Lanner

Lanner uCPE NCA-4020

Installation Guide

Verified Intel Select Solution for Universal Customer Premises Equipment (uCPE)

Version: 1.0

Date of Release: 2018-12-18

CHAPTER 1 - INTEL® SELECT SOLUTIONS OVERVIEW

Intel® Select Solutions are verified hardware and software stacks that are optimized for specific software workloads and are designed to be easy to deploy. The objective of the installation guide is to provide end users with detailed, step-by-step instructions so they can quickly and efficiently deploy the solution.

1.1 Hardware Configuration

Table 1: Intel® Select Solution for uCPE HW Configuration

NCA-4020	Base Platform	Plus Platform
Processor Number	Intel Xeon D-2183IT	Intel Xeon D-2187NT
Processor Spec.	16 Cores 22M Cache 2.20 GHz	16 Cores 22M Cache 2.00 GHz
Intel® QuickAssist Technology	N/A	100G Crypto/Comp.
Memory	16 GB DDR4 2400 MHz, 4 * 4 GB Total of 16 GB	64 GB DDR4 2667 MHz, 4 * 16 GB Total of 64 GB
NIC Module	4 x 10 GbE (XL710 NIC module)	4 x 10 GbE (XL710 NIC module)
Storage	Intel SSD DC S3700 Series 800GB	Intel SSD DC S3700 Series 800GB

1.2 Enhanced Auto Installer Package

The Enhanced Auto Installer package can be used for installing an Ubuntu* operating system image and all required SW stacks (refer to Table 2). This auto-installation procedure only supports booting in Unified Extensible Firmware Interface (UEFI) mode.

The Best Known Configuration (BKC) auto-installer package has two parts:

- The **OS part** contains a link to an International Standards Organization (ISO) disk image file for downloading OS files:

ubuntu-16.04.3-server-amd64.iso

- The **SW package** includes firmware, drivers, utilities, and some configuration scripts. This package will be updated in every BKC cycle.
- Intel® Xeon® Scalable Processor Platform Auto-Installer for Ubuntu* v16.04.3 (Document Number 596273) includes instructions to create an Ubuntu 16.04.3 installation with auto scripts that pull correct versions of software stacks and preload the specific drivers and firmware on the system to jump start testing and evaluation.

Table 2: Intel® Select Solution uCPE Software Version

	Ingredient	SW Version Details
OS	Ubuntu	Ubuntu 16.04.3 (kernel 4.4.0-116-generic)
APPs	DPDK	DPDK 18.02
Hypervisor	KVM/QEMU	2.5.0
Libvirt	Libvirt	Libvirt1.3.1
Drivers	Intel® QAT	1.7-L.1.0.5-25
	I40e	2.4.6
	Ixgbe	5.3.6

1.3 Terminology

Term	Description
BKC	Best Known Configuration
ISO	International Standards Organization (disk image file)
OS	Operating System
UEFI	Unified Extensible Firmware Interface
USB	Universal Serial Bus
Intel® DDIO	Intel® Data Direct I/O Technology
Intel® QAT	Intel® QuickAssist Technology
Intel® TXT	Intel® Trusted Execution Technology
Intel® UPI	Intel® Ultra Path Interconnect
Intel® VT	Intel® Virtualization Technology
SR-IOV	Single Root I/O Virtualization
SOC	System-on-a-Chip
uCPE	Universal Customer Premises Equipment
VMX	Virtual Machine Extensions
MGMT	Management

1.4 Reference Documents

Table 3: Reference Documents

Document	Document No./Location
Intel® Xeon® Scalable Processor Platform Auto-Installer for Ubuntu* v16.04.3	596273
Intel® QuickAssist Technology Software for Linux* - Getting Started Guide – HW Version 1.7	336212
Benchmarking Methodology for Network Interconnect Devices	https://tools.ietf.org/html/rfc2544
Benchmarking Terminology for Network Interconnection Devices	https://tools.ietf.org/html/rfc1242
Device Reset Characterization	https://tools.ietf.org/html/rfc6201

CHAPTER 2 - ENHANCED AUTO INSTALLATION STEPS

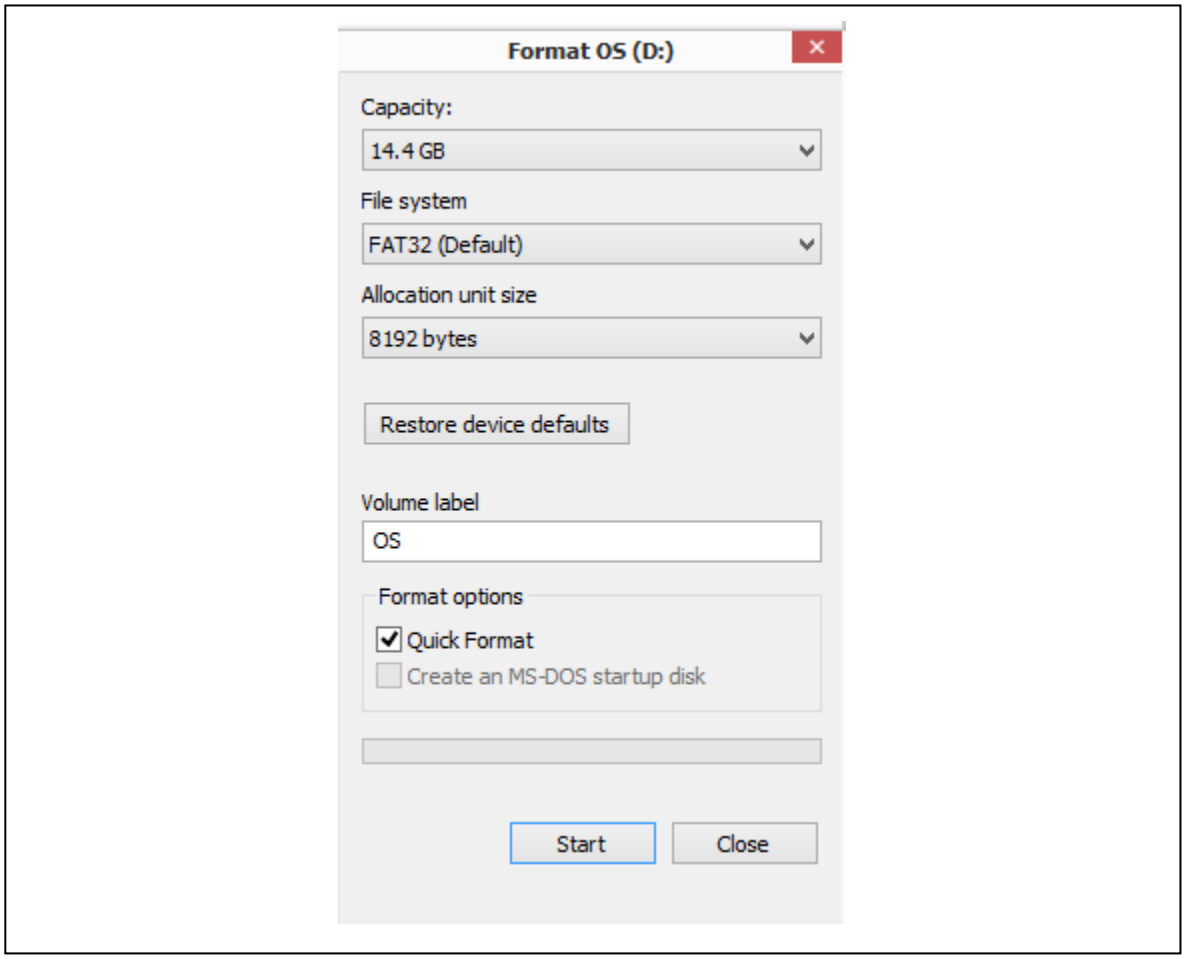
2.1 Preparation

Prepare a USB flash drive that has capacity greater than 8 GB.

In Windows*, format the USB drive as follows:

1. Connect the drive to a USB port.
2. Open Windows Explorer (or File Explorer).
3. Right-click on the USB drive and select Format.
4. Under **File System**, choose **FAT32 (Default)**.
5. In the **Volume Label** box, enter **OS** in upper case.

