Nex Gen Media Server (NGMS)

v1.6.4

User Guide

Contents

1. Introduction

Nex Gen Media Server (NGMS) is a real-time streaming media server with capabilities to stream live content, record, and host a complete media library. NGMS can be used as part of a live broadcasting system, cloud based media distribution infrastructure, video conferencing server, or directly embedded in a set-top box or mobile device. The Media Server consists of the following components.

NGMS – Stand-alone Media Server Component used to handle a single live broadcast or media source which can be disseminated to multiple clients. NGMS can be started from the command line to process a single media resource such as a live input capture or a static media file for delivery to a variety of client devices.

NGMP – The Nex Gen Media Server Web Portal is responsible for handling all client requests for media resources. Clients can be network media players, mobile phones, web browsers, IP Set-top-boxes, etc. NGMP is used to intelligently adapt and format live and stored media content to match the media client specific capabilities. It can selectively launch and manage NGMS child processes to handle client media session requests.

NGVX – The Nex Gen Video Conferencing Exchange is a SIP (Session Initiation Protocol) based Video Conferencing Server. NGVX is a Java component which is distributed independently of NGMS. NGVX requires NGMS to provide all media processing services.

NGMS is also available as a software library for direct integration into a third-party application. Please refer to the **NGMS Core Library Developer API Description** document (*NGMS_api.pdf*) for more information.

2. Software Contents

The NGMS download bundle contains several components. The key components are listed below.

2.1. bin/ngmsmgr

NGMP Web Portal. *ngmsmgr* is used to service client media requests and selectively launch and control NGMS child processes.

2.2. bin/ngmxcode

An executable wrapper for any third-party transcoders accessed through the *libxcode* interface layer. This executable provides transcoding services for *ngms*.

2.3. bin/ngms

A wrapper start-up script used to start the ngmsbin executable.

2.4. bin/ngmsbin

The streaming server binary built with transcoding support via the *libxcode* interface layer.

2.5. lib/libngms.so

The NGMS core framework packaged as a shared library which is used by the *ngms program*.

2.6. lib/libxcode.so

The transcoder interface layer library. This library may contain third-party codec encoder and decoder implementations.

2.7. etc/ngms.conf

The NGMS streaming configuration file. The '—*conf*' command line argument is used to load the streaming configuration. Any command-line arguments to *ngms* may supersede the same parameters defined in the configuration file.

2.8. etc/ngmsmgr.conf

The NGMP configuration file. This file is loaded on default by the *ngmsmgr* program.

2.9. etc/devices.conf

A basic device profile configuration file used by NGMS and NGMP. The device profile configuration is used to identify and classify streaming clients with their preferred format capabilities when they connect to NGMS via HTTP.

2.10. etc/pip.conf

An example PIP (Picture In Picture) configuration file. The PIP configuration file can be used by NGMS to specify dynamic PIP characteristics such as PIP motion, dynamic alpha, cropping, and picture padding adjustments.

2.11. etc/xcode.conf

An example transcoding configuration file. The transcoding configuration file is used by NGMS to configure any decoding and encoding properties.

2.12. bin/startmgr.sh

Startup script used to control NGMP. This script is used to start and stop the NGMS Web Portal.

2.13. bin/examplestart.sh

An example script used to show how to script NGMS. NGMS can be scripted by external applications to provide on-demand streaming services.

2.14. bin/mediaconvert.sh

A utility script which uses ffmpeg to convert media container files and video and audio encodings.

Unix Note: To run *ngms* from the command line you may need to instruct the system where to find any NGMS required shared libraries. For eg., from a Linux shell do:

export LD_LIBRARY_PATH=./lib; ./ngms –h

On Darwin (Mac OS), from a shell, do:

export DYLD_LIBRARY_PATH=./lib; ./ngms –h

3. NGMS Command Line Reference

3.1. General purpose arguments

3.1.1. **–conf** - Configuration file path. The default configuration file is '*etc/ngms.conf'*, The configuration file contains many settings which cannot be controlled via command line arguments.

3.1.2. **–help** - Show program usage and help.

3.1.3. **–license** - License file input path. The default license file path is *etc/license.dat*.

3.1.4. **--liccheck** – Validate the installed license. This argument can be included with other arguments that specify the license location, such as "*—license*" or "*—conf*".

3.1.5. **–licgen** – Create a license request information string. The output string can be copied and pasted in text form to request a streaming license from *ngmsvid.com*.

3.1.6. **–log** - An optional log file path. The log file can also be specified in the configuration file. If omitted, all log output is sent to *stdout*.

3.1.7. **–pid** - Output PID file to specified path.

3.1.8. **--verbose** – Enable verbose logging. This is the same as giving the *"-v"* option. Each single instance of "*-v*" increases the verbosity level. For eg. "*-vv*" is equivalent to *"–-verbose=3*", which increases the log verbosity from the default value of "*1*".

3.2. Arguments controlling general stream output

3.2.1. **–stream** - Enable stream output mode.

3.2.2. **--avsync** – Audio / Video sync base adjustment (positive or negative) in seconds expressed as a float. This argument is valid when processing both audio and video elementary streams. The audio video sync value is added to audio timestamps. A positive value will delay the audio with respect to the video.

When using multiple parallel output encodings, each output stream can be given it's own unique a/v offset by suffixing this argument with the output encoding index. For eg., to assign an a/v offset to stream output ***2*** use '*—avsync**2**=*'. Any index specific values will take precedence over the base '*—avsync=*' value.

3.2.3. **--conf** – Configuration input file path. If no file path is given the path '*etc/ngms.conf'* is used. The configuration file can contain parameters which may not be configurable using the command line.

3.2.4. **--delay** – Delay buffering factor applied to audio and video stream output. Value should be expressed as a positive float in seconds. This option is useful when processing live input, which

subsequently needs to be buffered to accommodate for poor input network conditions.

3.2.5. **--firaccept** – Controls how Full Intra Refresh (FIR) requests are handled by the application. The following behaviors are simultaneously affected with this parameter.

3.2.5.1. Controls the reception of RTCP Feedback type Full Intra Refresh messages (RTCP FB FIR), as defined in RFC 5104. The default value is 1, RTCP FB FIR request handling enabled. If set to *0*, RTCP FB FIR requests coming from the receiver of the RTP output stream will be ignored. This value can be individually controlled by the configuration file parameter "*FIRRTCPInputHandle*r".

3.2.5.2. Controls the behavior of IDR request messages for the local encoder. In addition to FIR messages received from RTCP, IDR requests can be internally generated when a client connects to a server published instance of the media output. An example is a connection to the HTTP URLs "*/tslive*", "*/flvlive*", "*/mkvlive*", the RTSP, or the RTMP server listener. The default value is 1, which enables an IDR request to the local encoder if a key-frame is needed to begin the format specific output. If set to 0, no internal IDR request will be dispatched to the local encoder. This value has no effect if the local output is not transcoded, if the configuration file parameter "*FIREncoder*" is disabled, or if the encoder does not support IDR requests. This value also does not affect the behavior of the "*videoForceIDR*" transcoding configuration parameter. This value can

be individually controlled by the configuration file parameter "*FIREncoderFromRemoteConnect*".

3.2.6. **--in** - Input file, media location, or capture device of the input media to be processed. If using separate video and audio devices, the video device should be placed in the first index. Up to two devices are permitted. Examples include:

3.2.6.1. "rtp://0.0.0.0:10000" – For an RTP listener on port 10000.

3.2.6.2. "*rtp://0.0.0.0:10000,10002*" For an RTP video listener on port 10000 and audio listener on port 10002. The video port should always be specified as the first port.

3.2.6.3. "*udp://127.0.0.1:41394*" For a direct UDP listener on port 41394 bound to the loopback interface only.

3.2.6.4. "*http://10.10.10.10:8080/tslive*" For requesting a live MPEG-2 TS encapsulated stream via HTTP. To load the media resource via SSL/TLS use "*https://10.10.10.10:8443/tslive*".

3.2.6.5. "*flv://10.10.10.10:8080/live.flv*" For requesting a remote FLV file via HTTP. Alternatively, this is interchangeable with the following command line: *–in=http://10.10.10.10:8080/live.flv –filter="type=flv"*. To load the resource via SSL/TLS use "*flvs://10.10.10.10:8443/live.flv*" or *–in=https://10.10.10.10:8443/live.flv –filter="type=flv"*.

3.2.6.6. "*http://10.10.10.10:8080/httplive/out.m3u8*" For requesting an HTTP Live stream from a remote server. To load the media resource via SSL/TLS use "*https://10.10.10.10:8443/httplive/out.m3u8*".

3.2.6.7. "*rtmp://10.10.10.10:1935/path/live.sdp*" For playing an RTMP stream with the name "*live.sdp*".

3.2.6.8. "*rtmp://0.0.0.0:1935*" For setting up an RTMP listener on the local interface port 1935 to accept a remote RTMP stream publisher. An RTMP publisher such as Flash Media Encoder can be used to provide a live input stream.

3.2.6.9. "*rtsp://10.10.10.10:554/live.sdp*" For playing a remote RTSP stream with the name "*live.sdp*".

3.2.6.10. *"/path/inputfile.mp4*" For reading from a container file such as an *.mp4*, *.3gp*, *.flv*, *.m2t*.

3.2.6.11. *"/path/inputfile.png*" For reading from an image format file such as an *.png* or *.bmp*.

3.2.6.12. *"/path/inputplaylist.m3u*" For reading and processing an input playlist.

3.2.6.13. *"/path/input.sdp*" For processing a Session Description Protocol file. The capture session will be setup to read from live capture according to the SDP syntax.

3.2.6.14. *"/path/pcapfile.pcap"* For reading from a pcap file.

3.2.6.15. "*eth0*" For raw pcap capture directly from a local interface.

3.2.6.16. *"/dev/framebuf0"* For reading directly from a block device, such as a video frame buffer, or audio samples buffer.

3.2.6.17. *"/dev/dummyvideo"* For reading from a dummy video frame buffer. This NGMS specific device expects "*ngmslib_onVidFrame*" to be called to provide video input frames according to the input filter configuration.

3.2.6.18. *"/dev/dummyaudio"* For reading from a dummy audio samples buffer. The NGMS specific device expects "*ngmslib_onAudSamples*" to be called to provide audio input samples according to the input filter configuration.

3.2.7. **--loop** – Enable looping of the input file.

3.2.8. **--noaudio** – Disable output of any audio elementary stream.

3.2.9. **--novideo** – Disable output of any video elementary stream.

3.2.10. **--overwrite** - Enable overwriting of recording output file if it already exists.

3.2.11. **--start** – Input file streaming start offset in seconds expressed as a float.

3.2.12. **–statusmax** – Maximum number of simultaneous HTTP Status connections. The default value is 0 (Status server disabled). The status server is used by the NGMP web portal to interrogate NGMS for real-time statistics. If enabled the status server is available on the default HTTP port of the "*/status*" URL.

3.2.13. **–streamstats** – Output file path of the stream output statistics file. Statistics include the overall throughput, average burst rate over the past 2 seconds, and past 8 seconds. RTP stream output will contain any RTCP Receiver Report metrics such as reported packet loss. TCP stream output will contain the current state of the output buffer. The value "*stdout*" or "*stderr*" can be used instead of an output file path.

By default, stream statistics are printed every 4 seconds. This interval can be adjusted by modifying the parameter "*outputStatisticsIntervalMs*" in the NGMS configuration file loaded via the *"—conf=*" command line parameter.

Stream statistics are also available for output via the status HTTP server as URI key value pair format parameters. The URL "*/status?streamstats*" can be used to access the stream statistics.

3.3. Arguments controlling Dynamic Update / Config Server

3.3.1. **--configsrv** – HTTP Dynamic configuration update interface server listening address and port string delimited by a colon. To enable a listener on port 8080 use "*8080*" or "http://*0.0.0.0:8080*". To enable a listener on localhost use "http://*127.0.0.1:8080*". To enable an SSL/TLS listener use "https://". An HTTP client can perform dynamic configuration updates by accessing the server at *http://[address]:[port]/config*.

