Quick Start Guide

Network and Edge Reference System Architectures - vRAN Setup with FlexRAN™ Software

FlexRAN™ software on a single server setup based on 4th Gen Intel® Xeon® Scalable processor platform with Intel® vRAN Boost

Authors

Abhijit Sinha

Guoshu Xu

Introduction

The Reference System Architectures (Reference System[1]) are forward-looking Kubernetes*-cluster cloud-native reference platforms aiming to ease the development and deployment of network and edge solutions.

The scope of this document is to walk-through, from software components and Kubernetes installation to onboard the **Intel® FlexRAN™ software**[2] on a single node cluster on a **4th Gen Intel® Xeon® Scalable processor**-based platform with **Intel® vRAN Boost**. The **Intel® FlexRAN™ software** can then be tested using the Timer and/or the xRAN test modes.

The Reference System Architectures allow deployment and customization of the required software in the form of Configuration Profiles mapped to specific use cases. For more details on this setup and other Configuration Profiles, refer to the User Guides listed in the Reference Documentation section.

Architecture and Setup

Figure 1 shows the architecture diagram of the Access Profile, which deploys infrastructure for the vRAN workload. The profile enables the Intel® SRIOV-FEC operator, SRIOV network plugins, and Intel® oneAPI libraries on a real-time OS for deploying Intel® FlexRAN™ software.

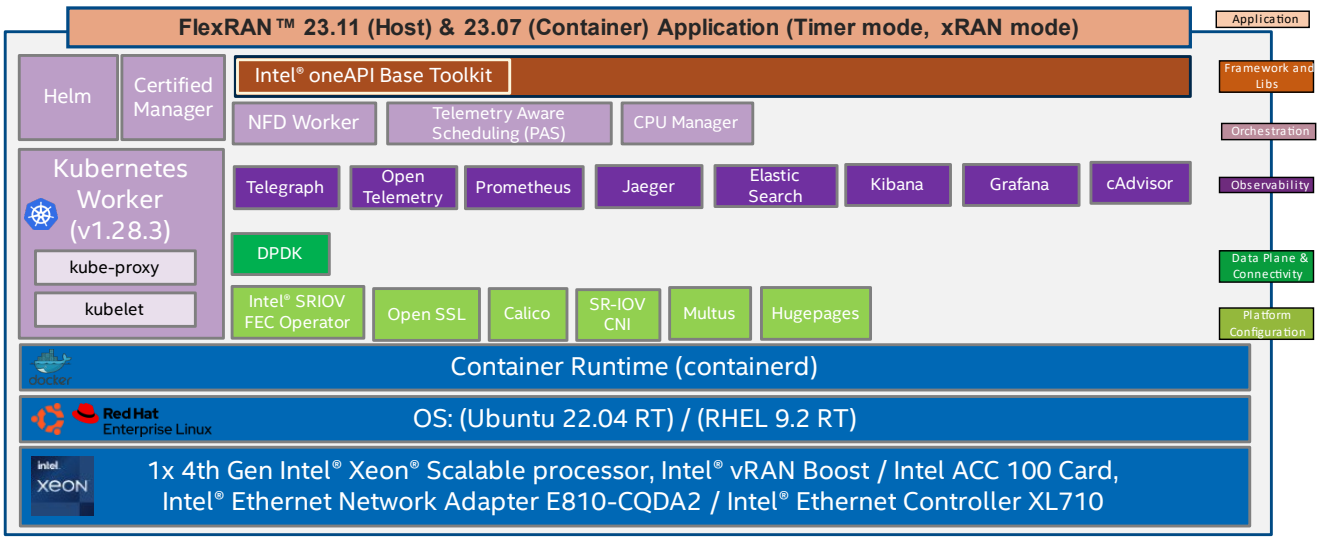

Figure 1: Architecture of FlexRAN™ software deployment using BMRA Access Profile

[1] In this document, "Reference System" refers to the Network and Edge Reference System Architecture.

