and actual JPEG image.

Create a JPEG picon to a target name. This command is not supported in ‘netxjson’.

```
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 Net-X-Base. Net-X-Base 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 Net-X-Base should be set up, so that it does not interfere with the other operations Net-X-Base is handling for capture and clipping.

Picon

Create a JPEG picon and return it as a mime/jpeg (Note, this command is not supported with netxjson).

```
http://127.0.0.1:1080/netx?request=picon&pisrc=d:/Record/
amberfin_0000572516.rtin&pidst=&pisize=10&piframe=250
```


NBA
WEDNESDAY

NetXSDI – MediaCMD

Start NetXSdi.exe
http://127.0.0.1:1080/netx?request=set&client=0&group=sdi&sdistart&channel=0

Stop NetXSdi.exe
http://127.0.0.1:1080/netx?request=set&client=0&group=sdi&sdistop&channel=0

Get status:
http://127.0.0.1:1080/netx?request=set&client=0&group=sdi&sdistatus&channel=0

Process MediaCmd
http://127.0.0.1:1080/netx?request=set&client=0&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=0&group=sdi&sdicmd&channel=0&play&speed=65520

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

Play at normal speed.

PlayAtSpeed

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

Returns 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=0&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=0&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, <Position> sub-element (text)

GetCurState

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

Internal Channels Only. Do not use yet.

Blank

http://localhost:1080/netx?request=set&client=0&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=0&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=0&group=sdi&sdicmd&channel=0&Trim&position=lPosition&st**art=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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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 method 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=0&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 an HTTPObject to send it through. Here is an 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=0&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=0&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 commands 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=0&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=0&group=sdi&sdicmd& (following the question mark) must 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

Basic Command (sdicmd)

netx?request=set&client=0&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 minute (specified as 1800 frames, non drop frame time code)
- 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 – string 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 format
- 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=0&group=sdi&sdicmd&play
– Normal play

netx?request=set&client=0&group=sdi&sdicmd&play&speed=32760
– Play at 50% forward speed

netx?request=set&client=0&group=sdi&sdicmd&play&speed=-65520
– Play at 100% reverse play speed

netx?request=set&client=0&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=0&group=sdi&sdicmd&pause
– Pause the channel

netx?request=set&client=0&group=sdi&sdicmd&stop
– Stop (E to E passthrough) the channel

netx?request=set&client=0&group=sdi&sdicmd&pause&position=1:00:00
– Seek to one minute

netx?request=set&client=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&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=0&group=sdi&sdigetstatus
– Returns an XML package including state, speed, position start and end points.

http://localhost:1080/netx?request=set&client=0&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=0&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=0&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=0&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 timeline, 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=0&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=0&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=0&group=sdi&sdinextdirentry&\&\Record
Returns: <locator>\OfflineMedia</locator>
http://localhost:1080/netx?request=set&client=0&group=sdi&sdinextdirentry&\&\OfflineMedia
Returns: <locator>\LocalMedia</locator>
http://localhost:1080/netx?request=set&client=0&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=0&group=sdi&sdinextdirentry&\Record\
Returns: <locator>\Record\..</locator>
http://localhost:1080/netx?request=set&client=0&group=sdi&sdinextdirentry&\Record\&\Record\..
Returns: <locator>\Record\Test.wav</locator>
http://localhost:1080/netx?request=set&client=0&group=sdi&sdinextdirentry&\Record\&\Record\
Test.wav
Returns: <locator>\Record\Test.avi</locator>
http://localhost:1080/netx?request=set&client=0&group=sdi&sdinextdirentry&\Record\&\Record\
Test.avi
Returns: <locator>END OF LIST</locator>

http://localhost:1080/netx?request=set&client=0&group=sdi&sdifileinfo
– Used to retrieve information on a specific file.

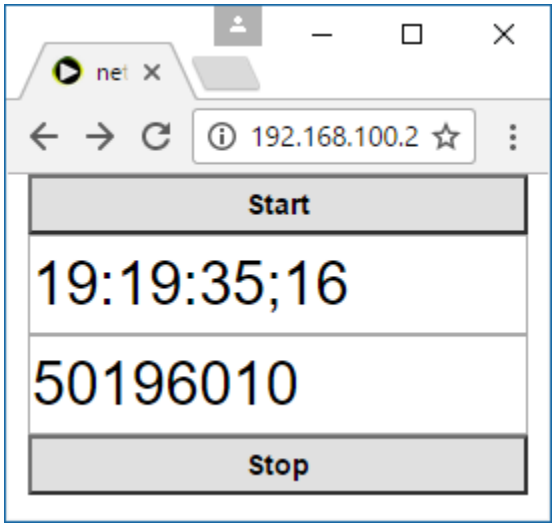
http://localhost:1080/netx?request=set&client=0&group=sdi&sdigeterrormsg&#
– Used to return one error message from the current list. The first call will not include an error number (just sdigeterrormsg). This will return an ErrorNumber to use to get the next message (sdigeterrormsg&202 for instance), as will each subsequent call. When all the error messages have been returned, it will return an ErrorNumber of -1.

Net-X-Time-Code

The Net-X-Time-Code server is a server that captures a UDP or RTP stream of time code metadata and converts it to an AJAX compatible HTTP server. The server records the latest time code and userbits, which can then be viewed within the default HTML page, or requested via AJAX requests. The base command for Net-X-Time-Code is “/netxtc?”. The base address is your machine's IP at port 1080. E.g.:

