APPS060-Streaming Outputs with ffmpeg Application Note

V1.0

2022-11-14

Table of Contents

1 Table of Abbreviations

Abbreviation:	Full Form:
T408	Codensity T408 Video Transcoder
TB	Thunderbolt 3 or 4
TB3	Thunderbolt 3
AI	Artificial Intelligence
API	Application Programming Interface
AUD	Access Unit Delimiters
AV	Audio Video
AV1	Alliance Open Media Video 1 Codec
CABAC	Context Adaptive Binary Arithmetic Coding
CBR	Constant Bit Rate
CPU	Central Processing Unit
CRF	Constant Rate Factor
DSP	Digital Signal Processing
EOF	End Of File
FIXQB	Fixed Quantization Parameter
FF	Form Factor
FPS	Frames Per Second
GOP	Group of pictures
HDR	High Dynamic Range
HHHL	Half Height, Half Length
HHFL	Half Height, Full Length
HLG	Hybrid Log Gamma
HRD	Hypothetical Reference Decoder
IE	Inference Engine
Mbps	Mega Bit per second
MBps	Mega Byte per second
NLP	Natural Language Processing
NPU	Neural Processing Unit
NVMe	Non-Volatile Memory express
PCIe	Peripheral Component Interconnect express
PPS	Picture Parameter Set
QP	Quoted Printable
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
RDO	Optimization
ROI	Region of interest

SDK	Software Development Kit
SEI	Supplemental Enhancement Information
SRIOV	Single Root I/O Virtualization
SPS	Sequence Parameter Set
VBR	Variable Bit Rat
VCL	Video Coding Layer
VPS	Video Parameter Set
VUI	Video Usability Information

2 References

Technical Note:

Specifications:

3 Background

3.1 Scope

This document presents the streaming output examples when using ffmpeg with NETINT Video Transcoder devices like T408, and Quadra. The ffmpeg streaming outputs are not dependent on the type of encoder in use. All streaming protocols supported by ffmpeg should interwork with NETINT Video Transcoder devices.

3.2 Intended Audience

This document is intended for engineers/developers/technicians operating hosts loaded with NETINT transcoders.

3.3 Compatibility

T408 / T432 Product	**Version**
Minimum Hardware Revision	-
Minimum Firmware Revision	-

4 FFmpeg outputs

4.1 Scope

This section lists the various output options when using ffmpeg for performing encoding and transcoding operations using NETINT Video Transcoder cards.

4.2 Output Types

When performing encoding and transcoding using ffmpeg the user can output to:
- null
- file
- stream output to local or remote destinations

This document will present ffmpeg examples when writing ffmpeg output to stream.

4.3 Streaming Options

ffmpeg is open source. All streaming options supported by ffmpeg should interwork with when using NETINT video transcoder device for encoding, and transcoding.