[2] Intel, the Intel logo, and FlexRAN™ are trademarks of Intel Corporation or its subsidiaries.

Hardware BOM

Following is the list of the hardware components that are required for setting up FlexRAN™ software:

Ansible Host	Laptop or server running a UNIX base distribution
Target Server	1x 4th Gen Intel® Xeon® Scalable processors (6421N) on Intel reference platform (code named Archer City) 1x 4th Gen Intel® Xeon® Scalable processors (6421N) on Intel Quanta SDP 1x 4th Gen Intel® Xeon® Scalable processors with Intel® vRAN Boost on Intel Quanta SDP
FEC Accelerator	Intel® vRAN Accelerator ACC100 Adapter on the target BBU server Note: The above is not required for 4th Gen Intel® Xeon® Scalable processor with Intel® vRAN Boost
Ethernet Adapter	Intel® Ethernet Network Adapter E810-CQDA2 or Intel® Ethernet Controller XL710 on the target server
Recommended BIOS	"Low Latency" BIOS configuration (Refer to Chapter 3.8 of BMRA User guide)

Software BOM

Following is the list of the software components that are required for setting up FlexRAN™ software:

FlexRAN™ Application	Intel® FlexRAN™ software v23.11 for bare metal (host) Intel® FlexRAN™ software v23.07 for container
OneAPI	Intel® oneAPI Base Toolkit (Base Kit)
Security	OpenSSL
Observability	Telegraf Open Telemetry Prometheus Grafana Jaeger cAdvisor
Acceleration/ Data Plane	DPDK 22.11
Connectivity	SRIOV-CNI
Operators & Device Plugins	Intel® SRIOV-FEC Operator Multus
Ethernet Drivers	i40e, ice, iavf
Container Runtime	containerd
Orchestration	K8s v1.28.3 Node Feature Discovery
OS	Ubuntu 22.04 LTS with real-time (kernel: 5.15.0.1036-realtime) RHEL 9.2 with real-time (kernel: 5.14.0-284.11.1.rt14.296.el9_2.x86_64)

For details about the software versions for the **Access Edge** Configuration Profile, refer to Chapter 4 of the BMRA User Guide listed in the Reference Documentation section.

FlexRAN Deployment using BMRA

Pre-Requisites

Before starting the deployment, perform the following steps:

- A fresh OS installation is expected on the controller and target nodes to avoid a conflict between the RA deployment process with the existing software packages. To deploy RA on the existing OS, ensure that there is no prior Docker or Kubernetes* (K8s) installations on the server(s).
- The hostname must be in lowercase, numerals and,' - 'format only for the target server.
    - For example: wrk-8 is acceptable, wrk_8, WRK8, Wrk^8 are not accepted as hostnames.
- The BIOS on the target server is set as per the recommended settings.

Deployment Setup

The FlexRAN™ software deployment as containers/POD, only needs one server for both timer and xRAN tests as shown in Figure 2. The bare metal deployment of FlexRAN™ software needs one server platform for timer mode tests and two server platforms for xRAN tests, where the second server emulates the Remote Radio Unit (oRU) as shown in Figure 3. The Ansible host is used for configuring and deploying BMRA on a set of target servers.