Figure 1. Formatting USB Drive



2.2 Copy Packages to the USB Flash Drive

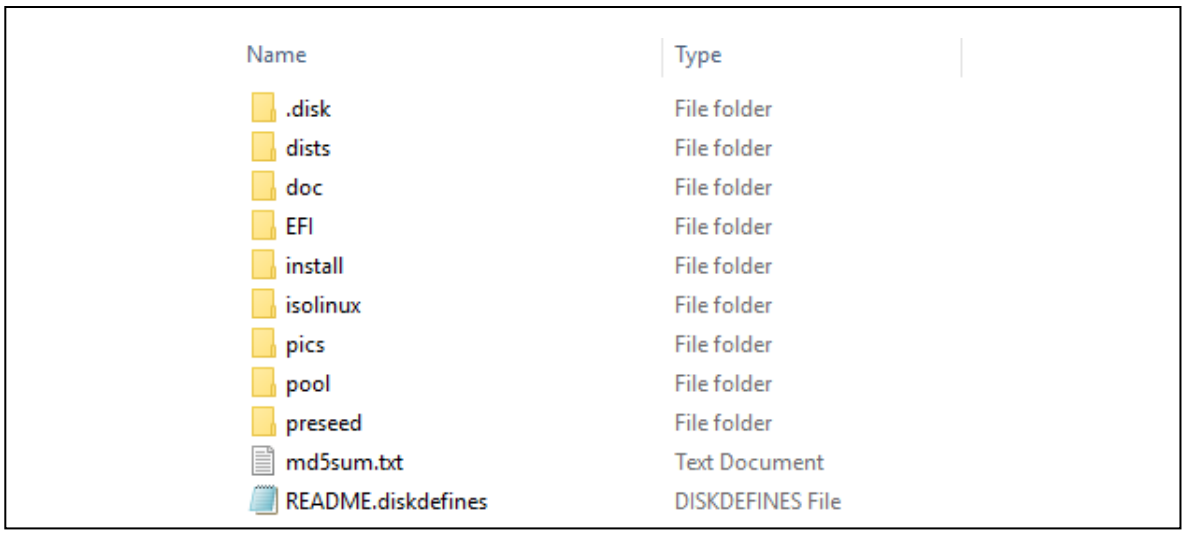
2.2.1 Copy the OS Package to the root of the USB Drive

1. Download Ubuntu v16.04.3 from the link:

- ubuntu-16.04.3-server-amd64.iso

2. Unzip the OS files.
3. Copy all contents except the [BOOT] and boot folders to the USB drive.

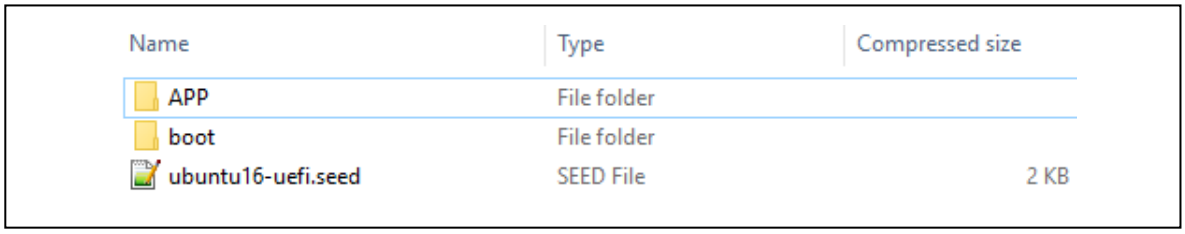
Figure 2. File listing of OS Install package



2.2.2 **Copy the SW Package to the root of the USB**

1. Download the SW Package (Document Number 596273) from Intel® IBL site
2. Go to the LinuxPackage folder included in the zip file containing this document.

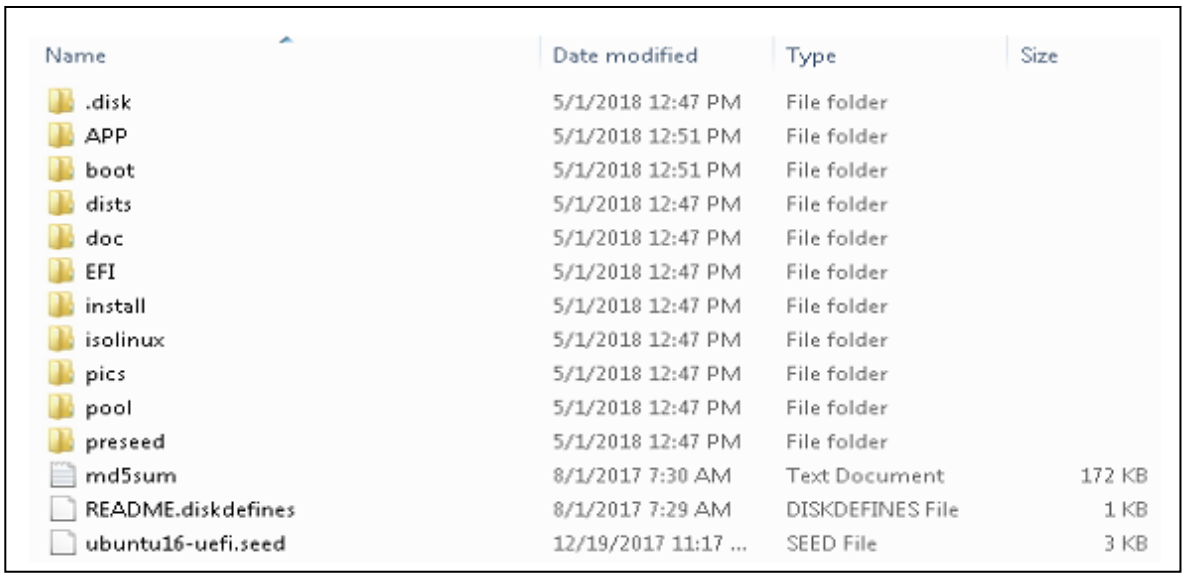
Figure 3. File listing of SW package



3. Unzip the files.

4. Edit ubuntu16-uefi.seed file to comment out line 16 as below
   - #d-i mirror/http/proxy string http://proxy-prc.intel.com:911
5. Copy all contents of the LinuxPackage folder to the root of the USB drive.
   - The final driver file structure on the USB drive should look like Figure 4.

Figure 4. Final USB Drive File Listing for Auto-Installer

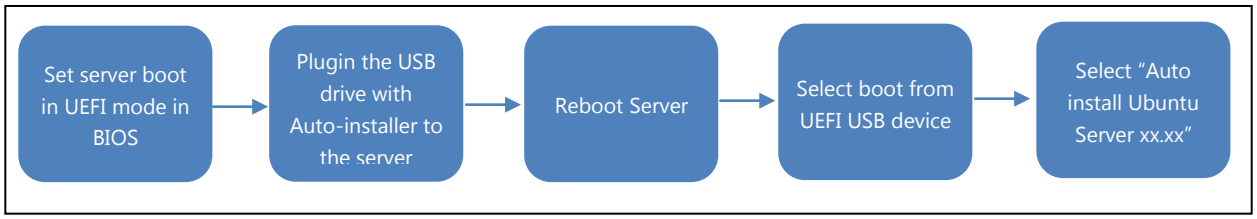


CHAPTER 3 - OPERATING SYSTEM INSTALLATION

3.1 OS Installation Overview

Figure 5 illustrates the general procedure for installing the OS.

Figure 5. OS Installation Procedure Overview



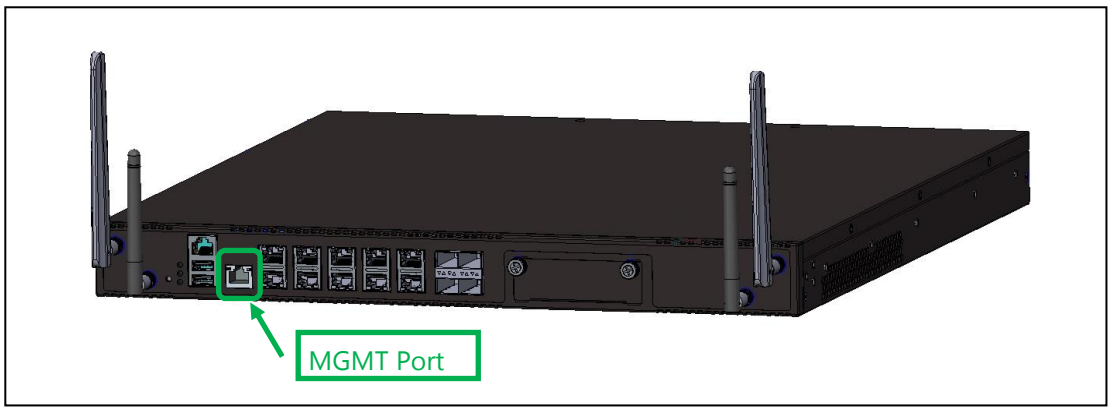
Note: **DO NOT UNPLUG THE USB** until the OS installation is complete!