http://127.0.0.1:1080

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



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 – time code
- ub - userbits
- frames – time code in frames
- ms - milliseconds

Start Capturing Time Code

Start capturing time code from the time code stream source

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

Stop Capturing Time Code

Stop capturing time code from the time code 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 file made after a copy
-p cardinfo - Return XML/JSON info on clips on a camera card
-p <profile> - Profile to use. Current profiles include:
> copy - copy the whole file
> wrap - re wrap file or part of a file
> index - create an RTIndex for a file
> getCopyInOut - get the extents required for a pfr, or use them with a temp file
> mp3-128kbps - Audio MP3 file
> mov-YCbCr8Bit - QuickTime MOV 8 bit uncompressed YcbCr file
> mov-dvcprohd - QuicktTime MOV DVCPro HD (1080/720)
> mp4-h264 - MPEG-4 h264 AAC Audio
> mxf-xdcam-720p - True XDCam MXF 8 channel audio
> mxf-dvcprohd-720p - MXF DVCPro HD 720p
> mxf-xdcam-1080i - True XDCam MXF 1080i 8 channel audio
> mxf-dvcprohd-1080i - MXF DVCPro HD 1080i 29/25 fps
> mxf-OP1a-MPEG - OpenMXF XDCam MPEG-2 16 channel audio
> mxf-OP1a-h264 - MXF h.264
> mxf-OP1a-HDF - MXF MPEG-2 HDF Standard
> mxf-as-11-sd-pal-dpp - MXF AS-11 SD PAL DPP
> mxf-as-11-sd-ntsc-dpp - MXF AS-11 SD NTSC DPP
> mxf-as-11-hd-dpp - MXF DPP AS-11 AVCi HD
> mov-proreshq - QuickTime MOV ProRes HQ
> mov-proreslt - QuickTime MOV ProRes LT

> mov-prores422 - QuickTime MOV ProRes 422
> mov-prores444 - QuickTime MOV ProRes 444(4)
> scaledown2000k - MP4 264 960x540, 2mbs, AAC
> scaledown500k - MP4 264 480x272, 0.5mbs, AAC
> 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
> TS-MPEG2 - MPEG-2 4:2:0/passthrough transport stream
> TS-h264 - h.264 4:2:0/passthrough transport stream
> 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
> aces - ACES image files
> dnxhd-mxf-720p - DNxHD 720p 50, 59, 60
> dnxhd-mxf-1080p - DNxHD 1080p 25, 29
> dnxhd-mxf-1080i - DNxHD 1080i 25, 29
> dnxhr-mxf-10-hq - DNxHR High Quality 10 bit
> dnxhr-mxf-8-hq - DNxHR High Quality 8 bit
> dnxhr-mxf-sq - DNxHR Standard Quality
> dnxhr-mxf-lq - DNxHR Low Quality

-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)
> mov - QuickTime MOV

-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)
-width <width> - output width of the video (only for arbitrary codecs like h264, hevc and prores)
-height <height> - output height of the video (in not set, the input size or codec size will be used)
-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
-dest <folder> - 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
-stereo - force a stereo pair (mix down) output
-aroute <12345678> - route channels to specific outputs
-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
-alignment - Alignment value for any temporary partial files, for GetCopyInOut profile
-m - Save the metadata in an XMP file
-fg - force the GUI on
-fc - force command line
NOTE: the parameters in [square brackets] are optional.

For JPEG picons
-pisrc <file-to-picon> - source fot the
-pidst <where-to-make-picon> - target folder and name
-pisize <size> - size of picon, 100%
-piframe <frame-offset-into-file> - frame to use to make the picon
-piskip <number-of-frames-to-skip> - if set, make a picon of each frame at this distance for the whole file
-width <width> - output width of the picon image
-ci -root -level - output level denotes verbosity

h26x Profiles:
10, 11, 12, 13, 20, 21, 22, 30, 31, 32, 40, 41, 42, 50, 51, 52

h26x Levels:
66 – Baseline
77 – Main
88 – Extended
100 – High
110 – High 10
122 – High 4:2:2
144 – High 4:4:4
166 – Advanced 4:4:4 Intra
188 – Advanced 4:4:4

NOTE: the parameters in [square brackets] are optional.
For further examples see:

https://www.drastic.tv/support-59/supporttipstechnical/295-netxcopy-pfr-clip-metadata
https://www.drastic.tv/support-59/supporttipstechnical/296-netxcopy-pfr-command-line
https://www.drastic.tv/support-59/supporttipstechnical/298-netxcopy-metadata-extraction
https://www.drastic.tv/support-59/supporttipstechnical/299-netxcopy-metadata-cardinfo

Discontinuity Sources In Net-X-Code

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

- PID continuity counter discontinuity
  - video
  - audio
  - dolby
  - timecode
  - metadata
  - metadata cc
  - metadata tc
  - video pts
  - audio pts
- Time code 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 upstream. The one exception is the time code discontinuity. Because the LBR/ABR are muxed, missing video/audio frames may not show up as discontinuities, as they occurred pre mux. The only indication (other than broken video/audio decodes) is that the time code jumps, so it is important to treat this as a full discontinuity.

Error Returns

#define netxErrorNone 0
No error, success. Normal response.

#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 exist.

#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 – Net-X-Copy

0 – Success
1 – unable to create file (EPERM)
2 – no such file or directory (ENOENT).
5 – read or write error (EIO)
6 – no such device or address (ENXIO)
12 – unable to allocate memory (ENOMEM)
22 – invalid argument (EINVAL)
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 than end position
261 – unable to set target file information
261 – DMF get bytes error
275 – general timecode error (deprecated)
276 – in timecode greater then out timecode
277 – tcin before first timecode in file
278 – timecode is invalid
279 – tcout after last timecode in file
290 – in timecode not in file range
291 – out timecode not in file range
1155 – insufficient permissions to use dmPut

Error Returns – Net-X-SDI

These are Microsoft style returns, with error/warning/info bits, and then areas within the DWORD/HRESULT return for source, area and type.