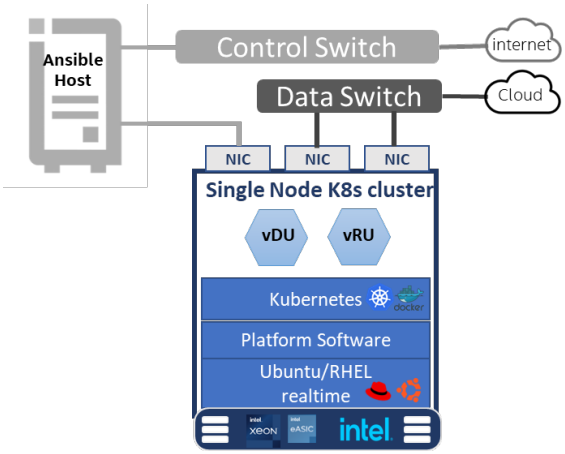

Figure 2: FlexRAN™ POD Deployment using BMRA

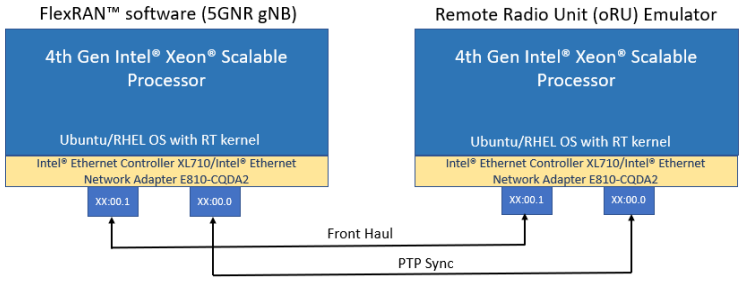

Figure 3: FlexRAN™ Bare Metal Deployment (xRAN test case)

Installation Flow for RA Deployment

Ansible playbooks are used to deploy the FlexRAN software and the necessary software packages using the Access Profile. Before the playbooks can be run, there are a few steps to prepare the environment and change relevant configuration options.

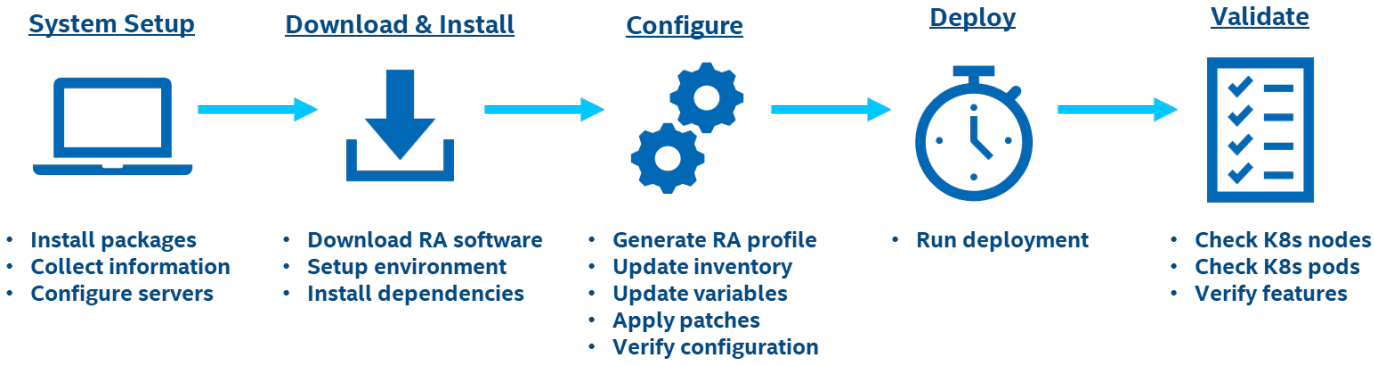

Figure 4: RA Deployment using Ansible Playbooks

Getting Started

Download the following files from the Intel® Developer Zone portal:

Files	Download URL
FlexRAN-23.11-L1.tar.gz_part0	https://cdrdv2.intel.com/v1/dl/getContent/794524
FlexRAN-23.11-L1.tar.gz_part1	https://cdrdv2.intel.com/v1/dl/getContent/794525
dpdk_patch-23.11.patch	https://cdrdv2.intel.com/v1/dl/getContent/794526

Note: The above files are only needed when you deploy FlexRAN™ software on bare metal. To obtain the files, make sure you have an account in the Intel® Developer Zone Portal. These can be downloaded to your laptop and later transferred to the Linux server as mentioned in the steps below.

Step 1 - Set Up the System

The steps described below assume that both the Ansible host and target server are running Ubuntu as the operating system. For RHEL, use 'yum' or 'dnf' as the package manager instead of 'apt'.

