Android Setup Application Note

V2.0

2021-12-29

Table of Contents

1 Table of Abbreviations

Abbreviation:	Full Form:
AOSP	Android Open-Source Project
KVM	Kernel-based Virtual Machine
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 Notes:

Specifications:

3 Background

This document describes the android environment setup guide.

3.1 Intended Audience

This document is intended to help engineers/technicians/developers integrating NETINT transcoders for a live stream application.

3.2 Compatibility

Software Compatibility

This guide is intended to be used with NETINT Codensity T408/T432 Video Transcoder software release v2.5.P or newer.

Hardware Compatibility

This guide is intended to be used with NETINT Codensity T408/T432 Video Transcoder running firmware release v2.5.P or newer.

Operation system

All OS supporting T408 Video Transcoder.

4 Android emulator Environment setup

4.1 System requirement

Require a 1T hard disk on Ubuntu server

Some needed package needs to be installed:

```
sudo apt-get install libssl-dev
sudo apt-get install m4
sudo apt-get install libncurses5
```

~~Run all preparation steps under root account(sudo su)~~

Don't run with sudo su during build

sudo su is only required during device passthrough

4.2 Download the AOSP code and Build it

```
mkdir android_work  && cd android_work
```
(android_work32 for 32 bit android)

```
repo init -u https://android.googlesource.com/platform/manifest -b android-9.0.0_r35
repo sync -c -j16
```

Note: -j4 for CPUs with less capability

```
source build/envsetup.sh
lunch aosp_x86_64-eng     (for 32bit: lunch aosp_x86-eng)
make –j16  (-j4)
```

4.3 Download and build android kernel

```
cd android_work
git clone https://android.googlesource.com/kernel/goldfish
```

```
cd goldfish
git checkout android-goldfish-4.4-dev
make ARCH=x86_64 x86_64_ranchu_defconfig
```

- Copy the attachment config file to replace you .config

You can also get it from https://1drv.ms/u/s!As9sncgYSSJhh3-RxB6qeTzfrkT7?e=DCd176

and then:

```
make ARCH=x86_64 -j16   (or -j4)
```

- Or you can do the following setting to your generated .config file:

Uncomment following two lines and set to y

```
CONFIG_BLK_DEV_NVME=y
CONFIG_NVMEM=y
CONFIG_AMD_IOMMU=y
```

Then:

```
make ARCH=x86_64 -j16   (or -j4)
```

note: both 32 and 64 bit android will use x86_64 for android kernel

4.4 Check if the android emulator can boot up

You can do it from the terminal. And then check whether the emulator can launch by below cmd.

On the AOSP(android_work or android_work32) folder:

```
source build/envsetup.sh
lunch aosp_x86_64-eng
```
(for 32bit: lunch aosp_x86-eng)

```
emulator -writable-system -netdelay none -netspeed full -gpu
off -no-window -qemu -enable-kvm -kernel
goldfish/arch/x86/boot/bzImage
```

if there is this message printed then you are good to continue:

4.5 Install the VFIO on your server

sudo apt-get install virt-manager python-spice-client-gtk

Reboot and go to the BIOS and enable virtualization option (vt-x, vt-d)

After boot up Modify the sudo vim /etc/default/grub

Add below:

```
GRUB_CMDLINE_LINUX_DEFAULT="intel_iommu=on"
GRUB_CMDLINE_LINUX="intel_iommu=on"
```

sudo update-grub

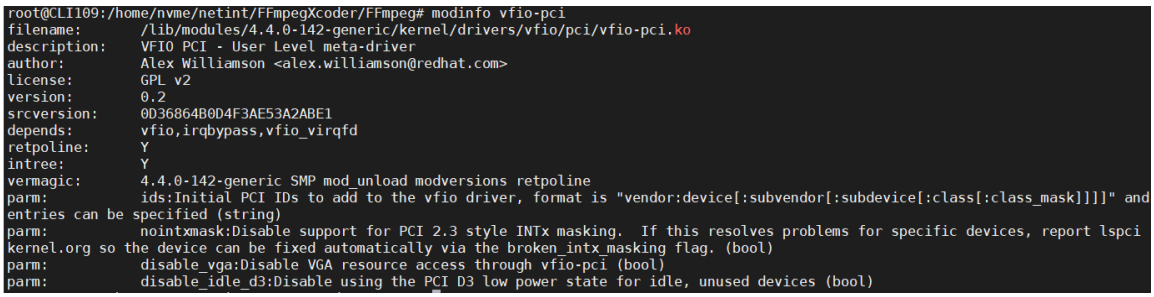
Then reboot again.

