Nex Gen Media Server (NGMS) Core Library

v1.4.2

Developer API Description

Contents

1. Introduction

The NGMS core library provides all of the streaming, capture, transport, transcoding services, which are used by NGMS. NGMS core is written in C and compiled with *gcc*. It can be statically or dynamically linked by your C / C++ application. The core library is self-contained and portable to many operating environments, such as Linux, Windows, OSX, Android, iOS. NGMS core can be used in Java applications by wrapping a JNI native layer over it.

2. Function Reference

The following NGMS core functions can be directly invoked by your application.

2.1. ngmslib_open

Opens and initializes a single NGMS instance.

2.2. ngmslib_close

Closes an NGMS instance which was previously opened.

2.3. ngmslib_stream

Starts the streaming service. The streaming service is used to:

2.3.1. Stream a container file from storage.

2.3.2. Stream a captured live network stream.

2.3.3. Stream live content from a local camera / microphone.

2.3.4. Send live captured content to a local framebuffer / speakers.

2.4. ngmslib_update

Updates a stream initiated with a prior call to *ngmslib_stream***.** Only transcoding parameters or timeout values may be updated.

2.5. ngmslib_onVidFrame

Supplies a raw input video frame from a local frame buffer / camera to the NGMS framework for processing.

2.6. ngmslib_onAudSamples

Supplies one ore more input audio samples from a local audio source / microphone to the NGMS framework for processing.

2.7. ngmslib_readVidFrame

Reads a single raw video output frame from the NGMS framework. The video frame can be rendered locally.

2.8. ngmslib_readAudSamples

Reads one or more captured raw audio output samples to be played by the local sound device.

2.9. ngmlib_registerReportCb

Registers a callback to be invoked from within the NGMS framework. This mechanism allows for reporting transmission statistics and remote reception statistics.

2.10. ngmlib_showSdp

Returns the SDP contents for the selected output configuration. The SDP can be referenced by an external control channel layer such as SIP.

2.11. ngmslib_captureNet

Starts the recording service.

2.12. ngmslib_createMp4

Used to create an mp4 container file from raw audio / video input.

2.13. ngmslib_extractMedia

Extracts raw audio / video input from a container file.

3. Startup Parameters

Startup configuration parameters are supplied to ngmslib_captureNet and ngmslib_stream in the data type “struct NGMSLIB_STREAM_PARAMS”. These parameters control all aspects of operation.

3.1. caphighprio – Set capture thread priority.

3.1.1. 1 - (Default) High Priority

3.1.2. 0 - Normal Priority

3.2. caprealtime - Set to one to enable download of capture input in real-time according to embedded input file timestamps. Useful for HTTP based capture, such as loading a static FLV file from a web server to be streamed in real-time.

3.3. capretrycntminone – Enable input method specific retry logic upon failure. Applicable to active capture methods such as HTTP.

3.3.1. 0 - (Default) No input retry.

3.3.2. 1 - Indefinite amount of retries.

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

3.4. capture_idletmt_ms – Input capture idle timeout in milliseconds for socket based packet capture (RTP, UDP). Default value is initialized in "*stream_start*" to 10000 ms.

3.4.1. 0 - No input timeout

3.5. capture_idletmt_1stpkt_ms – Input capture idle timeout for first packet in milliseconds for socket based packet capture (RTP, UDP). Default value is initialized in "*stream_start*" to 30000 ms.

3.5.1. 0 - No input timeout

3.6. capture_max_rtpplayoutdelayms - Maximum RTP play-out buffer time controlling wait time for any out of order or lost packet. Default value is 500ms.

3.7. confpath – Optional configuration file path. Default is "*etc/ngms.conf*". The configuration file can be used to load default settings. "*struct NGMSLIB_STREAM_PARAMS*" settings override file configuration settings.

3.8. dir –input storage directory path for recording capture.

3.9. extractpes - Controls de-mux of each packetized elementary Stream and subsequent re-mux.

3.9.1. 0 - (Default) Use direct replay of input MPEG-2 TS stream without decomposing each PES frame.

3.9.2. 1 – Decompose and recompose each PES frame. This value is automatically turned on if transcoding is enabled, or for non MPEG-2 TS based capture.

