installation guide on

WEDGE-100BF-32X

Work Package:	6
Task Item:	Task 1
Nature of Deliverable:	T (Technical report)
Dissemination Level:	Public

Authors: GÉANT RARE Team <rare@lists.geant.org>

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).

Abstract
This document presents the RARE/WEDGE-100BF-32xX installation guide used by the European Testbed currently deployed within the R&E community in the frame of GN4-3 project. The major distinction of this Testbed is that it encompasses P4 hardwares. It will soon be possible for research projects to validate at a European scale, P4 algorithms, but also possible for them to test network services based on RARE (aka GN4-3/WP6/T1) P4 router.

Table of Contents

Executive Summary

This document objective is to describe P4 switch WEDGE-100BF-32X in the context of WP6-T1/RARE project in 3 perspectives :

- A full WEDGE-100BF-32X MASTER TEMPLATE installation.
- A WEDGE-100BF-32X installation based on MASTER TEMPLATE
- Describe related Ansible playbook used to deploy a RARE-WEDGE-100BF-32X router

1 Introduction

WP6-T1/RARE team is currently deploying a European testbed that has the particularity to encompass P4 equipment. For now there is only one model of P4 hardware being rolled out which is the EDGECORE WEDGE-100-BF-32X that is powered by the TOFINO Network Processor Unit.

The partners involved in this testbed deployment are GÉANT, Jisc, KIFU, RENATER, SWITCH,UMU/REDIRIS.

The RARE team needs this testbed in order to test all the features set being developed by the project in the frame of the GN4-3 Work Package 6 Task 1.

This document is the RARE-WEDGE-100BF-32X installation guide following the 3 perspectives listed above.

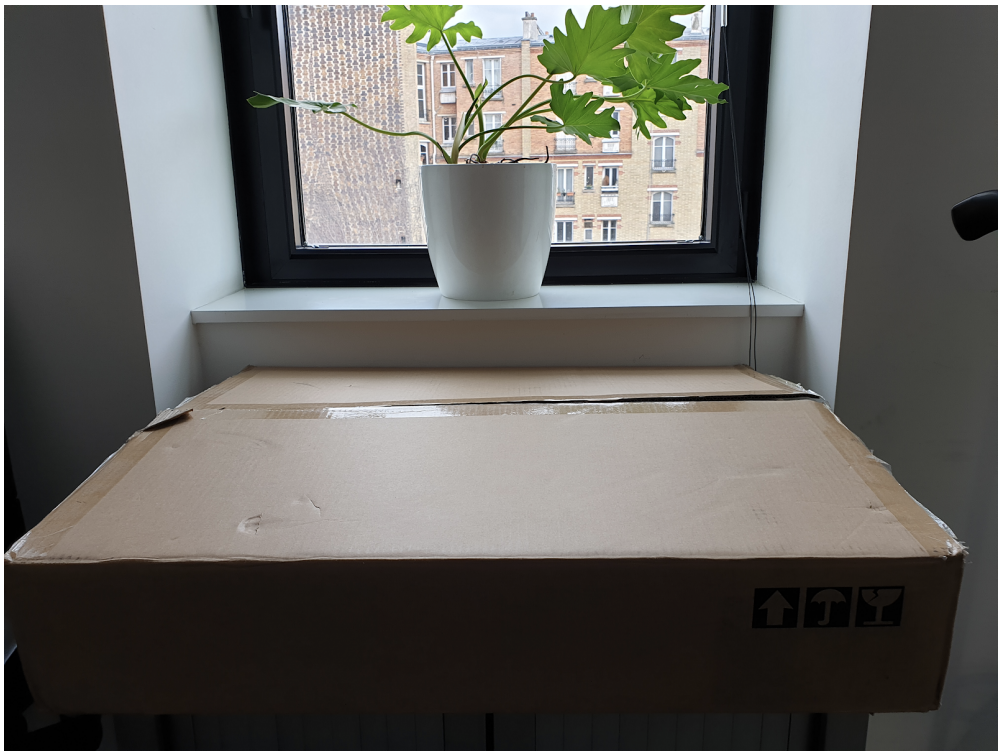
fyi Note that this procedure also applies to WEDGE-100BF-64X that has twice the number of ports. However, it does not have the two on-board Ethernet ports. Therefore the CPU port is only available via PCIe through the use of INTEL bf_kpkt port.

