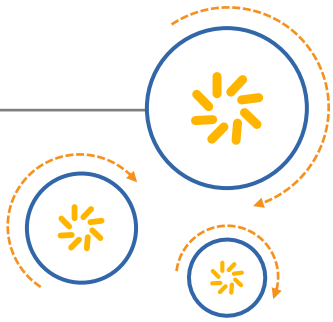
Qualcomm Technologies, Inc.



QCA_Networking_2017.SPF.5.0 CS.1

Release Notes

80-YA934-4 Rev. A

May 17, 2017

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	May 2017	Initial release

Contents

Tables

1 Introduction

This document provides details on the QCA_Networking_2017.SPF.5.0 CS.1 release.

Despite being downloaded from the Qualcomm ChipCode™ portal, the Qualcomm Atheros Support site, or embedded on Equipment received from Qualcomm Atheros, Inc. ("QCA") or its affiliates, the QCA_Networking_2017.SPF.5.0 CS.1 software release ("SW Package") shall be considered Deliverables and is subject to the terms and conditions of the Qualcomm Atheros, Inc. Limited Use Agreement ("Agreement"). The applicable Use Period, as that term is defined in the Agreement, for the SW Package starts on the Effective Date of your Agreement or the date you received the SW Package, whichever is later, and expires on May 16, 2018 (unless a different Use Period for the SW Package is specified in the Agreement, in which case the Use Period in the Agreement shall prevail). By receiving and/or using the SW Package, you acknowledge and agree that your use of the SW Package is subject to the terms and conditions of the signed Agreement. If you do not agree to the terms of the Agreement, have not signed such Agreement, or have not received the written approval from QCA or its affiliates set forth below, you shall immediately delete the SW Package from all storage media and destroy any and all copies made.

If you wish to utilize the SW Package commercially, you must obtain written approval from QCA or its affiliates, which may be obtained by contacting QCA or its affiliates by email at QCA.Contracts@qualcomm.com or via facsimile to (408) 516-5825, that the SW Package is approved for use under the terms of an applicable definitive agreement in effect between you and QCA or its affiliates or you must enter into a new definitive license agreement relating to the SW Package.

Information published by QCA or its affiliates regarding any third-party information does not constitute a license to use such information or endorsement thereof. QCA or its affiliates provides any such third party information as-is, without any representation, warranty, or indemnity, either express or implied. Use of such information may require a license from a third party under the intellectual property rights of such third party, or a license from QCA or its affiliates under the intellectual property rights of QCA or its affiliates. Users assume all risk of any use of such third party information.

1.1 Purpose

These release notes accompany the QCA_Networking_2017.SPF.5.0 CS.1 release. They describe new and changed features, download and installation procedures, and known and resolved problems in the hardware and software. It is recommended that these release notes are read in conjunction with the list of documents presented in section 1.2.

The QCA_Networking_2017.SPF.5.0 release aggregates these SPs:

- IPQ8064.ILQ.5.0
- IPQ4019.ILQ.5.0
- QCA9531.ILQ.5.0 (applies to QCA9531, QCA9558, and QCA9563 chipsets)

NOTE:

- The QCA_Networking_2017.SPF.5.0 CS.1 release interoperates with CSRmesh™. For the CSRmesh software and documentation, refer to https://www.csrsupport.com/Qwin/csrmesh. Customers that do not have a CSR™ account must register before they can access the software and documentation. Refer to https://www.csrsupport.com/register.php for details on registering an account.
- 802.11ad functionality has stability issues, especially in the multi-STA scenarios. For 802.11ad customers, an additional 802.11ad firmware image with the stability issues resolved will be released later by the relevant customer engineering team.

QCA_Networking_2017.SPF.5.0 contains SPs that relate to the version 5.0 of the 2017 Qualcomm® networking software products.

This release is:	**QCA_Networking_2017.SPF.5.0 CS.1**
The release version is:	QCA_Networking_2017.SPF.5.0.r1-00008-STD.PROD-1
The Linux Foundation hosted open source label (the CAF_TAG) that corresponds to this release is:	caf_AU_LINUX_QSDK_RELEASE_ENDIVE_CC_TARGET_ALL.0.1.719.031.xml caf_AU_LINUX_QSDK_RELEASE_ENDIVE_MIPS_AA_TARGET_ALL.0.2.3173.029.xml
Qualcomm ChipCode™ distribution tag (Use this tag to check out the code from git repository)	r00008.1

This release contains two different CAF tags intended for different SoCs:

AU_LINUX_QSDK_RELEASE_ENDIVE_CC_TARGET_ALL	This code is based on upstream OpenWRT version 15.05.1 (codename Chaos Calmer) with Linux kernel 3.14.77
AU_LINUX_QSDK_RELEASE_ENDIVE_MIPS_AA_TARGET_ALL	This code is based on upstream OpenWRT version 12.09 (codename Attitude Adjustment) with Linux kernel 3.3.8

1.2 Related documentation

Refer to the *QCA_NETWORKING_2017-SPF.5.x Related Documents for Reference* (80-YA760-2) for a list of all documents to refer for more information on using this release.

