Part 6. Validate the Setup

Table of contents

In this section, you will validate the ARC-OTA setup using bi-directional UDP.

Step 1: Add the SIM User Profile

Modify the following files:

- `oai_db.sql`

  There are 3 UEs pre-configured in this file. To find them, search for `001010000000001` and add or edit them as needed.

- `./targets/PROJECTS/GENERIC-NR-5GC/CONF/vnf.sa.band78.fr1.273PRB.Aerial.conf`

  Modify this file on the gNB server if you want to change the MCC and MNC in the gNB config file.

Step 2: Setup the UE and SIM Card

For reference, use the following: SIM cards – 4G and 5G reference software (open-cells.com)

Program the SIM Card with the Open Cells Project application “uicc-v2.6”, which can be downloaded here.

Use the ADM code specific to the SIM card. If the wrong ADM is used 8 times, the SIM card will be permanently locked.

```
sudo ./program_uicc --adm 12345678 --imsi 001010000000001 --isdn 00000001 --acc 0001 --key fec86ba6eb707ed08905757b1bb44b8f --opc C42449363BBAD02B66D16BC975D77CC1 -spn "OpenAirInterface" --authenticate Existing values in USIM ICCID: 89860061100000000191 WARNING: iccid luhn encoding of last digit not done USIM IMSI: 208920100001191 USIM MSISDN: 00000191 USIM Service Provider Name: OpenCells191 Setting new values Reading UICC values after uploading new values ICCID: 89860061100000000191 WARNING: iccid luhn encoding of last digit not done USIM IMSI: 001010000000001 USIM MSISDN: 00000001 USIM Service Provider Name: OpenAirInterface Succeeded to authentify with SQN: 64 set HSS SQN value as: 96
```

CUE Configuration Setup

Install the “Magic IPERF” application on the UE:

1. To test with CUE, a test SIM card with **Milenage** support is required. The following must be provisioned on the SIM card and must match the Core Network settings: mcc, mnc, IMSI, Ki, OPc.

2. The APN on the CUE should be configured according to the Core Network settings.

3. Start the DNS. Core Network should assign a mobile IP address and DNS. If DNS is not assigned, set the DNS with the other Android app.

Step 3. Running End-to-End OTA

This section describes how to run end-to-end traffic from the UE to the edge Core Network.

> ⓘ **Note**
>
> The UE can connect as close as 2-3 meters, with a maximum range of 10-15 meters. The connection distance outside of buildings has not been unverified.

Start CN5G Core Network

Use the following commands to start the CN5G Core Network.

```
sudo sysctl net.ipv4.conf.all.forwarding=1 sudo iptables -P FORWARD ACCEPT cd ~/oai-cn5g docker compose up -d
```

Start theCN5G Edge Application

After the CN5G is started, use the `oai-ext-dn` container to run IPERF.

```
docker exec -it oai-ext-dn /bin/bash
```

Start Aerial cuBB on the gNB

Execute the following command to set up the cuBB container.

```
# Run on host: start a docker terminal docker exec -it $AERIAL_CUBB_CONTAINER /bin/bash
```

Follow the Aerial cuBB documentation to build and run the cuphycontroller. The following instructions are for building and setting up the environment for running cuphycontroller. The following commands must be run from inside the cuBB container.

```
cd /opt/nvidia/cuBB export cuBB_SDK=$(pwd) mkdir build && cd build cmake .. make -j
```

The Aerial cuPHY configuration files are located in the `/opt/nvidia/cuBB/cuPHY-CP/cuphycontroller/config` directory. For ARC-OTA 1.3, the setup has been validated with `cuphycontroller_P5G_FXN.yaml` for the the Gigabyte server and with `cuphycontroller_P5G_FXN_R750.yaml` for the Dell R750 server.

Before running the cuphycontroller, edit the `cuphycontroller_<xyz>.yaml` configuration file to point to the correct MAC address of the ORU and the correct PCIe address of the FH interface on the gNB.

```
sed -i "s/ nic:.*/ nic: 0000:b5:00.0/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_<xyz>.yaml sed -i "s/ dst_mac_addr:.*/ dst_mac_addr: 6c:ad:ad:00:02:02/" ${cuBB_SDK}/cuPHY-CP/cuphycontroller/config/cuphycontroller_<xyz>.yaml
```

When the build is done and the configuration files are updated, exit the container. Run the following to commit the changes to a new image. The name of the image will be used later in the Docker Compose configuration.

```
docker commit $AERIAL_CUBB_CONTAINER cubb-build:23-4
```

Creating the NVIPC Source Code Package

In the latest release, NVIPC is no longer included in the OAI repository. You must copy the source package from the Aerial cuBB container and add it to the OAI build files. Execute the following to create the `nvipc_src.<data>.tar.gz` file.

```
docker exec -it $AERIAL_CUBB_CONTAINER ./cuPHY-CP/gt_common_libs/pack_nvipc.sh docker cp $AERIAL_CUBB_CONTAINER:/opt/nvidia/cuBB/cuPHY-CP/gt_common_libs/nvipc_src.<date>.tar.gz ~/openairinterface5g
```

This will create and copy the `nvipc_src.<data>.tar.gz` archive that is required to build OAI L2+ for Aerial. Instructions can also be found in the Aerial FAPI.