2 RARE-WEDGE Master template installation

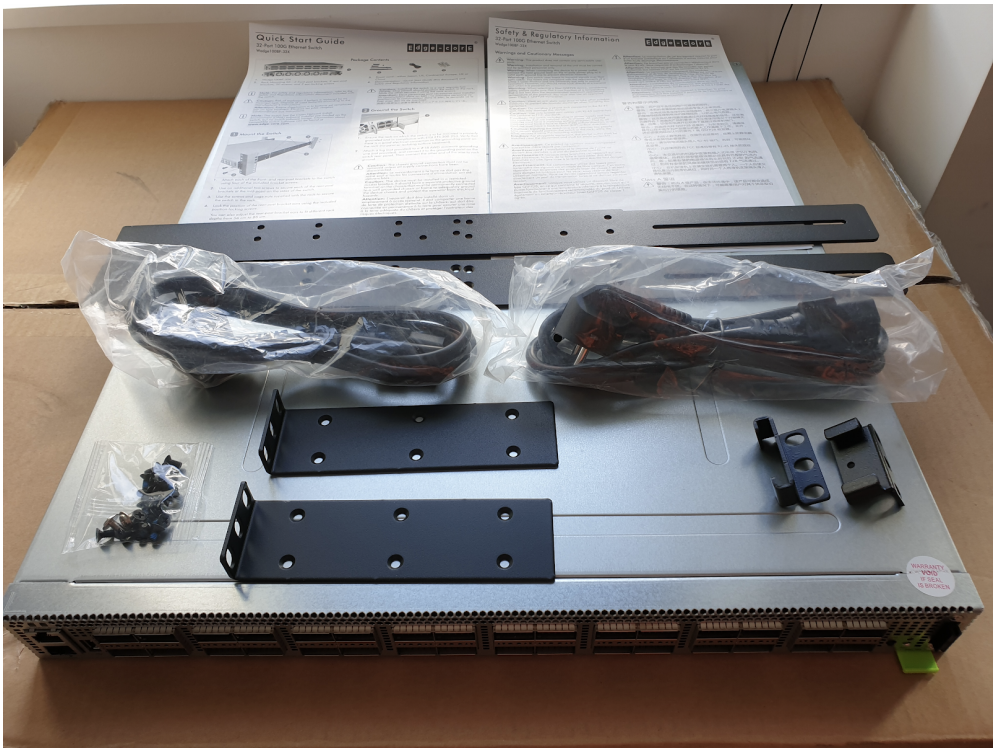
2.1 Unboxing

2.1.1 Parcels

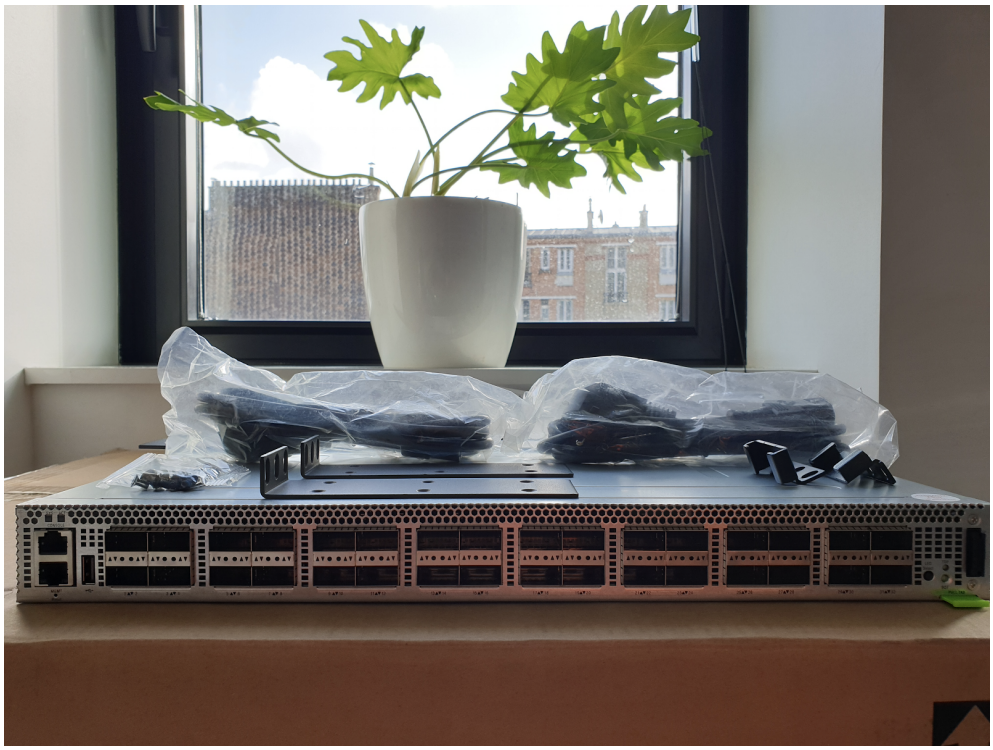
The parcel size is adapted to 1RU: L x l x H : 75 x 60 x 15 (cm)



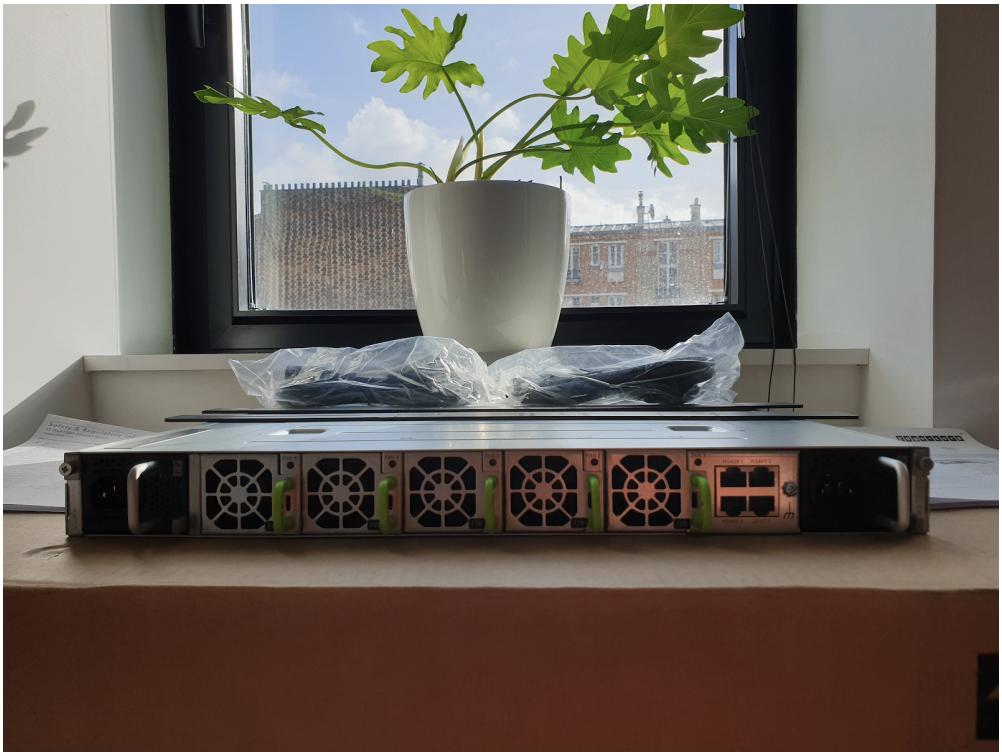
2.1.2 What's in the box



2.1.3 Front panel



2.1.4 Back panel



2.2 Initial OpenBMC and Ethernet management ports

The WEDGE has 2 main components:

- First component: BMC
- Second component: MAIN CPU board

Each of one of them has their own out of band management ports.

During first boot:

- Connect to BMC via console port and configure BMC OOBM IP address
- Then connect to MAIN CPU from BMC using `sol.sh` script and configure MAIN CPU OOBM IP address

At this stage, you can connect the WEDGE to your LAN via the management ethernet and you'll get access to the MAIN CPU running ONIE.

⚠ BMC and MAIN CPU are sharing an internal hub and have respectively ip address 192.168.0.1 and 192.168.0.2 therefore you can bounce to MAIN CPU from BMC via sol.sh script but it also means that you cannot assign 192.168.0.0/24. Additionally, ONIE boot loader has a default password less root account. Therefore it is recommended to connect the switch in a protected environment.

2.2.2 OpenBMC configuration via serial console port

The OpenBMC is accessible via the console port. You'll have to use a roll-over cable for that purpose.