3.10. favdelay – 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.

3.10.1. 0 - (Default) No additional queuing delay.

3.11. faoffset_m2tpktz – MPEG-2 TS specific timestamp packetization adjustment in seconds applied to audio program stream. This value is used to adjust the PTS and DTS values of the audio elementary stream.

3.11.1. 0 - (Default) Automatic assignment depending on input stream characteristics.

3.12. favoffsets – Audio / Video sync base adjustment in seconds given as a float. Only valid if processing both audio and video elementary streams. The avoffset value is added to the 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 its own unique a/v offset by adjusting the appropriate array index variable.

3.12.1. 0 - (Default) No baseline audio / video sync adjustment.

3.13. favoffsetrtcp – 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 only respect the timestamp of the first packet received instead.

3.13.1. 0 - (Default)

3.14. flvlive - HTTP FLV encapsulated live content streaming server listening address and port string delimited by a colon. For eg. "*127.0.0.1:8080*" or "*8080*" to denote "*0.0.0.0:8080*". A client will access the server at *http://[address]:[port]/flvlive*.

3.15. flvrecord – The output filename where to record the streamed content. Media will be encapsulated in FLV format.

3.16. fps – Input frame rate override. This value will override any fps from a container file or video / audio sequence headers. An fps may need to be provided if no frame rate parameters are present in the input stream.

3.17. fraudqueueslots – Input audio capture queue number of slots. The default value is specific to the input audio codec.

3.18. frvidqueueslots – Input video capture queue number of slots. The default value is specific to the input video codec.

3.19. frtcp_rr_intervalsec – RTCP Receiver Report interval in seconds. A non-zero value enables transmission of Receiver Reports to the sender.

3.19.1. 0 - (Default) Receiver reports disabled.

3.20. frtcp_sr_intervalsec – RTCP Sender Report transmission interval in seconds. Default is 5 seconds.

3.20.1. 0 - Disables transmission of Sender reports.

3.21. httplive – HTTP Live Streaming Server listening address and port string delimited by a colon. For eg. ("*127.0.0.1:8080*" or "*8080*" to denote "*0.0.0.0:8080*". An HTTP Live client (iPhone, Safari) will access the server at *http://[address]:[port]/httplive*.

3.22. 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, to create a temporary in-memory file system on Linux do: "*sudo mount -t tmpfs -o size=102400K tmpfs /usr/local/ram/*".

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

3.24. httpliveurlprefix – Output URL hostname which gets written to any .m3u8 playlist file for each TS file. Each playlist item will be prepended by the specified URL hostname to allow serving of transport media files via an alternate host or virtual URL.

3.25. httplivemax – Maximum number of HTTP Live client sessions. Default value is 4.

3.26. httplive_chunkduration – HTTP Live chunk duration in seconds. Default value is 10 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.

3.27. ignoreencdelay – Controls encoder specific frame latency audio / video sync adjustment.

3.27.1. 0 - (Default) Automatically adjusts audio / video sync to account for encoder latency.

3.27.2. 1 - Disable any adjustment to audio / video sync based on encoder latency.

3.28. inputs - Input file or capture device. Up to two devices are permitted. If using separate video and audio devices, the video device should be placed in the first index. Examples include:

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

3.28.2. "*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.

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

3.28.4. "*http://10.10.10.10:8080/tslive*" For requesting a live MPEG-2 TS encapsulated stream via HTTP.

3.28.5. "*flv://10.10.10.10/test.flv*" For requesting a remote FLV file via HTTP. Alternatively this is interchangeable with the following command-line: *–in=http://10.10.10.10/test.flv –filter="type=flv"*.

3.28.6. "*http://10.10.10.10/httplive/out.m3u8*" For requesting an HTTP Live stream from a remote server.

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

3.28.8. "*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.

3.28.9. "*rtsp://10.10.10.10:554/live.sdp*" For playing a remote RTSP stream.

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

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

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

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

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

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

3.28.16. *"/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.

3.28.17. *"/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.

3.29. islive – Set to 1 to indicate that input is a live capture as opposed to a container file on local storage.

3.30. licfilepath – License file input path. Default value is "*etc/license.dat*".