1.3 Changes in release packages

These changes have been made in recent release packages, starting with the QCA_Networking_2016.SPF.3.0 release:

1. Release packages are now located in individual feed directories.

   Previously, Qualcomm Technologies release packages were located in an output directory such as bin/ipq806x/packages/. They are now located in their own feed directory; automation scripts and manual testers must use the respective feed's directory.

2. Reset_to_factory_settings.sh has been removed. The equivalent upstream command is: **jffs2reset -y; reboo0074**

3. Some utilities used in previous releases are no longer included:

No longer included:	Use this instead for this release:
devmem2	devmem (from busybox)
mtd-utils	ubi-utils
radvd	dnsmasq
pure-ftpd	ftpd (from busybox)
tftp-hpa	tftpd (loadable ipk from busybox)

4. Some packages are backported from upstream Barrier-Barrier code:

   Upstream Chaos-Chalmer has removed support (e.g., multiwan, rp-pppoe, isc-dhcp)

Since these changes were introduced in QCA_Networking_2016.SPF.3.0 release, only customers that previously used QCA_Networking_2016.SPF.2.0 release will observe these changes. For customers that previously used QCA_Networking_2016.SPF.3.0 release, no changes in release packages are observed.

2 IPQ8064.ILQ.5.0

This chapter describes details regarding the IPQ8064.ILQ.5.0 SP that is part of this SPF package.

2.1 Supported hardware

This section describes the hardware boards that are compatible with the SP.

Hardware board	Comments
AP161.1	IPQ8069 design for 3 radios
AP160.1	IPQ8069 design for 3 radios (Enterprise)
AP160.2	IPQ8069 design for 3 radios + 2.5 G Ethernet (Enterprise)
CUS239.5	QCA9994-based 802.11ac 5 GHz; 4x4 configuration + Qorvo 5 GHz (Qualcomm® Connected SmartHome)
CUS239.7	QCA9994-based 802.11ac 5 GHz; 4x4 configuration + Qorvo 5 GHz (Enterprise)
CUS260.5	QCA9994-based 802.11n 2.4 GHz; 4x4 configuration + Skyworks 2.4 GHz (Connected SmartHome)
CUS260.7	QCA9994-based 802.11n 2.4 GHz ; 4x4 configuration + Skyworks 2.4 GHz (Enterprise)
CS.CAS01.1	QCA9994-based 802.11ac 5 GHz; 4x4 configuration + Qorvo 5 GHz (lower band)
CS.CAS01.3	QCA9994-based 802.11ac 5 GHz; 4x4 configuration + Qorvo 5 GHz (higher band)
CUS240.7	QCA9994-based 802.11ac 2.4 GHz 4x4 configuration + QFE1922 2.4 GHz (Enterprise)
CUS238.7	QCA9994-based 802.11ac 5 GHz 4x4 configuration + QFE1952 5 GHz (Enterprise)
XB242.2	QCA9889 based 802.11n 2.4GHz/5GHz scan radio (Enterprise)
AP148.3	IPQ8068 for Skyworks (Connected SmartHome)
AP148,5	IPQ8068 for Skyworks (Enterprise)
AP148.6	IPQ8068 for CUS238/CUS240 (Connected SmartHome)
AP148.7	IPQ8068 for CUS238/CUS240 (Enterprise)
CUS239.1	QCA9990-based 802.11n 5 GHz; 4x4 configuration + Skyworks 5 GHz (Connected SmartHome)
CUS239.2	QCA9990-based 802.11n 5 GHz; 3x3 configuration + Skyworks 5 GHz (Connected SmartHome)
CUS239.3	QCA9990-based 802.11n 5 GHz; 4x4 configuration + Skyworks 5 GHz (Enterprise)
CUS239.4	QCA9990-based 802.11n 5 GHz; 3x3 configuration + Skyworks 5 GHz (Enterprise)
CUS260.1	QCA9990-based 802.11n 2 GHz; 4x4 configuration + Skyworks 2.4 GHz (Connected SmartHome)
CUS260.2	QCA9990-based 802.11n 2 GHz; 3x3 configuration + Skyworks 2.4 GHz (Connected SmartHome)
CUS260.3	QCA9990-based 802.11n 2 GHz; 4x4 configuration + Skyworks 2.4 GHz (Enterprise)
CUS260.4	QCA9990-based 802.11n 2 GHz; 3x3 configuration + Skyworks 2.4 GHz (Enterprise)
CS.BL01.1	QCA9992-based 802.11n 2 GHz; 3x3 configuration + Skyworks 2.4 GHz (Enterprise)
CS.BL01.1	QCA9992-based 802.11n 2 GHz; 2x2 configuration + Skyworks 2.4 GHz (Enterprise
CUS238.1	QCA9990 4x4 configuration + QFE1952-based 802.11n 5 GHz (Connected SmartHome)
CUS238.3	QCA9990 4x4 configuration + QFE1952-based 802.11n 5 GHz (Enterprise)