3.3.2. **--configmax** – Maximum number of simultaneous HTTP dynamic configuration update connections. The default value is 0 (config server disabled). The config server interface can be used to update the running configuration such as encoder configuration parameters. This can be used to adjust stream output bitrate, GOP size, fps, force I frame insertion, adjust audio volume, etc. If enabled, the config server is available on the default HTTP port of the "*/config*" URL.

3.4. Arguments controlling MPEG-DASH Stream Output

3.4.1. **--dash** – MPEG-DASH live content streaming server listening address and port string delimited by a colon. To enable a listener on port 8080 use "*8080*" or "http://*0.0.0.0:8080*". To enable a listener on localhost use "http://*127.0.0.1:8080*". To enable an SSL/TLS listener use "https://". A client will access the server at *http://[address]:[port]/dash* or *http://[address]:[port]/dash/default.mpd* . Several versions of the Media Playlist Description (MPD) are available at the following URLS:

3.4.1.1. *http://[address]:[port]/dash/seglist.mpd* – An MPD using the *SegmentList* tag referencing the media encapsulated into mp4 segments. This is the same as the *default.mpd*.

3.4.1.2. *http://[address]:[port]/dash/segtemplate.mpd* – An MPD using the *SegmentTemplate* tag referencing the media encapsulated into mp4 segments.

3.4.1.3. *http://[address]:[port]/dash/tsseglist.mpd* – An MPD using the *SegmentList* tag referencing the media encapsulated into Transport Stream (.ts) segments. The option '*—dashts'* should be enabled.

3.4.1.4. *http://[address]:[port]/dash/tssegtemplate.mpd* – An MPD using the *SegmentTemplate* tag referencing the media encapsulated into Transport Stream (.ts) segments. The option '*—dashts'* should be enabled.

3.4.2. **--dashdir** – Output directory path where DASH media stream segments are stored. The default directory is "*html/dash*". It is recommended that a temporary in-memory file system be used to store these temporary chunk files. For eg, on linux to create a temporary in-memory file system do: "*sudo mount -t tmpfs -o size=102400K tmpfs /usr/local/ram/*".

3.4.3. **--dashfileprefix** – Output segment file prefix of any temporary MPEG-DASH segmented media file. The default prefix is "*out*".

This is the file prefix which will be written to any MPD file and which will be part of any URL request by an MPEG-DASH client.

3.4.4. **--dashurlhost** – Output URL hostname which will be written to an *MPD* file for each media hyperlink. Each hyperlink will be prepended by the specified URL hostname to allow serving of transport media files via an alternate host or virtual URL.

3.4.5. **--dashfragduration** – The movie fragment duration in seconds of each mp4 MOOF box contained within the mp4 media segment. The default value is 0.5 seconds. This value should generally not exceed more than 1 second.

3.4.6. **--dashmaxduration** – The maximum mp4 media segment duration in seconds. The default value is 10.0 seconds.

3.4.7. **--dashminduration** – The minimum mp4 media segment duration in seconds. The default value is 5.0 seconds. If this value is set to 0, then the media segmentor will produce segments according to the value of *'dashmaxduration'*. If this value is greater than 0 then the media segmentor will attempt to chunk segments at key-frame boundaries only after the minimum media duration has elapsed. Each segment should begin on a video key-frame.

3.4.8. **--dashmoof** – Enable or disable mp4 segment creation. This value is enabled by default if the MPEG-DASH server listener is enabled. To disable mp4 segment creation use *'—dashmoof=0'*.

3.4.9. **--dashts** – Enable or disable Transport Stream (*.ts*) segment creation. This value is disabled by default unless both the MPEG-DASH server listener and the HTTPLive segmentor are enabled. To enable Transport Stream segment creation use *'—dashts' or '—dashts=1'*.

3.4.10. **--dashtsduration** – MPEG-DASH Transport Stream (*.ts*) segment chunk duration in seconds. This option is effectively the same as '*—httplivechunk'* which controls the segment chunk duration for HTTPLive streaming.

3.4.11. **--dashuseinit** – Enable or disable use of a media stream initializer segment. This value is enabled by default resulting in the use of an initializer segment. To disable the media initializer segment use *'—dashuseinit=0',*

3.5. Arguments controlling FLV Stream Output

3.5.1. **--flvdelay** – Delay factor applied to audio and video stream output for FLV format encapsulation. Value should be expressed as a positive float in seconds. This setting will cause a client media player to pre-buffer the live content and allow it to be treated like a static file. Without this value set some media players may continue to show a buffering or loading icon for the video. The default value is 1.0 second.

3.5.2. **--flvlive** – HTTP FLV encapsulated live content streaming server listening address and port string delimited by a colon. To enable a listener on port 8080 use "*8080*" or "http://*0.0.0.0:8080*". To enable a listener on localhost use "http://*127.0.0.1:8080*". To enable an SSL/TLS listener use "https://". A client will access the server at *http://[address]:[port]/flvlive*.

3.5.3. **--flvrecord** – The output filename for recording output content to a file. The recorded media will be encapsulated using the FLV file format. An *".flv"* suffix will be appended to the output file if none is given. The command will fail if an output codec is not supported by the container file format.

When using multiple parallel output encodings, each individual output stream can be recorded by suffixing this argument with the output encoding index. For eg., to record output stream ***2*** use '—*flvrecord**2***='.

3.6. Arguments controlling HTTPLive Streaming

3.6.1. **--httplive** – HTTP Live Streaming Server listening address and port string delimited by a colon. To enable a listener on port 8080 use "*8080*" or "http://*0.0.0.0:8080*". To enable a listener on localhost use "http://*127.0.0.1:8080*". To enable an SSL/TLS listener use "https://". An HTTP Live client (iPhone, iPad, Safari) will access the server at *http://[address]:[port]/httplive*.

3.6.2. **--httplivebw** – Sets the HTTPLive published stream output bitrate(s) when using transcoding to produce multiple bitrate specific streams. The bitrate is pulished in the master HTTPLive playlist as the "BANDWIDTH" field and is expressed in Kb/s. If this parameter is omitted, the default bitrate of the output .ts stream is the encoder configuration bitrate multiplied by a factor of 1.15 to account for the audio stream and any packetization overhead. Multiple stream output bitrates are specified as a CSV. For eg. '*—httplivebw="300, 600"* ' to denote 300Kb/s and 600Kb/s.

3.6.3. **--httplivechunk** – HTTP Live segment chunk duration in seconds. The default value is 10.0 seconds. This value will influence the overall delay of media delivery from the capture source. Generally, a value between 5 and 15 seconds is acceptable.

3.6.4. **--httplivedir** – Output directory path where MPEG-2 TS stream output chunks are stored. The default directory is "*html/httplive*". It is recommended that a temporary in-memory file system be used to store these temporary chunk files. For eg, on linux to create a temporary in-memory file system do: "*sudo mount -t tmpfs -o size=102400K tmpfs /usr/local/ram/*".

3.6.5. **--httplivefileprefix** – Output chunk file prefix of any temporary HTTP Live chunk media file. The default prefix is "*out*". This is the file prefix which will get written to any .m3u8 playlist file and which will be part of any URL request by a httplive client.

3.6.6. **--httpliveurlhost** – Output URL media host prefix which gets written to any *.m3u8* playlist file for each TS file. Each playlist item will be prepended by the specified URL host prefix to allow serving of transport media files via an alternate host or virtual URL. An example would be "*https://httplive.cdn.mycompany.com:8080/httplive*".

3.7. Arguments controlling Auto-Format Adaptation Server

3.7.1. **--live** – HTTP Auto-Format Adaptation Web Server listening address and port string delimited by a colon. The Auto-Format Adaptation Server listener provides a single URL to access the different available delivery formats according to the client User-Agent. To enable a listener on port 8080 use "*8080*" or "http://*0.0.0.0:8080*". To enable a listener on localhost use "http://*127.0.0.1:8080*". To enable an SSL/TLS listener use "https://". A media player will access the server at *http://[address]:[port]/live*. The server will automatically return the appropriate media format content type according to the client User-Agent type. The User-Agent lookup is performed according to the device type definition found in "*etc/devices.conf*".

Note: The same HTTP Server resource pool is used to service all HTTP based requests such as *"/tslive"*, "*/httplive*", "*/*flvlive", *"/dash"*, and *"/live*". The maximum number of HTTP listeners is limited by the "*maxConn*" configuration file setting. The default value is 20 and maximum value is 100.

3.8. Arguments controlling Matroska / WebM Stream Output

3.8.1. **--mkvlive** – HTTP Matroska / WebM encapsulated live content streaming server listening address and port string delimited by a colon. To enable a listener on port 8080 use "*8080*" or "http://*0.0.0.0:8080*". To enable a listener on localhost use "http://*127.0.0.1:8080*". To enable an SSL/TLS listener use "https://". A client will access the server at *http://[address]:[port]/mkvlive*.

This option is the same as "*—webmlive*".

3.8.2. **--mkvdelay** – Delay factor applied to audio and video stream output for Matroska / WebM format encapsulation. Value should be expressed as a positive float in seconds. This setting will cause a client media player to pre-buffer the live content and allow it to be treated like a static file. Without this value set some media players may continue to show a buffering or loading icon for the video. The default is value is 1.0 second.

3.8.3. **--mkvrecord** – The output filename for recording output content to a file. The recorded media will be encapsulated using the Matroska file format. An *".mkv"* suffix will be appended to the output file if none is given. The command will fail if an output codec is not supported by the container file format.

When using multiple parallel output encodings, each individual output stream can be recorded by suffixing this argument with the output encoding index. For eg., to record output stream ***2*** use '—*mkvrecord**2***='.

3.9. Arguments controlling UDP / RTP Stream Output

3.9.1. **–mtu** - Path MTU (Maximum Transimission Unit) of the payload data in bytes. This is in addition to any RTP, UDP, IP and other packet headers.

3.9.2. --**noseqhdrs** – Disable inclusion of video specific sequence headers within the codec specific transport bit-stream. For an H.264 output stream this option disables including the SPS / PPS preceding each key-frame.

3.9.3. **--out** - Output destination or file name. If this option is absent then the output string takes the value of the *"--stream=*" argument.

Multiple UDP / RTP recipients can be specified by suffixing this argument with an incrementing output index. For eg., to output to two destinations use '—out**1**=' and '—*out**2***='. If performing transcoding to produce multiple parallel output encodings, each destination output index will correspond to the same encoder output index. Up to four output destinations can be given.

Examples include:

3.9.3.1. "*rtp://127.0.0.1:5004*" For RTP output to port 5004. The default media encapsulation is MPEG-2 TS.

3.9.3.2. "*udp://127.0.0.1:5004*" For direct UDP output to port 5004. The default media encapsulation is MPEG-2 TS.

3.9.3.3. "*rtp://127.0.0.1:5004,5006*" For RTP output using the RTP/AVP profile where the video elementary stream is sent on port 5004 and the audio elementary stream is sent on port 5006. Use the "*—transport=rtp*" argument to enable codec native RTP encapsulation.

3.9.3.4. "s*rtp://127.0.0.1:5004,5006*" For Secure RTP output using the RTP/SAVP profile where the video elementary stream is sent on port 5004 and the audio elementary stream is sent on port 5006. Use the "*—transport=rtp*" argument to enable codec native RTP encapsulation.

3.9.3.5. "*/path/outputfile.m2t*" For writing the output to local storage using MPEG-2 TS encapsulation.

Note: Alternatively the "*—flvrecord*" parameter can be used to save the stream output into an FLV container file.

3.9.4. **--rtcpavsync** – Audio / Video offset in seconds given as float which is advertised in RTCP sender reports. RTP / RTCP receivers may

choose to ignore this value and may instead only respect the timestamp of the first packet received.

3.9.5. **–rtcpports** – RTCP output port number(s). The default behavior is to use an RTCP UDP port number one greater than the RTP port. This parameter can be used to setup non-default RTCP port number(s). The RTCP ports specified with this parameter apply only to direct RTP output and not to RTSP initiated RTP streams. The following example shows video and audio output to two destinations, the first using default RTCP ports, and the second multiplexing the RTCP channels on the RTP channels. "*—out=rtp://10.10.10.10:2000,2002*" "***--rtcpports=2001,2003***" *out2=rtp://10.10.10.10:3000,3002*" "***--rtcpports2=3000,3002***"

3.9.6. **--rtcpsr** – RTCP Sender Report transmission interval in seconds. The default interval is 5 seconds. Set to 0 to disable RTCP Sender Reports. This value overrides the configuration file parameter "*RTCPSenderReportInterval*".