3.31. live – HTTP Web Server listening address and port string delimited by a colon. For eg. "*127.0.0.1:8080*" or "*8080*" to denote "*0.0.0.0:8080*". A media player will access the server at *http://[address]:[port]/live*. The server will return the appropriate media format content type according to the client User-Agent type.

Note: The same HTTP Server resource pool is used to service all HTTP based requests such as *"/tslive"*, *"/httplive*" and *"/live*". If using non-default port numbers, only one common listening *address:port* will be chosen based on the order given above.

3.32. livemax – Maximum number of client sessions for the *"/live"* URL Default value is 4.

3.33. loop – Controls looping of an input file. Set to 1 to loop content.

3.34. mtu - Maximum Transmission Unit. Specify to use non-default MTU.

3.35. noaud – Set to 1 to disable any audio elementary stream output stream.

3.36. noIncludeSeqHdrs – Set to 1 to 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.

3.37. novid – Set to 1 to disable any video elementary stream output.

3.38. outputs - Output destination or file name.

Multiple UDP / RTP recipients can be specified the output in its own unique array index. If performing transcoding to produce multiple parallel output encodings, each destination output index will correspond to the same encoder output index.

Examples include:

3.38.1. "*rtp://127.0.0.1:5004*" For RTP output to port 5004

3.38.2. "*udp://127.0.0.1:5004*" For direct UDP output to port 5004

3.38.3. *"/path/outputfile.m2t*" For writing the output to local storage.

3.38.4. "*rtp://127.0.0.1:5004,5006*" For RTP output where the video elementary stream is sent on port 5004 and the audio elementary stream is sent on port 5006.

3.39. outaudbufdurationms - Audio output raw sample frame buffer size in milliseconds. Valid for raw audio output to be read with *ngmslib_readAudSamples*. Defaults to 1000 ms.

3.40. overwritefile – set to 1 to force overwriting of recording file if it already exists.

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

3.42. promisc – set to 1 to use PCAP promiscuous capture

3.43. rawxmit – use raw (non – socket) based output.

3.44. rtmplive - 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.

3.45. rtmplivemax – Maximum number of RTMP server sessions. Default value is 4.

3.46. rtmpfp9 – Set to 1 to disable use of digital signature in RTMP handshake. RTMP.

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

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

3.49. rtplivemax – Maximum number of RTP output sessions. A single session comprises of both RTP video and audio even if using different ports. Default value is 4.

3.50. rtsplive - 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.

3.51. rtsplivemax – Maximum number of RTSP server sessions. Default value is 4.

3.52. sdpoutpath – SDP output file path of the published output session.

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

3.54. streamflags – Controls stream output characteristics. This variable is of type NGMS_STREAMFLAGS_T.

3.54.1. NGMS_STREAMFLAGS_DEFAULT - Default behavior.

3.54.2. NGMS_STREAMFLAGS_PREFERAUDIO - Gives real-time preference to delivery of the audio elementary stream over video. When this flag is set all queued input audio samples will be sent to output without regard to playout time stamps. Video input frames will only be sent to output if no queued input audio samples are present. This setting is preferable when streaming from a local microphone to ensure smooth and un-interrupted audio delivery. The video frame rate may decrease when the system becomes loaded.

3.54.3. NGMS_STREAMFLAGS_AV_SAME_START_TM - Video and audio playout start time will be both set to the input playout time stamp of the first video or audio input frame time. This flag is preferred if the input timestamps are generated from the local clock, such as when streaming from a local framebuffer or microphone. Without this flag, video and audio playout start times are tracked individually.

3.55. strfilters – 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 values delimited by the '=' character.

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

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

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

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

3.55.1.1.3. **"h263"** - H.263.

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

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

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

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

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

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

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

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

3.55.1.1.12. **“rgba”** - RGBA8888 (32 bits per pixel, 8 bit alpha mask).

3.55.1.1.13. **“bgra”** - BGRA8888 (32 bits per pixel, 8 bit alpha mask).

3.55.1.1.14. **“rgb565”** - RGB565 (16 bits per pixel, 8 bit alpha mask).

3.55.1.1.15. **“bgr565”** - BGR565 (16 bits per pixel, 8 bit alpha mask).