Hardware board	Comments
CUS240.1	QCA9990 4x4 configuration + QFE1952-based 802.11n 2.4 GHz (Connected SmartHome)
CUS240.3	QCA9990 4x4 configuration + QFE1952-based 802.11n 2.4 GHz (Enterprise)
AP160.3	IPQ8069 design for 3 radios + 802.11ad support
QCA6320.M2.SP03	QCA6320 based 802.11ad WIGIG module
XB112.2	AR9380 802.11n-based 2.4 GHz 3x3 configuration (Connected SmartHome)
RDP0317	IPQ8069.AP161.2.QCA9994.CUS238.7.QCA9994.CUS240.8.QCA9889.XB242.2.RDP IPQ8069 AP161 (AK 3.0 3 PCIe Ent 4 layer) + CAS (CUS238 (4x4 5GHz, MP) + CUS240 (3x3 2.4GHz MP) + QCA9889 XB242(1x1 2.4/5GHz 11ac DB iPA PCIe)
RDP0330	IPQ8065.AP161.1.QCA9984.CUS238.5.QCA9985.CUS240.9.RDP
RDP0329	IPQ8065.AP161.1.AR9287.HB97.1.QCA9886.XB.BSR01.1.RDP

NOTE: 802.11ad functionality has stability issues, especially in the multi-STA scenarios. For 802.11ad customers, an additional 802.11ad firmware image with the stability issues resolved will be released later by the relevant customer engineering team.

2.2 Restrictions

Table 2-1 lists the restrictions on the software while using it for testing.

NOTE: 802.11ad functionality has stability issues, especially in the multi-STA scenarios. For 802.11ad customers, an additional 802.11ad firmware image will be released later by the relevant customer engineering team.

Table 2-1 Restrictions on the software

Channel Conditions	**For peak performance results, the AP, STA, and channel conditions must be configured to:**		
	Attribute	**Value**	
	Image	See Section 1.1 for the release version	
	AP	AP160 + CUS239	AP160 + CUS260
	STA	AP160 + CUS239 in WDS mode	AP160 + CUS260 in WDS mode
	RSSI of AP at STA	0x43	0x40
	RSSI of STA at AP	0x43	0x40
	Attenuation	32 dB per chain	40 dB per chain