3.9.7. **–rtpclock** – The RTP output clock rate in Hz. To set the video stream clock to 24KHz and the audio to 48KHz use *'—rtpclock="v=24000,a=48000" ' or '--rtpclock="24000,48000" '*. This option can be used to override the default output clock obtained from the input media or SDP file.

3.9.8. **--rtpmaxptime** – Sets the amount of audio frames aggregated into a single RTP output packet payload for audio codecs which permit compounding multiple audio frames. The value is expressed as a

duration in ms. For an audio codec producing an audio frame every 20ms, *"—rtpmaxptime=40*" can be used to include two audio frames into a single RTP packet payload.

3.9.9. **–rtppayloadtype** – The RTP output payload type(s). To set the video payload type to 112, and the audio to 96 use *'—rtppayloadtype="v=112,a=96" ' or '–rtppayloadtype="112,96" '*. The default RTP payload type values are codec specific.

3.9.10. **–rtppktzmode** – The RTP output codec specific packetization mode. This value can be used to control a video packetization mode such as for H.264 (RFC 3984). An H.264 NAL packetization mode can take the values 0, 1, 2, with the default set to 1. When using packetization mode 0 ensure to use an encoding parameter "*videoSliceSizeMax"* set to less than the payload MTU.

3.9.11. **–rtpssrc** – The RTP output SSRC(s). To set the video SSRC to 0x01020304 and the audio to 0x01020305 use *'—rtpssrc="v=0x01020304,a=0x01020305" ' or '—-rtpssrc="0x01020304,0x01020305" '*. Instead of the base 16 notation show in this example, the SSRC can be expressed in standard base 10 notation. If this option is not provided, the default RTP SSRC values are generated randomly.

3.9.12. **–rtpusebindport** – Enable use of the same UDP/RTP local capture port(s) as the source UDP port for outgoing RTP packets. This setting is disabled by default and enabled for any video conference endpoints added through the PIP interface.

3.9.13. **--sdp** – SDP output file path of the published output session description.

3.9.14. **--srtpkey** – One ore more base64 encoded SRTP key(s) used to provide confidentiality and authentication of the outgoing media stream(s). If providing more than one key, the base64 strings should be separated by a comma ‘,’. The key before the comma is used for the video media and the key after the comma for the audio session. For eg., *“--srtpkey=mLBdSdX031vbI9rLOV5foVs5rbdobPl8/S2x2Xp0,x6N4T8dNTVfjyh1TU6XHUZ5SgbctnSTui8GwQyIC”*. This argument is only applicable if the output streaming method is “srtp://”. If this argument is omitted, a session key(s) will be created using the local pseudo random number generator.

When using multiple RTP recipients, multiple RTP recipient specific keys can be specified by suffixing this argument with an incrementing output index. For eg., to output to two destinations use ‘—srtpkey**1**=’ and ‘—*srtpkey**2***=’.

3.9.15. **–stunrequest** – Issue STUN binding requests on any RTP / RTCP outbound socket. An optional parameter can follow specifying the STUN policy.

3.9.15.1. **"1"** - Always send STUN binding requests. This is the default STUN policy if an optional parameter is omitted. For eg., '—stunrequest=1'.

3.9.15.2. **"2"** - Send STUN binding requests only if first receiving a STUN binding request on the RTP / RTCP socket. For eg., '—stunrequest=2'.

3.9.16. **–stunrequestuser** – Specify the STUN USERNAME attribute value used in STUN binding requests. The STUN username should be a base64 encoded string. The presence of this argument without the *'–stunrequest'* argument present will implicitly specify a STUN policy equivalent of *'–stunrequest=2'*.

If providing unique usernames for the video and audio channels, the base64 strings should be separated by a comma ','. The value before the comma is used for the video media and the value after the comma for the audio session. For eg., *"stunrequestuser=DfLbHAOOqteRaikppk9FyRFr, eWMFv3WdKONzJprjpbrzRjfY".*

3.9.17. **–stunrequestpass** – Specify the STUN password used for computing a MESSAGE-INTEGRITY HMAC STUN attribute in STUN binding requests. The STUN password should be a base64 encoded string. The presence of this argument without the *'–stunrequest'* argument present will implicitly specify a STUN policy equivalent of *'–stunrequest=2'*.

If providing unique passwords for the video and audio channels, the base64 strings should be separated by a comma ‘,’. The value before the comma is used for the video media and the value after the comma for the audio session. For eg., *“stunrequestpass=DfLbHAOOqteRaikppk9FyRFr, eWMFv3WdKONzJprjpbrzRjfY”.*

3.9.18. –stunrequestrealm – Specify the STUN realm used for computing a MESSAGE-INTEGRITY HMAC STUN attribute in STUN binding requests. If the STUN realm, password, and username are all present then STUN long-term credentials are used according to RFC 5389, otherwise this argument is ignored. The presence of this argument without the *‘–stunrequest’* argument present will implicitly specify a STUN policy equivalent of *‘–stunrequest=2’*.

3.9.19. --transport - Output transport protocol.

3.9.19.1. “*m2t*” - (Default) Use MPEG-2 Transport Stream encapsulation.

3.9.19.2. “*rtp*” - Use Video / Audio codec native RTP specific encapsulation type.

3.10. Arguments controlling Picture In Picture (PIP)

3.10.1. –**pip** – (PIP) Picture In Picture media source. Set the input path of the media to be used as the PIP. Any input media format used with the *"—in*" parameter can also be used as a PIP input media source, such as a capture from a live source. A .bmp, or .png with Alpha mask can be used for a still image PIP source.

3.10.2. –**pipalphamax** – (PIP) Picture In Picture maximum alpha masking value. Range is from 0 – 255, with 255 being the default, resulting in no transparency. This value caps the maximum alpha transparency of any pixel if the PIP image contains an alpha mask. A lower value results in greater transparency of the PIP.

3.10.3. –**pipalphamin** – (PIP) Picture In Picture minimum alpha masking value. Range is from 0 – 255, with 0 being the default. This value caps the minimum alpha transparency of any pixel if the PIP image contains an alpha mask. A greater value results in greater less transparency of the PIP.

3.10.4. –**pipbefore** – If this argument is present, the PIP will be overlayed into the main picture prior to any scaling directive specified in the *"—xcode*" argument. By default, PIP placement is done after any scaling of the main picture.

3.10.5. **–pipconf** – File path of a PIP configuration file defining PIP characteristics. The PIP configuration file can contain directives for picture cropping, padding, PIP motion, and PIP transitioning. For usage examples refer to the file "*etc/pip.conf*".

3.10.6. **--piphttp** – PIP control interface server listening address and port string delimited by a colon. To enable a listener on port 8080 use "*8080*" or "http://*0.0.0.0:8080*". To enable a listener on localhost use "http://*127.0.0.1:8080*". To enable an SSL/TLS listener use "https://". An HTTP client will access the server at *http://[address]:[port]/pip*.

3.10.7. **–piphttpmax** – Maximum number of simultaneous HTTP PIP control connections. The default value is 0 (PIP Control interface disabled). The PIP control server can be used to dynamically add and remove a PIP.

3.10.8. **--pipxcode** – PIP formatting configuration passed as a quoted comma separated list of key value pairs. These options take the same format as the *"—xcode*" parameters documented under "*Transcoder Configuration*". Only options specific to PIP output dimensions, scaling type, cropping, padding, and frame rate (applicable for non static PIP formats) are processed.

3.10.9. **–pipx** – The horizontal (*x axis*) placement of the left edge of the PIP relative from the left edge of the main picture.

3.10.10. **–pipxright** – The horizontal (x axis) placement of the right edge of the PIP relative from the right edge of the main picture. This can be used instead of "*—pipx*" to place a PIP with regard to the right edge.

3.10.11. **–pipy** – The vertical (*y*) placement of the top edge of the PIP relative from the top edge of the main picture.

3.10.12. **–pipybottom** – The vertical (*y axis*) placement of the bottom edge of the PIP relative from the bottom edge of the main picture. This can be used instead of "*—pipy*" to place a PIP with regard to the bottom edge.

3.10.13. **–pipybottom** – The vertical (*y axis*) placement of the bottom corner of the PIP relative from the bottom edge of the main picture. This can be used instead of "*—pipy*" to place a PIP with regard to the bottom edge.

3.11. Arguments controlling RTMP Stream Output

3.11.1. **--rtmp** - RTMP Server listening address and port string delimited by a colon. For eg. "*127.0.0.1:1935*" or "*1935*" to denote "*0.0.0.0:1935*". An RTMP client will access the server at "*rtsp://[address]:[port]*". If the optional port argument is omitted, the default listening port is *1935*.

3.12. Arguments controlling RTSP Stream Output

3.12.1. **--rtsp** - RTSP Server listening address and port string delimited by a colon. For eg. "*127.0.0.1:1554*" or "*1554*" to denote

"*0.0.0.0:1554*". An RTSP client will access the server at "*rtsp://[address]:[port*]". If the optional port argument is omitted, the default listening port is *1554*.

Note: The default listener port 1554 is different from the standard RTSP port 554, which requires root user access privileges.

3.13. Arguments controlling MPEG-2 TS Stream Output

3.13.1. **--tslive** – HTTP MPEG-2 Transport Stream live content streaming server listening address and port string delimited by a colon. To enable a listener on port 8080 use "*8080*" or "http://*0.0.0.0:8080*". To enable a listener on localhost use "http://*127.0.0.1:8080*". To enable an SSL/TLS listener use "https://". An MPEG-2 Transport Stream client will access the server at *http://[address]:[port]/tslive*.

3.14. Arguments controlling Transcoding

3.14.1. **--xcode** – The path of the transcoder configuration file or a quoted comma separated list of transcoder configuration parameters expressed as key value pairs.

The transcoder configuration file defines any transcoder parameters using key value pair format. A reference configuration file "*etc/xcode.conf"* is supplied with the NGMS installation bundle. Each configuration key can be specified in either long or

abbreviated notation. Abbreviated notation is preferred when passing a transcoder parameter list via the command line.

3.14.1.1. Video output parameters

3.14.1.1.1. "**videoCodec"** (or "**vc**") - Video Codec.

3.14.1.1.1.1. "**h264**" - H.264

3.14.1.1.1.2. **"h263"** - H.263

3.14.1.1.1.3. **"h263+"** - H.263+

3.14.1.1.1.4. **"mpg4"** - MPEG-4 Part 2

3.14.1.1.1.5. **"vp8"** - VP8

3.14.1.1.1.6. "**rgba**" - RGBA8888 (32 bits per pixel with 8 bit alpha mask)

3.14.1.1.1.7. **"bgra**" - BGRA8888 (32 bits per pixel with 8 bit alpha mask)

3.14.1.1.1.8. "**rgb565**" - RGB565 (16 bits per pixel)

3.14.1.1.1.9. "**bgr565**" - BGR565 (16 bits per pixel)

3.14.1.1.1.10. "**rgb**" - RGB888 (24 bits pr pixel)

3.14.1.1.1.11. "**bgr**" - BGR888 (24 bits pr pixel)

3.14.1.1.1.12. "**nv12**" - NV12 (12 bits per pixel)

3.14.1.1.1.13. "**nv21**" - NV21 (YUV420SP) (12 bits per pixel)

3.14.1.1.1.14. "**passthru**" or "**same**" - Pass-thru transcoding. Only available if multiple output encodings are enabled.

3.14.1.1.1.15. "**yuv420sp**" - YUV420SP (NV21) (12 bits per pixel)

3.14.1.1.1.16. "**yuv420p**" - YUV420P (12 bits per pixel)

3.14.1.1.1.17. "**yuva420p**" - YUVA420P (20 bits per pixel) 8 bit alpha mask.

3.14.1.1.2. "**cropBottom"** (or "**cropb**") - Number of pixels to crop at bottom edge of picture. Default 0.

3.14.1.1.3. "**cropLeft**" (or "**cropl**") - Number of pixels to crop at left edge of picture. Default 0.

3.14.1.1.4. "**cropRight"** (or "**cropr**") - Number of pixels to crop at right edge of picture. Default 0.