This document will present examples for the following streaming outputs –

1] HLS
2] UDP
3] RTMP
4] SRT

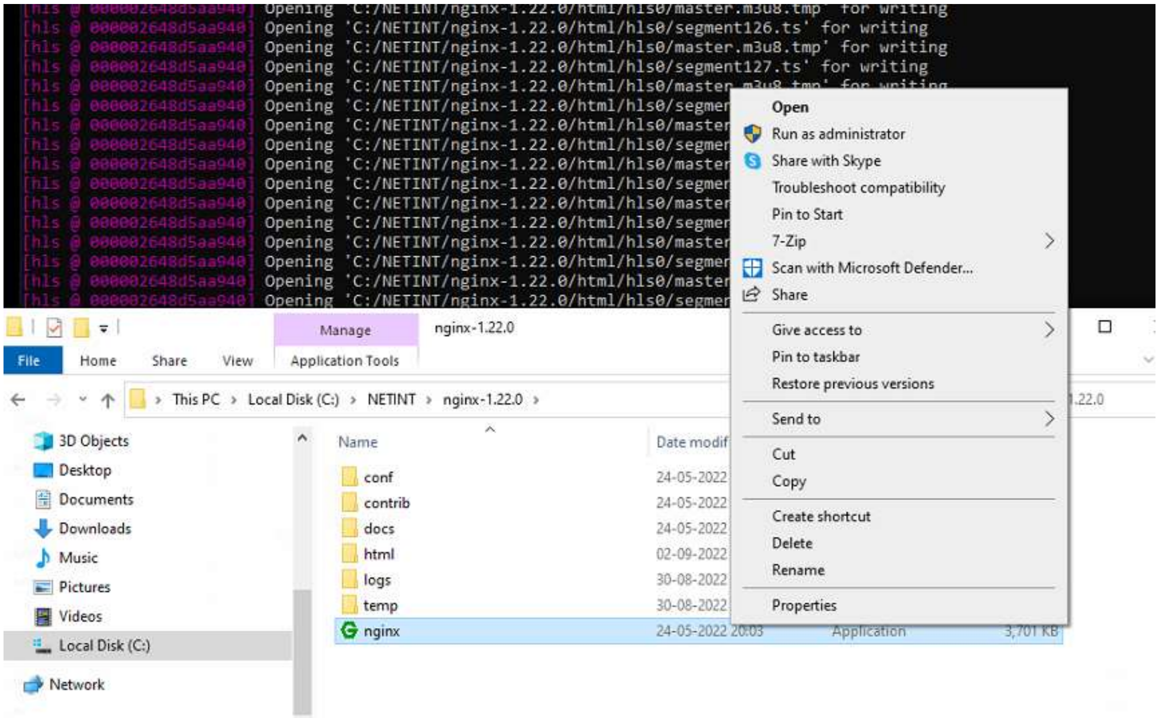
4.4 Web Server

Most streaming protocols will require a web server running on the host being used for streaming. Nginx is a free open source server and can be used when streaming using RTMP, HLS.

4.4.1 Installing nginx

4.4.1.1 On a Windows Host

1] Download and run a Nginx server on the host. Link - http://nginx.org/en/download.html . Unzip the Nginx download and then run the 'nginx' application as administrator.



2] Create a folder 'hls0' in the nginx->html folder, for example, nginx-1.22.0/html/hls0 folder in above snapshot.

4.4.1.2 On Ubuntu Linux host

Install nginx server and then modify the nginx.conf file. Refer Appendix A of this document for sample nginx.conf file and rtmp editing notes.

```
sudo apt install libnginx-mod-rtmp
```

```
sudo apt install nginx
```

Check firewall status:

```
sudo ufw status
```

If it replies ‘`Status: inactive`’ then skip the next step.

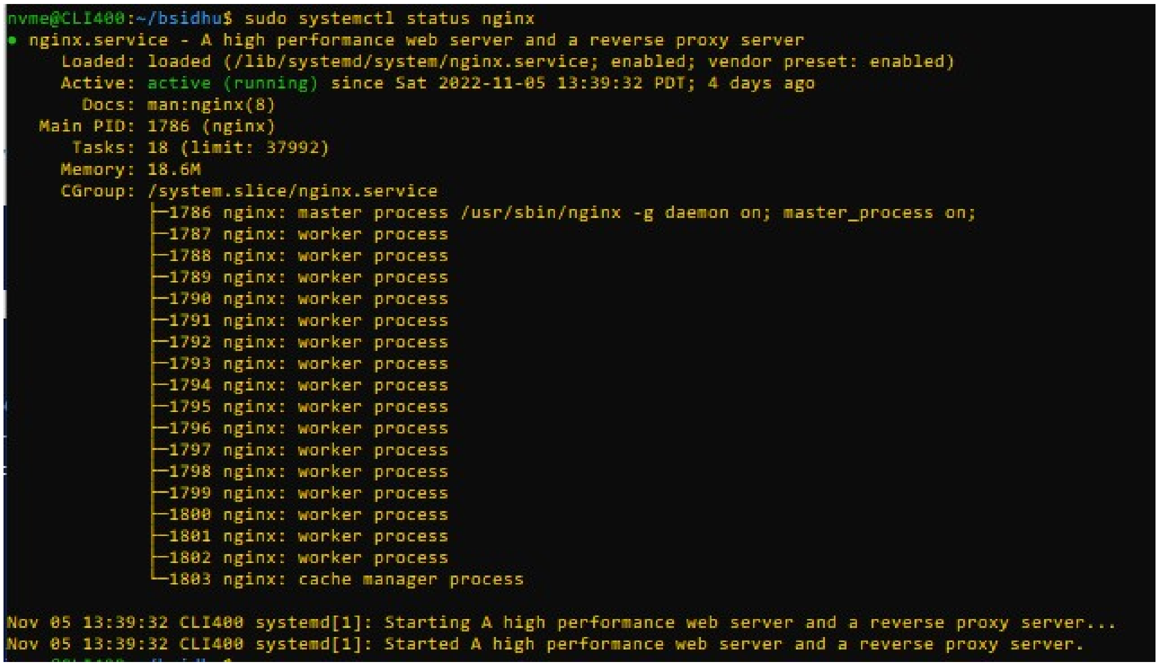
```
sudo ufw allow 'Nginx Full' 22 80 8080 1935 1801
```