Console settings:

```
9600 baud, 8 data bits, no parity and 1 stop bit.
```

The default BMC account is:

```
user: root
password: 0penBmc
```

⚠ OpenBMC is installed in nvram. This means that all modifications are not persistent after a reboot.

2.2.3 Accessing the ONIE bootloader via serial console port

When the device is powered on for the first time, the serial console (settings 9600/8N1) provides access to the OpenBMC prompt. The default credentials are username `root` with password `0penBmc`. After logging in, execute the following command at the BMC prompt

```
root@bmc:~# wedge_power.sh reset; sol.sh
```

This will reset the host CPU and connect the user to the console of the system (not to be confused with the console of the BMC). As a result, the user will be presented with the ONIE prompt

```
ONIE:~ #
```

As specified, the MAIN CPU initially has only ONIE boot loader installed. By default, ONIE is configured to spawn (extensively) boot requests in order to retrieve its firmware. It is therefore recommended to stop this auto-discovery process by issuing:

```
onie-stop
```

fyi ONIE installation process has now stopped. We will resume this auto-discovery process once ONIE ONL image recommended by INTEL is built.

2.3 ONIE firmware factory reset

In order to have a coherent deployment process, it is recommended to perform a ONIE firmware factory reset. As you recall ONIE by default is accessible via a password less `root` account. This means that for any reasons if the switch has been exposed to an unprotected environment the ONIE boot loader could have been compromised.


The following section describes ONIE firmware update. The section below is available for convenience, avoiding us from switching between various documents.

2.3.1 Getting ONIE firmware

Grab the ONIE firmware:

- from RENATER orebuilt ONIE rescue IMAGE
- use RENATER prebuilt ONIE self-upgrade image
- or create your own ONIE image using ONIE GitHub instructions.

2.3.2 Create the bootable ONIE image

Convert the ISO into a bootable image and load it onto a USB device.

On MACOS:

```
hdutil convert -format UDRW -o \
onie-recovery-x86_64-accton_wedge100bf_32x-r0.img \
onie-recovery-x86_64-accton_wedge100bf_32x-r0.iso
```

Find the partition on which your USB drive is mounted. In this example, `/dev/disk2` corresponds to the USB device. Use the diskutil list command as shown here:

```
MBP-de-Frederic:~ loui$ diskutil list
/dev/disk0 (internal):
   #:                       TYPE NAME                    SIZE       IDENTIFIER
   0:      GUID_partition_scheme                        500.3 GB   disk0
   1:                        EFI EFI                     314.6 MB   disk0s1
   2:                 Apple_APFS Container disk1         500.0 GB   disk0s2

…

disk5
```

```
   1:                        0xEF                        213.0 KB    disk5s1
```

In this precise case USB is `/dev/disk5`, Unmount the USB using:

```
sudo diskutil unmount /dev/disk5s1
```

Burn the image on the USB drive using the dd command:

```
sudo dd \
     if=./onie-recovery-x86_64-accton-wedge100bf_32x-r0.img.dmg \
     of=/dev/disk5 bs=1m
```

2.3.3 ONIE firmware update

- Connect to BMC
- Insert USB into the USB port on the WEDGE front panel
- At the the BMC prompt enter the MAIN CPU and type:

```
wedge_power.sh reset; sol.sh
Power reset micro server … Done
```

- During the reboot process from the MAIN CPU, press eth <ESC? or <DEL> key to enter the Aptio Setup Utility. In the Boot menu, set Boot mode select to LEGACY and set Boot Option #1 to USB Device.
- Press f4 to save and exit
- When you reach the ONIE boot screen under GRUB select ONIE EMBED MODE, which installs ONIE
- When the ONIE installation is done, remove the USB
- When the ONIE prompt appears (user:root no password) type:

```
onie-stop
```

- verify that the ONIE installation was successful by checking the ONIE log file:

```
tail /var/log/onie.log
```

2.4 INTEL recommended ONL installation

All necessary information is coming from OpenNetworkLinux Github and on Intel® P4 Studio SDE 9.3.0 release ONL patch and install script zip INTEL Connectivity Forum portal. In order to build ONL you'd need to have a machine with 40G of disk free space, docker, binfmt-support and at least 4G of ram and 4G of swap.

ONL compilation can be performed on a different host that the WEDGE. Just make sure the machine you are using for this step is compliant to the prerequisites listed above.

2.4.1 Getting the right ONL commit

ONL used as base Operating System in this deployment.

It is **MANDATORY** to use the version recommended by INTEL. You can grab this information from the SDE release notes downloaded from INTEL CONNECTIVITY RESEARCH PROGRAM. As we are installing SDE 9.3.0 we are thus using ONL commit 1537d8334d5fc9364f1ce6a44e26eb247e42f2e6.

Clone ONL from Github:

```
git clone https://github.com/opencomputeproject/OpenNetworkLinux.git
git checkout 1537d8334d5fc9364f1ce6a44e26eb247e42f2e6
```

2.4.2 WEDGE ONL compilation

Dowload ONL patch from INTEL CONNECTIVITY FORUM resource page (needs private access subject to NDA).

```
cd OpenNetworkLinux
git apply onl.patch
# SDE 9.3.0 uses ONL9 which is based on Debian 9
export VERSION=9
# make ONL adopt systemd
echo 'export INIT=systemd' >> setup.env
# vi edit ./OpenNetworkLinux/builds/any/rootfs/stretch/standard/standard.yml
# add user p4, [login:p4,password:p4]
make docker
…
<perfect time to grab a coffee>

docker/tools/onlbuilder -9
apt-cacher-ng
source setup.env
make amd64
```

Obviously ONL compilation is not necessary each time a WEDGE is deployed. Compilation is only needed if you need to install a INTEL SDE that requires a specific commit.