NSS	▪ The software is designed to work with the default configuration. Thus software configures GMAC0/GMAC1 to operate only in 1000 M mode. If the board is reconfigured to connect GMAC0/GMAC1 to a PHY instead of the switch, then software must be altered to allow speed-switching to lower Ethernet speeds. The PHY ID mapping used by the software must also be changed. ▪ Only IPv4/IPv6 + TCP/UDP flows are accelerated through fast path ▪ LAG: The Linux bonding driver configuration is supported in fast path through the bonding driver sysfs interface: ▫ Mode: 802.3ad (4) – xmit_hash_policy: layer2 (0), layer2+3 (2) ▫ Mode: Balance-xor (2) – xmit_hash_policy: layer2 (0), layer3+4 (1), layer2+3 (2) ▪ Flows that require ALG support are not accelerated by NSS in this release (except for TFTP (port 69)) ▪ Interfaces configured with NSS qdiscs must have a leaf node configured as the default node for enqueue (using the 'set_default' qdisc parameter). If qdisc structures are created without a default, no packets are transmitted; this includes management packets such as ARP. ▪ IGMP/MLD snooping for QCA8337N does not support MLDv1 completely. Qualcomm Technologies provides a software implementation for IGMP/MLD snooping ▪ IPsec does not work with fragmentation ▪ Tunnel support (6RD, DS-Lite) expects at least the initial few packets in the LAN > WAN direction to push the base fast path rule. Further flows for these tunnels can be initiated on either the WAN or LAN network. ▪ It is recommended that link detection be disabled for eth0 in case eth0 is used as part of static LAG bond group between the IPQ806x chip and QCA8337 switch chip. Link detection can be controlled with ethtool commands: ▫ ethtool --set-priv-flags eth0 linkpoll off ▫ To check current status: ethtool --show-priv-flags eth0
802.11ad Channels	▪ Performance on 802.11ad channel 1 is limited by PHY performance. Refer to the *QCA6320 RFIC Chip Device Revision Guide* (80-Y9072-4) for additional information. 802.11ad performance testing must be conducted on 60 GHz channels 2 or 3.
SQI and RSSI reporting	▪ On STA mode, the 802.11ad driver provides the detected signal quality (SQI) in response to a BSS scan or Link Quality status query. The SQI returns a value of 0, 20, 40, 60, or 100 which can be used to drive a 5-level 'bars' icon. Actual RSSI reporting is not supported; the return value is always -44. RSSI reporting will be supported in future driver update.
WPS	▪ The 802.11ad driver is not compatible with the implementation of Wi-Fi Protected Setup (WPS) in Windows. The GCMP encryption method used for 802.11ad is not supported by the Windows User Interface including the connection manager, when communicating with NDIS 6.3 drivers as used in Windows 7, 8.1, and 10. ▪ Qualcomm provides an Independent Hardware Vendor (IHV) mechanism which bypasses the Windows connection manager (and internal supplicant) and allows alternative security negotiation. ▪ The IHV mechanism enables a secured connection, but the Windows APIs provided supporting WPS do not work properly. Therefore WPS operation for establishing a secure connection is not functional. ▪ Qualcomm is pursuing this issue with Microsoft.
ETSI certification test	▪ Enable CCA threshold using the iwpriv command to successfully clear the ETSI certification check for Direct Attach radio.

DBDC repeater and TBTC	▪ Dynamically changing primary radio configuration and `alwaysprimary` configuration during runtime is not supported.
	▪ If the platform is tri-radio and has NSS Wi-Fi offload support, for TBTC repeater feature to work, NSS Wi-Fi offload must be disabled using UCI commands. Enter the following UCI configuration to disable NSS Wi-Fi offload mode.
	`uci set wireless.qcawifi=qcawifi`
	`uci set wireless.qcawifi.nss_wifi_olcfg=0`
	`uci commit`
	▪ By default, wifi0 is set as primary radio. On tri-radio board, if wifi0 is disabled, and if wif1 and wifi2 are configured in DBDC Repeater mode, then user must explicitly set wifi1 or wifi2 as primary radio.

2.3 Power management

The following power management features are enabled by default in this build:

Qualcomm® Krait™ frequency and voltage scaling on both cores	Krait frequency and voltage are scaled up and down based on CPU load. ▪ Performance governor is turned ON during the boot process ▪ On-demand governor is turned ON post boot ▪ Voltage/frequency settings used vary by process variations ▪ Values for fast, typical, and slow parts are taken from the Krait PVS tables ▪ L2 frequencies and voltages are scaled in synch with the Krait frequencies and voltages; they also are selected based on process variations ▪ The lowest frequency Krait0 is allowed to go to is 800 MHz (which solves a problem on some test cases where the power management code does not get enough bandwidth to pull itself from the low frequency mode); this limitation has little impact on power dissipation in the idle states
Krait clock gating is enabled	Software detects the inactivity in parts of the Krait and shuts down the clock to these parts accordingly
Frequency and voltage scaling of the multi-threaded network accelerator engines processors (Ubi32)	Core frequencies are scaled up and down based on load between 110 and 800 MHz; voltage is scaled between 1.05 and 1.15 V
Frequency scaling of the FABRICs	FABRICS frequencies are scaled up and down based on the load detected by the CPU. Only the APPS and NSS FABRICs are scaled. The rest of the FABRICS are left at their nominal frequency settings set at boot. VDD_CX rail to which the FABRICs are connected is scaled between 1.1 and 1.15 V accordingly. Details of the frequency values used in the scaling for each of the FABRICs are provided in the *IPQ806x Power Management Application Note*.

- VDD_CX and multi-threaded network accelerator engines processors voltage rails are combined on the AP reference design

Many power management features can be enabled and disabled with user commands (see the *IPQ806x Power Management Application Note (*80-Y6477-2)). In particular, the following guidelines apply:

- Auto scaling can be turned ON and OFF on multi-threaded network accelerator engines processors and Kraits by user commands. All features are turned ON by default.
- Details on power management in general, including frequencies and voltages used in the scaling, and information on user commands to enable and disable power management features and power configuration settings and default values.