Check if ‘nginx’ is running:

```
ps -aef |grep nginx
```

```
nvme@CLI400:~/bsidhu$ ps -aef |grep nginx
root        1786       1  0 Nov05 ?        00:00:00 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
www-data    1787    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1788    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1789    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1790    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1791    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1792    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1793    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1794    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1795    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1796    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1797    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1798    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1799    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1800    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1801    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1802    1786  0 Nov05 ?        00:00:00 nginx: worker process
www-data    1803    1786  0 Nov05 ?        00:00:00 nginx: cache manager process
nvme       50650   50381  0 07:11 pts/0    00:00:00 grep --color=auto nginx
```

```
sudo systemctl status nginx
```



```
nvme@CLI400:~/bsidhu$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2022-11-05 13:39:32 PDT; 4 days ago
       Docs: man:nginx(8)
   Main PID: 1786 (nginx)
      Tasks: 18 (limit: 37992)
     Memory: 18.6M
     CGroup: /system.slice/nginx.service
             ├─1786 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             ├─1787 nginx: worker process
             ├─1788 nginx: worker process
             ├─1789 nginx: worker process
             ├─1790 nginx: worker process
             ├─1791 nginx: worker process
             ├─1792 nginx: worker process
             ├─1793 nginx: worker process
             ├─1794 nginx: worker process
             ├─1795 nginx: worker process
             ├─1796 nginx: worker process
             ├─1797 nginx: worker process
             ├─1798 nginx: worker process
             ├─1799 nginx: worker process
             ├─1800 nginx: worker process
             ├─1801 nginx: worker process
             ├─1802 nginx: worker process
             └─1803 nginx: cache manager process

Nov 05 13:39:32 CLI400 systemd[1]: Starting A high performance web server and a reverse proxy server...
Nov 05 13:39:32 CLI400 systemd[1]: Started A high performance web server and a reverse proxy server.
```

The ‘nginx’ status must be ‘active’ before proceeding to use streaming options list HLS, RTMP.

Create a folder ‘hls0’ in /var/www/html/.

```
sudo mkdir -p /var/www/html/hls0
```

‘nginx’ server setup is now complete.

4.5 Streaming Ports

Ensure that any network port used in the ffmpeg streaming commands is accessible (open) on the network for stream data to flow. For examples in this document the network ports 80, 22, 1935, 1801, 1900 need to be open on both source and destination hosts.

4.6 Useful Links and Reference

1] https://ottverse.com/hls-packaging-using-ffmpeg-live-vod/
2] https://stackoverflow.com/questions/62977448/minimal-srt-stream-example-with-ffmpeg
3] http://nginx.org/en/download.html
4] http://trac.ffmpeg.org/wiki/StreamingGuide

5] https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-20-04
6] https://stackoverflow.com/questions/50967706/how-to-compile-ffmpeg-with-enabling-libsrt

5 HLS

5.1 Scope

HLS – HTTP Live Streaming is a popular, safe, reliable mode of delivering video and audio over the internet. This section provides an example of using HLS with ffmpeg.