2.4.3 WEDGE ONL installation

After ONL compilation all the artefacts will be located in:

- RELEASE/stretch/amd64/
  - ONIE installer:
  - **ONL-HEAD_ONL-OS9_2021-01-04.1043-1537d83_AMD64_INSTALLED_INSTALLER**
- REPO/stretch/packages/binary-amd64/
  - kernel header debian package:
    onl-kernel-4.14-lts-x86-64-all_1.0.0_amd64.deb

In our setup DHCP is used to bootstrap WEDGE ONIE installation :

- static IP address allocation based on MAC address
- with BOOTP/DHCP default-url
  http://<your_web_server>/ONL-HEAD_ONL-OS9_2021-01-04.1043-1537d83_AMD64_INSTALLED_INSTALLER

```
# DHCP example

host HOST_237 {
  hardware ethernet 00:90:fb:65:d6:3c;
  fixed-address 193.49.159.237;
  option default-url =
"http://194.57.4.109/ONL-HEAD_ONL-OS9_2020-02-18.0846-3127f40_AMD64_INSTALLED_INSTALLER";
}
```

Reactivate ONIE autodiscovery bootstrap process:

```
onie-discovey-start
…
<WEDGE should kickstart ONIE installation using DHCP default_url>
…
<perfect time to grab another coffee>
```

⚠ ONIE discovery process is convenient, however make sure connectivity between the WEDGE ethernet management port and the HTTP server. Otherwise, the switch would start to spawn (tons of) spurious requests to the whole Internet using a predefined list of URL. If public IP is used and that outgoing traffic toward the Internet is possible from the WEDGE management port it is therefore recommended to install the firmware manually.

Or install firmware manually: (194.57.4.109 is the http server)

```
onie-nos-install \
http://194.57.4.109/ONL-HEAD_ONL-OS9_2021-01-04.1043-1537d83_AMD64_INSTALLED_INSTALLER
…
<perfect time to grab another coffee>
```

2.4.4 WEDGE ONL final configuration

The default ONL account is:

```
user: root
```

```
password: onl
```

⚠ Once the MAIN CPU root account is changed it will be applicable also to the console access.

Check kernel version:

```
root@localhost:~# uname -r
4.14.151-OpenNetworkLinux
root@localhost:~#
```

Based on the command output above, Install kernel header package from:

REPO/stretch/packages/binary-amd64/

```
dpkg -i onl-kernel-4.14-lts-x86-64-all_1.0.0_amd64.deb
```

Link the kernel header:

```
ln -s /usr/share/onl/packages/amd64/onl-kernel-4.14-lts-x86-64-all/mbuilds/ \
      /lib/modules/4.14.151-OpenNetworkLinux/build
```

Disable daily update:

```
# check systemd is enabled
ps --no-headers -o comm 1

# disable systemd timer
systemctl stop apt-daily.timer
systemctl disable apt-daily.timer
systemctl stop apt-daily.service
systemctl disable apt-daily.service

systemctl stop apt-daily-upgrade.timer
systemctl disable apt-daily-upgrade.timer
systemctl stop apt-daily-upgrade.service
systemctl disable apt-daily-upgrade.service
systemctl daemon-reload
systemctl reset-failed
```

2.5 INTEL SDE installation

2.5.1 Getting the latest SDE

- Grab the INTEL latest SDE from INTEL CONNECTIVITY FORUM. As of this document writing the latest SDE is version 9.3.0.
- As we are deploying SDE on the WEDGE we need to also download WEDGE 100BF 32X Board Support Package (BSP) from INTEL CONNECTIVITY FORUM. We download then BSP 9.3.0 version corresponding to SDE 9.3.0.

2.5.2 INTEL ONL SDE dependencies installation

Previous SDE installation prior to SDE 9.3.0 had a dependencies.tar.gz file. Since SDE 9.3.0 this has to be installed by reading the instructions in **Intel® P4 Studio SDE 9.3.0 release ONL patch and install script** zip file.

In this tarball, there is also a nested `dependencies.tar.gz` tarball containing all the necessary dependency packages needed by the SDE 9.3.0.

```
# as root (obviously)

tar xzf dependencies.tar.gz
cd dependencies
./install.sh
…
<perfect time to grab another coffee>
```

2.5.3 INTEL SDE installation

We assume that you install the SDE using `root` account and you are at `/root` directory.

untar BSP 9.3.0 tarball

```
tar xzf bf-reference-bsp-9.3.0.tgz -C /opt
```

untar SDE 9.3.0 tarball

```
tar xzf bf-sde-9.3.0.tgz -C /opt
```

set SDE installation environment in `~/.profile`

```
export SDE=/opt/bf-sde-9.3.0
```

```
export SDE_INSTALL=$SDE/install
export PATH=$SDE_INSTALL/bin:$PATH
source ~/.profile
```

install SDE with BSP using p416_examples profile

```
cd $SDE/p4studio_build/
./p4studio_build.py --skip-dependencies \
                    --use-profile p416_examples_profile \
                    --bsp-path /root/bf-reference-bsp-9.3.0
```

⚠ We assume you installed the dependencies using ONL tarball so no need to download and reinstall dependencies again.

2.6 INTEL SDE final settings

2.6.1 INTEL additional tools

If you had not done it preciously, set SDE installation environment in `~/.profile`

```
export SDE=/opt/bf-sde-9.3.0
export SDE_INSTALL=$SDE/install
export PATH=$SDE_INSTALL/bin:$PATH
source ~/.profile
```

Not sure where is it now, but you had the possibility to download useful tools that were essentially wrapper around SDE tools: (p4_builb.sh)

```
tar xzf ba-102-tools-2020-02-10.tgz -C /opt
```

2.6.2 Enable Ethernet/PCIe CPU Port

There are 2 possibilities in order to use P4 CPU PORT:

- Ethernet CPU PORT (port **64**, 65, **66**, 67)
- PCIe CPU PORT (port **192**)

By default, as per our experience among 8 WEDGE-BF100-32X, Ethernet CPU ports are not enabled. In order to enable them you have to reboot the switch from MAIN CPU and get access to the BIOS and activate them from there following this procedure described in the INTEL CONNECTIVITY FORUM.

⚠ keep in mind that is applicable only for WEDGE-100BF-32X, for WEDGE-100BF-65X Ethernet CPU PORTs are not available.

Once enabled from the BIOS, you should be able to activate them from the MAIN CPU access

```
# P4 PORT 66
ip link set enp4s0f0 mtu 8192
ip link set enp4s0f0 up
# P4 PORT 64
ip link set enp4s0f1 mtu 8192
ip link set enp4s0f1 up
```

In order to enable Ethernet CPU_PORT you need to first load bf_drv driver

```
$SDE_INSTALL/bin/bf_kdrv_mod_load $SDE_INSTALL
```

and for PCIe CPU_PORT you need to first load bf_kpkt driver:

```
$SDE_INSTALL/bin/bf_kpkt_mod_load $SDE_INSTALL
```

You should be then be able PCIe CPU PORT 192:

```
ip link set enp6s0 mtu 8192
ip link set enp6s0 up
```

⚠ In ONL 8, the device was called `bf_pci0` , in ONL 9, the device has a different name. In our case it is called either `enp6s0`, or `enp5s0` when Ethernet ports are not activated via the BIOS.