Ansible Host

System Setup

1. Install the necessary packages (some might already be installed):

```
# sudo apt update
# sudo apt install -y python3 python3-pip openssh-client git build-essential
# pip3 install --upgrade pip
```

2. Generate an SSH keypair if needed (check /root/.ssh/):

```
# ssh-keygen -t rsa -b 4096 -N "" -f ~/.ssh/id_rsa
```

3. Copy the public key to the target server:

```
# ssh-copy-id root@<target IP>
```

4. Verify password-less connectivity to the target server:

```
# ssh root@<target IP>
```

Target Servers

The following steps are required for all the target nodes: FlexRAN™ software node and oRU node.

1. Install Ubuntu 22.04 or RHEL 9.2 with Real-Time (RT) kernel. You can follow the steps here as a reference for Ubuntu.
2. Verify that the kernel is tagged as a real-time kernel.

```
# uname -ri
5.15.0-1036-realtime x86_64
```

3. Install necessary packages (some might already be installed).

```
# sudo apt install -y python3 openssh-server lshw
```

4. As part of the configuration in Step 3, information about PCI devices for SR-IOV and FEC accelerator must be specified.
5. Find the relevant Network PCI IDs (bus:device.function) using 'lspci' and note down the IDs for later when configuring host_vars on the Ansible host.

```
# lspci | grep Eth
18:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 01)
18:00.1 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 01)
```

6. Find the FEC accelerator card's PCI IDs (domain:bus:device.function) using 'lspci' and confirm that the device ID is '0d5c' and note it down for later when configuring host_vars on the Ansible host.

```
# lspci -nnD | grep -i acc
0000:f7:00.0 Processing accelerators [1200]: Intel Corporation Device [8086:57c0]
```

The following steps (7 and 8) are only needed for the Bare Metal deployment:

7. Copy the FlexRAN™ software packages and merge them into one final package.

```
# mkdir -p /opt/cek/intel-flexran/
# cat FlexRAN-23.11-L1.tar.gz_part0 FlexRAN-23.11-L1.tar.gz_part1 > FlexRAN-23.11.tar.gz
```

8. Extract the FlexRAN-23.11 software, follow the ReadMe.txt, and install the FlexRAN™ software.

```
# cd /opt/cek/intel-flexran/
# tar -xvf FlexRAN-23.11.tar.gz
# cat ReadMe.txt
# ./extract.sh
```

Note: During the installation, all EULA must be reviewed and "manually" accepted on the terminal screen.

Step 2 - Download and Install

Download & Install

Ansible Host

1. Download the source code from the GitHub repository for the Reference System server.

```
# git clone https://github.com/intel/container-experience-kits/
# cd container-experience-kits
# git checkout v24.01
```

2. Set up Python* virtual environment and install dependencies.

```
# python3 -m venv venv
# source venv/bin/activate
# pip3 install -r requirements.txt
```

3. Install Ansible dependencies for the Reference System.

```
ansible-galaxy install -r collections/requirements.yml
```

4. Unzip and copy the DPDK patch (needed only for Bare Metal deployment)

```
# mkdir -p /opt/patches/flexran/dpdk-stable-22.11.1/
# cp dpdk_patch-23.11.patch /opt/patches/flexran/dpdk-stable-22.11.1/
```

Step 3 - Configure

The **Access Edge** configuration profile is used for **FlexRAN™** software deployment.

Configure

Configuring BMRA for FlexRAN™ Software

Ansible Host

1. Generate the configuration files for Access Profile.

```
# export PROFILE=access
# make k8s-profile PROFILE=${PROFILE} ARCH=spr
```

2. Update the **inventory.ini** file to match the target server's hostname. The values for *<bbu hostname>* and *<bbu IP>* must be updated to match the target system.