3.55.1.1.16. **“yuv420p”** - YUV420p (YUV 4:2:0 Y,UU, VV planar).

3.55.1.1.17. **“nv12”** - YUV420sp (YUV 4:2:0 Y, UV, UV semi-planar).

3.55.1.1.18. **“nv21”** - YUV420sp (YUV 4:2:0 Y, VU, VU semi-planar).

3.55.1.1.19. **“pcm”** - PCM 16 bit signed little endian.

3.55.1.1.20. **“alaw”** - PCM 8 bit a-law.

3.55.1.1.21. **“ulaw”** - PCM 8 bit mu-law.

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

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

3.55.2. Input capture filters useful for PCAP based capture.

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

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

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

3.55.2.4. **"dstport"** - Destination Port.

3.55.2.5. **"srcport"** - Source Port.

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

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

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

3.55.3. Input audio stream capture filters.

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

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

3.55.4. Input video stream capture filters.

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

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

3.55.5. Input raw format filters.

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

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

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

3.55.6. Stream specific recording control.

3.55.6.1. **"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.

3.56. **strstartoffset** – Input file streaming start offset in seconds.

3.57. strxcode - Transcoder configuration passed as a quoted comma separated list of key values.

3.58. transproto - Output transport protocol.

3.58.1. "*m2t*" - (Default) Use MPEG-2 Transport Stream encapsulation.

3.58.2. "*rtp*" - Use Video / Audio codec RTP specific encapsulation type.

3.59. tslive – HTTP MPEG-2 Transport Stream live content streaming server listening address and port string delimited by a colon. For eg. "*127.0.0.1:8080*" or "*8080*" to denote "*0.0.0.0:8080*". An MPEG-2 Transport Stream client will access the server at *http://[address]:[port]/tslive*.

3.60. tslivemax – Maximum number of MPEG-2 Transport Stream HTTP client sessions. Default value is 4.

3.61. tslivepwd – Optional password to access live MPEG-2 Transport Stream. The password is supplied by the client on the URL with the parameter "*?pass=xxx*"

3.62. uselocalpcr – Controls output MPEG-2 Transport stream timing mechanism.

3.62.1. 0 - (Default) Use direct timestamps (PTS / DTS) contained in the captured or stored input stream.

3.62.2. 1 - Set to 1 to force to use system clock for streaming MPEG-2 Transport streams.

3.63. verbosity – Log output level

3.63.1. 1 - Normal

3.63.2. 2 - High

3.63.3. 3 – Very High

3.64. xmithighprio – Set streaming transmitter thread priority.

3.64.1. 1 - (Default) High Priority

3.64.2. 0 - Normal Priority

4. Transcoding Configuration

Transcoding settings are supplied within the "*strxcode*" parameter as a quoted comma separated list of key values delimited by the '=' character.

4.1. Video output parameters

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

4.1.1.1. "**h264**" - H.264

4.1.1.2. **"h263"** - H.263

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

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

4.1.1.5. **"vp8"** - VP8

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.1.7. "**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.

4.1.8. "**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.

4.1.9. "**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.

4.1.10. "**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.

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

4.1.12. "**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.

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

4.1.13.1. 0 – (Default) Uses encoder specific output bitrate tolerance.

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

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

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

4.1.16.1. 0 – (Default) Input frame timestamp is obtained from input transport mechanism (MPEG-2 TS PTS / DTS, RTP Timestamp).

4.1.16.2. 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.

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

4.1.17.1. 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.

4.1.17.2. 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.

4.1.17.3. -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.

4.1.18. "**videoOutputConstantFps**" (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. Default value is 400ms.

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

4.1.19.1. 0 – slowest (highest quality)

4.1.19.2. 1 – slow (high quality)

4.1.19.3. 2 – medium (medium quality)

4.1.19.4. 3 -- fast (low quality)

4.1.19.5. 4 -- fastest (lowest quality)

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

4.1.20.1. 0 – lowest (highest speed)

4.1.20.2. 1 – low (fast speed)

4.1.20.3. 2 – medium (medium speed)

4.1.20.4. 3 -- high (slow speed)

4.1.20.5. 4 -- highest (lowest speed)

4.1.21. "**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*".