4.6 Check and insmod the VFIO module on your server

run below cmd:

```
modprobe vfio
modprobe vfio-pci
modinfo vfio-pci
```

If success it will print below info:



4.7 VFIO PCI T408 to KVM

Running “lspci” to identify your T408 pci number (Ex:0000:01:00.0) and sub-number: (1d82 0202) then run below cmd:

```
echo "0000:01:00.0" > /sys/bus/pci/drivers/nvme/unbind
```

```
echo 1d82 0202 > /sys/bus/pci/drivers/vfio-pci/new_id

chown fpga:fpga  /dev/vfio/1
```

Modify /etc/security/limits.conf

To add

```
@fpga       hard       memlock 15928456

@fpga       soft       memlock 15928456
```

note: some hosts require this to run under sudo su

Replace @fpga with your user name and replace 15442440 with your system memory size

Reference: https://www.kernel.org/doc/Documentation/vfio.txt

4.8 Start Android Emulator using VFIO PCI T408 card

On the AOSP(android_work or android_work32) folder.

```
source build/envsetup.sh

lunch aosp_x86_64-eng  (for 32bit: lunch aosp_x86-eng)

emulator -memory 4096 -partition-size 4096 -writable-system -
netdelay none -netspeed full -gpu off -show-kernel -no-window -qemu
-enable-kvm -device vfio-pci,host=0000:01:00.0 -kernel
goldfish/arch/x86/boot/bzImage
```

note: use your own pci address, 0000:01:00.0 is just an example

4.9 Check the card status when Android Emulator boot up

after emulator boot up, open another terminal and run:

```
source build/envsetup.sh
```

```
lunch aosp_x86_64-eng  (for 32bit: lunch aosp_x86-eng)

adb shell

ls /dev/

ls /dev/block/
```

if it can see the /dev/nvme0 and dev/block/nvme0n1 it means the environment is ready.

4.10 Build /hardware/interfaces/nidec (Must build on android source code)

On android AOSP folder build

Copy nidec folder to aosp/hardware/interfaces/

(You can also get nidec from https://1drv.ms/u/s!As9sncgYSSJhh3-RxB6qeTzfrkT7?e=DCd176

Or

https://networkintelligence-my.sharepoint.com/:f:/g/personal/qing_wang_netint_ca/EtaOynGKhPNMgxP8LegtEtsBGnFFZRQI4ZmNMPD5fvB1hg?e=Cl9oEN

aosp_change folder)

```
source build/envsetup.sh

      lunch aosp_x86_64-eng  (for 32bit: lunch aosp_x86-eng)

mmm aosp/hardware/interfaces/nidec/1.0
```

Copy the below generate files to andriod env:

```
adb root

adb remount

adb push aosp/out/target/product/generic_x86_64/system/lib/vndk-28/android.hardware.nidec\@1.0.so /system/lib/vndk-28/
```

```
adb push aosp/out/target/product/generic_x86_64/system/lib64/vndk-28/android.hardware.nidec\@1.0.so /system/lib64/vndk-28/

adb push aosp/out/target/product/generic_x86_64/system/lib64/android.hardware.nidec\@1.0.so /system/lib64/

adb push aosp/out/target/product/generic_x86_64/system/lib/android.hardware.nidec\@1.0.so /system/lib/

adb push aosp/out/target/product/generic_x86_64/system/lib64/android.hardware.nidec\@1.0.so /vendor/lib64/

adb push aosp/out/target/product/generic_x86_64/vendor/lib64/hw/android.hardware.nidec\@1.0-impl.so /vendor/lib64/hw/

adb push aosp/out/target/product/generic_x86_64/vendor/lib64/hw/android.hardware.nidec\@1.0-impl.so /vendor/lib64/

adb push aosp/out/target/product/generic_x86_64/vendor/bin/hw/android.hardware.nidec\@1.0-service /vendor/bin/hw/

adb push aosp/out/target/product/generic_x86_64/vendor/etc/init/android.hardware.nidec\@1.0-service.rc /vendor/etc/init/
```