For performance-related tests such as RFC2544, Qualcomm Technologies recommends disabling auto scaling. While several improvements have been made in auto-scaling algorithms, the RFC2544 0% stipulation is still quite demanding, especially when starting traffic burst. To disable auto scaling, execute these commands at the prompt:

echo "userspace" > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
echo "userspace" > /sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
echo "1725000" > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
echo "1725000" > /sys/devices/system/cpu/cpu1/cpufreq/scaling_setspeed
echo 0 > /proc/sys/dev/nss/clock/auto_scale
echo 800000000 > /proc/sys/dev/nss/clock/current_freq

NOTE: 802.11ad functionality has stability issues, especially in the multi-STA scenarios. For 802.11ad customers, an additional 802.11ad firmware image will be released later by the relevant customer engineering team.

The addition of the 60 GHz 802.11ad radio to the AP can add a considerable load on the IPQ806x processor. CPU interrupt affinity can be used to assign 11ad interrupt processing to a CPU core to improve load balancing. To do this, the customer must identify the IRQ number used by the 802.11ad driver. The 802.11ad interrupt is mapped to one of the PCIE interrupts, and 802.11ad uses the PCIE Message Signaled Interrupt (MSI) mechanism. The IRQ number can be identified as follows:

1. Run standalone 802.11ad traffic
2. Issue the following command a couple of times:

   cat /proc/interrupts | grep pcie

   The counter of the relevant interrupt increments in the above, which must identify the relevant interrupt number. For example, to set interrupt affinity for 802.11ad to core 1 with IRQ number 67 on the SR99 reference design, use the following command:

   echo 2 > /proc/irq/67/smp_affinity

   The addition of the 60 GHz 802.11ad radio to the access point can add a considerable load on the IPQ8065 processor. CPU interrupt affinity can be used to assign 802.11ad interrupt processing to a CPU core to improve load balancing. To perform this task, the customer must identify the IRQ number used by the 11ad driver. Identify the IRQ number by issuing the following command:

   cat /proc/interrupts | grep wil6210

   For example, to set interrupt affinity for 802.11ad to core 1 with IRQ number 90 on the IPQ8065.AP161.1.QCA9984.CUS238.5...CUS240.5.QCA6320.M2.SP03.2 reference design, use the command:

   echo 2 > /proc/irq/90/smp_affinity

2.4 Build and load the image for IPQ8064.ILQ.5.0

This product includes software developed by the University of California, Berkeley and its contributors.

An SPF and its custom branches are created and managed in CreatePoint in the same manner as a single SP distribution. A single git repository is created as a publicly available open source software version control system. This git repository:

- Is used to record the release history for the entire codebase of a corresponding SPF
- Adds each SPF distribution with its own custom branch and manages each as a new snapshot of the SPF codebase

An SPF distribution contains the same SP and SI build command files contained in the individual SP distributions. The primary difference the SPF includes more than one instance of the same type of SI and can include more than one SP.

The file set for each software product (SP) is contained in a directory that has the same name as the product (such as IPQ8064.ILQ.5.0) and consists of the following files:

- Binary files to program into flash memory
- Build files that generate the binary files
- Command files that program the binary files into flash memory

Released SP images are available for download from the ChipCode portal for proprietary-based code. For open source code, obtain the files from the Linux Foundation at the Code Aurora Forum.

To build and load the SP image on the device, do the following:

1. Download the Qualcomm Technologies proprietary code from ChipCode (see section 2.4.1).
2. Download other components from external websites by QSDK while building the default configuration (see section 2.4.2).
3. Generate the firmware by doing the following:
   a. Reassemble the code (see section 2.4.3.1).
   b. Create the QSDK build (see section 2.4.3.2).
   c. Build a complete firmware image (see section 2.4.3.3).
4. Install the image in the flash memory of the device and boot using the image from the flash (see section 2.4.4).

It is recommended that you be familiar with the structure of directories that contain the SP images for the different subsystems before you download the code and build the images for loading. For more information, refer to the *QCA_Networking_2017.SPF.5.0 Product Family Overview* (80-YA935-1). For each SP included in an SPF, SP binary files are generated from the SI binary files of only a subset of the included SIs. In an SPF, some SIs may support multiple SPs while others may only support one SP.

Starting with the QCA_Networking_2016.SPF.4.0 CS release, all WiGig components are not a part of the APPS image and are distributed as loadable installable packages (ipks). Install them manually after the image is loaded from flash memory. In addition, wilserver and LogCollector are integrated to the Dandelion distribution.