The auto-installer takes about 30 minutes to complete. The system reboots two times during auto-installation.

3.2 Detailed Installation Steps

1. System's **MGMT** port connect to the network with dhcp available.

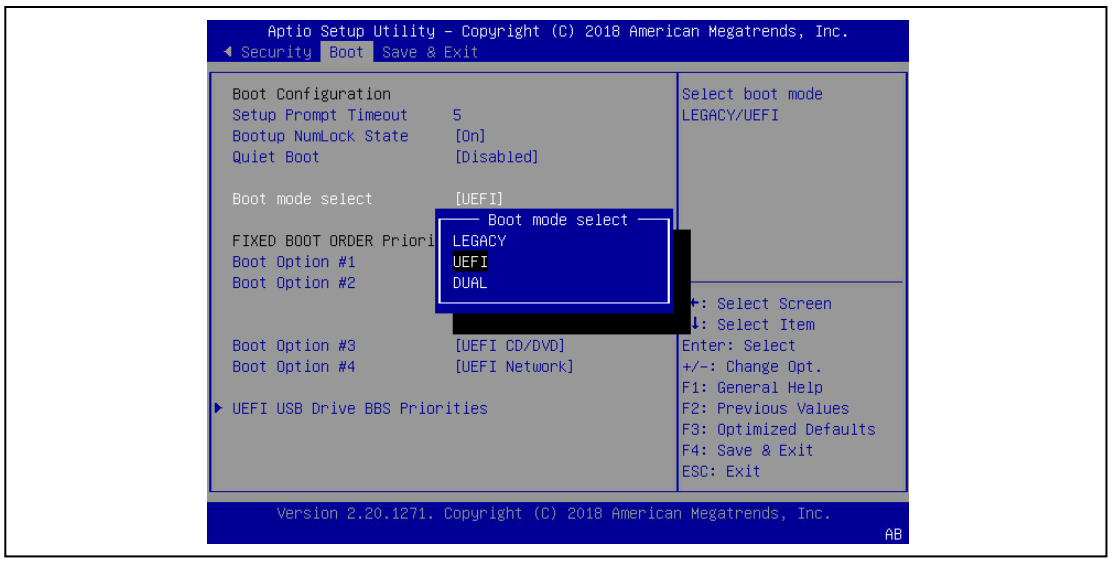
Figure 6. Location of NCA-4020's MGMT port



2. Enter BIOS setup on the system where you want to install the OS.
3. Select '**Boot**' menu.

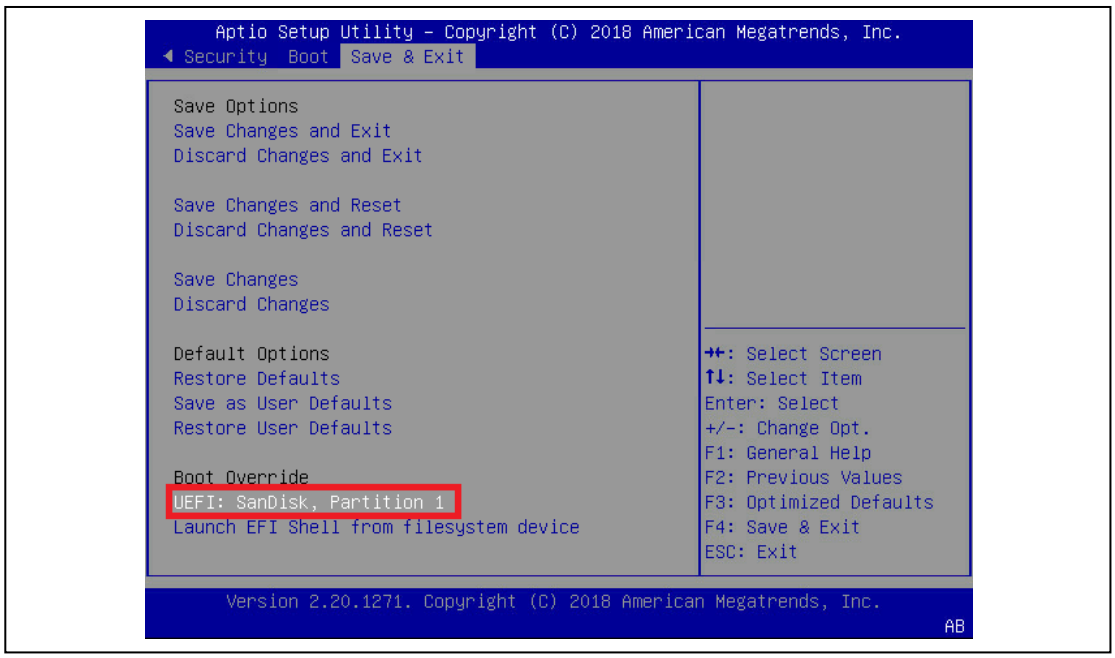
4. Set **Boot Mode** to **UEFI** (refer to Figure 7.)

Figure 7. Boot Option Setting in BIOS



5. Select '**Save & Exit**' menu.
6. From **Boot Override**, scroll down to the USB drive and select it.
   - **Note:** The manufacturer and model of your USB drive may not match Figure 8.

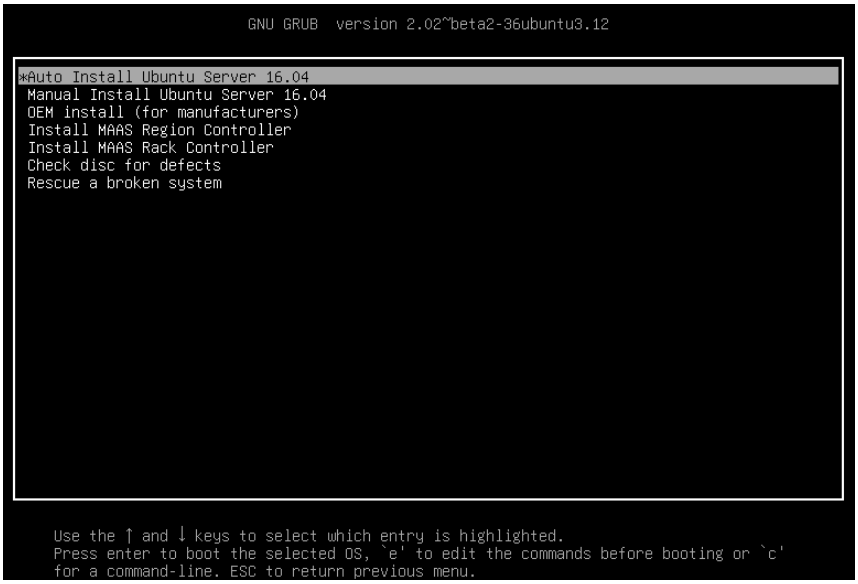
Figure 8. Select USB Drive to Use for Boot



   - If BIOS settings are configured correctly to boot from USB, the Auto Install menu will display the grub menu as shown in Figure 9.