   Note: For xRAN test mode on Bare Metal deployment, a separate oRU node is required as shown in Figure 2. For xRAN tests in POD mode, the vDU the vRU containers is deployed on the BBU node and you can comment-out the oRU node.

```
# vim inventory.ini

[all]
<bbu hostname>      ansible_host=<bbu IP> ip=<bbu IP> ansible_user=root
<oru hostname>      ansible_host=<oru IP> ip=<oru IP> ansible_user=root
localhost           ansible_connection=local ansible_python_interpreter=/usr/bin/python3

[vm_host]

[kube_control_plane]
<bbu hostname>

[etcd]
```

```
<bbu hostname>

[kube_node]
<bbu hostname>

[oru]
<oru hostname> #Comment this out for POD deployment

[k8s_cluster:children]
kube_control_plane
kube_node

[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

3. Update the host_vars filenames with the target machine's hostnames.

```
# cp host_vars/node1.yml host_vars/<bbu hostname>.yml
# cp host_vars/node1.yml host_vars/<oru hostname>.yml #Only in case of xRAN test mode in BM
```

To utilize features depending on SR-IOV, FEC accelerator, host_vars must be updated with information about the PCI devices on the target server. The example below can be used as a reference for the configuration but should be updated to match the correct PCI IDs of the target servers.

4. Update *host_vars/<bbu_hostname>.yml* with PCI device information specific to the target servers. You need two PFs and a minimum of four VFs per PF.

```
# vim host_vars/<bbu hostname>.yml
dataplane_interfaces:
  - bus_info: "18:00.0"
    pf_driver: "iavf"
    default_vf_driver: "vfio-pci"
    sriov_numvfs: 4
  - bus_info: "18:00.1"
    pf_driver: "iavf"
    default_vf_driver: "vfio-pci"
    sriov_numvfs: 4
```

5. Make the below changes for enabling the DPDK patch and adding the FEC acc card in *host_vars/<bbu_hostname>.yml*.

```
# vim host_vars/<bbu hostname>.yml
fec_acc: "dddd:bb:ss.f" # Wireless FEC H/W Accelerator Device (e.g. ACC100/ACC200) PCI ID
dpdk_local_patches_dir: "/opt/patches/flexran"
dpdk_local_patches_strip: 1
```

6. Make sure that the Intel® QAT is turned off on targets in *host_vars/<bbu_hostname>.yml*.

```
# vim host_vars/<bbu hostname>.yml
update_qat_drivers: false
openssl_install: false
```

7. Make sure the below parameters are set correctly in *group_vars/all.yml*.

```
# vim group_vars/all.yml
profile_name: access
configured_arch: spr
preflight_enabled: true
intel_sriov_fec_operator_enabled: false (we set this to 'false' as we use sriov device
plugin for enabling FEC device in this release)
```

8. (Optional) In case the CPU SKU is not listed, add in the *group_vars/all.yml*.

```
# vim group_vars/all.yml
unconfirmed_cpu_models: ['6443N']
```

9. Set the FlexRAN™ test mode in *group_vars/all.yml* as per your testing need.

```
# vim group_vars/all.yml
intel_flexran_enabled: true # if true, deploy FlexRAN
intel_flexran_mode: "timer" # supported values are "timer" and "xran"
```

10. Set the FlexRAN™ deployment mode as HOST or POD in *group_vars/all.yml* based on the deployment model.

```
# vim group_vars/all.yml
intel_flexran_type: "pod" # supported values are "host" and "pod"
```

11. Set the below network interfaces in *group_vars/all.yml* for xRAN testing mode (ignore it for timer mode tests).

```
# vim group_vars/all.yml (only for xran test mode. Refer to Figure 1 for more info)
intel_flexran_bbu_front_haul: "0000:43:00.0"
intel_flexran_bbu_ptp_sync: "0000:43:00.1"
intel_flexran_oru_front_haul: "0000:4b:00.0"
intel_flexran_oru_ptp_sync: "0000:4b:00.1"
```

12. If the server is behind a proxy, update *group_vars/all.yml* by updating and uncommenting the lines for `http_proxy`, `https_proxy`, and `additional_no_proxy`.