5.2 Pre-Requisite

As per Section 4.4, Section 4.5, and Appendix A.

5.3 ffmpeg streaming example – file input and hls output

Run the following command on the source host.

```
ffmpeg -hide_banner -c:v h264_ni_logan_dec -i input_h264.mp4 -c:v
h264_ni_logan_enc -xcoder-params bitrate=6000000:RcEnable=1 -f
hls -hls_segment_filename C:/NETINT/nginx-
1.22.0/html/hls0/segment%d.ts -hls_time 4 -master_pl_name
master.m3u8 -hls_flags delete_segments C:/NETINT/nginx-
1.22.0/html/hls0/master.m3u8
```

Note – The above example shows hls0 path on a Windows host. A user will update the hls0 paths in above example to match the host nginx installation on their Linux or Windows host. For a Linux host the path is usually /var/www/html/hls0/ .

Next, open a VLC player to view the network stream. General format is
`http://<ip address>/hls0/master.m3u8`
where, ip address is the IP address of the streaming source host

6 UDP

6.1 Scope

UDP – User Datagram Protocol is a fast and simple streaming protocol. This section provides an example of using UDP with ffmpeg.

6.2 Pre-Requisite

As per Section 4.5.

6.3 ffmpeg streaming example – file input and udp output

Run the following command on the source host.

```
ffmpeg -hide_banner -re -i inputfile.mp4 -c:v h264_ni_logan_enc -f mpegts "udp://localhost:1900?overrun_nonfatal=1"
```

The above command is pushing the stream to destination host localhost:1900. The user will update the destination IP and port to match their use case.

Next, on the destination host receive the udp stream. The following example has ffmpeg receiving the udp stream as input.

```
ffmpeg -hide_banner -i udp://0.0.0.0:1900 -f null -
```

Caller –

```
nvme@CL1400:~/bsidhu$ ffmpeg -hide_banner -f concat -i list.txt -c:v h264_ni_quadra_enc -maxrate 4000k -bufsize 8000k -g 50 -f mpegts
 "udp://localhost:1900?overrun_nonfatal=1"
Input #0, concat, from 'list.txt':
  Duration: N/A, bitrate: N/A
    Stream #0:0: Video: h264 (High), yuv420p(progressive), 1920x1080, 24 fps, 24 tbr, 1200k tbn, 48 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> h264 (h264_ni_quadra_enc))
Press [q] to stop, [?] for help
[h264_ni_quadra_enc @ 0x5613b6f6a3c0] Session state: 0 allocate frame fifo.
[h264_ni_quadra_enc @ 0x5613b6f6a3c0] pix_fmt is 0, sw_pix_fmt is -1
[h264_ni_quadra_enc @ 0x5613b6f6a3c0] sw_pix_fmt assigned to pix_fmt was 0, is now -1
[h264_ni_quadra_enc @ 0x5613b6f6a3c0] p_param->hwframes = 0
[h264_ni_quadra_enc @ 0x5613b6f6a3c0] dts offset: 7, gop_offset_count: 0
Output #0, mpegts, to 'udp://localhost:1900?overrun_nonfatal=1':
  Metadata:
    encoder         : Lavf58.45.100
    Stream #0:0: Video: h264 (h264_ni_quadra_enc), yuv420p, 1920x1080, q=2-31, 200 kb/s, 24 fps, 90k tbn, 24 tbc
    Metadata:
      encoder         : Lavc58.91.100 h264_ni_quadra_enc
frame= 5489 fps=273 q=-0.0 size=  130274kB time=00:03:48.00 bitrate=4680.7kbits/s speed=11.3x
```