⚠ bf_kdrv and bf_kpkt are mutually exclusive you cannot load both of them at the same time. If you plan to use Ethernet CPU_PORT you can just use bf_kdrv, if you plan to use PCIe CPU_PORT you can use bf_kpkt, this will make “PCIe based” network interface appear. From different platform/OS point of view this interface can be named `bf_pci0`, `enp6s0`, `enp5s0` or `ens1` (on ubuntu STORDIS BF2556X-1T P4 switch)

2.7 RARE installation

2.7.1 OpenJDK JAVA installation

As FreeRouter is written in JAVA, JRE needs to be installed. JDK is necessary if you need to recompile FreeRouter from source. Generally, you won’t need to recompile it. Therefore Java Runtime Environment is enough. There are various ways to install JRE:

- manual installation
- using Operating system stock JRE

Manual installation

Manual installation is required if you really need a specific version of JAVA environment recommended by your organization or if you aim for an installation without INTERNET connection. In this example we are using JDK 13.0.2, but you can use any JAVA version starting from JAVA 8.

Download JAVA JDK from JDK home page. In our deployment we use JDK 13.0.2 GA release.
untar the openjdk-13.0.2_linux-x64_bin.tar.gz into `/root` directory:

```
tar xzf openjdk-13.0.2_linux-x64_bin.tar.gz
```

Set JAVA environment variable in ~/.bashrc:

```
export JAVA_HOME=/root/jdk-13.0.2
export PATH=$JAVA_HOME/bin:$PATH
```

Operating system JRE (for debian based system such as ONL)

This is the recommended way to install JAVA, as it is ubiquitous and simple.

```
apt-get update
apt-get upgrade
apt-get install default-jre-headless --no-install-recommends
```

2.7.2 Get RARE software release

Since 2020 september, RARE is proposing 3 fully dataplanes that are now **OpenSource**. 2 of them are P4 dataplane and 1 is based on DPDK, we hope soon to add more dataplane support.

Clone RARE software from GÉANT Bitbucket. (eduGain and special authorization is not needed any more ! :-))

```
cd /root
git clone https://bitbucket.software.geant.org/scm/rare/rare.git
```

2.8 RARE final installation & verification

2.8.1 freeRouter control plane manual installation

```
mkdir -p ~/freeRouter/bin ~/freeRouter/lib ~/freeRouter/etc
~/freeRouter/log
cd ~/freeRouter/lib
wget http://freerouter.nop.hu/rtr.jar
```

```
tree freeRouter
freeRouter
├── bin   # binary files
├── etc   # configuration files
├── lib   # library files
└── log   # log files
```

2.8.2 freeRouter control plane configuration

freeRouter uses 2 configuration files in order to run, these configuration files for one freeRouter control plane can be in `~/freeRouter/etc`.

- freeRouter hardware configuration file: `freerouter-hw.txt`

```
int eth0 eth 0000.1111.00fb 127.0.0.1 22710 127.0.0.1 22709
tcp2vrf 2323 v1 23
tcp2vrf 9080 v1 9080
```

⚠ Basically:
- v1 is the default vrf. The name is arbitrary.
- Telnet access seen from port 23 network namespace is accessible via port 2323 from UNIX host
- Same apply for port 9080 which is freeRouter API messages connection toward Dataplane
- If you need SSH you'll need to add `tcp2vrf 2002 v1 22`

- freeRouter software configuration file: `freerouter-sw.txt`

```
hostname tna-freerouter
buggy
!
!
vrf definition v1
 exit
!
interface ethernet0
 description freerouter@P4_CPU_PORT[enp6s0]
 no shutdown
 no log-link-change
 exit
!
interface sdn1
 description freerouter@sdn1[enp0s3]
 mtu 9000
 macaddr 0072.3e18.1b6f
 vrf forwarding v1
 ipv4 address 192.168.0.131 255.255.255.0
 ipv6 address 2a01:e0a:159:2850::666 ffff:ffff:ffff:ffff::
 ipv6 enable
 no shutdown
 no log-link-change
 exit
!
!
!
!
```

```
!
!
!
!
!
!
!
!
!
!
server telnet tel
 security protocol telnet
 no exec authorization
 no login authentication
 vrf v1
 exit
!
server p4lang p4
 export-vrf v1 1
 export-port sdn1 0 10
 interconnect ethernet0
 vrf v1
 exit
!
!
end
```

Explanation:

- Name of the router is:

```
hostname tna-freerouter
```

- buggy: Enable hidden command in the CLI. Some of them can be intrusive or induce more CPU usage. In normal operation you don't need it.

```
buggy
```

- In freeRouter world, there is no default, global implicit VRF. Everything~~Erything~~ is explicitly declared inside a VRF.

```
!
vrf definition v1
 exit
!
```

- freeRouter connection to Dataplane via API Control message.

```
!
interface ethernet0
 description freerouter@P4_CPU_PORT[enp6s0]
 no shutdown
 no log-link-change
 exit
```

```
!
```

- freeRouter basic telnet access server (notice that it is inside VRF `v1`).

```
!
server telnet tel
 security protocol telnet
 no exec authorization
 no login authentication
 vrf v1
 exit
!
```

- freeRouter dataplane secret sauce: `p4lang server` stanza and `sdn<x>` interface

```
!
interface sdn1
 description freerouter@sdn1[enp0s3]
 mtu 9000
 macaddr 0072.3e18.1b6f
 vrf forwarding v1
 ipv4 address 192.168.0.131 255.255.255.0
 ipv6 address 2a01:e0a:159:2850::666 ffff:ffff:ffff:ffff::
 ipv6 enable
 no shutdown
 no log-link-change
 exit
!
!
server p4lang p4
 export-vrf v1 1
 export-port sdn1 132 10
 interconnect ethernet0
 vrf v1
 exit
!
```

⚠ On WEDGEBF10032X, TOFINO have internal interface identifier tagged with the **D_P** field.

From a `bf_shell` session:

```
bfshell> ucli
Cannot read termcap database;
using dumb terminal settings.
bf-sde> pm
bf-sde.pm> sho
bf-sde.pm> show -a 1/0
-----+----+---+----+-------+----+---+---+---+--------+---------------+---------------+-
PORT |MAC |D_P|P/PT|SPEED  |FEC |RDY|ADM|OPR|LPBK    |FRAMES RX      |FRAMES TX      |E
-----+----+---+----+-------+----+---+---+---+--------+---------------+---------------+-
1/0  |23/0|132|3/ 4|10G    |NONE|YES|ENB|UP |  NONE  |         263577|         264667|
bf-sde.pm>
```

So port TOFINO port 1/0, which is port number 1 on the WEDGEBF10032X front panel has `D_P` id 132. Please keep in mind that before using a port on the WEDGE100BF32X you need to plug QSFP28 on the switch front panel with optical pigtail SFP28 form factor. In the example above, we put a DAC100G --> 10G.