7. Select **Auto Install Ubuntu Server**.

Figure 9. Grub Menu

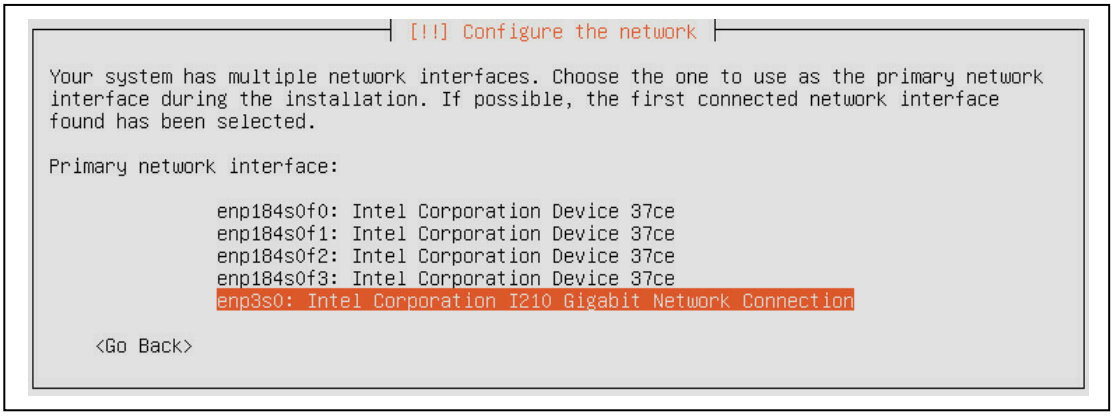


The installer starts loading installation components and detecting Ethernet devices on the system.

8. Choose the primary network interface to be used for network connection during the installation (refer to Figure 10).

▶

Figure 10. Network Device Selection Page



9. Continue with the installation to select the disk to install and partition, etc.

▶ The system will reboot after OS installation is complete. The system starts up automatically in Ubuntu*, performs auto-configuration, installs drivers, and reboots again. After the second reboot, BKC installation is complete.

CHAPTER 4 - DEFAULT SETTINGS

You can customize some server settings after auto-installation is complete.

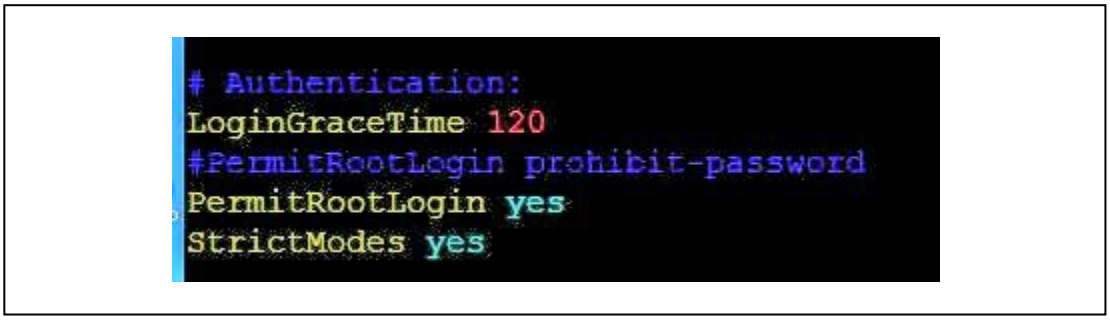
Note: Changes to server settings will take effect after the server reboots.

4.1 Account Settings

Default settings and passwords are described in this chapter.

- The auto installed Linux* system will have a root account with password "password".
- The auto installed Linux* system will have a user account "pid" with password "password". Use this account for remote login.
- The auto installed Linux* system has unassigned-hostname by default.
- The auto installed Linux root remote login is disabled because of security reasons. Please enable it if needed by editing /etc/ssh/sshd_config and setting "PermitRootLogin yes" as shown in Figure 11.

Figure 11. Editing /etc/ssh/sshd_config



4.2 Changing Passwords

To customize passwords, edit the ubuntu-xx.seed file in the root directory of the OS as follows:

d-i passwd/root-password password password

d-i passwd/root-password-again password password

4.3 Time Zone Settings

To change the default time zone, edit the ubuntu-xx.seed file in the root directory of the OS.

Note: Currently the default time zone is Asia/Shanghai.

d-i time/zone string Asia/Shanghai

4.4 Auto Logon User

The default user name in the ubuntu-xx.seed file is the default user for logon.

And SMBIOS type11 OEM String was "Recovery BIOS".

CHAPTER 5 - PERFORMANCE TEST PROCEDURE

The Data Plane Development Kit (DPDK) is a core platform technology component of the Intel® Select Solution for uCPE reference design for both base and plus configurations.

As such, it is expected that a compliant platform must implement DPDK software and meet the performance metrics as defined in this section.

5.1 Detecting Hardware/Software Configurations

There is a script named Ubuntu.sh to verify that your hardware configuration, host OS version, and software match the specifications in Sections 1.1 and 1.2. This script is located in the '/opt/APP/script directory. This script documented in Appendix B of this document.

To test that your configuration matches the specifications for the Intel® Select Solution for uCPE **base** HW and SW configuration, type in a console window:

./Ubuntu.sh NCA-4020 base > NCA-4020_Ubuntu.log

To test that your configuration matches the specifications for the Intel® Select Solution for uCPE **plus** uCPE HW and SW configuration, type in a console window:

./Ubuntu.sh NCA-4020 plus > NCA-4020_Ubuntu.log

5.2 Packet Processing Performance Requirements

Intel® Select Solution for uCPE base and plus configuration-compliant platform solutions must demonstrate a minimum packet processing performance as specified in Section 5.2.2.1 and Section 5.2.2.2 implementing the Data Plane Development Kit (DPDK) to optimize packet processing performance.
In order to validate conformance to this requirement, benchmarking must be performed using the DPDK L3 Forwarding application. Information on the DPDK L3fwd test case can be found at the web link:

http://www.dpdk.org/doc/guides/sample_app_ug/l3_forward.html

The RFC2544 zero packet loss test case (https://tools.ietf.org/html/rfc2544) is used to validate conformance. This test is used to determine the target platform throughput as defined in RFC1242 (https://tools.ietf.org/html/rfc1242). For this requirement, the RFC2544 test uses DPDK L3fwd as the test application.

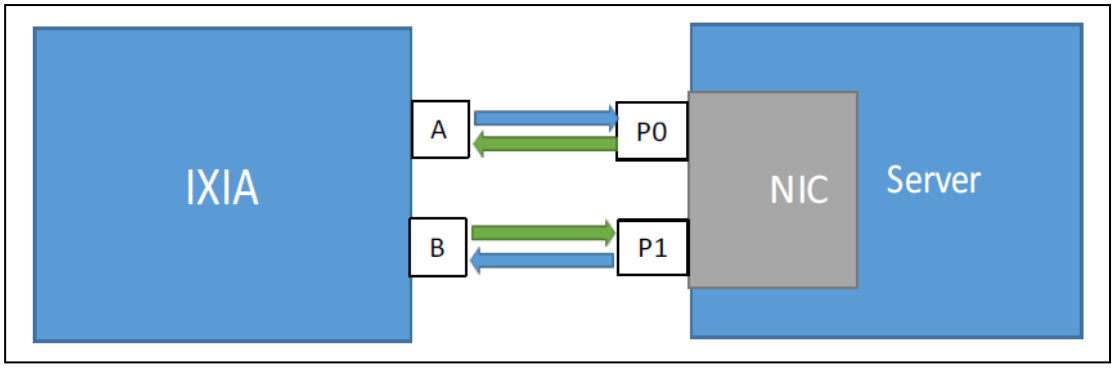
Note: RFC6201 (https://tools.ietf.org/html/rfc6201) updates both RFC2544 and RFC1242 with respect to resets.

5.2.1 Packet Processing Test Procedure

The summary of the test procedure is:

1. Send a specific number of frames at a specific rate from the test equipment through the Device Under Test (DUT) and then back to the test equipment.

2. Count the frames that are transmitted back by the DUT.
3. If the number of sent frames is not equal to the count of received frames, reduce the number of sent frames and rerun the test.
   - The throughput is the fastest rate at which the count of test frames transmitted by the DUT is equal to the number of test frames sent to it by the test equipment.
   -
   - Figure 12. Test Setup (with IXIA* Packet Generator)
   -



5.2.2 **Packet Processing Performance Requirements**

5.2.2.1 Packet Processing Performance Requirements for Plus Platform

- **NIC:** NCS2-IXM407 Module, 4-port 10GbE SFP+ NIC Module
- **Test Configuration:**
  - Four ports of the NIC are used.
  - Each port has one queue.
  - One queue is assigned per logical core.
  - Total four logical cores, four queues, and four ports.
- **Test Parameters:** ./build/l3fwd -l 1-5 -n 4 -- -p 0xf --config="(0,0,1),(1,0,2),(2,0,3),(3,0,4)"
- **DPDK Test:** DPDK L3fwd RFC2544 zero packet loss test
- **Performance Requirement:** 100% line rate with packet size 256B
- **Conformance:** Required

5.2.2.2 Packet Processing Performance Requirements for Base Platform

- **NIC:** NCS2-IXM407 Module, 4-port 10GbE SFP+ NIC Module
- **Test Configuration:**
  - Two ports of the NIC are used.
  - Each port has one queue.
  - One queue is assigned per logical core.
  - Total two logical cores, two queues, and two ports.
- **Test Parameters:** ./build/l3fwd -l 1-4 -n 4 -- -p 0x3 --config="(0,0,1),(1,0,2)"
- **DPDK Test:** DPDK L3fwd RFC2544 zero packet loss test
- **Performance Requirement:** 100% line rate with packet size 128B
- **Conformance:** Required

5.3 Intel® QAT Performance Requirement

5.3.1 **Intel® QAT Performance Sample Code**

Compliance is measured by the Intel® QuickAssist Technology (Intel® QAT) Performance sample code that is cpa_sample_code cipher and hash benchmark utility included as part of the Intel® QAT software package installation.