2.4.1 Download packages available through ChipCode

Qualcomm Technologies proprietary code is available from ChipCode.

A web/GUI interface and a secure git server both allow access to this code. Browse available packages and obtain the download URL at https://chipcode.qti.qualcomm.com/ (see https://chipcode.qti.qualcomm.com/helpki/cloning-code-from-a-repository for more information).

See https://chipcode.qti.qualcomm.com/helpki for more information on installation and configuration of the correct version of git and OpenSSL on both Windows and Linux platforms that is required to support the authentication methods used by ChipCode.

2.4.2 Download packages from external websites

These components are downloaded by QSDK while building the default configuration for the profiles, QSDK may be further customized to download additional components; this table lists only the components that are necessary for at least one of the QSDK 2.0 default profiles. This list does not include the packages obtained from ChipCode as described in section 2.4.1.

Table 2-2 Packages available from external sites

Package
1.0.4.3.arm
LuaSrcDiet-0.12.1.tar.bz2
MPlayer-1.1.1.tar.xz
Python-2.7.9.tar.xz
alsa-lib-1.0.28.tar.bz2
alsa-utils-1.0.28.tar.bz2
argp-standalone-1.3.tar.gz
arptables-v0.0.4.tar.gz
attr-20150220.tar.gz
autoconf-2.69.tar.xz
automake-1.15.tar.xz
backports-20160121.tar.bz2
bc-1.06.95.tar.bz2
binutils-2.24.tar.bz2
binutils-linaro-2.24.0-2014.09.tar.xz
bison-3.0.2.tar.xz
bluez-5.30.tar.xz
bridge-utils-1.5.tar.gz
busybox-1.23.2.tar.bz2
bzip2-1.0.6.tar.gz
cmake-2.8.12.2.tar.gz
coccinelle-1.0.0-rc24.tar.gz
curl-7.40.0.tar.bz2
db-4.7.25.NC.tar.gz
dbus-1.9.14.tar.gz
dnsmasq-2.73.tar.xz
dosfstools-3.0.28.tar.gz
dropbear-2015.67.tar.bz2
e2fsprogs-1.42.12.tar.gz
e2fsprogs-1.42.8.tar.gz
elfutils-0.161.tar.bz2
ethtool-3.18.tar.xz
expat-2.1.0.tar.gz
fcgi-2.4.0.tar.gz
ffmpeg-2.6.2.tar.bz2
file-5.25.tar.gz
findutils-4.4.2.tar.gz
firewall-2015-07-27-980b7859bbd1db1e5e46422fccccbce38f9809ab.tar.gz
flex-2.5.39.tar.bz2
fstools-2016-01-10-96415afecef35766332067f4205ef3b2c7561d21.tar.gz
gcc-linaro-4.8-2014.04.tar.xz
gdb-linaro-7.6-2013.05.tar.bz2
gdbm-1.11.tar.gz

Package
gengetopt-2.22.6.tar.gz
glib-2.44.1.tar.xz
gmp-5.1.3.tar.xz
gmp-6.0.0a.tar.xz
i2c-tools-3.1.2.tar.bz2
iozone3_420.tar
iperf-2.0.5.tar.gz
iproute2-4.0.0.tar.xz
iptables-1.4.21.tar.bz2
iputils-s20101006.tar.bz2
iw-4.3.tar.xz
jansson-2.7.tar.gz
json-c-0.12.tar.gz
jsonfilter-2014-06-19-cdc760c58077f44fc40adbbe41e1556a67c1b9a9.tar.gz
libelf-0.8.13.tar.gz
libffi-3.0.13.tar.gz
libgcrypt-1.6.1.tar.bz2
libgpg-error-1.12.tar.bz2
libical-1.0.tar.gz
libiwinfo-2015-06-01-ade8b1b299cbd5748db1acf80dd3e9f567938371.tar.gz
libmad-0.15.1b.tar.gz
libmnl-1.0.3.tar.bz2
libnetfilter_conntrack-1.0.4.tar.bz2
libnfnetlink-1.0.1.tar.bz2
libnl-3.2.21.tar.gz
libogg-1.3.2.tar.xz
libpcap-1.5.3.tar.gz
libtheora-1.1.1.tar.bz2
libtool-2.4.tar.gz
libubox-2015-11-08-10429bccd0dc5d204635e110a7a8fae7b80d16cb.tar.gz
libvorbis-1.3.5.tar.xz
libxml2-2.9.2.tar.gz
linux-atm-2.5.2.tar.gz
linux-firmware-17657c3.tar.bz2
linux-ramdump-parser-v2-2008-12-18.tar.bz2
lua-5.1.5.tar.gz
lzma-4.65.tar.bz2
lzo-2.08.tar.gz
m4-1.4.17.tar.xz
make-ext4fs-2015-05-01.tar.gz
mbedtls-1.3.15-gpl.tgz
mcproxy-2014-12-31-b7bd2d0809a0d1f177181c361b9a6c83e193b79a.tar.bz2