4.1.22. "**videoFpsDenumerator**" (or "v**fd**") - Output frame rate denominator.

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

4.1.24. "**videoGOPMaxMs**" (or "**vgmax**") – Encoder max GOP in milliseconds. The actual frame count passed to the encoder is obtained based on the specified frame rate (*videoFps*).

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

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

4.1.26.1. 0 – (Default) Use automatic encoder specific configuration.

4.1.26.2. 1 - Minimal encoder lag configuration (0). This value should be used for real-time encoding. Note that the "*videoThreads*" (encoder thread control count) may also influence the actual encoder frame lag.

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

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

4.1.27.1. H.264 Profiles

4.1.27.1.1. 0 (Default) H.264 High

4.1.27.1.2. 66 – H.264 Baseline

4.1.27.1.3. 77 – H.264 Main

4.1.27.1.4. 100 – (Default) H.264 High

4.1.27.2. 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.

4.1.27.2.1. 0 - (Default) Simple Profile

4.1.27.2.2. 240 - Advanced Simple Profile (Profile value 15 (0x0f) << 4)

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

4.1.28. "**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.

4.1.29. "**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.

4.1.30. "**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.

4.1.31. "**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.

4.1.32. "**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.

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

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

4.1.34.1. "**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.

4.1.34.2. "**cqp**" - Rate control utilizing a constant quantizer for P-frames. A valid target quantizer "*vq*" 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.

4.1.34.3. "**crf**" - Rate control utilizing a constant Rate Factor. A valid target quantizer "*vq*" 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.

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

4.1.35.1. 1 – SWS_FAST_BILINEAR

4.1.35.2. 2 – SWS_BILINEAR

4.1.35.3. 3 - SWS_BICUBIC

4.1.35.4. 4 - SWS_GAUSS

4.1.35.5. 5 - SWS_SINC

4.1.36. "**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, default value is 40.

4.1.37. "**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.

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

4.1.38.1. 0 – (Default) Encoder will choose a default value. Values greater than 1 will usually incur additional encoder frame output lag. For real-time minimal encoder latency should be set to 1.

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

4.1.39.1. 0 – (Default) disables frame up-sampling / frame duplication.

4.1.39.2. 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.

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

4.1.40.1. 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.

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

4.1.41.1. 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*".

4.2. Audio output parameters

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

4.2.1.1. "**aac**" – AAC

4.2.1.2. "**ac3**" – A53 / AC3

4.2.1.3. "**amr**" – AMR-NB

4.2.1.4. "**g711a**" – G.711 alaw

4.2.1.5. "**g711u**" – G.711 mulaw

4.2.1.6. "**vorbis**" – Vorbis

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

4.2.1.8. "**ulaw**" – PCM 8 bit mulaw

4.2.1.9. "**alaw**" – PCM 8 bit alaw

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

4.2.2.1. 0 – (Default) Uses encoder default setting.

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

4.2.3.1. 0 – (Default) enable audio transcoding only if output codec does not match input codec type, sample rate, and channel configuration.

4.2.3.2. 1 - Force audio transcoding.

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

4.2.4.1. 0 – (Default) Reuses input sampling rate.

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

4.2.5.1. 0 – (Default) – Reuses input channel count.

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

4.2.6.1. 0 – (Default) No volume adjustment.

4.2.6.2. $0 < n < 256$ - Decrease volume by factor of $(8 – \log2(n))$.

4.2.6.3. 256 – Base volume setting resulting in no volume adjustment.

4.2.6.4. $256 > n > 65536$ - Increase volume by factor of $(\log2(n) – 8)$.