Setting up interface on TOFINO is not in the scope of this document. You'll find the relevant information in the INTEL WEDGE configuration guide.

In this example in `server p4`:

- we activate message toward the dataplane for VRF v1
- we consider interface `sdn1` and map it to `TOFINO 1/0` port that has `D_P` identifier `132`.
- `10` parameter after `132` stands for 10G
- `ethernet0` is the interconnection port with the dataplane. (`packet_in`/`packet_out`)
- `server p4lang` operate inside vrf `v1`
    - this latter is important, only if we need to expose server lang network context into VRF `v1`.
    - For security purpose we can juts isolate `p4lang server` and create a dedicated solely in this occasion. (i.e VRF `my_p4_server+_lang_vrf` and not VRF `v1`)

2.8.3 RARE bf_router.p4 compilation

Depending on RARE profile and CPU PORT there are 2 variants:

- RARE with MPLS profile and Ethernet CPU PORT: port 64

```
/root/tools/p4_build.sh \
            -I ~/rare/100-WEDGE-100BF-32X/p4src \
            -DHAVE_COPP \
            -DHAVE_INACL \
            -DHAVE_OUTACL \
            -DHAVE_MPLS \
            -DHAVE_BRIDGE \
            ~/rare/100-WEDGE-100BF-32X/p4src/bf_router.p4
```

- RARE with MPLS profile and PCIe CPU PORT: port 192

```
/root/tools/p4_build.sh \
            -I ~/rare/100-WEDGE-100BF-32X/p4src \
```

```
            -DHAVE_COPP \
            -DHAVE_INACL \
            -DHAVE_OUTACL \
            -DHAVE_MPLS \
            -DHAVE_BRIDGE \
            -D_WEDGE100BF32X_ \
            ~/rare/100-WEDGE-100BF-32X/p4src/bf_router.p4
```

- RARE with SRv6 profile and Ethernet CPU PORT: port 64

```
/root/tools/p4_build.sh \
            -I ~/rare/100-WEDGE-100BF-32X/p4src \
            -DHAVE_COPP \
            -DHAVE_INACL \
            -DHAVE_OUTACL \
            -DHAVE_SRV6 \
            -DHAVE_BRIDGE \
            -DHAVE_NAT \
            -DHAVE_PBR
            ~/rare/100-WEDGE-100BF-32X/p4src/bf_router.p4
```

- RARE with SRv6 profile and PCIe CPU PORT: port 192

```
/root/tools/p4_build.sh \
            -I ~/rare/100-WEDGE-100BF-32X/p4src \
            -DHAVE_COPP \
            -DHAVE_INACL \
            -DHAVE_OUTACL \
            -DHAVE_SRV6 \
            -DHAVE_BRIDGE \
            -DHAVE_NAT \
            -DHAVE_PBR \
            -D_WEDGE100BF32X_ \
            ~/rare/100-WEDGE-100BF-32X/p4src/bf_router.p4
```

2.8.4 Start INTEL bf_switchd

```
cd $SDE
./run_switchd -p bf_router
```

2.8.5 Start RARE-FreeRTR control-plane

You can use the following manual script in order to start FreeRouter: (to be done once after each reboot)

```
echo 1 > /proc/sys/net/ipv6/conf/all/disable_ipv6
echo 1 > /proc/sys/net/ipv6/conf/default/disable_ipv6
```

```
# set CPU_PORT UP, promiscuous mode and jumbo MTU
ip link set enp6s0  up
ifconfig enp6s0 promisc
ip link set dev enp6s0 up mtu 10240

# Disable TCP offload
export TOE_OPTIONS="rx tx sg tso ufo gso gro lro rxvlan txvlan rxhash"

for TOE_OPTION in $TOE_OPTIONS; do
    /sbin/ethtool --offload enp6s0 "$TOE_OPTION" off &> /dev/null
done
```

Launch freeRouter explicitly:

```
java -jar lib/rtr.jar routersc etc/freerouter-hw.txt \
                               etc/freerouter-sw.txt
```

2.8.6 Start freeRouter control-plane using bitbucket environment

You can use freeRouter installation from Bitbucket RARE repository. In order to get the latest freeRouter version, use the following script:

```
cd ~/rare
./update-jar.sh
```

This will update freeRouter latest version into:

```
~/rare/REFERENCE-LAB-MODEL/0000-topology-A/bin/
```

Depending on the CPU PORT the special file needs to be adjusted in RARE-FreeRTR startup script.

```
cd ~/rare/100-WEDGE-100BF-32X/0001-vpn-over-bgp-isis-sr-operation
vi bin/setup_core1.sh
# if Ethernet is used
FREERTR_ETH0="enp4s0f1"
#if PCIe is used
FREERTR_ETH0="enp6s0"
```

```
cd ~/rare/100-WEDGE-100BF-32X/0001-vpn-over-bgp-isis-sr-operation
# start RARE-FreeRTR
make
# if you want to stop RARE-FreeRTR
make clean
```

⚠ Please note that in this case freeRouter configuration is in:

```
~/rare/WEDGE100BF32X/0001-vpn-over-bgp-isis-sr-operation/run/core1
ls -l
total 6016
-rw-r--r-- 1 p4 p4     343 Sep 16 17:37 core1-hw.txt
-rw-r--r-- 1 p4 p4    8335 Sep 23 18:28 core1-sw.txt
```

2.8.7 Start RARE interface between freeRouter and P4 dataplane

In this case we are launching RARE interface between FreeRouter control plane and TOFINO P4 dataplane. Default profile is MPLS:

```
cd ~/rare/100-WEDGE-100BF-32X/bfrt_python
./bf_forwarder.py
```

3 RARE WEDGE installation from template

This installation is meant to install the WEDGE:

- in a situation without Internet connectivity
- in a situation where you a multiple subsequent installation

In “RARE-WEDGE Master template installation” section, we presented a full installation procedure that is usually done once. After the first time installation all the artefacts needed resulting from previous steps don’t not need to be recompiled again.