Package
mdadm-3.2.5.tar.xz
minicom-2.7.tar.gz
miniupnpd-1.9.20150609.tar.gz
mklibs_0.1.35.tar.gz
mm-common-0.9.7.tar.xz
mpc-1.0.2.tar.gz
mpfr-3.1.2.tar.bz2
mtd-utils-1.5.1-92686f212c9a4e16891c6a3c57629cbf4f0f8360.tar.gz
nat46-6.tar.xz
ncurses-5.9.tar.gz
netifd-2015-12-16-245527193e90906451be35c2b8e972b8712ea6ab.tar.gz
ntfs-3g_ntfsprogs-2014.2.15.tgz
odhcp6c-2015-07-29.tar.bz2
odhcpd-2015-11-19.tar.bz2
opencore-amr-0.1.3.tar.gz
openssl-1.0.2g.tar.gz
openswan-2.6.41.tar.gz
opkg-9c97d5ecd795709c8584e972bfdf3aee3a5b846d.tar.gz
opus-1.1.tar.gz
patch-2.7.5.tar.xz
patchelf-0.8.tar.bz2
perl-5.20.2.tar.gz
pkg-config-0.29.tar.gz
pm-utils-1.4.1.tar.gz
ppp-2.4.7.tar.gz
procd-2015-10-29.1-d5fddd91b966424bb63e943e789704d52382cc18.tar.gz
quagga-0.99.22.4.tar.xz
quilt-0.63.tar.gz
readline-6.3.tar.gz
rng-tools-5.tar.gz
rp-pppoe-3.11.tar.gz
readline-6.2.tar.gz
rpcd-2016-04-13-73aea9b8b621a1ce034bc6ee00c9d058a40c8a3d.tar.gz

Package
rstp-2011-10-11-434d24bae108dbb21461a13a4abcf014afa8b029.tar.gz
rtl8712u.bin
samba-3.6.25.tar.gz
scons-2.3.1.tar.gz
sed-4.2.2.tar.bz2
sigma-dut-2016-01-29-410fccab4c10c452d78023479f0db8301cb883fd.tar.gz
speex-1.2rc1.tar.gz
sqlite-autoconf-3081101.tar.gz
squashfs4.2.tar.gz
sysfsutils-2.1.0.tar.gz
sysstat-11.0.4.tar.xz
tcpdump-4.5.1.tar.gz
trace-cmd-v2.4.2.tar.gz
u-boot-2014.10.tar.bz2
uClibc++-0.2.4.tar.bz2
uClibc-0.9.33.2.tar.bz2
ubi-utils-1.5.1.tar.gz
ubox-2015-11-22-c086167a0154745c677f8730a336ea9cf7d71031.tar.gz
ubus-2015-05-25-f361bfa5fcb2daadf3b160583ce665024f8d108e.tar.gz
uci-2015-08-27.1.tar.gz
uhttpd-2015-11-08-fe01ef3f52adae9da38ef47926cd50974af5d6b7.tar.gz
usign-2015-05-08-cf8dcdb8a4e874c77f3e9a8e9b643e8c17b19131.tar.gz
ustream-ssl-2015-07-09-c2d73c22618e8ee444e8d346695eca908ecb72d3.tar.gz
util-linux-2.25.2.tar.xz
wireless_tools.29.tar.gz
xl2tpd-devel-20150930.tar.gz
xtables-addons-2.5.tar.xz
xz-5.2.1.tar.bz2
yaffs2_android-2008-12-18.tar.bz2
zlib-1.2.8.tar.gz

2.4.3 Generate the firmware for IPQ8064.ILQ.5.0

To generate a firmware image, reassemble the code, create the QSDK build, and create a complete firmware image. This section describes the procedure to generate the firmware.

2.4.3.1 Reassemble the code

The first step is to reassemble the code from ChipCode and the Linux Foundation and generate the QSDK framework. The example illustrated in this section assumes that all packages listed in sections 2.4.1 and 2.4.2 are obtained using the **git clone** command and placed in the top-level directory:

1. Enter the following commands to reassemble the code and generate the QSDK framework:

```
$ git clone <chipcode-distro>
$ cd <chipcode directory>
$ git checkout r00008.1
```

2. After the copy of the existing Git repository is completed, the directories in which the files are present must be changed as described in the following table before the repo command is run.