5. Configuration Examples

5.1. Live capture to local video frame buffer and audio samples buffer.

5.1.1. Setup an MPEG-2 TS encapsulated RTP stream listener on port 5004. The video elementary stream should be decoded to RGB888 format at a 320 x 240 resolution and the output frame rate should

not exceed 24/1 fps. The audio elementary stream should be decoded to PCM 16 bit signed Little Endian at 44.1KHZ with one mono channel.

```
NGMSLIB_STREAM_PARAMS_T params;

ngmslib_open(&params);

params.islive=1;
params.inputs[0] = "rtp://0.0.0.0:5004";
params.strfilters[0] = "type=m2t";
params.strxcode="videoCodec=rgb,vx=320,vy=240,videoFps=24,audioCodec=pcm,audioSampleRate=44100,audioChannels=1";

/*
 Note:  this is a blocking call which does not return until the default input  timeout is reached or ngmslib_close is called asynchronously.
*/
ngmslib_stream(&params);

ngmslib_close(&params);
```

5.1.2. Setup a video and audio RTP listener. The incoming video is H.264 at 90KHZ on UDP port 5004 sent over RTP. The incoming audio is AAC at 44.1KHZ with 2 channels (stereo) on UDP port 5006 sent over RTP. The video content is decoded into RGB565 format at a vertical resolution of 240px with the input aspect ratio preserved

and the output frame rate should not exceed 24/1fps. The audio elementary stream should be decoded to mu-law at 8KHZ with one mono channel. The output audio buffer is set to 200ms in play-out duration.

```
NGMSLIB_STREAM_PARAMS_T params;

ngmslib_open(&params);


params.islive=1;
params.inputs[0] = "rtp://0.0.0.0:5004,5006";
params.strfilters[0] = "type=h264,clock=90000,dstport=5004";
params.strfilters[1] =
"type=aac,clock=44100,channels=2,dstport=5006";
params.strxcode="videoCodec=rgb565,vy=240,videoFrameRate=2
4,audioCodec=ulaw,audioSampleRate=8000,audioChannels=1";
params.outaudbufdurationms = 200;


ngmslib_stream(&params);

ngmslib_close(&params);
```

5.2. Live output from local video and audio device

5.2.1. Stream local video and audio content. Video frames are input by asynchronous calls to *ngmslib_onVidFrame*. Each video frame is expected in YUV420p format at a resolution of 1024x768. Audio samples are input by asynchronous calls to *ngmslib_onAudSamples*. Audio is expected at PCM 16 bit signed Little Endian at 8KHZ with one mono channel. Video is encoded into H.264 baseline profile at 300Kb/s at a resolution of 320x240 and a frame rate of 24/1. Encoder speed/quality tradeoff of is set to 1 (medium), with minimal encoder look-ahead latency, and GOP between 1-2 sec. Audio is encoded into G.711 a-law. The output video and audio is encapsulated in an MPEG-2 TS stream sent to 10.10.10.10:5004 over UDP/RTP.

```
NGMSLIB_STREAM_PARAMS_T params;

ngmslib_open(&params);

params.output="rtp://10.10.10.10:5004";
params.transproto="m2t";
params.inputs[0] = "/dev/dummyvideo";
params.strfilters[0] = "type=yuv420p,size=1024x768";
params.inputs[1] = "/dev/dummyaudio";
params.strfilters[1] = "type=pcm,clock=8000,channels=1";
params.strxcode="videoCodec=h264,videoBitrate=300,videoWidth
=320,videoHeight=240,videoFps,videoEncoderSpeed=1,videoLook
aheadFramesMin1=1,videoThreads=1,videoOutputConstantFps=-
1,videoGOPMinMs=1000,videoGOPMaxMs=2000,videoProfile=66,
audioCodec=g711a";
```

```
/*
 Note:  this is a blocking call which does not return until
ngmslib_close is called asynchronously.
*/
ngmslib_stream(&params);
```