Therefore we will avoid recompilation of:

- ONL 9 commit `1537d8334d5fc9364f1ce6a44e26eb247e42f2e6`
- SDE-9.3.0

Finally, the ONIE firmware update described in the previous section is only possible on site.

fyi Essentially, this procedure is the same as the previous section. Instead we re-used all the artifacts above just by copying them onto the same switch.

3.1 ONL 9 artefacts

3.1.1 ONL 9 ONIE installer

After ONL compilation all the ONIE install will be located in:

- RELEASE/stretch/amd64/
    - ONIE installer:
    - **ONL-HEAD_ONL-OS9_2021-01-04.1043-1537d83_AMD64_INSTALLED_INSTALLER**

fyi ONL image is the version recommended by INTEL. Technically each SDE release is associated to an ONL commit version. If don't want to rebuild ONL release and that you are willing to use external image, you can re-use own ONL image required INTEL.

3.1.2 ONL 9 kernel header

After ONL compilation all the kernel header will be located in:

- REPO/stretch/packages/binary-amd64/
    - kernel header debian package:
      onl-kernel-4.14-lts-x86-64-all_1.0.0_amd64.deb

3.2 INTEL SDE artefacts

3.2.1 ONL SDE dependencies installation

In the previous section "Getting the right ONL commit", we should have downloaded ONL dependencies file from ONL Support Files Download. (There was onl.patch in the tarball)

In this tarball, there is also a nested `dependencies.tar.gz` tarball containing all the necessary dependency packages needed by the SDE 9.3.0.

```
# as root (obviously)

tar xzf dependencies.tar.gz
cd dependencies
./install.sh
…
<perfect time to grab another coffee>
```

3.2.2 Unzip SDE 9.3.0

```
tar xzf bf-sde-9.3.0.tgz
```

3.2.3 INTEL SDE 9.3.0 install folder

After SDE compilation all the main needed artefacts are located in:

```
$SDE_INSTALL
```

Therefore copy `install` folder resulting from the previous compilation into the `$SDE` folder that you untared in the previous section.

3.2.4 INTEL SDE 9.3.0 pkgsrc folder

INTELT P4 helper tools supplied also need some files in `pkgsrc` folder that must be copy into `$SDE` folder.

```
$SDE/pkgsrc/p4-build/
$SDE/pkgsrc/p4-examples/tofino
```

fyi A RARE-SDE tarball has been created. All the artefacts listed above are included in it.

4 Conclusion

This document presented the RARE WEDGE-100BF-32X installation. It presented 3 ways to stage a fully functional switch. While the full installation master template is useful for Network engineers, the second method is focused on providing a “unplugged” installation.

The last method’s objective is to provide a “zero touch” deployment procedure of the “unplugged” installation.

References

Wedge 100B ONIE Bootloader Download
ONL Support Files Download
WEDGE 100B user guide
OpenNetworkLinux GitHub
Port 64 on Wedge 100BF-32x
Send packet to/from CPU

Glossary

DPP	Data Plane Programming
FPGA	Field Programmable Gate Array
IX	Internet eXchange point
NOS	Network Operating System
P4	Programming Protocol-Independent Packet Processors - programming language
TCO	Total Cost Ownership

5 Annexes

5.1 SDE 9.3.0 Caveat

5.1.1 BRI issue: bfrtTable.py

There is an issue with BRI for certain programs. This is described here. Download bfrtTable.py.zip, unzip it and copy it into /opt/bf-sde-9.3.0/install/lib/python3.4/bfrtTable.py.

5.1.2 SDE 9.3.0 post installation

```
cd $SDE_INSTALL/lib/python2.7/site-packages/tofino/google/
ln -s /usr/local/lib/python2.7/dist-packages/protobuf-3.6.1- py2.7.egg/google/protobuf/
protobuf
```

5.1.3 SDE 9.3.0 dependency fetch

Dependencies can be fetched using the following commands.

fyi curl / libcurl deb are mentioned are security downloads and hence are mentioned (**up12**). The version from apt (**up10**) won't match then. When we downloaded these 2 libraries, the version downloaded were **up13**. You'll have to modify the version download in **install.sh** helper script provided by INTEL.

```
egrep "dpkg -i" ./install.sh  | awk -F '[_ ]+' '/dpkg -i / {print $3}' | xargs apt-get
download

wget http://deb.debian.org/debian/pool/main/z/zlib/zlib_1.2.8.dfsg.orig.tar.gz
wget https://mirror.ibcp.fr/pub/apache/thrift/0.13.0/thrift-0.13.0.tar.gz
wget http://deb.debian.org/debian/pool/main/libn/libnl3/libnl3_3.2.27.orig.tar.gz
wget https://dl.bintray.com/boostorg/release/1.67.0/source/boost_1_67_0.tar.gz
wget http://deb.debian.org/debian/pool/main/g/graphviz/graphviz_2.40.1.orig.tar.gz
wget
https://github.com/protocolbuffers/protobuf/releases/download/v3.6.1/protobuf-python-3.6.1.tar
.gz
wget
https://files.pythonhosted.org/packages/9d/0d/197f4a023269b5018054c5c2def0dd33b8dee04cdab6c606
54182d2b0fbb/ctypesgen-1.0.2-py2.py3-none-any.whl
wget
https://files.pythonhosted.org/packages/f9/d3/955738b20d3832dfa3cd3d9b07e29a8162edb480bf988332
f5e6e48ca444/setuptools-44.0.0-py2.py3-none-any.whl
wget
https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/pycrc16/crc16-
0.1.1.tar.gz
wget
https://files.pythonhosted.org/packages/c1/63/47d4bc4e4bfae29e00ff9256b52dfbf945b409804cfadf95
714c113b8efb/jsl-0.2.4.tar.gz
wget
https://files.pythonhosted.org/packages/fb/af/ce7b0fe063ee0142786ee53ad6197979491ce0785567b6d8
be751d2069e8/coverage-4.5.2.tar.gz
wget
https://files.pythonhosted.org/packages/02/24/f73045afb049295b34ac55aaf6ea1592604cda3749632a22
e563e66604a3/Cython-0.29.3.tar.gz
wget
https://files.pythonhosted.org/packages/1f/9e/7b2ff7e965fc654592269f2906ade1c7d705f1bf25b7d469
fa153f7d19eb/futures-3.2.0.tar.gz
wget
https://files.pythonhosted.org/packages/d8/55/221a530d66bf78e72996453d1e2dedef526063546e131d70
bed548d80588/wheel-0.32.3.tar.gz
wget
https://files.pythonhosted.org/packages/dd/bf/4138e7bfb757de47d1f4b6994648ec67a51efe58fa907c1e
11e350cddfca/six-1.12.0.tar.gz
```

```
wget
https://files.pythonhosted.org/packages/58/b9/171dbb07e18c6346090a37f03c7e74410a1a56123f847efe
d59af260a298/jsonschema-2.6.0.tar.gz
wget
https://files.pythonhosted.org/packages/e3/24/c35fb1c1c315fc0fffe61ea00d3f88e85469004713dab488
dee4f35b0aff/simplejson-3.16.0.tar.gz
wget https://pypi.python.org/packages/source/T/Tenjin/Tenjin-1.1.1.tar.gz

git clone https://github.com/google/grpc.git
cd grpc
git checkout v1.17.0
git submodule update --init --recursive
tar czf grpc.tar.gz grpc

git clone https://github.com/p4lang/PI.git
cd PI
git checkout 4546038f5770e84dc0d2bba90f1ee7811c9955df
git submodule update --init --recursive
tar czf PI.tar.gz PI
```