3.14.1.1.5. "**cropTop**" (or "**cropt**") - Number of pixels to crop at top edge of picture. Default 0.

3.14.1.1.6. "**padAspectRatio**" (or "**padaspect**") - Set to 1 to preserve original aspect ratio of when adding padding pixels. Default 0.

3.14.1.1.7. "**padBottom**" (or "**padb**") - Number of pixels to add as border at bottom edge of picture. The frame output resolution is preserved but the original picture is scaled down to accommodate the padding. Default 0.

3.14.1.1.8. "**padLeft**" (or "**padl**") - Number of pixels to add as border at left edge of picture. The frame output resolution is preserved but the original picture is scaled down to accommodate the padding. Default 0.

3.14.1.1.9. "**padRight**" (or "**padr**") - Number of pixels to add as border at right edge of picture. The frame output resolution is preserved but the original picture is scaled down to accommodate the padding. Default 0.

3.14.1.1.10. "**padTop**" (or "**padt**") - Number of pixels to add as border at top edge of picture. The frame output resolution is preserved but the original picture is scaled down to accommodate the padding. Default 0.

3.14.1.1.11. "**padColorRGB**" (or "**padrgb**") - RGB Color value of padding pixels. The default value is *0x000000* (black). For eg. *0xffffff* is the RGB value for white.

3.14.1.1.12. "**videoBitrate**" (or "**vb**") - Video output bitrate in Kilobits per sec. This value is relevant for codecs that use bit rate based rate control ("*btrt*"), such as H.264 when it is not using "*cqp*" or "*crf*" based rate control.

The "*videoBitrate"* parameter should always be specified if HTTPLive output is enabled for multiple parallel encodings to provide adaptive bitrate support. This is applies even if an encoder rate control method other than "btrt" is used. The value of "*videoBitrate*" is used to describe each bitrate specific output stream in the master HTTPLive playlist to enable bitrate stream switching.

3.14.1.1.13. "**videoBitrateTolerance**" (or "**vbt**") - Video output bandwidth variance tolerance in Kilo bits per sec. Applicable only if "*videoBitrate*" is set.

3.14.1.1.13.1. 0 – (Default) Uses encoder specific output bitrate tolerance.

3.14.1.1.14. "**vbvBufferSize**" (or "**vbvbuf**") - Video output video buffer verifier buffer size in Kilobits. This value is relevant for codecs that support it.

3.14.1.1.15. "**vbvMaxRate**" (or "**vbvmax**") - Video output video buffer verifier maximum bitrate in Kilobits per sec. This value is relevant for codecs that support it.

3.14.1.1.16. "**vbvMinRate**" (or "**vbvmin**") - Video output video buffer verifier minimum bitrate in Kilobits per sec. This value is relevant for codecs that support it.

3.14.1.1.17. "**videoInputConstantFps**" (or "**vcfrin**") - Controls input frame rate type.

3.14.1.1.17.1. 0 – (Default) Input frame timestamp is obtained from input transport mechanism (MPEG-2 TS PTS / DTS, RTP Timestamp).

3.14.1.1.17.2. 1 – Input frame rate is always constant according to fps automatically obtained from video codec specific sequence header information, or "*fps*" command line hint. The effective input frame timestamp may be automatically adjusted if it drifts beyond a threshold of the actual input frame time.

3.14.1.1.18. "**videoOutputConstantFps**" (or "**vcfrout**") - Controls output frame rate type.

3.14.1.1.18.1. 0 – (Default) Output frame rate timestamp will be the same as the corresponding input frame time stamp. The output fps (*videoFrameRate*) will not be

exceeded even if the input video frame rate is higher than the configured output.

3.14.1.1.18.2. 1 – Output frame rate is always constant according to (*videoFrameRate*). The output frame timestamp may be automatically internally adjusted if it drifts beyond the threshold (*videoOutputFpsDriftTolerance)* of the input frame time.

3.14.1.1.18.3. -1 – Output frame rate timestamp is always the same as the corresponding input frame time stamp, regardless of the output fps (*videoFrameRate*). If this value is omitted, the configured output fps (*videoFrameRate*) will not be exceeded even if the input video frame rate is higher than the configured output.

3.14.1.1.19. "**videoOutputFpsDriftTolerance**" (or "**vcfrtol**") - If "*videoOutputConstantFps"* is enabled, this value controls the constant frame rate timestamp tolerance in milliseconds. The tolerance is the limit on the deviation of the constant frame rate time stamp with the actual input frame timestamp. If the tolerance is exceeded, the output frame timestamp is reset to the input time stamp. The default value is 400ms.

3.14.1.1.20. "**videoEncoderSpeed**" (or "**vf**") - Encoder speed / quality trade-off presets. This is mutually exclusive with the setting "*videoEncoderQuality*".

3.14.1.1.20.1. 0 – slowest (highest quality)

3.14.1.1.20.2. 1 – slow (high quality)

3.14.1.1.20.3. 2 – medium (medium quality)

3.14.1.1.20.4. 3 – fast (low quality)

3.14.1.1.20.5. 4 – fastest (lowest quality)

3.14.1.1.21. "**videoEncoderQuality**" (or "**vqual**") - Encoder quality / speed trade-off presets. This is mutually exclusive with the setting "*videoEncoderSpeed*".

3.14.1.1.21.1. 0 – lowest (highest speed)

3.14.1.1.21.2. 1 – low (fast speed)

3.14.1.1.21.3. 2 – medium (medium speed)

3.14.1.1.21.4. 3 – high (slow speed)

3.14.1.1.21.5. 4 – highest (lowest speed)

3.14.1.1.22. "**videoFpsNumerator**" (or "v**fn**") - Output frame rate numerator. The Output frame rate value is passed to the encoder to determine the bitrate when using non-qp based encoding. The actual output frame rate may deviate from the supplied (*videoFpsNumerator / videoFpsDenumerator*) value depending on the encoder specific configuration, such as settings of "*videoInputConstantFps*", "*videoUpsampling*".

3.14.1.1.23. **"videoFpsDenumerator**" (or "**vfd**") - Output frame rate denominator.

3.14.1.1.24. "**videoFps**" (or "**vfr**") - Output Frame rate expressed as a floating point. For eg "*videoFps=29.97"* is equivalent to *"videoFpsNumerator=2997,videoFpsDenumerator=100".*

3.14.1.1.25. "**videoForceIDR**" (or "**vidr**") - If present, sends an IDR request to the underlying encoder. This flag should not be present in the initial transcoding configuration but can be used to explicitly request an IDR during an encoder re-configuration update.

3.14.1.1.26. "**videoGOPMaxMs**" (or "**vgmax**") – Encoder max Group Of Pictuers (GOP) in milliseconds. The actual frame count passed to the encoder is obtained based on the specified frame rate (*videoFps*). The value "*infinte*" can be used to request the encoder to only produce a single IDR. The "*videoForceIDR*" parameter can subsequently be used

to request an IDR when performing an encoder configuration update.

3.14.1.1.27. "**videoGOPMinMs**" (or "**vgmin**") – Encoder min Group Of Pictures (GOP) in milliseconds. The actual frame count passed to the encoder is obtained based on the specified frame rate (*videoFps*)

3.14.1.1.28. "**videoLookaheadFramesMin1**" (or "**vlh**") - Encoder specific look-ahead configuration. If > 0, value passed to encoder is (n – 1).

3.14.1.1.28.1. 0 – (Default) Use automatic encoder specific configuration.

3.14.1.1.28.2. 1 - Minimal encoder lag configuration (0). This value should be used for real-time encoding. Note that the "*videoThreads*" (encoder thread control count) and "videoDecoderThreads" (decoder thread control count) may also influence the actual transcoder frame lag.

3.14.1.1.28.3. n (n – 1) lag encoder configuration.

3.14.1.1.29. "**videoProfile**" (or "**vp**") - Encoder codec specific profile.

3.14.1.1.29.1. H.264 Profiles

3.14.1.1.29.1.1. 0 - (Default) H.264 High

3.14.1.1.29.1.2. 66 – H.264 Baseline

3.14.1.1.29.1.3. 77 – H.264 Main

3.14.1.1.29.1.4. 100 – (Default) H.264 High

3.14.1.1.29.2. MPEG-4 Part 2 Profiles

This 8 bit value is a combination of the profile and level. The profile is the 4 most significant bits, the level is the 4 least significant bits.

3.14.1.1.29.2.1. 0 - (Default) Simple Profile

3.14.1.1.29.2.2. 240 - Advanced Simple Profile (Profile value 15 (0x0f) << 4)

3.14.1.1.29.2.3. n - (Profile:15 (0x0f) << 4) | (Level:1 (0x01)). For eg. Profile 15, Level 1 (15 << 4 | 1) = 241

3.14.1.1.30. "**videoQuantizer**" (or "**vq**") - Video target quantizer for codecs which support it. "*videoQuantizer*" should be used in-place of "*videoBitrate*" (target bitrate) when using "*cqp*" or "*crf*" based rate control. For H.264, a valid

quantizer range is from 10-51, with lower values giving higher quality. The quantizer value is for P-frames.

3.14.1.1.31. "**videoQuantizerBRatio**" (or "**vqbratio**") - Video target quantizer output ratio for B frames relative to P frames. A value > 1 produces an average B frame with higher quantizer (lower quality) than an average P frame. H.264 default is 1.25.

3.14.1.1.32. "**videoQuantizerIRatio**" (or "**vqiratio**") - Video target quantizer output ratio for I frames relative to P frames. A value < 1 produces an average I frame with a lower quantizer (higher quality) than an average P frame. H.264 default is 0.71.

3.14.1.1.33. "**videoQuantizerMax**" (or "**vqmax**") - The maximum output quantizer of generated P frames. This can be specified to have a supporting encoder produce output frames within a permissible quality threshold. Quantizer Range is codec specific. For H.264: 10 – 51. For MPEG-4: 2 - 31. For VP8: 4 – 63.

3.14.1.1.34. "**videoQuantizerMin**" (or "**vqmin**") - The maximum output quantizer of generated P frames. This can be specified to have a supporting encoder produce output frames within a permissible quality threshold. Quantizer Range is codec specific. For H.264: 10 – 51. For MPEG-4: 2 - 31. For VP8: 4 – 63.

3.14.1.1.35. "**videoQuantizerDiff**" (or "**vqdiff**") - The difference in quantization of consecutive output frames.

3.14.1.1.36. "**videoRateControl**" (or "**vrc**") - Video output rate control type.

3.14.1.1.36.1. "**btrt**" - Bit rate based rate control. This is the default rate control type. A valid "*videoBitrate*" bit rate should also be given if using bitrate based rate control. If using a low "*videoBitrateTolerance*" bit rate tolerance value, this method is usually able to produce video output with the least standard deviation of output bandwidth, which is best suitable for streaming on congested network links.

3.14.1.1.36.2. "**cqp**" - Rate control utilizing a constant quantizer for P-frames. A valid target quantizer "*videoQuantizer*" should also be given. This method strives to produce video output where each P frame is encoded to a similar quality, leading to very controlled video output quality at the expense of some bandwidth wasting. Video output bandwidth can have very high standard deviation depending on the scene complexity of the input video.

3.14.1.1.36.3. "**crf**" - Rate control utilizing a constant Rate Factor. A valid target quantizer "*videoQuantizer*" should also be given. This method is very similar to "*cqp*" but strives to be more intelligent in the allocation of available bandwidth between high complexity and low complexity frames. For network streaming, "*crf*" based rate control is generally preferred instead of "*cqp*" in terms of standard deviation of video output bitrate.

3.14.1.1.37. "**videoScalerType**" (or "**vsc**") - Resolution scaler type The default value is 3. Order should be from fastest to slowest, with fastest being 1. Libswscale scalers are listed below:

3.14.1.1.37.1. 1 – SWS_FAST_BILINEAR

3.14.1.1.37.2. 2 – SWS_BILINEAR

3.14.1.1.37.3. 3 - SWS_BICUBIC

3.14.1.1.37.4. 4 - SWS_GAUSS

3.14.1.1.37.5. 5 - SWS_SINC

3.14.1.1.38. "**videoISceneAggressivity**" (or "**vsi**") - Frame scene cut insertion aggressivity. This value is encoder specific.