```
# vim group_vars/all.yml
## Proxy configuration ##
http_proxy: "http://proxy.example.com:port"
https_proxy: "http://proxy.example.com:port"
additional_no_proxy: ".example.com,mirror_ip"
```

13. (Required) Apply required patches for Kubespray.

```
# ansible-playbook -i inventory.ini playbooks/k8s/patch_kubespray.yml
```

14. (Optional) It is recommended that you check dependencies of components enabled in group_vars and host_vars with the package dependency checker.

```
# ansible-playbook -i inventory.ini playbooks/preflight.yml
```

15. (Optional) Verify that Ansible can connect to the target server, by running the below command and checking the output generated in the **all_system_facts.txt** file.

```
# ansible -i inventory.ini -m setup all > all_system_facts.txt
```

Step 4 -Deploy

Ansible Host

Now the Reference System can be deployed by using the following command:

```
# ansible-playbook -i inventory.ini playbooks/access.yml --flush-cache
```

(Optional) If the playbook fails or if you want to clean up the environment to run a new deployment, you can optionally use the provided Cluster Removal Playbook to remove any previously installed Kubernetes and related plugins.

```
# ansible-playbook -i inventory.ini playbooks/redeploy_cleanup.yml
```

Deploy

Step 5 -Validate

Ansible Host

Validate

1. To interact with the Kubernetes CLI (kubectl), start by connecting to the target node in the cluster, which can be done using the following command:

```
# ssh root@<target ip>
```

2. Once connected, the status of the Kubernetes cluster can be checked.

```
# kubectl get nodes -o wide
# kubectl get pods -A
```

```
~kubectl get pods -A
NAMESPACE       NAME                                                READY   STATUS    RESTARTS      AGE
cadvisor        cadvisor-6qxft                                      2/2     Running   0             10m
cert-manager    cert-manager-6b7f85876b-4b76k                       1/1     Running   0             55m
cert-manager    cert-manager-cainjector-65bd7466f-vv4k4             1/1     Running   0             55m
cert-manager    cert-manager-webhook-64b99f54b7-449dm               1/1     Running   0             55m
cosign-system   policy-controller-policy-webhook-5c49dcf7d7-7b6lx   1/1     Running   0             51m
cosign-system   policy-controller-webhook-789f85ffcd-9zf5x          1/1     Running   0             51m
default         flexran-dockerimage-release                         2/2     Running   0             38m
kube-system     calico-kube-controllers-5c5b57ffb5-k5dlm            1/1     Running   0             55m
kube-system     calico-node-x6fcv                                   1/1     Running   0             56m
kube-system     container-registry-77bc8c848d-66brh                 2/2     Running   0             52m
kube-system     coredns-5c469774b8-frnbw                            1/1     Running   0             55m
kube-system     dns-autoscaler-5cc59c689b-kfzrg                     1/1     Running   0             55m
kube-system     kube-apiserver-as09-33-ac                           1/1     Running   0             51m
kube-system     kube-controller-manager-as09-33-ac                  1/1     Running   2             57m
kube-system     kube-multus-ds-amd64-cj4lr                          1/1     Running   0             55m
kube-system     kube-proxy-bpmrk                                    1/1     Running   0             56m
kube-system     kube-scheduler-as09-33-ac                           1/1     Running   1             57m
kube-system     kubernetes-dashboard-5bf8857fc6-scvfr               1/1     Running   0             55m
kube-system     kubernetes-metrics-scraper-75d7f948-jc6pg           1/1     Running   0             55m
kube-system     node-feature-discovery-master-6fb7d46677-5cwtg      1/1     Running   0             42m
kube-system     node-feature-discovery-worker-gvb82                 1/1     Running   1 (42m ago)   42m
kube-system     sriov-net-dp-kube-sriov-device-plugin-amd64-hcgvd   1/1     Running   0             39m
monitoring      elasticsearch-master-0                              1/1     Running   0             10m
monitoring      grafana-6c7fdf4d47-svq4g                            1/1     Running   0             11m
monitoring      jaeger-collector-d66fd7c59-rmr49                    1/1     Running   0             8m33s
monitoring      jaeger-query-647c486bdf-m8hwb                       1/1     Running   0             8m32s
monitoring      kibana-kibana-585c44546b-w84bp                      1/1     Running   0             5m40s
monitoring      kube-state-metrics-7685d5fccf-j8kf9                 3/3     Running   0             12m
monitoring      node-exporter-9zmdt                                 2/2     Running   0             12m
monitoring      opentelemetry-operator-64f5fc868d-f94s7             2/2     Running   0             7m58s
monitoring      otel-agent-cadvisor-collector-sbvsl                 1/1     Running   0             6m34s
monitoring      otel-agent-telegraf-collector-bb65c                 1/1     Running   0             6m33s
monitoring      otel-gateway-collector-69f6647f6f-dmr29             1/1     Running   0             6m36s
monitoring      prometheus-adapter-76f7bbb67b-7lkj6                 1/1     Running   0             11m
monitoring      prometheus-adapter-76f7bbb67b-c45n6                 1/1     Running   0             11m
monitoring      prometheus-k8s-0                                    2/2     Running   0             11m
monitoring      prometheus-operator-58b55c7c48-7f9vh                2/2     Running   0             11m
monitoring      telegraf-k54gj                                      2/2     Running   0             10m
observability   jaeger-operator-557d9d7f7f-t7tg9                    2/2     Running   0             8m49s
```

Deployment of FlexRAN™ software to be used in an end-to-end network is concluded here. The testing of stand-alone timer mode and xRAN is described below.

Target Server

5.1 FlexRAN™ software on Bare Metal validation steps

Below are the steps to validate FlexRAN tests on Bare Metal.

5.1.1 Testing FlexRAN™ software in Timer Mode on the target:

You need two terminal windows on the target for running the FlexRAN™ software L1 and L2 applications.

1. Run the FlexRAN™ software L1 app.