5.2 SDE 9.2.0 Caveat

5.2.1 dependencies compilation error

dependencies installation failed due to a missing package in the `onl.patch` file.

```
root@bud-onl:~/bf-sde-9.1.0# apt-get install pkg-config
```

5.3 SDE 9.1.0 Caveat

5.3.1 run_switchd -p <my-program.p4> is stuck

In some cases run_switchd is stuck:

```
root@bud-onl:~/bf-sde-9.1.0# ./run_switchd.sh -p bf_router
Using SDE /root/bf-sde-9.1.0
Using SDE_INSTALL /root/bf-sde-9.1.0/install
Setting up DMA Memory Pool
Using TARGET_CONFIG_FILE /root/bf-sde-9.1.0/install/share/p4/targets/tofino/bf_router.conf
Using                                                                          PATH
/root/bf-sde-9.1.0/install/bin:/root/jdk-13.0.2/bin:/root/bf-sde-9.1.0/install/bin:/usr/local/
sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/lib/platform-config/current/onl/bin:/lib/pl
atform-config/current/onl/sbin:/lib/platform-config/current/onl/lib/bin:/lib/platform-config/c
urrent/onl/lib/sbin
Using LD_LIBRARY_PATH /usr/local/lib:/root/bf-sde-9.1.0/install/lib:
Install Dir: /root/bf-sde-9.1.0/install (0x56526f6dfa60)
bf_sysfs_fname /sys/class/bf/bf0/device/dev_add
kernel mode packet driver present, forcing kernel_pkt option!
Install dir: /root/bf-sde-9.1.0/install (0x56526f6dfa60)
```

```
bf_switchd: system services initialized
bf_switchd:                                  loading                                  conf_file
/root/bf-sde-9.1.0/install/share/p4/targets/tofino/bf_router.conf...
bf_switchd: processing device configuration...
Configuration for dev_id 0
  Family        : tofino
  pci_sysfs_str : /sys/devices/pci0000:00/0000:00:03.0/0000:05:00.0
  pci_domain    : 0
  pci_bus       : 5
  pci_fn        : 0
  pci_dev       : 0
  pci_int_mode  : 1
  sbus_master_fw: /root/bf-sde-9.1.0/install/
  pcie_fw       : /root/bf-sde-9.1.0/install/
  serdes_fw     : /root/bf-sde-9.1.0/install/
  sds_fw_path   : /root/bf-sde-9.1.0/install/
  microp_fw_path:
bf_switchd: processing P4 configuration...
P4 profile for dev_id 0
num P4 programs 1
  p4_name: bf_router
  p4_pipeline_name: pipe
    libpd:
    libpdthrift:
    context: /root/bf-sde-9.1.0/install/share/tofinopd/bf_router/pipe/context.json
    config: /root/bf-sde-9.1.0/install/share/tofinopd/bf_router/pipe/tofino.bin
  Pipes in scope [0 1 2 3 ]
  diag:
  accton diag:
  Agent[0]: /root/bf-sde-9.1.0/install/lib/libpltfm_mgr.so
  non_default_port_ppgs: 0
  SAI default initialize: 1
bf_switchd: library /root/bf-sde-9.1.0/install/lib/libpltfm_mgr.so loaded
bf_switchd: agent[0] initialized
Tcl server started..
Tcl server: listen socket created
Tcl server: bind done on port 8008, listening...
Tcl server: waiting for incoming connections...
2020-02-29 03:20:02.566436 BF_PLTFM ERROR - Error Getting the Board info from BMC. Hence
exiting ****

root@bud-onl:~/bf-sde-9.1.0#
```

Solution:
Not sure why, `usb0` from the BMC is not activated. BMC and MAIN CPU are communication together via their respective `usb0` port. Therefore:

- log into BMC
- Enable `usb0` using: `ip link set usb0 up`

.

5.3.2 Cannot access BIOS is order to enable Ethernet port

In order to access the BIOS in order to activate the onboard Ethernet interfaces, you need to connect via BMC and bounce to the MAIN CPU and reboot from MAIN CPU.

5.3.3 ONL compilation with ixgbe ethernet driver as loadable module

On WEDGE-BF100-32X, there is a possibility to activate the 2x10GE on board ethernet built into the MAIN CPU board. The procedure is described INTEL connectivity forum portal.

These are intel Ethernet ports using `ixgbe` driver Linux module. However, if you issue an `lsmod` command ixgbe driver does not appear in the list. This is because during ONL 9 compilation this driver is compiled and set to be built in the kernel. (`lsmod` is listing only dynamically loadable modules)

For unknown reasons, these ethernet ports are going down and a reboot seems to be the only solution to reactivate them.

Having the possibility to load/unload the module dynamically can be useful as it can prevents the WEDGE reboot.

```
git clone https://github.com/opencomputeproject/OpenNetworkLinux.git
git checkout 3127f40b3c4d9122ba09eb40326286e1710a13af
# Get onl.patch from INTEL ONL support page
git apply onl.patch

# edit
# ~/OpenNetworkLinux/packages/base/any/kernels/4.14-lts/configs/x86_64-all/x86_64-all.config

…
# Find CONFIG_IXGBE
# set it to "m"
…
CONFIG_IXGBE=m
…

# save & exit file

# SDE 9.3.0 uses ONL9 which is based on Debian 9
export VERSION=9
make docker
…
<perfect time to grab a coffee>
```