Listener -

```
nvme@CLI400:~$ ffmpeg -hide_banner -i udp://0.0.0.0:1900 -f null -
Input #0, mpegts, from 'udp://0.0.0.0:1900':
  Duration: N/A, start: 1.691667, bitrate: N/A
  Program 1
    Metadata:
      service_name    : Service01
      service_provider: FFmpeg
    Stream #0:0[0x100]: Video: h264 (High) ([27][0][0][0] / 0x001B), yuv420p(tv, progressive), 1920x1080, 24 fps, 24 tbr, 90k tbn, 48 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> wrapped_avframe (native))
Press [q] to stop, [?] for help
Output #0, null, to 'pipe:':
  Metadata:
    encoder         : Lavf58.45.100
    Stream #0:0: Video: wrapped_avframe, yuv420p, 1920x1080, q=2-31, 200 kb/s, 24 fps, 24 tbn, 24 tbc
    Metadata:
      encoder         : Lavc58.91.100 wrapped_avframe
frame= 2852 fps=284 q=-0.0 size=N/A time=00:01:58.83 bitrate=N/A speed=11.8x
```

7 RTMP

7.1 Scope

RTMP – Real-Time Messaging Protocol is a reliable mode of delivering video and audio over the internet. This section provides an example of using RTMP with ffmpeg.

7.2 Pre-Requisite

As per Section 4.4, Section 4.5, and Appendix A.

7.3 ffmpeg streaming example – file input and rtmp output

Run the following command on the source host.

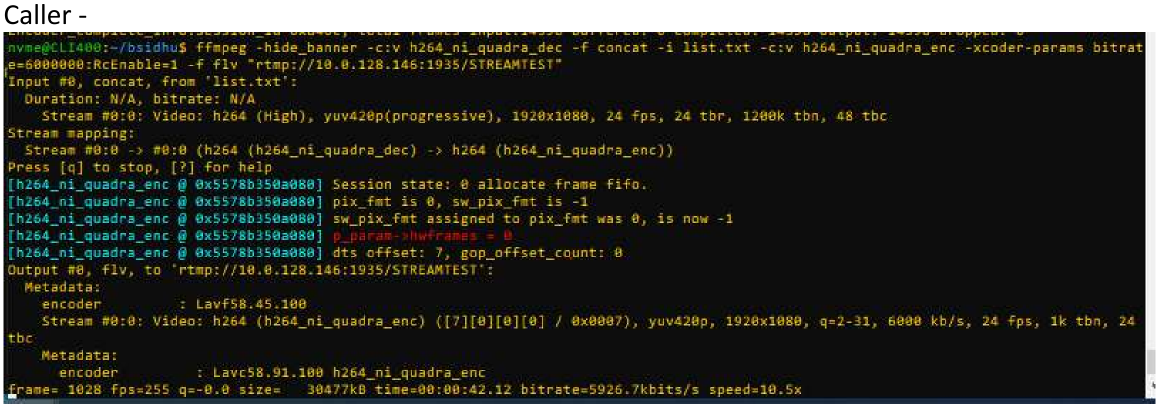
```
ffmpeg -hide_banner -re -c:v h264_ni_logan_dec -i input_h264.mp4
-c:v h264_ni_logan_enc -xcoder-params bitrate=6000000:RcEnable=1
-f flv -i rtmp://192.168.1.10:1935/STREAMTEST
```

The above command is pushing the stream to destination host 192.168.1.10. The user will update the destination IP and port to match their use case.

Next, on the destination host, in this example 192.168.1.10, receive the RTMP stream. The following example shows ffmpeg receiving the rtmp stream as input.

```
ffmpeg -hide_banner -i rtmp://@:1935/STREAMTEST -f null -
```