Build gNB Docker Image

In ARC-OTA 1.3, the OAI image is built in two steps. Instructions can be found in the Aerial FAPI.

> **Note**
>
> When building a Docker image, the files are copied from the filesystem into the image. After you build the image, you must make changes to the configuration inside the container.

Pre-build Steps for OAI L2

1. When building OAI L2 for Aerial 23-4, remove line 50 and 51 in the `docker/Dockerfile.gNB.aerial.ubuntu20` file before building. (ARC-OTA 1.3. includes the `OAI 2024.w15` tag and Aerial CUDA-Accelerated RAN (Layer 1) 23-4). This will be merged to OAI `develop` branch in the future.

2. In the same file, add `moreutils` on line 79.

```
moreutils \
```

3. Remove the pinning of `p7_thread` in the OAI FAPI integration (`nfapi/oai_integration/aerial/fapi_vnf_p5.c`). To do so, remove lines 58-64. Core 8 is occupied by L1 on the Gigabyte server and on the Dell the R750 server. This will be merged to the OAI `develop` branch in the future.

```
#CPU_SET(8, &cpuset); #s = pthread_setaffinity_np(pthread_self(), sizeof(cpu_set_t), &cpuset); #if (s != 0) # printf("failed to set afinity\n");
```

4. Ensure the the FAPI polling thread is run on an isolated core by modifying line 585 of the `nfapi/oai_integration/aerial/fapi_nvIPC` file.

   On the Gigabyte server, use core 10:

```
stick_this_thread_to_core(10)
```

   And on the Dell R750, use core 21:

```
stick_this_thread_to_core(21)
```

5. There is also a memory leak in the `OAI 2024.w15` tag used with ARC 1.3. This is already fixed on the `develop` branch of OAI. The current ARC release has been verified by applying this MR on top of the 2024.w15 `OAI 2024.w15` tag.

```
cd ~/openairinterface5g/ wget https://gitlab.eurecom.fr/oai/openairinterface5g/-/merge_requests/2747.patch git
```

```
apply 2747.patch
```

Use the below commands to build the OAI L2 code:

```
cd ~/openairinterface5g/ docker build . -f docker/Dockerfile.base.ubuntu20 --tag ran-base:latest docker build . -f docker/Dockerfile.gNB.aerial.ubuntu20 --tag oai-gnb-aerial:latest
```

This will create two images:

- `ran-base:latest` includes the environment to build the OAI source code. This will be used then building the `oai-gnb-aerial:latest` image.
- `oai-gnb-aerial:latest` will be used later in the Docker Compose script to create the OAI L2 container.

Running gNB with Docker Compose

ARC-OTA 1.3 includes a Docker Compose configuration for running gNB software. Refer to the OAI instructions on how to run with Docker Compose. The Docker Compose configuration is located in the following file path:

```
~/openairinterface5g/ci-scripts/yaml_files/sa_gnb_aerial/docker-compose.yaml
```

Before starting the gNB, you will need to update the `docker-compose.yaml` file.

1. On line 27, switch the cuBB image to the image that was built, committed, and tagged in the previous steps.

```
image: cubb-build:23-4
```

1. On line 30, add `sudo` to the command.

```
command: bash -c "sudo rm -rf /tmp/phy.log && sudo chmod +x
/opt/nvidia/cuBB/aerial_l1_entrypoint.sh &&
/opt/nvidia/cuBB/aerial_l1_entrypoint.sh"
```

1. On line 37, if you follow the guide, change the `cpu_set` for the OAI container to cores “13-20” for a Gigabyte server and cores “23,25,27,29,31,33,35,37” for a Dell R750 server.
    1. Gigabyte server:

    ```
    cpu_set: "13-20"
    ```

    2. Dell R750 server:

    ```
    cpu_set: "23,25,27,29,31,33,35,37"
    ```

2. On line 60, add a volume for the `oai.log` file.

1. On line 61 (after the volumes), add a command for executing the `nr-softmodem`. This will replace the default command that is integrated in the docker image. This new command will set the priority and create an `oai.log` file with time stamp.

```
command: bash -c "chrt -f 99 /opt/oai-gnb/bin/nr-softmodem -O /opt/oai-
gnb/etc/gnb.conf | ts | tee /var/log/aerial/oai.log"
```