```
#define ERR_SEVERITY_SUCCESS             0x00000000UL
#define ERR_SEVERITY_INFORMATIONAL       0x40000000UL
#define ERR_SEVERITY_WARNING             0x80000000UL
#define ERR_SEVERITY_ERROR               0xC0000000UL

#define ERR_CUSTOMER_CODE                0x20000000UL

#define FACILITY_AVHAL                   0x08010000UL
#define FACILITY_MEDIAFILE               0x08020000UL
#define FACILITY_VVW                     0x08030000UL
#define FACILITY_VVWEXT                  0x08040000UL
#define FACILITY_VVWINT                  0x08050000UL
#define FACILITY_VVWNET                  0x08060000UL
#define FACILITY_VVWCTL                  0x08070000UL
#define FACILITY_TCS                     0x08080000UL
#define FACILITY_CLIP                    0x08090000UL
#define FACILITY_UTIL                    0x080A0000UL
#define FACILITY_MEM                     0x08100000UL
#define FACILITY_MEDIABASE               0x08110000UL
#define FACILITY_DRASTIC                 0x0FFF0000UL

#define DEFAULT_FACILITY                 FACILITY_DRASTIC

//
// Generic sub areas for errors to occur
//
#define VVWSUB_HANDLE                              0x0000
#define VVWSUB_CHANNEL                             0x1000
#define VVWSUB_MEMORY                              0x2000
#define VVWSUB_FILE                                0x3000
#define VVWSUB_READ                                0x4000
#define VVWSUB_WRITE                               0x5000
#define VVWSUB_HARDWARE                            0x6000
#define VVWSUB_MEDIACMD                            0x7000
#define VVWSUB_DFRAME                              0x8000