Note: The Compress and Verify feature in Intel® QAT is required to meet the performance requirements specified in Table 4 and Table 5. Appendix A contains instructions for enabling Compress and Verify.

Table 4. Intel® QAT Performance Requirements

SKU	Compress and Verify[1]	Encryption[2]	RSA[3]
Intel® Select Solution for uCPE Plus Platform	55 Gb/s	100 Gb/s	100 K sign/s

NOTE: [1]Performance to be measured at 8 KB packet size

[2]Performance to be measured at 4 KB packet size

[3]Performance to be measured at 2 KB packet size

The test can be accomplished by executing the command:

/opt/APP/driver/QAT/build/cpa_sample_code > NCB-4020_qat.log

Note: Refer to the Intel® QuickAssist Technology Software for Linux* - Getting Started Guide – HW Version 1.7, available at https://01.org/intel-quickassist-technology. (Table 3) Ensure that the Intel® QAT driver is located at /opt/APP/driver/QAT.

5.3.2 OpenSSL Speed

In addition to the bulk crypto performance requirements, both Intel® Select Solution for uCPE base and plus configuration-compliant platform solutions must demonstrate a minimum OpenSSL performance requirement as measured by executing OpenSSL speed benchmark testing.

An Intel® Select Solution for uCPE base platform must demonstrate 20 Gbps and a plus platform 100 Gbps running the OpenSSL Speed AES 128-CBC-HMAC-SHA1. Similarly, it must demonstrate 20K sign operations per second and 100 sign operations per second for RSA 2048 benchmarks. The tables below summarize the requirement.

Table 5. Intel® QAT Performance Requirements

SKU	AES128-CBC-HMAC-SHA1[1]	RSA 2048[1]
Intel® Select Solution for uCPE Plus Platform	100 Gb/s	100 K sign/s

NOTE: [1]Performance to be measured at 16 KB packet size

This section contains instructions for setting up the requirements on running OpenSSL speed.
Detailed instructions for the following procedures are documented in this weblink: https://github.com/01org/QAT_Engine/blob/master/README.md.

- Building the Intel® QAT Driver
- Building OpenSSL
- Building the Intel® QAT OpenSSL Engine
- Running the openssl speed application

Note: The Intel® QAT engine for the openssl build command line example:

```
./configure \
--with-qat_dir=/opt/APP/driver/QAT \
--with-openssl_dir=/root/speed/test/openssl \
--with-openssl_install_dir=/usr/local/ssl \
--enable-upstream_driver \
--enable-usdm \
--disable-qat_lenstra_protection
```

Alternatively, the QAT_speed.sh script can be used to download, build OpenSSL, and run the OpenSSL speed test for the algorithm that is required for conformance. A result file named openssl.log will be generated.

APPENDIX A INSTALLING INTEL® QAT WITH COMPRESS AND VERIFY ENABLED

Some versions of Intel® QuickAssist Technology do not have the Compress and Verify feature installed by default. To install Intel® QAT with Compress and Verify enabled:

1. To remove the QAT driver run the following
   - service qat_service stop
   - cd /opt/APP/driver/QAT
   - ./installer.sh u
   - cd ..
   - rm –rf /opt/APP/driver/QAT
   - rm /usr/lib64/libqat_s.so
   - rm /usr/lib64/libusdm_drv_s.so
2. Upgrade the QAT driver to version 1.7L.4.2.0-0012 or later
   a. Navigating to https://01.org/zh/intel-quickassist-technology to download the latest QAT driver for Linux
   b. Copy the file to '/opt/APP/driver' directory then untar it
      - mkdir /opt/APP/driver/QAT
      - cd /opt/APP/driver/QAT
      - tar zxf ../qat1.7.l.x.x.x-xxxxx.tar.gz
3. To rebuild and install the new QAT driver, run installer.sh with option 2.
4. To build the sample application, run installer.sh with option 6.
5. Restart the QAT service by entering:
   - service qat_service restart
6. CNV data from the log file should include:

```
----------------------------------------
API                     Data_Plane
Session State           STATELESS
Algorithm               DEFLATE
Huffman Type            STATIC
Mode                    ASYNCHRONOUS
Use CNV                 YES
CNV Enabled             YES
Direction               COMPRESS
Packet Size             8192
Compression Level       1
```