For the x264 encoder, the value is from 1 .. 100, where 1 is least aggressive, 100 is most aggressive. The default value is 40.

3.14.1.1.39. "**videoSliceSizeMax**" (or "**vslmax**") – Encoder maximum size of each frame slice in bytes. This value is encoder and codec specific. A value less than the MTU should result in multiple slices per frame.

3.14.1.1.40. "**videoThreads**" (or "**vth**") - Encoder specific number of threads.

3.14.1.1.40.1. 0 – (Default) Encoder will choose a default value. Values greater than 1 will usually incur additional encoder frame output lag. This value should be set to 1 for real-time minimal encoder latency.

3.14.1.1.41. "**videoDecoderThreads**" (or "**vthd**") - Decoder specific number of threads.

3.14.1.1.41.1. 0 – (Default) Decoder will choose a default value. Values greater than 1 could potentially incur additional decoder frame output lag. This value should be set to 1 for real-time minimal decoder latency.

3.14.1.1.42. "**videoUpsampling**" (or "**vup**") - Controls video up-sampling / frame duplication logic.

3.14.1.1.42.1. 0 – (Default) disables frame up-sampling / frame duplication.

3.14.1.1.42.2. 1 - Enable frame up-sampling in-order to adhere to specified output frame rate. Up-sampling is only enabled if "*videoOutputConstantFps*" is set to 1.

3.14.1.1.43. "**videoWidth**" (or "v**x**") - Horizontal output resolution in pixels.

3.14.1.1.43.1. 0 – Default. If "*videoWidth"* is set and "*videoHeight*" is not set, the input picture aspect ratio will be preserved, and the horizontal resolution will be set to "*videoWidth"*. If both "*videoWidth*" and "*videoHeight*" are not set, the input picture dimensions will be preserved.

3.14.1.1.44. "**videoHeight**" (or "v**y**") - Vertical output resolution in pixels.

3.14.1.1.44.1. 0 – Default. If "*videoHeight*" is set and "*videoWidth*" is not set, the input picture aspect ratio will be preserved, and the vertical resolution will be set to "*videoHeight*".

3.14.1.2. Audio output parameters

3.14.1.2.1. "**audioCodec**" (or "**ac**") - Audio codec.

3.14.1.2.1.1. "**aac**" – AAC

3.14.1.2.1.2. "**ac3**" – A53 / AC3

3.14.1.2.1.3. "**amr**" – AMR-NB

3.14.1.2.1.4. "**g711a**" – G.711 alaw

3.14.1.2.1.5. "**g711u**" – G.711 mulaw

3.14.1.2.1.6. "**silk**" – SILK

3.14.1.2.1.7. "**vorbis**" – Vorbis

3.14.1.2.1.8. "**pcm**" – PCM signed 16bit Little Endian

3.14.1.2.1.9. "**ulaw**" – PCM 8 bit mulaw

3.14.1.2.1.10. "**alaw**" – PCM 8 bit alaw

3.14.1.2.2. "**audioBitrate**" (or "**ab**") – Audio output bandwidth per channel in bits per second.

3.14.1.2.2.1. 0 – (Default) Uses encoder default setting.

3.14.1.2.3. "**audioForce**" (or "**af**") – Force audio output transcoding even if output codec matches input codec type, sample rate, and channel configuration.

3.14.1.2.3.1. 0 - (Default) enable audio transcoding only if output codec does not match input codec type, sample rate, and channel configuration.

3.14.1.2.3.2. 1 - Force audio transcoding.

3.14.1.2.4. "**audioSampleRate**" (or "**ar**") – Audio output sampling frequency in HZ.

3.14.1.2.4.1. 0 – (Default) Reuses input sampling rate.

3.14.1.2.5. "**audioChannels**" (or "**as**") – Audio output channels.

3.14.1.2.5.1. 0 – (Default) – Reuses input channel count.

3.14.1.2.6. "**audioVolume**" (or "**av**") – Audio output volume adjustment.

3.14.1.2.6.1. 0 – (Default) No volume adjustment.

3.14.1.2.6.2. 0 < n < 256 - Decrease volume by factor of (8 – log2(n)).

3.14.1.2.6.3. 256 – Base setting resulting in no volume adjustment.

3.14.1.2.6.4. 256 > n > 65536 - Increase volume by factor of (log2(n) – 8).

3.14.2. Encoding to produce multiple parallel outputs.

NGMS is able to produce multiple encoder outputs for the same input media. The same "*xcode*" parameter string is used to define multiple encoder configurations by assigning a unique output index to any configuration parameter. For eg., to define a second encoded output stream with a video bitrate of 500 Kb/s, include the index "**2"** following the parameter defining the bitrate, such as "*videoBitrate**2**=500"*. The second encoded stream will use the same properties as the first one, with the specified bitrate being unique. Up to four unique output instances can be defined.

To enable pass-thru encoding of the original input stream the codec type "*videoCodec=passthru"* is used, such as "***videoCodec2=passthru**,videoCodec2=h264,videoBitrate2=500"* . If "*videoCodec2=passthru"*, or "*videoCodec3=passthru*" is specified, the pass-thru output will always be placed into the first index.

Alternatively, "*passthru=on*" can be specified, which turns on pass-thru encoding into the first output index.

To access each additional encoder output stream the corresponding output index should be specified in the format specific client request. For eg., to access the encoder output stream defined with the index "*2*", "*/2*" should be appended to the format specific request for the stream. Such as "*http:///[NGMS IP]:[8080]/tslive**/2"***, or "*rtsp:///[NGMS IP]:1554/stream/**2"*.

Multiple encoder outputs can be used to provide adaptive bitrate streaming. If HTTPLive output is enabled, NGMS will automatically create a master playlist containing multiple output stream descriptions for HTTPLive adaptive bitrate stream switching.

3.15. Arguments controlling capture and recording

3.15.1. **–capture** - Enable live input stream capture.

3.15.2. **--in** - Input file, media location, or capture device. If using separate video and audio devices, the video device should be placed in the first index. Up to two devices are permitted. Refer to the stream output description for examples.

3.15.3. **–out** - Output destination or file name. "*—out*" or "*—stream*" is used in conjunction with "*—capture*" to re-stream a live input. Refer to the section describing arguments controlling stream output for output examples and semantics.

3.15.4. **--dir** – Input storage directory path for recording capture. If this option is absent any recorded file will be written to the NGMS home directory.

3.15.5. **--filter** – Capture specific input filter string. Each filter corresponds to the appropriate index of the input array. Filter parameters are supplied as a quoted comma separated list of key value pairs delimited by the "=" character.

3.15.5.1. Input capture filter required for all input stream types.

3.15.5.1.1. **"type"** - Input Stream designation type.

3.15.5.1.1.1. **"h264"** - H.264 over RTP using NAL packetization (RFC 3984).

3.15.5.1.1.2. **"mpg4"** - MPEG-4 Part 2 over RTP (RFC 3016).

3.15.5.1.1.3. **"h263"** - H.263.

3.15.5.1.1.4. **"h263+"** - H.263+ or H.263 plus.

3.15.5.1.1.5. **"aac"** - AAC over RTP (RFC 3640).

3.15.5.1.1.6. **"amr"** - AMR over RTP (RFC 3267).

3.15.5.1.1.7. **"g711u"** - G.711 mu-law over RTP.

3.15.5.1.1.8. **"g711a"** - G.711 a-law over RTP.

3.15.5.1.1.9. **"raw"** - Raw input. Raw stream data can be recorded "as-is" and is not dependant on codec specific packetization.

3.15.5.1.1.10. **"rgb"** - RGB888 (24 bits per pixel).

3.15.5.1.1.11. **"bgr"** - BGR888 (24 bits per pixel).

3.15.5.1.1.12. **"rgba"** - RGBA8888 (32 bits per pixel, 8 bit alpha mask).

3.15.5.1.1.13. **"bgra"** - BGRA8888 (32 bits per pixel, 8 bit alpha mask).

3.15.5.1.1.14. **"rgb565"** - RGB565 (16 bits per pixel, 8 bit alpha mask).

3.15.5.1.1.15. **"bgr565"** - BGR565 (16 bits per pixel, 8 bit alpha mask).

3.15.5.1.1.16. **"yuv420p"** - YUV420p (YUV 4:2:0 Y,UU, VV planar).

3.15.5.1.1.17. **"nv12"** - YUV420sp (YUV 4:2:0 Y, UV, UV semi-planar).

3.15.5.1.1.18. **"nv21"** - YUV420sp (YUV 4:2:0 Y, VU, VU semi-planar).

3.15.5.1.1.19. **"pcm"** - PCM 16 bit signed little endian.

3.15.5.1.1.20. **"alaw"** - PCM 8 bit a-law.

3.15.5.1.1.21. **"ulaw"** - PCM 8 bit mu-law.

3.15.5.1.1.22. **"m2t"** - MPEG-2 Transport Stream. This type should be used when downloading a live stream within an MPEG-2 TS container via HTTP.

3.15.5.1.1.23. **"flv"** - FLV file. This type should be used when downloading a live stream within an FLV container via HTTP.

3.15.5.2. Input capture filters useful for PCAP based capture.

3.15.5.2.1. **"dst"** - Destination IP Address.

3.15.5.2.2. **"src"** - Source IP Address.

3.15.5.2.3. **"ip"** - Source or Destination IP Address.

3.15.5.2.4. **"dstport"** - Destination Port.

3.15.5.2.5. **"srcport"** - Source Port.

3.15.5.2.6. **"port"** - Source or Destination Port.

3.15.5.2.7. **"pt"** - RTP specific Payload Type.

3.15.5.2.8. **"ssrc"** - RTP specific SSRC.

3.15.5.3. Input audio stream capture filters.

3.15.5.3.1. **"clock"** - Clock Rate in HZ (required for most Audio stream).

3.15.5.3.2. **"channels"** - Number of audio channels. (Defaults to 1).

3.15.5.4. Input video stream capture filters.

3.15.5.4.1. **"fps"** - FPS of input video stream.

3.15.5.4.2. **"clock"** - RTP Timestamp Clock Rate in HZ.

3.15.5.5. Input raw format filters.

3.15.5.5.1. **"width"** - Video input horizontal resolution.

3.15.5.5.2. **"height"** - Video input vertical resolution.

3.15.5.5.3. **"size"** - Video input resolution given as a single string delimited by "*x*" For eg. ("size=640x480").

3.15.5.6. Stream specific recording control.

3.15.5.6.1. **"file"** - Output path of recording file. If 'file' is not explicitly given the output file name will be automatically generated based on the input stream characteristics.

3.15.6. **--firxmit** – Controls how RTCP Feedback type Full Intra Refresh messages (RTCP FB FIR) as defined in RFC 5104, are sent from the application. The following behaviors are simultaneously affected with this parameter.

RTCP FB FIR messages will not be sent to the originator of a multicast RTP transmission unless the configuration file parameter "*RTCPReplyToMulticastStream*" is enabled.

3.15.6.1. Controls the transmission of RTCP FB FIR messages if requested by the local decoder. The default value is 1, FIR messages may be sent to the remote input capture transmitter if required by the decoding process. If set to 0, FIR messages requested by the local decoder are disabled. This value can be individually controlled by the configuration file parameter "*FIRSendFromDecoder*".

3.15.6.2. Controls the transmission of RTCP FB FIR messages if requested by the local RTP receiver. The default value is 0, FIR messages are not sent to the remote input capture transmitter if packet loss has led to corruption of the video bit-stream. If set to 1, FIR messages may be sent if the input video bit-stream corruption has been detected. This value can be individually controlled by the configuration file parameter "*FIRSendFromInputCapture*".

This value is automatically set to 1 if the input is an SDP file containing the Codec Control Message Full Intra Refresh video media attribute. For eg., *"a=rtcp-fb:* ccm fir".*

3.15.6.3. Controls the transmission of RTCP FB FIR messages when a client connects to a server published instance of the media output and the local output is not transcoded. An example is a connection to the HTTP URLs "/tslive", "/flvlive", "mkvlive", the RTSP, or the RTMP server listener. The default value is 1, which enables generation of an RTCP FB FIR message to the remote input capture transmitter if a key-frame is needed to begin the format specific output. If set to 0, no FIR message will be generated. This value has no effect if the local output is transcoded. This value can be individually controlled by the configuration file parameter "*FIRSendFromRemoteConnect*".