Caller -



Listener -

```
nvme@nvme-cli452:~$ ffmpeg -hide_banner -i rtmp://@:1935/STREAMTEST -f null -
Input #0, flv, from 'rtmp://@:1935/STREAMTEST':
  Metadata:
    Server          : NGINX RTMP (github.com/arut/nginx-rtmp-module)
    displayWidth    : 1920
    displayHeight   : 1080
    fps             : 24
    profile         :
    level           :
  Duration: 00:00:00.00, start: 280.292000, bitrate: N/A
    Stream #0:0: Data: none
    Stream #0:1: Video: h264 (High), yuv420p(tv, progressive), 1920x1080, 5999 kb/s, 24 fps, 24 tbr, 1k tbn, 48 tbc
Stream mapping:
  Stream #0:1 -> #0:0 (h264 (native) -> wrapped_avframe (native))
Press [q] to stop, [?] for help
Output #0, null, to 'pipe:':
  Metadata:
    Server          : NGINX RTMP (github.com/arut/nginx-rtmp-module)
    displayWidth    : 1920
    displayHeight   : 1080
    fps             : 24
    profile         :
    level           :
    encoder         : Lavf58.45.100
    Stream #0:0: Video: wrapped_avframe, yuv420p, 1920x1080, q=2-31, 200 kb/s, 24 fps, 24 tbn, 24 tbc
    Metadata:
      encoder         : Lavc58.91.100 wrapped_avframe
frame= 1917 fps=270 q=-0.0 size=N/A time=00:01:19.87 bitrate=N/A speed=11.3x
```

8 SRT

8.1 Scope

SRT – Secure Reliable Transport is open source and provides a secure mode of delivering video and audio over the internet. This section provides an example of using SRT with ffmpeg.

8.2 Pre-Requisite

As per Section 4.4, Section 4.5, and Appendix B.

ffmpeg will need to be recompiled with –enable-libsrt support to use SRT as streaming output protocol. User can run the command 'ffmpeg -protocols' to learn if SRT is supported.

8.3 ffmpeg streaming example – file input and srt output

Run the following command on the source host as Caller.

```
ffmpeg -hide_banner -re -c:v h264_ni_logan_dec -i input_h264.mp4
-c:v h264_ni_logan_enc -xcoder-params bitrate=6000000:RcEnable=1
-maxrate 4000k -bufsize 8000k -g 50 -f mpegts
"srt://localhost:1900?mode=caller&pkt_size=1316"
```

The above example uses localhost:1900 as destination. The user will update the destination IP and port to match their use case

Receive SRT as Listener on destination host. The following example shows ffmpeg receiving the srt stream as input.

```
ffmpeg -i srt://0.0.0.0:1900?mode=listener -c:v copy -c:a copy -f
null /dev/null
```

Caller -

```
nvme@CLI400:~/bsidhu$ ffmpeg -hide_banner -c:v h264_ni_quadra_dec -f concat -i list.txt -c:v h264_ni_quadra_enc -xcoder-params bitrat
e=6000000:RcEnable=1 -maxrate 4000k -bufsize 8000k -g 50 -f mpegts "srt://localhost:1900?mode=caller&pkt_size=1316"
Input #0, concat, from 'list.txt':
  Duration: N/A, bitrate: N/A
    Stream #0:0: Video: h264 (High), yuv420p(progressive), 1920x1080, 24 fps, 24 tbr, 1200k tbn, 48 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (h264_ni_quadra_dec) -> h264 (h264_ni_quadra_enc))
Press [q] to stop, [?] for help
[h264_ni_quadra_enc @ 0x556a1272cd00] Session state: 0 allocate frame fifo.
[h264_ni_quadra_enc @ 0x556a1272cd00] pix_fmt is 0, sw_pix_fmt is -1
[h264_ni_quadra_enc @ 0x556a1272cd00] sw_pix_fmt assigned to pix_fmt was 0, is now -1
[h264_ni_quadra_enc @ 0x556a1272cd00] p_param->hwframes = 0
[h264_ni_quadra_enc @ 0x556a1272cd00] dts offset: 7, gop_offset_count: 0
Output #0, mpegts, to 'srt://localhost:1900?mode=caller&pkt_size=1316':
  Metadata:
    encoder         : Lavf58.45.100
    Stream #0:0: Video: h264 (h264_ni_quadra_enc), yuv420p, 1920x1080, q=2-31, 6000 kb/s, 24 fps, 90k tbn, 24 tbc
    Metadata:
      encoder         : Lavc58.91.100 h264_ni_quadra_enc
frame= 1162 fps=257 q=-0.0 size=   35229kB time=00:00:47.70 bitrate=6049.1kbits/s speed=10.5x
```