4.11 Build libxcoder (Must build on android source code)

On android AOSP folder build

```
source build/envsetup.sh

lunch aosp_x86_64-eng  (for 32bit: lunch aosp_x86-eng)

mmm  ../FFmpegXcoder/libxcoder/source
```

Copy the below generate files to andriod env

```
adb root

adb remount
```

```
adb push aosp/out/target/product/generic_x86_64/system/lib64/libxcoder.so /system/lib64/

adb push aosp/out/target/product/generic_x86_64/system/lib/libxcoder.so /system/lib/

adb push aosp/out/target/product/generic_x86_64/system/lib/vndk-28/libxcoder.so /system/lib/vndk-28/

adb push aosp/out/target/product/generic_x86_64/system/lib64/vndk-28/libxcoder.so /system/lib64/vndk-28/

adb push aosp/out/target/product/generic_x86_64/vendor/bin/ni_rsrc_mon /system/bin/
```

Note: this should be the relative path to FFmpegXoder/libxcoder/source

It will report error if the path is absolute(ie: /home/fpga/FFmpegXcoder/libxcoder/source)

4.12 Push security change to environment

(You can also get nidec from https://1drv.ms/u/s!As9sncgYSSJhh3-RxB6qeTzfrkT7?e=DCd176

Or https://networkintelligence-my.sharepoint.com/:f:/g/personal/qing_wang_netint_ca/EtaOynGKhPNMgxP8LegtEtsBGnFFZRQI4ZmNMPD5fvB1hg?e=Cl9oEN

aosp_change folder)

```
adb root
adb remount
adb push mediacodec.policy /system/etc/seccomp_policy/
adb push manifest.xml vendor/
```

4.13 Prepare ffmepg4.3 environment

Download android-ndk-r21d from https://developer.android.com/ndk/downloads

You can also get it from https://1drv.ms/u/s!As9sncgYSSJhh3-RxB6qeTzfrkT7?e=DCd176

Download android-sdk_r24.4.1-linux from https://dl.google.com/android/android-sdk_r24.4.1-linux.tgz?utm_source=androiddevtools.cn&utm_medium=website

You can also get it from https://1drv.ms/u/s!As9sncgYSSJhh3-RxB6qeTzfrkT7?e=DCd176

Unpack it on your home dir

unzip android-ndk-r21d-linux-x86_64.zip

Now you should have the ndk folder

```
fpga@CLI206:~$ ls -laht |grep ndk
      drwxr-xr-x  13 fpga fpga 4.0K Sep 28 19:02 android-ndk-r21d
```

4.14 Build ffmpeg 4.3 from the FFmpegXcoder

Build the libxcoder:

```
sh build.sh -a
```

Build the ffmpeg 4.3

```
fpga@CLI206:~/FFmpegXcoder$ cd FFmpeg-n4.3
fpga@CLI206:~/FFmpegXcoder/FFmpeg-n4.3$  make clean
fpga@CLI206:~/FFmpegXcoder/FFmpeg-n4.3$  sh build_ffmpeg.sh -a --shared
fpga@CLI206:~/FFmpegXcoder/FFmpeg-n4.3$  make install
fpga@CLI206:~/FFmpegXcoder/FFmpeg-n4.3$  sudo ldconfig
```

The you will see the build out lib&bin on ./android/x86_64/lib and ./android/x86_64/bin

Note: if the build failed with **unaccepted sdk license** from build.log:

1. record the name of the license and
2. go to the android-sdk-linux/tools
3. ./android list sdk –a**l**l And check for the **ID** of the license you previously recorded
4. ./android update sdk –u -a –t **ID1**,**ID2**... or ./android update sdk –u -a (will take longer)

4.15 Push all the libs and bin to android.

Note: adb can only run when there is an emulator running in another terminal

Start another terminal, under aosp folder:

```
source build/envsetup.sh
lunch aosp_x86_64-eng    (for 32bit: lunch aosp_x86-eng)
adb root
adb remount
```