APPENDIX B UBUNTU.SH

```
#!/bin/bash
#
# Revision Rev1.2 Script based on Reference Design Rev1.0
#
# Purpose - Checking the Hardware/Firmware and Software per Intel Select Brand Reference
# Design Implementation for uCPE
# Help - ubuntu.sh customer_name base/plus
base=1
plus=1
```

```
check=0
if [ "$#" -ne 2 ]; then
	echo "Usage: ubuntu.sh <Customer_Name> <base/plus>"
	check=1
fi
echo ""
echo "================="
if [ "$2" = "base" ]; then
	echo "Intel Select Base for uCPE"
fi;
if [ "$2" = "plus" ]; then
	echo "Intel Select Plus for uCPE"
fi;
if [ "$check" = "1" ]; then
	echo "Checking for Conformance"
fi;
echo "================="
echo ""
CustomerName=$1
str="_hwsw.log"
file=$CustomerName$str
echo ""
echo "INFO:- File $file will be generated"
echo "INFO:- File $file is generated" > $file
echo "INFO:- ubuntu.sh for uCPE revision 1.1" > $file
date >> $file
echo "" >> $file
echo "=================" >> $file
if [ "$2" = "base" ]; then
	echo "Intel Select Base for uCPE on Ubuntu" >> $file
fi;
if [ "$2" = "plus" ]; then
	echo "Intel Select Plus for for uCPE on Ubuntu" >> $file
fi;
if [ "$check" = "1" ]; then
	echo "Checking for Conformance" >> $file
fi;
echo "=================" >> $file
echo "" >> $file
echo ""
echo "=== CPU Info ==="
lscpu | grep Model
lscpu | grep "CPU(s)"
lscpu | grep Stepping
failstr=""
failplus=""
failbase=""
Microcode=$(cat /proc/cpuinfo | grep microcode | sed -n '1p' | awk '{print $2}')
Microcode=$(cat /proc/cpuinfo | grep microcode | sed -n '1p' | tr -d ' ' | cut -f2 -d":")
echo "microcode: $Microcode"
CpuInfo_Platform=$(lscpu | grep Model | sed -n '2p' | awk '$2 == "Intel(R)" {print $4}')
CpuInfo_SKU=$(lscpu | grep Model | sed -n '2p' | awk '$3 == "Intel(R)" {print $5}')
CPU_Pass=1
CpuXD=$(lscpu | grep Model | sed -n '2p' | awk '$3 == "Intel(R)" {print $5}'|cut -d'-' -f1)
if [ "$2" = "base" ] || [ "$check" = "1" ]; then
	if [ "$CpuXD" = "D" ]; then
		echo "CPU SKU is Xeon D Family"
		CPU_Pass=1
	else
		failbase="== CPU - not Xeon D Family"
		failstr="== CPU - not Xeon D Family"
		base=0
		CPU_Pass=0;
	fi;
fi;
if [ "$2" = "plus" ] || [ "$check" = "1" ]; then
	if [ $CpuInfo_SKU = "D-2177NT" ] || [ $CpuInfo_SKU = "D-2187NT" ]; then
		echo "CPU SKU $CpuInfo_SKU discovered"
	else
```

```
            failplus="== CPU - not D-2187NT or D-2177NT =="
            failstr="== CPU - not D-2187NT or D-2177NT =="
            plus=0
            CPU_Pass=0;
        fi;
fi;
if [ $CPU_Pass -eq 1 ]; then
        echo ".....Passed $CpuInfo_SKU "
else
        echo ".....Failed $CpuInfo_SKU "
fi;
echo "" >> $file
echo "=== SCOREBOARD SUMMARY " >> $file
echo "=== HARDWARE " >> $file
if [ "$CPU_Pass" = "1" ]; then
        echo "= CPU PASS $CpuInfo_SKU" >> $file
else
        echo "= CPU FAIL $failstr" >> $file
fi;
failstr=""
echo ""
echo "=== MEM Info ==="
cat /proc/meminfo | grep MemTotal
Mem_Pass=0
MemSize=$(cat /proc/meminfo | grep MemTotal | awk '{print $2}')
if [ "$2" = "base" ]; then
        if [ $MemSize -lt 16000000 ]; then
            base=0
        else
            Mem_Pass=1
        fi;
fi;
if [ "$2" = "plus" ]; then
        if [ $MemSize -ge 60000000 ]; then
            Mem_Pass=1
        else
            plus=0
        fi;
fi;
if [ $Mem_Pass -eq 1 ]; then
        echo ".....Passed $MemSize"
else
        echo ".....Failed: Totol Memory $MemSize "
fi;
MemArrange=$(dmidecode -t Memory Device | grep 'Size')
n=$(echo "$MemArrange" | wc -l) #24
MemArrangeCPU1=$(dmidecode -t Memory Device | grep 'Size' | sed -n 1,"$n"p | sed 's/[^0-9]*//g');
MemNumCPU1=$(echo "$MemArrangeCPU1" | wc -w)
MemSizeCPU1=$(echo $MemArrangeCPU1 | cut -f2 -d" ")
MemArrangeSpeedCPU1=$(dmidecode -t Memory Device | grep 'Speed' | sed -n 1,"$n"p | sed 's/[^0-9]*//g');
MemNumSpeedCPU1=$(echo "$MemArrangeSpeedCPU1" | wc -w)
MemSpeedCPU1=$(echo $MemArrangeSpeedCPU1 | cut -f2 -d" ")
echo CPU1: $MemNumCPU1 x $MemSizeCPU1 GB $MemSpeedCPU1 Hz
MemArrange_Pass=0
if [ $MemSizeCPU1 -ge 16 ] && [ $MemNumCPU1 -ge 4 ]; then
        MemArrange_Pass=1
fi
if [ $MemSizeCPU1 -ge 16 ] && [ $MemNumCPU1 -ge 8 ]; then
        MemArrange_Pass=1
fi
if [ $MemNumCPU1 -eq 0 ]; then
        MemArrange_Pass=0
fi
if [ $MemArrange_Pass -eq 1 ]; then
        echo ".....Passed"
else
        echo ".....Failed"
        base=0
        plus=0
```

```
        failstr="$failstr""== MEM - Fail Memory Population =="
        failbase="$failbase""== MEM - Fail Memory Population =="
        failplus="$failplus""== MEM - Fail Memory Population =="
        Mem_Pass=0;
fi;
if [ "$Mem_Pass" = "1" ]; then
        echo "= MEM PASS $MemSize" >> $file
else
        echo "= MEM FAIL $MemSize" >> $file
fi;
echo ""
echo "=== NIC ==="
lspci | grep Ethernet

faillstr=""
pcie_qat_pass=1
if [ "$check" = "1" ] || [ "$2" = "plus" ]; then
        echo ""
        echo "=== QAT HW ==="
        Count_37c8=0
        echo ""
        lspci | grep "37c8"
        lspci | grep "37c8" | { while read -r a;
                do
                        QATAddr=$(echo $a | awk '{print $1}')
                        DeviceID=$(cat /sys/bus/pci/devices/0000\:$QATAddr/device)
                        if [ "$2" = "1" ]; then
                                echo " $QATAddr "
                        fi
                        if [ "$DeviceID" = "0x37c8" ]; then
                                Count_37c8=$((Count_37c8+1))
                        fi
                done
                if [ $Count_37c8 -ge 3 ]; then
                        echo "Intel'(R)' QAT Engines.....Passed"
                else
                        echo "Only $Count_37c8 x Intel'(R)' QAT Engines"
                        base=0
                        plus=0
                        pcie_qat_pass=0
                fi;
                if [ "$pcie_qat_pass" = "1" ]; then
                        echo "= QAT HW PASS $Count_37c8 Engine" >> $file
                else
                        echo "= QAT HW FAIL $Count_37c8 Engine" >> $file
                fi
        }
fi

SSD_Pass=0
SSD_size=$(parted -l | egrep 'INTEL|Disk' | sed -n '2p' | awk '{print $3}'| sed 's/GB//g' )
ssd_vendor=$(parted -l | egrep 'Model|Disk' | sed -n '1p' | awk '{print $3}')
echo ""
echo "=== SSD ==="
if [ "$ssd_vendor" = "INTEL" ]; then
        SSD_Pass=1
fi
if [ "$check" = "1" ] || [ "$2" = "plus" ]; then
        if [ $SSD_size -lt 512 ]; then
                failstr="$failstr""== SSD - size less than 512GB =="
                failplus="$failplus""== SSD - size less than 512GB =="
                plus=0
                SSD_Pass=0
        else
                SSD_Pass=1
        fi
fi
if [ "$check" = "1" ] || [ "$2" = "base" ]; then
        if [ $SSD_size -lt 256 ]; then
```

```
            failstr="$failstr""== SSD - size less than 256GB =="
            failplus="$failplus""== SSD - size less than 256GB =="
            plus=0
            base=0
            SSD_Pass=0
        else
            SSD_Pass=1
        fi
fi
if [ $SSD_Pass -eq 1 ]; then
        echo ".....Passed"
else
        echo ".....Failed"
fi;
if [ "$SSD_Pass" = "1" ]; then
        echo "= SSD HW PASS $ssd_vendor $SSD_size GB" >> $file
else
        echo "= SSD HW FAIL $ssd_vendor $SSD_size GB" >> $file
fi;
echo ""
echo "=== BIOS ==="
dmidecode -t bios | grep Version
dmidecode -t bios | grep "Release Date:"
BIOS_Pass=0
BIOSVersion=$(dmidecode -t bios | grep Version | awk '{print $2}')
BIOSDateY=$(dmidecode -t bios | grep "Release Date:" | awk '{print $3}' | cut -f3 -d"/")
BIOSDateM=$(dmidecode -t bios | grep "Release Date:" | awk '{print $3}' | cut -f1 -d"/")
BIOSDateD=$(dmidecode -t bios | grep "Release Date:" | awk '{print $3}' | cut -f2 -d"/")
biosfstr="$BIOSDateY""$BIOSDateM""$BIOSDateD"
if [ $BIOSDateY -gt 2018 ]; then
        BIOS_Pass=1
elif [ $BIOSDateY -eq 2018 ] && [ $BIOSDateM -gt 2 ]; then
        BIOS_Pass=1
elif [ $BIOSDateY -eq 2018 ] && [ $BIOSDateM -eq 2 ] && [ $BIOSDateD -ge 8 ]; then
        BIOS_Pass=1
fi;
check_tool=$(dpkg -l | grep msr | awk '{print $2}')
if [ "$check_tool" = "msr-tools" ]; then
        echo "msr-tools has been installed"
else
        apt -y install msr-tools
fi;
modprobe msr
mcu=$(rdmsr 0x8b -f 48:32)
echo "MCU revision is ${mcu}"
mcu_dec=`echo "$((16#$mcu))"`
mcu_min=`echo "$((16#43))"`
if [ $mcu_dec -ge $mcu_min ]
        BIOS_Pass=1
else
        BIOS_Pass=0
        biosfstr="$biosfstr""== $mcu"
fi
if [ $BIOS_Pass -eq 1 ]; then
        echo ".....Passed $BIOSVersion dated $biosfstr"
else
        echo ".....Failed: BIOS $BIOSVersion dated $biosfstr"
fi;
uncore_scale=$(rdmsr 0x620)
stringlen=${#uncore_scale}
freq=$(rdmsr 0x620| sed 's/./ /g' | awk '{print $2}')
echo "msr 0x620 = $uncore_scale"
if [ $stringlen -lt 4 ]; then
        echo "Uncore Scaling Enabled.....Failed:BIOS Configuration"
else
        freq=$(expr substr "$uncore_scale" 1 4)
        max=$(expr substr "$freq" 3 2)
        min=$(expr substr "$freq" 1 2)
        if [ "$min" = "$max" ] || [ $min -ge $max ]; then
```

```