Listener –

```
nvme@CLI400:~$ ffmpeg -hide_banner -i srt://0.0.0.0:1900?mode=listener -f null -
Input #0, mpegts, from 'srt://0.0.0.0:1900?mode=listener':
  Duration: N/A, start: 1.691667, bitrate: N/A
  Program 1
    Metadata:
      service_name    : Service01
      service_provider: FFmpeg
    Stream #0:0[0x100]: Video: h264 (High) ([27][0][0][0] / 0x001B), yuv420p(tv, progressive), 1920x1080, 24 fps, 24 tbr, 90k tbn, 48 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> wrapped_avframe (native))
Press [q] to stop, [?] for help
Output #0, null, to 'pipe:':
  Metadata:
    encoder         : Lavf58.45.100
    Stream #0:0: Video: wrapped_avframe, yuv420p, 1920x1080, q=2-31, 200 kb/s, 24 fps, 24 tbn, 24 tbc
    Metadata:
      encoder         : Lavc58.91.100 wrapped_avframe
frame= 6929 fps=260 q=-0.0 size=N/A time=00:04:48.70 bitrate=N/A speed=10.9x
```

9 Appendix A – Sample nginx.conf file

On an Ubuntu Linux host the nginx.conf file is generally located in the /etc/nginx folder.

9.1 Sample nginx.conf file when using rtmp

The default nginx.conf file may not have the 'rtmp' block. A user will need to edit to add the block per their use case. It is important that port 1935 on the Linux host is accessible through the firewall i.e. port 1935 is reachable so streaming data can reach the internet users.

```
nvme@nvme-cli452:/etc/nginx$ cat nginx.conf
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        ##
        # SSL Settings
        ##
```

```
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping
SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        ##
        # Logging Settings
        ##

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        ##
        # Gzip Settings
        ##

        gzip on;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json
application/javascript text/xml application/xml
application/xml+rss text/javascript;

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}

rtmp {
        server {
                listen 1935;

                application STREAM22 {
                        live on;
                }

                        application STREAMTEST {
                        live on;
                        publish_notify on;
                }
```

```
        }
}

#mail {
#       # See sample authentication script at:
#       #
http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
#
#       # auth_http localhost/auth.php;
#       # pop3_capabilities "TOP" "USER";
#       # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
#       server {
#               listen     localhost:110;
#               protocol   pop3;
#               proxy      on;
#       }
#
#       server {
#               listen     localhost:143;
#               protocol   imap;
#               proxy      on;
#       }
#}
```

9.2 Sample nginx.conf file when using HLS

The default nginx.conf file may not have the ‘rtmp’ block. A user will need to edit to add the block per their use case. It is important that port 1935 on the Linux host is accessible through the firewall i.e. port 1935 is reachable so streaming data can reach the internet users.

```
nvme@CLI400:/etc/nginx$ cat nginx.conf
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {
```

```
##
# Basic Settings
##

sendfile on;
tcp_nopush on;
tcp_nodelay on;
keepalive_timeout 65;
types_hash_max_size 2048;
# server_tokens off;

# server_names_hash_bucket_size 64;
# server_name_in_redirect off;

include /etc/nginx/mime.types;
default_type application/octet-stream;

##
# SSL Settings
##

ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
ssl_prefer_server_ciphers on;

##
# Logging Settings
##

access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log;

##
# Gzip Settings
##

gzip on;

# gzip_vary on;
# gzip_proxied any;
# gzip_comp_level 6;
# gzip_buffers 16 8k;
# gzip_http_version 1.1;
```

```
    # gzip_types text/plain text/css application/json application/javascript text/xml
application/xml application/xml+rss text/javascript;

    ##
    # Virtual Host Configs
    ##

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}

rtmp {
    server {
        listen 1935;
        chunk_size 4096;

        application live {
            live on;
            record off;
            hls on;
            hls_nested on;
            hls_path /var/www/html/hls0;
            hls_fragment 15s;
        }
    }
}


#mail {
#    # See sample authentication script at:
#    # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
#
#    # auth_http localhost/auth.php;
#    # pop3_capabilities "TOP" "USER";
#    # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
#    server {
#        listen     localhost:110;
#        protocol   pop3;
#        proxy      on;
#    }
#
#    server {
#        listen     localhost:143;
```

```
#           protocol   imap;
#           proxy      on;
#     }
#}
```