#define VVWSUB_INTERNAL                            0xE000
#define VVWSUB_SOFTWARE                            0xF000

//
// Specific errors
//
enum {
       VVWERR_DLIST,        // Handling dlist
       0             0x00
       VVWERR_DOESNTEXIST,  // Handles, pointers, drivers
       VVWERR_UNKNOWN,      // Channel
       VVWERR_INVALID,      // Handles, pointers
       VVWERR_BAD,          // Handles, pointers, mediacmd, dframe
       VVWERR_NULL,         // Pointer
       VVWERR_SELECTION,    // Channel select, in/out, clip
       VVWERR_LOCAL,        // Memory
```

```
        VVWERR_GLOBAL,       // Memory
        VVWERR_SHARED,       // Memory, resource
        VVWERR_FREE,         // Memory, resource
        10           0x0A
        VVWERR_TOOSMALL,     // Memory area to small
        VVWERR_OPEN,         // File open
        VVWERR_END,          // End of File
        VVWERR_READ,         // File, memory
        VVWERR_WRITE,        // File, memory
        VVWERR_CLOSE,        // File
                     16           0x10
        VVWERR_NOTFOUND,     // File, resource
        VVWERR_NOTOPEN,      // File
        VVWERR_PARAM,        // Calling
        VVWERR_PARAM1,       // Calling
               20
        VVWERR_PARAM2,       // Calling
        VVWERR_PARAM3,       // Calling
        VVWERR_PARAM4,       // Calling
        VVWERR_PARAM5,       // Calling
        VVWERR_INDIRECTION,  // Recursion
        VVWERR_ADD,          // Adding to list, tcspace, clipspace
        VVWERR_DEL,          // Deleting/Removing from list, tcspace, clipspace
        VVWERR_FIND,         // Finding
        VVWERR_CALLBACK,     // User error
        VVWERR_FILEEXISTS,   // Creating existing file                                    30
        VVWERR_LOCKED,       // Lock exclusive by someone else
        VVWERR_NOALLOC,      // No Allocation available
               0x20
        VVWERR_EXCEPTION,    // Caused and exception
        VVWERR_SEEK,         // Unable to seek
        VVWERR_NOT_READY,    // File has not yet finished write
        VVWERR_NOT_ENOUGH_DATA, // File has not yet finished write
        VVWERR_NOT_SUPPORTED, // Not a supported operation
        VVWERR_TOLARGE,      // Too many too big
        VVWERR_NONEAVAIL,    // A request param was not available
        VVWERR_BAD_FORMAT,   // Item format not correct                                   40
               0x28
        VVWERR_OUTOFRANGE,
// Sentinal
        __VVWERR_SENTINAL = 0xFFF   // last possible error
};
#define VVWERR_MASK  0xFFFFUL
```

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 and its source, and is 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”
  - “meta tc”
  - “meta cc”
  - “metadata”
  - “audio pts”
  - “video pts”
- 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 Net-X-Copy

Net-X-Copy 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 time code 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

Net-X-Base (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 Net-X-Cmd/Base to communicate on
multicast – string – the multicast address for Net-X-Base and Net-X-Cmd to use to find each other (default 230.7.7.7)

Net-X-Cmd

Geometry – string – position of UI
forcegui – dword – optional, force the GUI on (1) or off (0)
commandip – string – IP address of the interface for Net-X-Cmd/Base to communicate on
videoip – string – IP address of the interface for Net-X-Code to capture network video streams on
autostart – if true, start capture of any enabled groups as soon as Net-X-Cmd 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 – if true, start capture of this group as soon as Net-X-Cmd 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 inline to mp4
mp4proxyfcc – the mp4 codec to use for the proxy
mp4proxymode – if 1 then use the proxy/downconvert
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, ts-hls, mxf, mxf-as02, mxf-xdcam, mxf-open, mp4, mp4-fmp4, mp4-dash, mov

Net-X-Copy

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

Net-X-Time-Code

address – group address of the time code stream
port – port for the time code stream
protocol – currently UDP or RTP

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

General Config: config.xml

The config.xml file contains the general configuration for the system, including codec, plugin and all the SDI channel information. This file can be found here:

Windows:

C:\ProgramData\Drastic\config.xml
Alt - C:\Documents and Settings\Drastic\config.xml

Linux:

/etc/Drastic/config.xml
Alt - /Home//Drastic/config.xml

OS-X:

/Library/Applications Support/Drastic/config.xml
Alt - /Home//Drastic/config.xml