            echo "Uncore Scaling Disabled.............Passed"
        else
            echo "Uncore Scaling Enabled.............Failed:BIOS Configuration"
            BIOS_Pass=0
            biosfstr="$biosfstr""== BIOS - Recommend Uncore Freq Scaling to be disabled=="
        fi;
fi;
if [ "$check" = "1" ] || [ "$2" = "plus" ]; then
        echo ""
        echo "=== QAT ==="
        QAT_Pass=0
        QATVersion0=$(cat /opt/APP/driver/QAT/versionfile | sed -n '3p' | cut -f2 -d"=")
        QATVersion1=$(cat /opt/APP/driver/QAT/versionfile | sed -n '5p' | cut -f2 -d"=")
        QATVersion2=$(cat /opt/APP/driver/QAT/versionfile | sed -n '7p' | cut -f2 -d"=")
        QATVersion3=$(cat /opt/APP/driver/QAT/versionfile | sed -n '9p' | cut -f2 -d"=")
        QATVersion4=$(cat /opt/APP/driver/QAT/versionfile | sed -n '11p' | cut -f2 -d"=")
        echo "Version: $QATVersion0.$QATVersion1.$QATVersion2.$QATVersion3-$QATVersion4"
        QAT_VER="$QATVersion0.$QATVersion1.$QATVersion2.$QATVersion3-$QATVersion4"
        if [ $QATVersion1 -gt 1 ]; then
            QAT_Pass=1
        elif [ $QATVersion1 -eq 1 ] && [ $QATVersion2 -gt 0 ]; then
            QAT_Pass=1
        elif [ $QATVersion1 -eq 1 ] && [ $QATVersion2 -eq 0 ] && [ $QATVersion3 -gt 3 ]; then
            QAT_Pass=1
        elif [ $QATVersion1 -eq 1 ] && [ $QATVersion2 -eq 0 ] && [ $QATVersion3 -eq 3 ] && [ $QATVersion4 -ge 42 ]; then
            QAT_Pass=1
        fi
        if [ $QAT_Pass -eq 1 ]; then
            echo ".............Passed $QAT_VER"
        else
            echo ".............Failed: $QAT_VER, QAT-L.1.0.3-42 Needed"
        fi
fi
echo ""
echo "=== DPDK ==="
DPDK_Pass=0
echo $(find /opt/APP/utility/ -maxdepth 1 -name 'dpdk*' | sed -n '1p' | cut -f5 -d"/")
DPDK_Ver=$(find /opt/APP/utility/ -maxdepth 1 -name 'dpdk*' | sed -n '1p' | cut -f5 -d"/")
DPDKVersion1=$(find /opt/APP/utility/ -maxdepth 1 -name 'dpdk*' | sed -n '1p' | cut -f5 -d"/" |
cut -f2 -d"-" | cut -f1 -d".")
DPDKVersion2=$(find /opt/APP/utility/ -maxdepth 1 -name 'dpdk*' | sed -n '1p' | cut -f5 -d"/" |
cut -f2 -d"-" | cut -f2 -d".")
if [ $DPDKVersion1 -gt 18 ]; then
        DPDK_Pass=1
elif [ $DPDKVersion1 -eq 18 ] && [ $DPDKVersion2 -ge 02 ]; then
        DPDK_Pass=1
fi
if [ $DPDK_Pass -eq 1 ]; then
        echo ".............Passed $DPDK_Ver"
else
        echo ".............Failed: $DPDK_Ver, DPDK-18.02 Needed"
fi
echo ""
echo "=== Driver ==="
nic_driver_pass=1
I40eVersion=$(modinfo i40e | grep -i version | sed -n '1p' | sed 's/[[:space:]]//g' | cut -f2 -d":");
echo "i40e: $I40eVersion"
I40eVersionNum=$(modinfo i40e | grep -i version | sed -n '1p' | grep -o '[0-9]*' | tr -d '\n')
if [ $I40eVersionNum -ge 246 ]; then
        echo ".............Passed"
else
        echo ".............Failed: i40e 2.4.6 Needed"
        nic_driver_pass=0
fi;
osfailstr=""
echo ""
echo "=== OS ==="
OS=$(cat /etc/issue | sed -n '1p' | awk '$1 == "Ubuntu" {print $2}')
echo "OS: $OS"
```

```
UbuntuVer1=$(cat /etc/issue | sed -n '1p' | awk '$1 == "Ubuntu" {print $2}' | cut -d"." -f1)
UbuntuVer2=$(cat /etc/issue | sed -n '1p' | awk '$1 == "Ubuntu" {print $2}' | cut -d"." -f2)
UbuntuVer3=$(cat /etc/issue | sed -n '1p' | awk '$1 == "Ubuntu" {print $2}' | cut -d"." -f3)
OS_Pass=0;
if [ $UbuntuVer1 -gt 16 ]; then
      OS_Pass=1
elif [ $UbuntuVer1 -eq 16 ] && [ $UbuntuVer2 -gt 4 ]; then
      OS_Pass=1
elif [ $UbuntuVer1 -eq 16 ] && [ $UbuntuVer2 -eq 4 ] && [ $UbuntuVer3 -ge 3 ]; then
      OS_Pass=1
fi;
if [ $OS_Pass -eq 1 ]; then
      echo ".....Passed $OS"
else
      echo ".....Failed: Require Ubuntu $OS Version 16.04.3"
fi;
echo ""
echo "=== KVM/QEMU ==="
virsh version
QemuVer1=$(virsh version | sed -n '4p' | cut -d" " -f4 | cut -d"." -f1)
QemuVer2=$(virsh version | sed -n '4p' | cut -d" " -f4 | cut -d"." -f2)
QemuVer3=$(virsh version | sed -n '4p' | cut -d" " -f4 | cut -d"." -f3)
Qemu_Pass=0;
if [ $QemuVer1 -gt 2 ]; then
      Qemu_Pass=1
elif [ $QemuVer1 -eq 2 ] && [ $QemuVer2 -gt 5 ]; then
      Qemu_Pass=1
elif [ $QemuVer1 -eq 2 ] && [ $QemuVer2 -eq 5 ] && [ $QemuVer3 -ge 0 ]; then
      Qemu_Pass=1
fi
if [ $Qemu_Pass -eq 1 ]; then
      echo ".....Passed $QEMU"
else
      echo ".....Failed: Require Qemu Version 2.5.0"
fi
echo ""
echo "=== Libvirt ==="
Lib_Pass=0;
LibVer1=$(libvirtd --version | cut -d" " -f3 | cut -d"." -f1)
LibVer2=$(libvirtd --version | cut -d" " -f3 | cut -d"." -f2)
LibVer3=$(libvirtd --version | cut -d" " -f3 | cut -d"." -f3)
if [ $LibVer1 -gt 1 ]; then
      Lib_Pass=1
elif [ $LibVer1 -eq 1 ] && [ $LibVer2 -gt 3 ]; then
      Lib_Pass=1
elif [ $LibVer1 -eq 1 ] && [ $LibVer2 -eq 3 ] && [ $LibVer3 -ge 1 ]; then
      Lib_Pass=1
fi;
libvirtfailstr="$LibVer1"."$LibVer2"."$LibVer3"
echo "LIBVIRT version is $libvirtfailstr"
if [ $Lib_Pass -eq 1 ]; then
      echo ".....Passed"
else
      echo ".....Failed: Require Libvirt 1.3.1"
fi;
SW_fail=0
echo "=== SOFTWARE/FIRMWARE " >> $file
if [ "$BIOS_Pass" = "1" ]; then
      echo "= BIOS PASS $BIOSVersion $biosfstr" >> $file
else
      echo "= BIOS FAIL $BIOSVersion $biosfstr" >> $file
      SW_Fail=1
fi;
if [ "$2" = "plus" ] || [ "$check" = "1" ]; then
      if [ "$QAT_Pass" = "1" ] && [ "$2" = "plus" ]; then
            echo "= QAT SW PASS $QAT_VER" >> $file
      else
            echo "= QAT SW FAIL $QAT_VER" >> $file
            SW_Fail=1
```

```
        fi
fi
if [ "$DPDK_Pass" = "1" ]; then
        echo "= DPDK SW PASS $DPDK_Ver" >> $file
else
        echo "= DPDK SW FAIL $DPDK_Ver" >> $file
        SW_Fail=1
fi;
if [ "$nic_driver_pass" = "1" ]; then
        echo "= NIC Driver PASS $I40eVersion" >> $file
else
        echo "= NIC Driver FAIL $I40eVersion" >> $file
        SW_Fail=1
fi;
if [ "$OS_Pass" = "1" ]; then
        echo "= OS PASS $OS" >> $file
else
        echo "= OS FAIL $OS" >> $file
        SW_Fail=1
fi;
if [ "$Qemu_Pass" = "1" ]; then
        echo "= KVM/QEMU PASS $QEMU" >> $file
else
        echo "= KVM/QEMU FAIL $QEMU" >> $file
        SW_Fail=1
fi;
if [ "$Lib_Pass" = "1" ]; then
        echo "= Libvirt PASS $libvirtfailstr" >> $file
else
        echo "= Libvirt FAIL $libvirtfailstr" >> $file
        SW_Fail=1
fi;
if [ "$SW_Fail" = "1" ]; then
        sw_string="Software stacks, BIOS, Intel Platform Technology - FAIL"
else
        sw_string="Software stacks, BIOS, Intel Platform Technology - PASS"
fi
echo "=== SCOREBOARD SUMMARY " >> $file
echo "" >> $file
echo "=== Intel Select Reference " >> $file
if [ "$check" = 1 ]; then
        if [ "$base" = "1" ]; then
                echo "= Intel Select Reference HW BOM Base - PASS || $sw_string = " >> $file
                else echo "= Intel Select Reference HW BOM Base - FAIL $failbase || $sw_string =" >> $file
        fi
        if [ "$plus" = "1" ]; then
                echo "= Intel Select Reference HW BOM Plus - PASS || $sw_string = " >> $file
        else
                echo "= Intel Select Reference HW BOM Plus - FAIL $failplus || $sw_string = " >> $file
        fi
else
        if [ "$2" = "base" ]; then
                if [ "$base" = "1" ]; then
                        echo "= Intel Select Reference HW BOM Base - PASS || $sw_string =" >> $file
                else
                        echo "= Intel Select Reference HW BOM Base - FAIL $failbase || $sw_string =">> $file
                fi
        fi
        if [ "$2" = "plus" ]; then
                if [ "$plus" = "1" ]; then
                        echo "= Intel Select Reference HW BOM Plus - PASS || $sw_string =" >> $file
                else
                        echo "= Intel Select Reference HW BOM Plus - FAIL $failplus || $sw_string =">> $file
                fi
        fi
fi
echo "" >> $file
echo "" >> $file
echo " -------- RAW DATA CAPTURES ---------" >> $file
```

```
echo "MSR 0x620 = " >> $file
rdmsr 0x620 >> $file
echo "MSR 0x621 = " >> $file
rdmsr 0x621 >> $file
echo "" >> $file
echo "MSR 0x1a0 = " >> $file
rdmsr 0x1a0 >> $file
echo "MSR 0x13a -f 0:0 = " >> $file
rdmsr 0x13a -f 0:0 >> $file
echo "MSR 0x198 = " >> $file
rdmsr 0x198 >> $file
echo "MSR 0x199 = " >> $file
rdmsr 0x199 >> $file
echo " -------- CPU/MEM INFO ---------" >> $file
echo "" >> $file
lscpu >> $file
echo "" >> $file
cat /proc/cmdline >> $file
echo "" >> $file
cat /proc/meminfo >> $file
uname -a >> $file
echo " -------- NIC INFO ---------" >> $file
echo "" >> $file
lspci | grep Eth >> $file
echo "" >> $file
ifconfig -a >> $file
echo "" >> $file
modinfo i40e >> $file
echo " -------- LSPCI INFO ---------" >> $file
echo "" >> $file
lspci | grep Eth >> $file
echo "" >> $file
lspci | grep Non >> $file
echo "" >> $file
lspci -vv >> $file
echo "" >> $file
lspci | grep 37c8 >> $file
echo "" >> $file
lspci -vv >> $file
echo " -------- DMIDECODE INFO ---------" >> $file
echo "" >> $file
dmidecode -t bios >> $file
echo "" >> $file
dmidecode >> $file
echo "" >> $file
parted -l>> $file
echo " -------- END OF FILE ---------" >> $file
echo " -------- Complete ---------"
```