10 Appendix B – Notes for libsrt needed for SRT support for ffmpeg

Install packages for libsrt. Link - https://stackoverflow.com/questions/50967706/how-to-compile-ffmpeg-with-enabling-libsrt . A successful compile and install from this link should look as follows.

```
nvme@CLI400:~/bsidhu/srt/build$ sudo make install
[ 14%] Built target srtsupport_virtual
[ 85%] Built target srt_virtual
[ 88%] Built target srt_static
[ 92%] Built target srt-tunnel
[ 96%] Built target srt-file-transmit
[100%] Built target srt-live-transmit
Install the project...
-- Install configuration: "Release"
-- Installing: /home/nvme/ffmpeg_build/lib/libsrt.a
-- Installing: /home/nvme/ffmpeg_build/include/srt/version.h
-- Installing: /home/nvme/ffmpeg_build/include/srt/srt.h
-- Installing: /home/nvme/ffmpeg_build/include/srt/logging_api.h
-- Installing: /home/nvme/ffmpeg_build/include/srt/access_control.h
-- Installing: /home/nvme/ffmpeg_build/include/srt/platform_sys.h
-- Installing: /home/nvme/ffmpeg_build/include/srt/udt.h
-- Installing: /home/nvme/ffmpeg_build/lib/pkgconfig/haisrt.pc
-- Installing: /home/nvme/ffmpeg_build/lib/pkgconfig/srt.pc
-- Installing: /home/nvme/ffmpeg_build/bin/srt-live-transmit
-- Up-to-date: /home/nvme/ffmpeg_build/bin/srt-live-transmit
-- Installing: /home/nvme/ffmpeg_build/bin/srt-file-transmit
-- Up-to-date: /home/nvme/ffmpeg_build/bin/srt-file-transmit
-- Installing: /home/nvme/ffmpeg_build/bin/srt-tunnel
-- Up-to-date: /home/nvme/ffmpeg_build/bin/srt-tunnel
-- Installing: /home/nvme/ffmpeg_build/bin/srt-ffplay
nvme@CLI400:~/bsidhu/srt/build$
```

11 Revision History

Version	Change	Create Date	Author
0.1	Document Created	November 9, 2022	Balwinder Sidhu
1.0	Issued Version	November 14, 2022	Balwinder Sidhu

12 Legal Notice

Information in this document is provided in connection with NETINT products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in NETINT's terms and conditions of sale for such products, NETINT assumes no liability whatsoever and NETINT disclaims any express or implied warranty, relating to sale and/or use of NETINT products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right.

A "Mission Critical Application" is any application in which failure of the NETINT Product could result, directly or indirectly, in personal injury or death. Should you purchase or use NETINT's products for any such mission critical application, you shall indemnify and hold NETINT and its subsidiaries, subcontractors and affiliates, and the directors, officers, and employees of each, harmless against all claims costs, damages, and expenses and reasonable attorney's fees arising out of, directly or indirectly, any claim of product liability, personal injury, or death arising in any way out of such mission critical application, whether or not NETINT or its subcontractor was negligent in the design, manufacture, or warning of the NETINT product or any of its parts.

NETINT may make changes to specifications, technical documentation, and product descriptions at any time, without notice. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications.

NETINT, Codensity, and NETINT Logo are trademarks of NETINT Technologies Inc. All other trademarks or registered trademarks are the property of their respective owners.

© 2022 NETINT Technologies Inc. All rights reserved.