3.15.6.4. Controls the transmission of RTCP FB FIR messages upon reception of an RTCP FB FIR message from a remote RTP receiver. The default value is 0, FIR requests are not sent to the remote RTP transmitter upon reception of an RTCP FB FIR. If set to 1, FIR requests may be sent if the local output is not transcoded and FIR

message has been received from a remote RTP receiver. This value can be individually controlled by the configuration file parameter "*FIRSendFromRemoteMessage*".

3.15.7. **--maxrtpdelay** - Maximum RTP play-out buffer time controlling wait time for any out of order or lost packet. The default value is 100 ms.

3.15.8. **--overwrite** - Enable overwriting of recording output file if it already exists.

3.15.9. **--pes** - Enable de-mux of each packetized elementary stream and subsequent re-mux.

3.15.9.1. Not specified - (Default) Use direct replay of input MPEG-2 TS stream without decomposing each PES frame.

3.15.9.2. Specified – Decompose and recompose each PES frame. This is automatically turned on if transcoding is enabled, or for non MPEG-2 TS based capture.

3.15.10. **--queue** – Input packet queue number of slots. Applicable for MPEG-2 Transport Stream capture and recording. Actual size in bytes = "*—queue*=" * 12408.

3.15.11. **--realtime** - Enable download of capture input in real-time according to embedded input file timestamps. Useful for HTTP

based capture, such as when loading a static FLV file from a web server to be streamed in real-time. If this option is not present, the input will be loaded without throttling and may overflow any input capture queue.

3.15.12. **--rtcprr** – RTCP Receiver Report interval in seconds. A non-zero value enables transmission of Receiver Reports to the sender. RTCP Receiver Reports are disabled by default. To enable RTCP Receiver Reports it is recommended to use a reasonable interval between 2.0 and 8.0 seconds. The interval of 5.0 seconds will be used If no value is specified. This value overrides the configuration file parameter "*RTCPReceiverReportInterval*".

RTCP messages will not be sent to the originator of a multicast RTP transmission unless the configuration file parameter "*RTCPReplyToMulticastStream*" is enabled.

3.15.13. **--rtmpnosig** – Disable use of a digital signature in RTMP handshake.

3.15.14. **--rtmppageurl** – Specify an optional *RTMPPageUrl* string parameter used in the protocol Connect packet when connecting as a stream client to an RTMP server.

3.15.15. **--rtmpswfurl** – Specify an optional *RTMPSwfUrl* string parameter used in the protocol Connect packet when connecting as a stream client to an RTMP server.

3.15.16. **--rtpframedrop** – The RTP input capture frame dropping policy. This policy applies to the way video frames are re-assembled from RTP input packetization.

3.15.16.1. **0** – Do not attempt to drop any frames when packet loss is detected for an incoming video stream. This policy may pass corrupted video frames to the local decoder or to any stream output if the stream is not transcoded.

3.15.16.2. **1** – Attempt to drop a video frame when packet loss is detected for an incoming video stream. This policy may discard reference and non-reference video frames if sufficient corruption of the video payload is detected. This is the default drop policy.

3.15.16.3. **2** – Attempt to drop a video frame when any packet loss is detected for an incoming video stream. This policy will discard any video frame that is corrupted due to packet loss.

3.15.17. **--audq** – Input audio capture queue number of slots. The default value is specific to the input audio codec.

3.15.18. **--vidq** – Input video capture queue number of slots. The default value is specific to the input video codec.

3.15.19. **--retry** – Enable input method specific retry logic upon failure. Applicable to active capture methods such as HTTP.

3.15.19.1. 0 - (Default) No input retry.

3.15.19.2. 1 - Indefinite amount of retries.

3.15.19.3. n - (n – 1) number of consecutive retries.

3.15.20. –stunrespond – Enable STUN binding responses to be sent when receiving STUN binding requests on listener RTP / RTCP sockets. This argument can be followed by an optional STUN password parameter such as *'—stunrespond=5rGs9Ow+ST/MM3Sc'*. The STUN password is used for computing a MESSAGE-INTEGRITY HMAC STUN attribute. If the optional password is not given as a parameter, the value of the input SDP attribute "a=ice-ufrag:" will be used as the password.

3.16. Arguments controlling Video Conferencing

NGMS can run as a video conferencing bridge for up to eight participants. A participant's video and audio streams are added to the video conference similar to how a PIP (Picture In Picture) is added through an HTTP(s) interface.

An audio mixer is used to combine multiple audio streams into a common audio feed. The mixer provides a unique output for each conference participant's output audio stream, removing the participant's own input audio stream from the mixed output.

The video conference can be broadcast to an unbounded number of read-only viewers using one of the many output formats supported by the server. The conference can be recorded for future replay.

The following command line arguments control video conferencing. Video conferencing mode is enabled if either the *'–conference'* or *'–mixer'* argument is present on the command line.

3.16.1. –conference - Enables video conferencing mode. Video Conferencing mode allows video and audio streams to be added and removed from the conference using the PIP HTTP(s) interface.

3.16.2. –in - Input media file or still image path to be used as default background of the conference overlay.

3.16.3. –layout - Sets the layout configuration of the arrangement of participant video on the main output overlay. Valid configurations are below:

3.16.3.1. "grid" - Grid layout. Video participants are arranged in a grid-like pattern. This is the default configuration.

3.16.3.2. "vad" - Active speaker switching based on VAD analytics. The active speaker's video will cover most of the output video overlay.

3.16.4. –mixer - Enables or disables the audio mixer. The audio mixer is enabled by default when the *'—conference'* argument is present. To disable the mixer use *'—mixer=0'*.

3.16.5. **–mixeragc** - Enables or disables audio Automatic Gain Control. AGC is enabled by default when using the audio mixer. To disable AGC use *'—mixeragc=0'*.

3.16.6. **–mixerdenoise** - Enables or disables audio denoise filtering. Audio denoise filtering is enabled by default when using the audio mixer. To disable denoise filtering use *'—mixerdenoise=0'*.

3.16.7. **–mixervad** - Enables or disables voice activity detection. VAD is enabled by default when using the audio mixer. To disable VAD use *'—mixervad=0'*.

3.16.8. **--piphttp** – PIP control interface server listening address and port string delimited by a colon. The PIP control interface is used to add and remove media sessions from the conference. To enable a listener on port 8080 use "*8080*" or "http://*0.0.0.0:8080*". To enable a listener on localhost use "http://*127.0.0.1:8080*". To enable an SSL/TLS listener use "https://". An HTTP client will access the server at *http://[address]:[port]/pip*.

3.16.9. **–piphttpmax** – Maximum number of simultaneous HTTP PIP control connections. The default value is 0 (PIP Control interface disabled). The PIP control server can be used to dynamically add and remove a PIP.

3.17. Arguments controlling broadcast control mode

3.17.1. **–broadcast** - Enables broadcast control mode.

3.17.2. **–listen** - HTTP local address and port defining local HTTP command web server. Used to override configuration file entry. For eg. *"—listen=http://127.0.0.1:8080*"

3.17.3. **–media** – Root directory for all media files exposed via the web server interface.

3.17.4. **–home** - NGMS Home installation directory.

3.17.5. **–dbdir** - Media database directory. The default value is *[media dir]/ngmsdb*. A non-default value may be needed if the media directory is a read-only location, such as a CD/DVD. The media database stores media file meta-information and preview thumbnails.

3.17.6. **–nodb** - Disables media database directory creation and updates.

3.18. Command line examples

3.18.1. Streaming media content

3.18.1.1. Stream a file via MPEG-2 TS encapsulation over RTP to a remote host listening on 10.10.10.10:5004. *"–transport=m2t*" is the default encapsulation and does not need to be specified.

./bin/ngms --in="path/inputfile.mp4" –stream=rtp://10.10.10.10:5004 –transport=m2t

3.18.1.2. Stream a file via native codec specific encapsulation RTP to a remote host listening at the IP address 10.10.10.10, with the video stream on port 5004 and the audio stream on port 5006.

./bin/ngms --in="path/inputfile.mp4" –stream=rtp://10.10.10.10:5004,5006 –transport=rtp

3.18.1.3. Stream a file via native codec specific encapsulation over RTP to a remote host at 10.10.10.10, with the video stream on port 5004 and the audio stream on port 5006. The stream is also available via MPEG-2 TS over HTTP ("*/tslive*"), HTTP Live ("*/httplive*"), both on port 8080, via RTMP on port 1935, and via RTSP on port 1554. The HTTP resources are also accessible via SSL/TLS on port 8443. Adding "*—live=*" allows automatic User-Agent based stream output method selection when connecting to the URL "*/live*".

./bin/ngms --in="path/inputfile.mp4" –stream=rtp://10.10.10.10:5004,5006 –transport=rtp –tslive=8080 –tslive=https://8443 –httplive=8080 –httplive=https://8443 –rtmp=1935 –rtsp=1554 –live=8080 –live=https://8443

3.18.1.4. Record a live MPEG-2 TS capture sent from localhost via UDP on port 41394. The content will be saved to *out.ts* and any prior file content will be automatically overwritten.

./bin/ngms --capture="udp://127.0.0.1:41394" --filter="type=m2t,file=out.ts" –overwrite

3.18.1.5. Record a live input capture sent from a remote host defined by the Session Description Protocol file *input.sdp*. The content will be saved in FLV format to the file *output.flv* and any prior file content will be automatically overwritten.

./bin/ngms --capture="input.sdp" --flvwrite=output.flv –overwrite

3.18.1.6. Stream a live input stream. The live MPEG-2 TS stream is captured via RTP on port 41394. The stream is sent via RTP to the remote host at 10.10.10.10 via MPEG-2 TS encapsulation. The stream is also available via HTTP Live on port 8080 and via RTSP on port 1554.

./bin/ngms --capture="udp://41394" --filter="type=m2t" –stream=rtp://10.10.10.10:5004 –httplive=8080 –rtsp=1554

3.18.1.7. Stream a live input stream described by a Session Description Protocol file. The stream is sent via RTP to a remote host at 10.10.10.10 via MPEG-2 TS encapsulation. The stream is also available via HTTP Live on port 8080 and via RTSP on port 1554.

./bin/ngms --capture="inputstream.sdp" –stream=rtp://10.10.10.10:5004 –httplive=8080 –rtsp=1554

3.18.1.8. Stream a file via MPEG-2 TS encapsulation over RTP to a remote host at 10.10.10.10:5004. *"–transport=m2t*" is the default encapsulation and does not need to be specified. The output is transcoded to H.264 at 300Kb/s, 320x240 at a constant 25fps. Audio output is AAC mono at 48KHz.

./bin/ngms --in="path/inputfile.mp4" –stream=rtp://10.10.10.10:5004 --xcode="videoCodec=264,videoBitrate=300,vx=320,vy=240,videoFps=25,videoOutputConstantFps=1,videoUpsampling=1,audioCodec=aac,audioBitrate=64000,audioSampleRate=44100,audioChannels=1"

3.18.1.9. Stream a live input stream described by a Session Description file an offer it for download via HTTP Live. The output is transcoded into two unique bitrates and automatically packaged into an adaptive bitrate compatible *.m3u8* playlist file available at the URL "*/httplive*". Each bitrate specific HTTP Live output stream is also available at the transcoding output specific index URL, such as "*/httplive/**1***" and "*/httplive/**2***".

*./bin/ngms --in="test.sdp" –httplive=http://8080 –httplive=https://8443 --xcode="videoCodec=264,videoBitrate=300,vx=320,vy=240,videoFps=25,videoOutputConstantFps=1,videoUpsampling=1,audioCodec=aac,audioBitrate=64000,audioSampleRate=44100,audioChannels=1,videoBitrate**2**=150"*

3.18.1.10. Stream and transcode a live input stream to be viewed by an HTML5 capable web browser such as Chrome. The video is encoded in VP8 and the audio in Vorbis and encapsulated in a WebM container format. The output can be viewed in an HTML5 capable web browser by connecting to http:///*[NGMS IP]:[8080]/mkv or* https:///*[NGMS IP]:[8443]/mkv*

./bin/ngms --in="test.sdp" --mkvlive=http://8080 –mkvlive=https://8443 --xcode="videoCodec=vp8,videoBitrate=300,vx=320,vy=240,videoFps=25,videoEncoderSpeed=3,videoOutputConstantFps=1,videoUpsampling=1,audioCodec=vorbis,audioBitrate=32000,audioSampleRate=44100,audioChannels=1"

3.18.2. Video Conferencing Setup

3.18.2.1. Setup a video conferencing server. The video output will be encoded using the VP8 codec at 15fps, and each input participant aspect ratio will be preserved in the output picture. The audio mixer will run at 16KHz and the default audio output will use the AAC codec. Participants can be added and removed by connecting to the PIP server listener on HTTP port 8080. The video conference output is available using */tslive* on port 8080 and *RTSP* on port 1035.