APPENDIX C QAT_SPEED.SH

```
#yum    install -y git

########################################
###
### download and build openssl 1.1.0e
###
########################################
Test_Dir=/root/ssl_speed
cd $Test_Dir
git clone https://github.com/openssl/openssl.git
cd openssl
#git tag
git checkout OpenSSL_1_1_0e
git describe --tag
./config --prefix=/usr/local/ssl
make depend
make
make install

########################################
###
### download and build QAT Engine patch
###
########################################


cd $Test_Dir/openssl
git clone https://github.com/01org/QAT_Engine.git
export OPENSSL_ENGINES=/usr/local/ssl/lib/engines-1.1
#make sure usdm_drv is installed
insmod /opt/APP/driver/QAT/quickassist/utilities/libusdm_drv/usdm_drv.ko

cd $Test_Dir/openssl/QAT_Engine
./autogen.sh
./configure \
--with-qat_dir=/opt/APP/driver/QAT \
--with-openssl_dir=$Test_Dir/openssl \
--with-openssl_install_dir=/usr/local/ssl \
--enable-upstream_driver \
--enable-usdm \
--disable-qat_lenstra_protection
make
make install


#####################################################
###
### Copy conf file and change the LimitDevAccess to 0
###
#####################################################


cp $Test_Dir/openssl/QAT_Engine/qat/config/c6xx/multi_process_optimized/*    $Test_Dir/

cd $Test_Dir
sed -e 's/LimitDevAccess = 1/LimitDevAccess = 0/g' ./c6xx_dev0.conf > /etc/c6xx_dev0.conf
echo "modify c6xx_dev0.conf"
sed -e 's/LimitDevAccess = 1/LimitDevAccess = 0/g' ./c6xx_dev1.conf > /etc/c6xx_dev1.conf
echo "modify c6xx_dev1.conf"
sed -e 's/LimitDevAccess = 1/LimitDevAccess = 0/g' ./c6xx_dev2.conf > /etc/c6xx_dev2.conf
echo "modify c6xx_dev1.conf"

##Restart qat service
echo "Restart qat service"
service qat_service restart
export LD_LIBRARY_PATH=$PATH:/usr/local/ssl/lib
```

```
cd $Test_Dir/openssl/apps
#./openssl engine -t -c -vvvv qat


echo "OpenSSL Speed test results log" >    $Test_Dir/openssl.log
echo "taskset -c 1 ./openssl speed -engine qat -async_jobs 150 -multi 1 rsa2048" >> $Test_Dir/openssl.log
taskset -c 1 ./openssl speed -engine qat -async_jobs 150 -multi 1 rsa2048 >> $Test_Dir/openssl.log
#cat openssl-rsa.log | grep "rsa " > rsa_sign.log
#awk '/rsa/ { print $6 }' rsa_sign.log


echo "taskset -c 1-4,23-26 ./openssl speed -engine qat -async_jobs 150 -evp aes-128-cbc-hmac-sha1 -multi 7"    >> $Test_Dir/openssl.log
taskset -c 1-4,23-26 ./openssl speed -engine qat -async_jobs 150 -evp aes-128-cbc-hmac-sha1 -multi 7 >> $Test_Dir/openssl.log
#cat openssl-bulk.log | grep evp > openssl2.log
#awk '/evp/ { print $7 }' openssl2.log
```

Learn More

Lanner Intel Select for uCPE: http://www.lannerinc.com/applications/sd-wan/intel-select-for-ucpe

Intel Select Solutions web page: https://builders.intel.com/intelselectsolutions

Intel Xeon D processor family: http://www.intel.com/xeond

Intel Select Solutions are supported by the Intel Builders Program: https://builders.intel.com

Note: * Intel, the Intel logo, and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.