Generally, the alt location is for users without administrator/super-user credentials. The ideal location is in the system area, so it is shared by all users, but per user config is also supported for machines that are locked down.

Automatic Proxy Generation Setup

Net-X-Code can create a low res proxy for streams captured from IP or SDI. Normally the proxy will be an MP4, but various types of multiple bitrate files are also supported as MP4, HLS/TS and DASH fMP4. To set up the proxy that will be generated for every stream that is recorded or converted, you will need to modify **config.xml**, in the location described above.
In config.xml, find the area
<DrasticConfig>
  <VVW>
    <Internal0>
Within the <Internal0> tag, there are four elements that control the automatic proxy:
<proxymode Value="1" Type="2" />
 0 – no automatic proxy

1 – create proxy on record/write
2 – create proxy on playback
3 – create proxy on read and write

<proxyfiletype Value="211" Type="2" />
197 – create MP4 proxies
210 – create multirate HLS/TS proxies
211 – create multirate MP4-Dash proxies
(note: multirate proxies can have just one rate/size/etc)

<proxycompressiontype Value="1635148593" Type="2" />
1635148593 – (avc1) h.264 compression

<proxydatarate Value="3000" Type="2" />
3000 – 3 megabits. Setting is in kilobits.

To setup one or more multirate TS/HLS or MP4/DASH proxies, see the multirate setup below.

Multirate TS/HLS and MP4/DASH setup

The multirate configuration is contained in the **config.xml** file, which can be located with the information described above. There is only one multirate setup, which will be applied to HLS or DASH, depending what file is specified to be created. The configuration keys can be found here:
<DrasticConfig>
<MediaReactor>
<PlugIns>
<MultiRate>
The first setting sets how many files will be created. Under the MuliRate key, there will be
<TotalMultiRateFiles Value=”3” Type=”2” />
The value contains the number of multirate files to create. In this case files are being recorded. Each multirate file will require a sub key of File# (eg, <File0>, <File1>, <File2>) under the <MultiRate> key. Within each <File#> key, the following settings are supported:

<Width Value="512" Type="2" />
The width of the file to be created. Should be a multiple of 8.

<Height Value="288" Type="2" />
The height to be created. Should be a multiple of 16, or a standard value like 1080

<BitRateAvg Value="3000" Type="2" />
The target average bit rate in kilobits per second

<BitRatePeak Value="3200" Type="2" />
The peak bit rate in kilobits per second

<Profile Value="77" Type="2" />
The h.264 profile to use while encoding (optional)

<Level Value="30" Type="2" />
The h.264 level to use while recording (optional)

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 Net-X-Code uses include:

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

7630 – TCP – Net-X-Cmd/Net-X-Code server command port
57500-57XXX – UDP = Communications server port
58500 – UDP – Net-X-Base multicast server port
58500 – TCP – Net-X-Cmd→Net-X-Base communication port (outgoing)
59000->50### – TCP – Net-X-Base<-Net-X-Cmd communications port, where ### is the max number of channels (incoming)

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

230.7.7.7 – Net-X-Code system discovery multicast address

The individual streams being captured or transmitted also use multicast 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:

- Net-X-Base hosts multicast at 230.7.7.7:58500 UDP for Net-X-Cmds to connect to
- Net-X-Cmd joins 230.7.7.7:(57500 +offset) and sends a message to Net-X-Base
- Net-X-Base responds back through multicast
- Net-X-Cmd receives the message and gets the Net-X-Base IP from the messages
- Net-X-Cmd makes a TCP connect from Net-X-Cmd <ip>:58500 to Net-X-Base <baseip>:59000-50###
- Net-X-Code is spawned, it connects back to the local Net-X-Cmd 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 Net-X-Code is working properly on your system. For most systems, 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 Net-X-Code, 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
```

This manual has been compiled to assist the user in their experience using the Net-X-Code API. It is believed to be correct at the time of writing, and every effort has been made to provide accurate and useful information. Any errors that may have crept in are unintentional and will hopefully be purged in a future revision of this document. We welcome your feedback.

Drastic Technologies Ltd
523 The Queensway, Suite 102
Toronto, ON, M8Y 1J7
Canada
P (416) 255 5636
F (416) 255 8780