```
# cd /opt/cek/intel-flexran/
# source set_env_var.sh -d
# cd bin/nr5g/gnb/l1
# ./l1.sh -e
```

2. Open another terminal on target to run the Test MAC app.

```
# cd /opt/cek/intel-flexran/
# source set_env_var.sh -d
# cd bin/nr5g/gnb/testmac
# ./l2.sh --testfile=spr-sp-eec/sprsp_eec_mu0_10mhz_4x4_hton.cfg
```

5.1.2 Testing FlexRAN™ software in xRAN mode:

You need three terminal windows on the target for running the FlexRAN™ software in xRAN mode.

1. Run the FlexRAN™ software L1 app

```
# cd /opt/cek/intel-flexran/
# source set_env_var.sh -d
# cd bin/nr5g/gnb/l1/orancfg/sub3_mu0_10mhz_4x4/gnb
# ./l1.sh -oru
```

2. Open another terminal on target to run the Test MAC app

```
# cd /opt/cek/intel-flexran/
# source set_env_var.sh -d
# cd bin/nr5g/gnb/testmac
# ./l2.sh --
testfile=../l1/orancfg/sub3_mu0_10mhz_4x4/gnb/testmac_clxsp_mu0_10mhz_hton_oru.cfg
```

3. You can then start the oRU server with the command below

```
# cd /opt/cek/intel-flexran/bin/nr5g/gnb/l1/orancfg/sub3_mu0_10mhz_4x4/oru
# ./run_o_ru.sh
```

5.2 FlexRAN™ software in POD validation steps (supported on 4th Gen Intel® Xeon® Scalable processor with Intel® vRAN Boost from 23.07.1 release)

Below are the steps to validate FlexRAN tests on Containers.

You can find the FlexRAN™ POD name using the below command:

```
# kubectl get pods -A | grep flexran
```

You can check the status of the FlexRAN™ container applications running in the POD using the below command:

```
# kubectl describe pod <flexran_pod-name>
```

5.2.1 Testing FlexRAN™ software in Timer Mode in POD:

After the containers are created in the POD, the timer mode test is running already.

1. The status of the L1 app can be checked using the below command:

```
# kubectl logs -f <flexran-pod-name> -c <flexran-l1-app>
For example: kubectl logs -f flexran-dockerimage-release -c flexran-l1app
```

2. The status of the L2 TestMAC app can be checked using the below command:

```
# kubectl logs -f <flexran-pod-name> -c <flexran-testmac-app>
For example: kubectl logs -f flexran-dockerimage-release -c flexran-testmac
```

5.2.2 Testing FlexRAN™ software in xRAN mode in POD:

You need three terminal windows on the target for running the FlexRAN™ software in xRAN mode. The example xRAN mode test case used here is "sub3_mu0_10mhz_4x4".

1. (Terminal 1) Run the FlexRAN™ software L1 app:

```
# kubectl exec -it <flexran-vdu-pod-name> -- bash
# cd flexran/bin/nr5g/gnb/l1/orancfg/sub3_mu0_10mhz_4x4/gnb/
# ./l1.sh -oru
```

2. (Terminal 2) Open another terminal on target to run the Test MAC app:

```
# kubectl exec -it <flexran-vdu-pod-name> -- bash
# cd flexran/bin/nr5g/gnb/testmac
# ./l2.sh --
testfile=../l1/orancfg/sub3_mu0_10mhz_4x4/gnb/testmac_clxsp_mu0_10mhz_hton_oru.cfg
```

3. (Terminal 3) Open another terminal and then start the oRU server:

```
# kubectl exec -it <flexran-vru-pod-name> -- bash
# cd flexran/bin/nr5g/gnb/l1/orancfg/sub3_mu0_10mhz_4x4/oru/

Note: Update the file run_o_ru.sh for the port BDF corresponding with your server port oRU
BDF, example <--vf_addr_o_xu_a "0000:ca:11.0,0000:ca:11.1" I am running a few minutes late;
my previous meeting is running over. --vf_addr_o_xu_b "0000:ca:11.2,0000:ca:11.3">

# ./run_o_ru.sh
```

For more info on test cases, refer to the Intel FlexRAN™ Docker hub.

Reference Documentation

The *Network and Edge Container Bare Metal Reference System Architecture User Guide* provides information and a full set of installation instructions for a BMRA.

The *Network and Edge Reference System Architectures Portfolio User Manual* provides additional information for the Reference Systems including a complete list of reference documents.

The Intel FlexRAN™ Docker hub provides additional information on running the FlexRAN™ software in a POD on 4th Gen Intel® Xeon® Scalable processors with Intel® vRAN Boost.

Other collaterals, including technical guides and solution briefs that explain in detail the technologies enabled in the Reference System are available in the following location: Network & Edge Platform Experience Kits.

Document Revision History

REVISION	DATE	DESCRIPTION
001	July 2022	Initial release.
002	October 2022	Updated Intel® FlexRAN™ software version to 22.07.0 with xRAN test mode and RHEL 8.6RT kernel support.
003	December 2022	Support for 4th Gen Intel® Xeon® Scalable processor with Intel® vRAN Boost CPU and Intel® FlexRAN™ software version updated to 22.07.3.
004	March 2023	Updated Intel® FlexRAN™ software version to 22.11 and added support for running FlexRAN™ software in a POD on the 3rd Gen Intel® Xeon® Scalable processor server.
005	July 2023	Updated Intel® FlexRAN™ software version to 23.03.
006	September 2023	Added support for running FlexRAN™ v23.03 software in a POD for timer and xRAN mode on the 4th Gen Intel® Xeon® Scalable processor server with Intel® vRAN Boost.
007	October 2023	Updated FlexRAN™ software version to 23.07 on both POD and bare metal deployment on the 4th Gen Intel® Xeon® Scalable processor server with Intel® vRAN Boost.
008	January 2024	Updated FlexRAN™ software version to 23.11 for bare metal deployment on the 4th Gen Intel® Xeon® Scalable processor server with Intel® vRAN Boost.

0124/DN/WIT/PDF 737687-008US