./bin/ngms --conference –piphttp=8080 --xcode="videoCodec=vp8,videoEncoderSpeed=3,videoFps=15,videoG

OPMaxMs=2000,videoGOPMinMs=1000,padAspectRatio=1,audioCodec=aac,audioSampleRate=16000" –tslive=8080 –rtsp=1035

The following HTTP request URL will add a conference participant with VP8 video output sent to 10.10.10.10:2000 using RTP payload type 97, and SILK audio output at 8KHz to port 2002 using RTP payload type 96.

"http://[NGMS IP]:8080/pip?pipstart&in=live1.sdp&out=rtp://10.10.10.10:2000,2002&rtppayloadtype=97,96,audioCodec=silk,audioSampleRate=8000"

*The conference can be viewed by c*onnecting to the */tslive* listener at *http://[NGMS IP]:8080/tslive.* The audio output available will be AAC at 16KHz. The RTP output to the conference participant will be encoded using SILK at 8KHz. Each audio participant will receive independently mixed audio output which does not contain it's own audio input samples.

3.18.3. Miscellaneous

3.18.3.1. Analyze video elementary stream within a container file.

./bin/ngms --analyze=inputfile.mp4

Found SPS timing: 2500000 Hz, tick 104271 Hz (23.976 fps)
inputfile.mp4 size: 119946.0KB, duration: 00:02:03.1231
2500000 Hz, tick 104271 Hz (23.9760 fps)

H.264 High profile (0x64) level 41
1920x816 CABAC poc:0 ref_frames:3 frame_mbs_only YUV420
slices:6077 vcl:6075 decode errors:0
I:170, P:1635, B:1148, SI:0, SP:0, SPS:1, PPS:1, SEI:3122, unknown:0
frames: 2953, 2.1 slices/fr GOP avg: 17.4, max: 24
I:170 84.6KB/fr, P:1635 47.5KB/fr, B:1148 24.3KB/fr
bandwidth avg: 7980.6Kb/s, max(1sec): 15821.9Kb/s, (300ms): 19265.8Kb/s

Add the --verbose option to dump the entire contents of the H.264 SPS / PPS and to enumerate and examine each NAL unit.

3.18.3.2. Create an mp4 file given a raw H.264 Annex B formatted file. The output here will take the name test.mp4.

./bin/ngms --create=input_annex_b.h264 –out=test.mp4

3.18.3.3. Create or add a track to an existing mp4 file given a raw AAC ADTS formatted input file.

./bin/ngms --create=input_adts.aac --out=test.mp4

--fps=[video frame rate] may need to be explicitly given if no timing information is contained within the SPS in the raw H.264 Annex B file.

3.18.3.4. Dump the entire box structure of an mp4 container file.

./bin/ngms --dump=test.mp4

Use --verbose to include detailed sample description box contents.

To dump the entire frame structure and summary of an MPEG2-TS file.

./bin/ngms --dump=test.m2t

To dump the decoding time-to-sample of an mp4 track to a file.

./bin/ngms --dump=test.mp4 --stts > test.stts

The dumped STTS info contains the sync between the audio and video track and can be manually edited to alter lip-sync. Use *--stts=[dumped stts file]* as an argument when creating an mp4 container file to preserve audio video sync.

3.18.3.5. Extract the raw H.264 and AAC contents of an mp4 container.

./bin/ngms --extract=test.mp4

This will create *test.h264* containing the H.264 NALs in Annex B format and *test.aac* in AAC ADTS format.

"*—extract"* can be used to convert *.flv* files to *.mp4* by extracting the raw contents and then creating an mp4 container. *.mp4* is preferred over *.flv* to allow fast-play of download files and better seeking during playback.

To extract the raw video and audio contents of an MPEG-2 TS file.

./bin/ngms --extract=test.m2t

If the MPEG-2 TS file was captured from broadcast TV, this will create *test.h262* containing the MPEG-2 elementary stream and *test.ac3* containing the *a52* (dolby digital) audio contents.

4. Streaming Transport Protocols

4.1. Stream Output via RTP

NGMS can stream output via RTP or SRTP (Secure-RTP) to both unicast and multicast addresses. The same stream can be output to multiple unicast addresses to allow display on multiple recipient clients.

4.2. Stream Output via RTSP

NGMS has a built-in RTSP server to allow clients to view live content via RTSP. Media is encapsulated and delivered via RTP over UDP or TCP interleaved mode. The server supports both RTSP and RTSP tunneled over HTTP. The following URL can be used directly by an RTSP player:

rtsp://[NGMS IP]:[1554]/rtsp/live

From any Web Browser the following URL can be used to automatically link to the RTSP server:

http://[[NGMS IP]:[8080]/live

or

http://[NGMS IP]:[8080]/rtsp

To customize the presentation of the RTSP HTML web page made available to clients, edit the file *html/rsrc/rtsp_embed.html.*

NGMS can also capture a live RTSP stream as a form of input.

4.3. Stream Output via RTMP

NGMS has a built-in RTMP server to allow clients to view live content via RTMP. Media is encapsulated via the RTMP protocol allowing an RTMP player such as Flash player to load and view live content. The following URL can be used directly by an RTMP player:

rtmp://[NGMS IP]:[1935]/rtmp/live

From any Web Browser the following URL can be used to view the live stream via an embedded Flash applet.

http://[NGMS IP]:[8080]/live

or

http://[NGMS IP]:[8080]/rtmp

To customize the presentation of the RTMP HTML web page made available to clients, edit the file *html/rsrc/rtmp_embed.html.*

NGMS can also load a live or pre-recorded RTMP in client mode or server mode as a form of input.

4.4. Stream Output via FLV encapsulation over HTTP

NGMS has a built-in FLV output server to allow clients to view live content via FLV encapsulation. Media is encapsulated via the FLV file format allowing an FLV file handler such as Flash player to load and view live content as if it were a static file. The following URL can be used directly by an FLV player:

http://[NGMS IP]:[8080]/flvlive

From any Web Browser the following URL can be used to view the live stream via an embedded Flash applet.

http://[NGMS IP]:[8080]/live

or

http://[NGMS IP]:[8080]/flv

Multiple HTTP listeners can be used to provide both HTTP and HTTPS listeners using the following options "*--flvlive=8080 –flvlive=https://8443 –live=8080 –live=https://8443*". A web browser connecting to "*https://[NGMS IP]:8443/flv*" *or* "*https://[NGMS IP]:8443/live*" will automatically receive a link to access the underlying FLV encapsulated media stream via SSL/TLS.

To customize the presentation of the FLV HTML web page made available to clients, edit the file *html/rsrc/http_embed.html.*

NGMS can also load a live or pre-recorded FLV encapsulated stream via HTTP or HTTPS as a form of input.

4.5. Stream Output via Matroska / WebM encapsulation

NGMS has a built-in Matroska / WebM output server to allow clients to view live content via Matroska / WebM encapsulation. A WebM media file contains VP8 video and Vorbis audio using Matroska encapsulation with the document type "*webm*". Media is encapsulated via the Matroska file format allowing a Matroska or WebM file handler such as the Chrome browser to load and view live content as if it were a static file. The following URL can be used directly by a Matroska or WebM aware Web Browser:

http://[NGMS IP]:[8080]/mkvlive

From a Matroska or Webm aware Web Browser the following URL can be used to view the live stream.

http://[NGMS IP]:[8080]/live

or

http://[NGMS IP]:[8080]/mkv

Multiple HTTP listeners can be used to provide both HTTP and HTTPS listeners using the following options "*--mkvlive=8080 –mkvlive=https://8443 –live=8080 –live=https://8443*". A web browser connecting to "*https://[NGMS IP]:8443/mkv*" *or* "*https://[NGMS IP]:8443/live*" will automatically receive a link to access the underlying Matroska / WebM encapsulated media stream via SSL/TLS.

To customize the presentation of the Matroska HTML web page made available to clients, edit the file *html/rsrc/mkv_embed.html.*

4.6. Stream Output via MPEG-DASH over HTTP

NGMS has a built-in MPEG-DASH compatible output server to allow DASH enabled clients to download and view live content. NGMS maintains and publishes one or more Media Playlist Description (MPD) files which include program stream meta-data. Media is encapsulated in ISO Base Media File Format (BMFF) using an mp4 container file containing movie fragment (MOOF)

boxes. The following URLs can be used to access an MPD containing the *SegmentList* XML element:

http://[NGMS IP]:[8080]/dash

or

http://[NGMS IP]:[8080]/dash/default.mpd

or

http://[NGMS IP]:[8080]/dash/seglist.mpd

The following URL can be used to access an MPD containing the *SegmentTemplate* XML element:

http://[NGMS IP]:[8080]/dash/segtemplate.mpd

Multiple HTTP listeners can be used to provide both HTTP and HTTPS listeners using the following options *"--dash=8080 –dash=https://8443*". A DASH enabled client can connect to "*https://[NGMS IP]:8443/dash/default.mpd*" *or* "*https://[NGMS IP]:8443/dash/default.mpd*" to access the published MPD.

4.6.1. Segment duration

NGMS creates distinct media segments with a duration between 5 and 10 seconds by default. Each segment should begin with a video key frame. To

decrease the real-time media delay, change the configuration line item "*dashmaxduration*" to a lower value.

4.6.2. Segment output location

When using NGMS to continuously host MPEG-DASH Streaming content it is recommended to change the directory of the continuously updated segments to be mapped to an in-memory file system. This will reduce unnecessary disk activity.

- *sudo mkdir /usr/local/ram*
- *sudo mount -t tmpfs -o size=102400K tmpfs /usr/local/ram/*
- Update the configuration line item *dashdir=/usr/local/ram/*

4.6.3. MPEG-DASH Adaptive Bitrate Streaming

NGMS will automatically produce multiple MPEG-DASH segmented output streams when multiple output transcoding output is enabled. Each published MPEG-DASH MPD will contain multiple representation elements describing each output stream.

4.7. Stream Output via MPEG-2 TS over HTTP

To stream across the internet or through firewalls any clients may need to receive a live broadcast over TCP / HTTP instead of UDP / RTP. Streaming over HTTP

can be used simultaneously while delivering content over RTP. NGMS uses the Content-Type: *video/x-mpeg-ts* in the HTTP headers.

The live stream can be accessed at the URL:

http://[NGMS IP]:[8080]/live

or

http://[NGMS IP]:[8080]/tslive

Multiple HTTP listeners can be used to provide both HTTP and HTTPS listeners using the following options "*--tslive=8080 –tslive=https://8443 –live=8080 –live=https://8443*".

NGMS can also load a live MPEG-2 TS stream via HTTP or HTTPS as a form of input capture.

4.8. HTTP Live Streaming

NGMS has full built-in support for HTTP Live Streaming. HTTP Live Streaming is a standard proposed by Apple which is used to deliver live media to any Apple mobile device such as iPhone, iPod, or iPad, as well as any Mac OS X (>=10.6) with Safari and QuickTime X. HTTP Live streaming works by segmenting a live output stream into small distinct chunks. These chunks are referenced from a continuously updated playlist file available for download via HTTP. This mechanism has an inherent delay of 15-35 seconds, depending on the configuration settings.

The live stream can be accessed from a supported Apple at the URL:

http://[[NGMS IP]:[8080]/live

or

http://[NGMS IP]:[8080]/httplive

Multiple HTTP listeners can be used to provide both HTTP and HTTPS listeners using the following options "*--httplive=8080 –httplive=https://8443 –live=8080 –live=https://8443*". A web browser connecting to "*https://[NGMS IP]:8443/httplive*" *or* "*https://[NGMS IP]:8443/live*" will automatically receive a link to access the underlying media via SSL/TLS.