If you do not have android source code you can directly use our build libs and bin (for android x86_64 platform)

Push the aosp build output:

```
adb push aosp/out/target/product/generic_x86_64/system/bin/ASharedBufferServer   /system/bin/
adb push aosp/out/target/product/generic_x86_64/system/bin/ni_rsrc_mon /system/bin/
```

```
adb push aosp/out/target/product/generic_x86_64/system/lib64/libxcoder.so /system/lib64/
```

(note for for 32bit :

```
adb push aosp/out/target/product/generic_x86/system/bin/ASharedBufferServer  /system/bin/

adb push aosp/out/target/product/generic_x86/system/bin/ni_rsrc_mon /system/bin/

adb push aosp/out/target/product/generic_x86/system/lib/libxcoder.so /system/lib/
```

)

Push the ffmpeg4.3 bin and lib build output:

```
adb push ./FFmpegXcoder/FFmpeg-n4.3/android/x86_64/bin/* /system/bin/

adb push ./FFmpegXcoder/FFmpeg-n4.3/android/x86_64/lib/* /system/lib64/
```

Make soft link on the android env for libraries:

```
source build/envsetup.sh

lunch aosp_x86_64-eng    (for 32bit: lunch aosp_x86-eng)

adb root

adb remount

adb shell

cd  /system/lib64 (for 32bit: cd  /system/lib)

ln -s libxcoder.so libxcoder.so.250R1F10

reboot the android system
```

Note: 250R1F10 is the version number, change according to your release version

4.16 Run the ffmpeg on android platfrom

(Only run once when android system boot up)

```
adb shell
./vendor/bin/hw/android.hardware.nidec@1.0-service &
 ni_rsrc_mon
```

run the ffmpeg:

FFMPEG COMMAND

5 robox Environment setup

5.1 System requirement

Require a 1T hard disk on Ubuntu server

Run all preparation steps under root account(sudo su)

#sudo passwd root

#sudo vi /etc/ssh/sshd_config

Find the PermitRootLogin and make it to yes.

PermitRootLogin = yes

sudo service ssh restart

5.2 Install the necessary tools

#sed -i "s/# deb-src/ deb-src/g" /etc/apt/sources.list

#apt-get update

#apt-get install build-essential cmake cmake-data debhelper dbus google-mock libboost-dev libboost-filesystem-dev libboost-log-dev libboost-iostreams-dev libboost-program-options-dev libboost-system-dev libboost-test-dev libboost-thread-dev libcap-dev libdbus-1-dev libdbus-cpp-dev libegl1-mesa-dev libgles2-mesa-dev libglib2.0-dev libglm-dev libgtest-dev liblxc1 libproperties-cpp-dev libprotobuf-dev libsdl2-dev libsdl2-image-dev lxc-dev pkg-config protobuf-compiler

#apt install build-essential

#apt install mesa-common-dev

#apt-get install docker.io

#apt-get install xvfb x11vnc dkms

Some libs need get from below link then install. (Just for X86. If you use ARM server pls change the setting from x86_64 to arm64)

https://1drv.ms/u/s!As9sncgYSSJhiA9L1Bxo1a2_wn83?e=rXyTYg

dkms install libdbus-cpp5_5.0.0+16.10.20160809-0ubuntu2_amd64.deb

dkms install libdbus-cpp-dev_5.0.0+16.10.20160809-0ubuntu2_amd64.deb

dkms install libprocess-cpp3_3.0.1-0ubuntu5_amd64.deb

dkms install libprocess-cpp-dev_3.0.1-0ubuntu5_amd64.deb

apt install xfce4 xfce4-* xrdp

#vmi ~/.xsession

Add "xfce4-session" on it.