5.2.2. Stream local video and audio content. Video frames are read from a local */dev/framebuffer* device at 30fps. Each video frame is expected in RGBA8888 format at a resolution of 1024x768. Audio samples are read from a local */dev/dsp* device. Audio is expected at PCM 16 bit signed Little Endian at 48KHZ with one mono channel. Video is encoded into H.264 baseline profile at 300Kb/s at a resolution of 320x240 and a frame rate of 24/1. Encoder speed/quality tradeoff of is set to 0 (high quality), with GOP between 1-2 sec. Audio is encoded into AAC-LC mono at 48KHZ with a target rate of 64Kb/s per channel. The output video is sent using NAL packetization via RTP port 5004. The output audio is sent using AAC-hbr packetization via RTP port 5006. The output is also available via MPEG-2 TS over HTTP on local port 8080 as well as HTTP Live iPhone streaming on port 8080.

```
NGMSLIB_STREAM_PARAMS_T params;

ngmslib_open(&params);

params.output="rtp://10.10.10.10:5004,5006";
```

```
params.transproto="rtp";
params.inputs[0] = "/dev/framebuffer";
params.strfilters[0] = "type=rgba,size=1024x768,fps=30";
params.inputs[1] = "/dev/dsp";
params.strfilters[1] = "type=pcm,clock=48000,channels=1";
params.tslive="8080";
params.httplive=”8080”;
params.strxcode="videoCodec=h264,videoBitrate=300,videoWidth
=320,videoHeight=240,videoFps,videoEncoderSpeed=0,videoOutp
utConstantFps=-
1,videoGOPMinMs=1000,videoGOPMaxMs=2000,videoProfile=77,
audioCodec=aac,audioSampeRate=48000,audioBitrate=64000,aud
ioChannels=1";

/*
 Note:  this is a blocking call which does not return until
ngmslib_close is called asynchronously.
*/
ngmslib_stream(&params);
```

6. Linking with NGMS Core Library

NGMS core is packaged as a stand-alone shared library which can be dynamically linked with your application. NGMS core is separate from any third-party open source or commercial encoders / decoders.

6.1. Xcode

libxcode.so is the wrapper for any third-party open source or commercial video and audio encoders and decoders. The out-of-the box *libxcode* library provided with NGMS core contains support for *libavcodec* (part of *Ffmpeg*), x264 (H.264 encoder) and FAAC (open source AAC encoder). *Libxcode* is licensed under different terms than NGMS core and is provided in open source form. The precise *libxcode* license may be either LGPL or GPL, depending on what code encompasses.

6.1.1. xcode interface - The *libxcode* interface is defined in *ixcode.h* If integrating a third-party transcoder you must modify the source of the following functions to add the appropriate transcoding support..

6.1.1.1. **ixcode_init_vid** - Called by NGMS to initialize the video encoder and decoder. A subsequent call prior to calling *ixcode_close_vid* will update the encoder running configuration. Depending on the encoder being used, only certain parameters may actually be updated.

6.1.1.2. **ixcode_init_aud** - Called by NGMS to initialize the audio encoder and decoder.

6.1.1.3. **ixcode_close_vid** - Called by NGMS to close and release video encoder and decoder resources.

6.1.1.4. **ixcode_close_aud** - Called by NGMS to close and release audio encoder and decoder resources.

6.1.1.5. **ixcode_frame_vid** - Called by NGMS to transcode a single video frame.

6.1.1.6. **ixcode_frame_aud** - Called by NGMS to transcode one or more audio frames or a series of raw samples.

6.2. System Architecture

NGMS core provides binary distributions for the following system architectures. An appropriate pre-compiled version of *libxcode,* and accompanying source code is provided as well.

6.2.1. Linux intel

NGMS core is packaged as a 32bit shared library (.so) . The library can be linked and executed on 64bit systems with the appropriate system i386 support libraries.

6.2.2. Linux ARM

NGMS core is packaged as a shared library (.so). A version of an ARM specific makefile (*config.mak.armeabi*) is provided to help in building third-party supporting code, such as any open source transcoders. The

accompanying makefile containes tunable configurations for a variety of common ARM processors.

6.2.2.1. CFG_THUMB - Set to 1 to enable gcc to generate code using the 16bit Thumb instruction set. (Disabled by default).

6.2.2.2. CFG_ARM_MFPU - Floating Point Unit type.

6.2.2.2.1. “neon” (Default) . Enables advanced SIMD NEON. Should be enabled if supported on the given hardware platform for optimal performance.

6.2.2.2.2. “vfp” - Vector Float Point. Enables low-cost single-precision and double precision floating point computations.

6.2.2.3. CFG_ARM_MFLOAT - Floating point type.

6.2.2.3.1. “soft” - (Default). Enables use of soft floating point types for optimal performance.

6.2.3. Windows

NGMS core is packaged as a dynamic library (.dll). The core library can be linked when building within the MinGW environment.

6.2.4. Mac

NGMS core is packaged as a 64bit shared library (.so).