To enable HTTP Live Streaming ensure that the iPhone / HTTPLive checkbox is selected in the UI. This will enable any media file, or a live captured stream to be available through the HTTP Live interface.

Live streaming media can also be opened from Safari on any Mac with Quicktime X. When opening the URL directly from Quicktime X use /httplive/out.m3u8 (eg. *http://10.10.10.10:8080/httplive/out.m3u8*).

NGMS can also load a live HTTPLive playtlist stream via HTTP or HTTPS as a form of input.

4.8.1. Chunk segment duration

NGMS creates the recommended duration of 10 seconds segment chunks by default. The last three segments are stored in the current .m3u8 playlist file. To decrease the real-time media delay, change the configuration line item "*httpliveduration*" from 10.0 seconds to 5.0. A value of less than 5 seconds duration is not recommended.

4.8.2. Chunk output location

When using NGMS to continuously host HTTP Live Streaming content it is recommended to change the directory of the continuously updated chunks to be mapped to an in-memory file system. This will reduce unnecessary disk activity.

- *sudo mkdir /usr/local/ram*
- *sudo mount -t tmpfs -o size=102400K tmpfs /usr/local/ram/*
- Update the configuration line item *httplivedir=/usr/local/ram/*

NGMS comes with integrated functionality to provide a complete HTTP Live Streaming solution by both automatically segmenting an output stream and hosting it via the bundled web server. This approach requires virtually no configuration. However, for large scale production environments which handle a large amount of HTTP requests it may be desirable to use a third-party web server such as Apache to deliver the media content to clients. In such case:

- Update the configuration line item "*httplivedir="* to an existing path within your web server directory tree.

- In the same directory, create your own index.html with the following tag placed inside the body tag. *<video controls autoplay src="out.m3u8"/>*

4.8.3. HTTPLive Adaptive Bitrate Streaming

NGMS will automatically produce multiple HTTPLive segmented output streams when multiple output transcoding output is enabled. The default HTTPLive HTTP request ("*http:///[NGMS IP]:8080/httplive")* index file will return an embedded video tag pointing to a master "*multi.m3u8"* playlist file. The master playlist file includes references to each bitrate specific playlist file for automatic selection by an HTTPLive client. Alternatively, an HTTPLive client can request a bitrate specific output stream by specifying its output index, such as "*http:///[NGMS IP]:[8080]/httplivelive/**2***."

5. NGMS Web Server

The NGMS Web Server is available over HTTP on the default port 8080. The same web services can be available on multiple *address:port* listeners as well as SSL/TLS.

5.1. The NGMS integrated Web Server is used to host the following virtual URLs when live stream output is enabled.

5.1.1. **/config** - Provides a dynamic runtime configuration interface. Available if "*—configsrv*" is given. The configuration interface is used by NGVX to perform runtime updates such as conference participant hold, STUN / ICE binding updates, and RTP streaming address changes. The configuration interface can also be used to adjust transcoder settings such as bitrate adjustments. The following URI parameter keys are supported.

5.1.1.1. **"xcode"** - An updated transcoder configuration passed as a comma separated list of key value pairs. Any running encoder configuration settings will be updated on the fly. The actual underlying running encoder must support this functionality.

5.1.1.2. **"reset" -** Set to 1 to initiate a full reset of the transcoder configuration. This will cause any transcoders to be closed and re-opened.

5.1.2. **/dash** - Loads the default MPEG-DASH MPD. Available if the *"—dash*" option is present. This URL is the same as "***/dash/default.mpd***".

5.1.2.1. **/dash/seglist.mpd** - Loads the MPD variant which uses the *SegmentList* element to describe the published media.

5.1.2.2. **/dash/segtemplate.mpd** - Loads the MPD variant which uses the *SegmentTemplate* element to describe the published media.

5.1.3. **/flv** - Returns an embedded Flash player to access live output using FLV encapsulation as if loading a static FLV file. Available if the "*—flvlive*" option is present.

5.1.4. **/flvlive** - Direct access to live media output encapsulated using FLV over HTTP. Available if the "*—flvlive*" option is present.

5.1.5. **/live** - Examines the client User-Agent HTTP Header and automatically chooses what output format to return. Available if the '—live' option is present.

The "*/live*" URL is intended to be the universal connection path to an input stream from any client. NGMS will adapt the output format of the stream to be the same as either "*/rtmp*", "*/rtsp*", "*/flvlive"*, or "/*httplive*". The specific stream format is determined based on the User-Agent header lookup of the device type configuration defined in *"/etc/devices.conf"*

5.1.6. **/httplive** - Direct access to live media output for Apple HTTP Live Streaming clients. Available if the *"—httplive*" option is present.

5.1.7. **/mkv** - Returns an embedded HTML5 video tag to access live output using Matroska / WebM encapsulation as if loading a static Matroska / WebM file. Available if the "*—mkvlive*" option is present.

The "***/mkv***" URL is synonomous with "***/webm***".

5.1.8. **/mkvlive** - Direct access to live media output encapsulated using Matroska / WebM over HTTP. Available if the "*—mkvlive*" or "*—webmlive*" option is present.

5.1.9. **/pip** - Allows for runtime configuration of a PIP (Picture In Picture). This URL is also used to add and remove participants to a video conference.

Configuration parameters are passed as URL key value pairs to the HTTP(s) GET request to the */pip* URL.

5.1.9.1. **"pipstart"** – If present will try to start a new PIP. The result of the operation is returned in the HTTP Response as a result code equal to the index of the new PIP. The result code should be used as the parameter to "pipstop=" to stop the PIP. A code < 0 indicates an error. For eg. *"http://[NGMS IP]:8080/pip?pipstart&in=liveinput.sdp&pipxright=0&xcode=videoWidth=160,videoHeight=120"* starts a new PIP overlay with dimensions of 160x120 pixels located at the upper right corner of the main overlay.

5.1.9.2. **"pipstop"** – Stops the PIP at the given index. The result of the operation is returned in the HTTP Response as a result code where 0 indicates success and a code < 0 indicates an error. For eg. "*http://[NGMS IP]:8080/pip?pipstop=1"* stops the PIP at index 1.

5.1.9.3. **"in"** – The PIP input media file path or SDP file to be processed. For eg., *"&in=liveinput.sdp*" or *"&in=logo.png*"

5.1.9.4. **"out"** – The RTP output destination of the video conference participant. This option is only valid if using video conferencing mode (if *'–conference'* or *'–mixer'* is present on the command line). For eg., *"&out=rtp://10.10.10.10:5004,5006"* .

5.1.9.5. **"rtppayloadtype"** – PIP RTP output stream payload type(s). To set the video payload type to 112, and the audio to 96 use *"&rtppayloadtype=112,96"*. The default RTP payload type values are codec specific. This option is only valid when using the *"&out="* parameter to add a video conference endpoint.

5.1.9.6. **"xcode"** – Any PIP formatting configuration passed as a list of key value pairs. These options take the same format as the *"—xcode*" command line parameters documented under "*Transcoder Configuration*". Only options specific to PIP output dimensions, scaling type, cropping, padding, and frame rate (applicable for non static PIP formats) are processed. PIP audio parameters are only valid if using the video conferencing mixer. For eg., to specify the PIP RTP output audio stream to be encoded at 8KHz with the SILK codec use *"&xcode=audioCodec=silk,audioSampleRate=8000"* .

5.1.9.7. **"pipalphamax"** – (PIP) maximum alpha masking value. Range is from 0 – 255, with 255 being the default, resulting in no transparency. This value caps the maximum alpha transparency of any pixel if the PIP image contains an alpha mask. A lower value results in greater transparency of the PIP.

5.1.9.8. **"pipalphamin** – (PIP) Picture In Picture minimum alpha masking value. Range is from 0 – 255, with 0 being the default. This value caps the minimum alpha transparency of any pixel if the PIP image contains an alpha mask. A greater value results in greater less transparency of the PIP.

5.1.9.9. **"pipx"** – The horizontal (*x axis*) placement of the left edge of the PIP relative from the left edge of the main picture.

5.1.9.10. **"pipxright"** – The horizontal (x axis) placement of the edge corner of the PIP relative from the right edge of the main picture.

5.1.9.11. **"pipy"** – The vertical (*y axis*) placement of the top edge of the PIP relative from the top edge of the main picture.

5.1.9.12. **"pipybottom"** – The vertical (y axis) placement of the bottom edge of the PIP relative from the bottom edge of the main picture.

5.1.9.13. **"pipzorder"** – The PIP z axis placement order as a signed integer. The default z-order index is 0. A higher z-order will result in the placement of the PIP covering any other PIPs with a lower z-order index.

5.1.10. **/rtmp** - Returns an embedded Flash player to access live output via RTMP. Available if the "*—rtmp*" option is present.

5.1.11. /rtsp - Returns a link to the live output available via RTSP. Available if the "*—rtmp*" option is present.

5.1.12. /status - Returns status information about the server. Available if "*—statusmax*" is set to > 0. The following URI parameter keys are supported to control which output is displayed. If no URI parameters are present, the '*output*' option is used as the default. For eg. the following URL can be used to view the current stream statistics: *"http://[NGMS IP]:[8080]/status?streamstats*".

5.1.12.1. "output" - Shows how many active output sessions are being serviced. The intention of the output status URL is to be used by the NGMP web portal to determine how many concurrent output sessions each stream processor is handling.

5.1.12.2. "streamstats" – Display the stream output statistics. Statistics include the overall throughput, average burst rate over the past 2 seconds, and past 8 seconds. RTP stream output will contain any RTCP Receiver Report metrics such as reported packet loss. TCP stream output will contain the current state of the output buffer. The statistics response is returned as URL key value pairs. The same statistics are available via the '*—streamstats*' command line argument.

5.1.13. /tslive - Direct access to live media output encapsulated over MPEG-2 TS. Available if the "*—tslive*" option is present.

5.2. When NGMS is run in broadcast control mode the following virtual URLs are hosted.

5.2.1. **/** - Returns a broadcast control panel used to control a live server broadcast.

6. Frequently Asked Questions

6.1. Failure to start

6.1.1. NGMS Fails to start due to the following linker error:

./bin/ngms: error while loading shared libraries: libngms.so:
cannot open shared object file: No such file or directory

or

./bin/ngms: error while loading shared libraries: libxcode.so:
cannot open shared object file: No such file or directory

This means that the system library loader cannot find the NGMS shared libs. The following command tells the shell where to look for shared libraries:

export LD_LIBRARY_PATH=./lib

6.1.2. NGMS Fails to start due to the following system message:

./bin/ngms: error while loading shared libraries: libngms.so: cannot restore segment prot after reloc: Permission denied

or

./bin/ngms: error while loading shared libraries: libxcode.so: cannot restore segment prot after reloc: Permission denied

On some systems certain kernel security extensions may prohibit the shared libraries from loading correctly. To override this do:

chcon -t texrel_shlib_t lib/libngms.so
chcon -t texrel_shlib_t lib/libxcode.so

6.1.3. Unable to find license file:

If you have just installed the license file 'license.dat' into "*etc\license.dat*" but NGMS fails to read the license ensure that you have the entry "*license=etc\license.dat*" in the NGMS configuration file "*etc\ngms.conf*". This may occur if you are starting NGMS by double-clicking from File Explorer.

6.2. Problems trying to loading SSL services.

6.2.1. Some clients may fail to load media links via HTTPS (SSL) if the media URL being referenced from a web page points to a different origin than the web page. For instance, if publishing HTTPLive

services over HTTPS, the URL *https://[NGMS FQDN / IP]/httplive* may fail to load any media from iOS devices if the *.m3u8* playlist contains a different originating URL. This can be addressed by setting the *"localHost"* configuration option in the NGMS configuration file to be consistent with the server's public FQDN.

In ngms.conf

*localhost=**httplive.cdn.mycompany.com***

An example NGMS command line to publish an HTTPLive stream using SSL on port 8443 would be:

*./bin/ngms --in=input.mp4 --httplive=https://0.0.0.0:8443 httpliveurlhost=https://**httplive.cdn.mycompany.com**:8443/httplive".*