/etc/init.d/xrdp restart

```
git clone https://github.com/anbox/anbox-modules.git
cd anbox-modules/
cp anbox.conf /etc/modules-load.d/
cp 99-anbox.rules /lib/udev/rules.d/
cp -rT ashmem /usr/src/anbox-ashmem-1
cp -rT binder /usr/src/anbox-binder-1
dkms install anbox-ashmem/1
dkms install anbox-binder/1
```

#chmod 777 /tmp

#vim /etc/systemd/logind.conf

Modify "UserTasksMax=200000"

#reboot

5.3 Download the anbox code and Build libxcoder on it.

(Just for X86. If you use ARM server pls change the setting from x86_64 to arm64)

```
$ cd $HOME/anbox-work$ repo init -u
https://github.com/anbox/platform_manifests.git -b anbox$ repo sync –
j16

$ Source build/envsetup.sh

$ lunch anbox_x86_64-userdebug

$ make –j8

$ mkdir vendor/

$ cp libxcoder vendor/

$ mmm vendor/libxcoder/source/

$ cp out/target/product/x86_64/system/lib64/libxcoder.so “your libs anb
bins folder”.

$ cp out/target/product/x86_64/system/bin/ASharedBufferServer “your
libs anb bins folder”.

$ cp out/target/product/x86_64/system/bin/ni_rsrc_mon “your libs anb
bins folder”.


Build ffmpeg pls reference the #3 chapter on ffmpeg-4.3.

The copy all the libs and bin to “your libs folder”.
```

5.4 Download the robox code and Build robox.

#git clone https://github.com/lag-linaro/robox.git -b release-phase2.3

#cd robox

#mkdir build

#cd build

#cmake ..

#make –j8

#make install

5.5 Start robox

Option1: (use already build out android.img)

You can get anroid.img from https://build.anbox.io/android-images/2018/07/19/ (for X86_64 server)

https://mirrors.huaweicloud.com/kunpeng/archive/kunpeng_solution/native/android_20191126.img (for arm64 server)

Option2:(build android.img by yourself)

You can build android image by yourself:

```
$ repo init -u https://github.com/anbox/platform_manifests.git -b anbox
$ repo sync -j4
$ . build/envsetup.sh
```

Then initialize the build using `lunch`.

```
$ lunch anbox_x86_64-userdebug
$ make -j8
$ cd $HOME/anbox-work/vendor/anbox$ scripts/create-package.sh \
$PWD/../../out/target/product/x86_64/ramdisk.img \
$PWD/../../out/target/product/x86_64/system.img
```

(replace `x86_64` with your target architecture)

This will create an *android.img* file in the current directory.

mount anroid.img /mnt

cd /mnt

#tar --numeric-owner -cf- . | docker import - android

Check if the image is installed on docker.

#docker images

#rmmod binder_linux ashmem_linux

export XDG_RUNTIME_DIR=/run/user/0 (make sure the /run/user/0 exist)

Enter to the robox root folder. Replace the robox on robox root folder.

https://1drv.ms/u/s!As9sncgYSSJhiA9L1Bxo1a2_wn83?e=rXyTYg

Option1: (share folder between ubuntu and robox)

Modify the robox below line --volume to your "`your libs and bins folder`". folder.

--volume=/home/nvme/netint/robox-work/robox-work/output_2.4.0_android7:/output_2.4.0_android7:rw

Option2: (copy libs and bins to robox)

Copy all the libs and bins to robox system /system/lib64/ and system/bin/ folder.

Chmod 777 libs and bin

Then:

```
./robox -v start 1
```

If you want to exit anbox pls use.

```
./robox -v stop 1
```

Entor the robox container:

```
docker exec -it instance1 sh
```

Run the cmd on robox container:

```
LD_LIBRARY_PATH=libs/ libs/bin/ASharedBufferServer &

LD_LIBRARY_PATH=libs/ libs/bin/ni_rsrc_mon

LD_LIBRARY_PATH=libs/ libs/bin/ffmpeg -y -hide_banner -nostdin -vsync 0 -c:v h264_ni_dec -i test/1280x720p_Basketball.264 -c:v h265_ni_enc output_9. h265
```

6 Revision History

Version	Change	Create Date	Author
1.0	Document Created	November 02,2020	Qing Wang
1.1		November 09,2020	Qing Wang
1.2		November 24, 2020	Jinge Li
1.3	Add robox	November 25, 2020	Qing Wang
1.4	Use ffmpeg 4.3	September 29,2021	Shawn Hu
1.5	Safec considerations	October 31, 2021	Qing Wang
2.0	Update as per new app note template	December 9, 2021	Balwinder Sidhu

7 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.

© 2021 NETINT Technologies Inc. All rights reserved.