After following the instructions, you should have the following images:

```
$ docker image ls REPOSITORY TAG IMAGE ID CREATED SIZE cubb-build 23-4
be7a5a94f2d3 10 seconds ago 26GB gitlab-
master.nvidia.com:5005/gputelecom/container/cubb Aerial-cuBB-container-
ubuntu22.04-23.04.0-Rel-23-4.256-x86_64 3a46cace77de 3 weeks ago 24.6GB oai-
gnb-aerial latest 0856b9969f42 3 weeks ago 4.88GB ran-base latest c5d060d23529 3
weeks ago 2.42GB
```

Docker Compose will start containers running cuBB and OAI L2+. The Docker Compose script includes an entry-point script for cuBB, which you need to modify before running. The script points at the `cuphycontroller_xxx.yaml` configuration that you want to run. This script is located in the following file path:

```
~/openairinterface5g/ci-scripts/yaml_files/sa_gnb_aerial/aerial_l1_entrypoint.sh
```

Before running Docker Compose, update the `aerial_l1_entrypoint.sh` file.

1. The latest ARC-OTA release does not have to build `gdrcopy`.

```
# cd "$cuBB_Path"/cuPHY-CP/external/gdrcopy/ || exit 1 # ./insmod.sh
```

1. In the latest ARC-OTA release, you are not root by default. Add `sudo` to the following, including `P5G_FXN_R750` if you are using a Dell R750 or P5G_FXN if you are using .

```
sudo -E nvidia-cuda-mps-control-d sudo -E echo start_server -uid 0 | sudo -E nvidia-cuda-mps-control sudo -E "$cuBB_Path"/build/cuPHY-CP/cuphycontroller/examples/cuphycontroller_scf P5G_FXN_R750
```

Before running the Docker Compose script, you also need to add root for some commands that are run before starting Aerial cuPHY Layer 1. To do so, edit the `ci-scripts/yaml_files/sa_gnb_aerial/docker-compose.yaml` file.

```
command: bash -c " sudo rm -rf /tmp/phy.log && sudo chmod +x /opt/nvidia/cuBB/aerial_l1_entrypoint.sh && /opt/nvidia/cuBB/aerial_l1_entrypoint.sh"
```

Also, OAI L2 must point to the correct configuration by editing the following row in the `ci-scripts/yaml_files/sa_gnb_aerial/docker-compose.yaml` file. All L2 configurations can be found in `targets/PROJECTS/GENERIC-NR-5GC/CONF`.

```
- ../../conf_files/gnb-vnf.sa.band78.273prb.aerial.conf:/opt/oai-gnb/etc/gnb.conf
```

You can now run ARC-OTA with the below command.

```
docker compose up -d // console of cuBB docker logs -f nv-cubb // console of oai docker logs -f oai-gnb-aerial // tail -f /var/log/aerial/oai.log
```

CUE Connecting to 5G Network

Take the CUE out of airplane mode to start attaching the UE to the network. Make sure that the CUE is in airplane mode before starting OAI L2 stack.

Observe 5G Connect Status

Refer to the Preamble log in the cuphycontroller console output.

Check the Core Network log or CUE log to see whether NAS authentication and PDU session succeeded.

Running E2E IPERF Traffic

Start `ping`, `iperf`, or other network app tests after the PDU session connects successfully.

You can install and run the “Magic IPerf” Android application on the CUE for this purpose.

Ping Test

Ping the UE from the CN:

```
docker exec -it oai-ext-dn ping 12.1.1.2
```

Ping from the UE to the CN:

```
ping -I 12.1.1.2 192.168.70.135
```

Iperf Downlink Test

Perform Iperf downlink test on the UE Side:

```
iperf3 -s
```

Perform Iperf downlink test on the CN5G Side:

```
# Test UDP DL docker exec -it oai-ext-dn iperf3 -u -P 20 -b 25M -t 60 -c 12.1.1.2 #Test UDP bidirectional docker exec -it oai-ext-dn iperf3 -u --bidir -P 10 -b 50M -t 60 -c 12.1.1.2 # Test TCP DL docker exec -it oai-ext-dn iperf3 -P 4 -b 100M -t 60 -c 12.1.1.2 #Test TCP bidirectional docker exec -it oai-ext-dn iperf3 --bidir -P 4 -b 100M -t 60 -c 12.1.1.2
```

IPERF Uplink Test

Perform Iperf uplink test on the UE Side:

```
iperf3 -s
```

Perform Iperf uplink test on the CN5G Side:

```
#UDP docker exec -it oai-ext-dn iperf3 -u -R -b 120M -t 60 -c 12.1.1.2 #TCP docker exec -it oai-ext-dn iperf3 -R -b 120M -t 60 -c 12.1.1.2
```

To stop the containers, use the following commands:

```
docker stop $OAI_GNB_CONTAINER docker rm $OAI_GNB_CONTAINER
```

Note

ARC-OTA is a P5G cellular network; specific enterprise switching/routing/firewalls/policies might need integration support to

enable access to the World Wide Web.