All Packages:	Use git to obtain the following files from ChipCode and copy them to the working QSDK top-level directory of the device:	Local directory path to files fetched by git from ChipCode:
	qsdk-qca-wifi qsdk-qca-wlan qsdk-ieee1905-security qsdk-qca-athdiag	NHSS.QSDK.5.0\apss_proc\out\proprietary\Wifi
	qca-lib qca-mcs-apps qsdk-qca-nss qca-nss-userspace.tar.bz2	NHSS.QSDK.5.0\apss_proc\out\proprietary\QSDK-Base
	qca-bluetopia.tar.bz2	NHSS.QSDK.5.0\apss_proc\out\proprietary\BLUETOPIA
	qca-wifi-fw-QCA9984_hw_1-WLAN.BL.3.5-00010-S-1.tar.bz2 qca-wifi-fw-AR900B_hw_2-WLAN.BL.3.5-00010-S-1.tar.bz2	WLAN.BL.3.5\cnss_proc\bin\hw.1 WLAN.BL.3.5\cnss_proc\bin\hw.2
	qca-wifi-fw-src-component-cmn-WLAN.BL.3.5-00010-S-1.tgz qca-wifi-fw-src-component-halphy_tools-WLAN.BL.3.5-00010-S-1.tgz	WLAN.BL.3.5\cnss_proc\src\components
	qca-wifi-fw-AR9887_hw_1-CNSS.PS.2.5-00006-S-1.tar.bz2 qca-wifi-fw-AR9888_hw_2-CNSS.PS.2.5-00006-S-1.tar.bz2	CNSS.PS.2.5

3. After copying the necessary files to the appropriate directories, enter the following commands to continue with the process of generating the QSDK framework:

```
$ repo init -u git://codeaurora.org/quic/qsdk/releases/manifest/qstak -b release
   -m caf_AU_LINUX_QSDK_RELEASE_ENDIVE_CC_TARGET_ALL.0.1.719.031.xml --repo-
   url=git://codeaurora.org/tools/repo.git --repo-branch=caf-stable
$ repo sync
$ mkdir -p qsdk/dl
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-qca-wifi/* qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-qca-wlan/* qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-ieee1905-security/*
   qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-qca-athdiag/* qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/QSDK-Base/qca-lib/* qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/QSDK-Base/qca-mcs-apps/* qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/QSDK-Base/qsdk-qca-nss/* qsdk
$ tar xjvf NHSS.QSDK.5.0/apss_proc/out/proprietary/QSDK-Base/qca-nss-
   userspace.tar.bz2 -C qsdk
```

```
$ tar xjvf NHSS.QSDK.5.0/apss_proc/out/proprietary/BLUETOPIA/qca-
   bluetopia.tar.bz2 -C qsdk
$ cp WLAN.BL.3.5/cnss_proc/bin/hw.1/* qsdk/dl
$ cp WLAN.BL.3.5/cnss_proc/bin/hw.2/* qsdk/dl
$ cp WLAN.BL.3.5/cnss_proc/src/components/* qsdk/dl
$ cp CNSS.PS.2.5/* qsdk/dl
```

For enterprise customers:

```
$ cp NHSS.QSDK.5.0/apss proc/out/proprietary/RBIN-NSS-ENTERPRISE/* qsdk/dl
```

For premium customers:

```
$ cp NHSS.QSDK.5.0/apss proc/out/proprietary/RBIN-NSS-RETAIL/* qsdk/dl
```

	Customers with Qualcomm® HY-FI™, WHC, WAPid, or WiGig packages: These files are fetched from ChipCode and copied to the working QSDK top-level directory (Applicable for Premium profiles only):	
HY-FI Customers:	hyfi-ipq	NHSS.QSDK.5.0\apss_proc\out\proprietary\Hyfi
WHC Customers:	qsdk-whc qsdk-whcpy	NHSS.QSDK.5.0\apss_proc\out\proprietary\Wifi
WAPid Customers:	qsdk-wapid	NHSS.QSDK.5.0\apss_proc\out\proprietary\Wapid
WiGig Customers:	qsdk-qca-wigig qsdk-wigig-utils	NHSS.QSDK.5.0\apss_proc\out\proprietary\Wigig

	HY-FI, WHC, WAPid, or WiGig packages customers: Run the additional code (Applicable for Premium profiles only):
HY-FI:	`$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Hyfi/hyfi-ipq/* qsdk`
WHC:	`$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-whc/* qsdk` `$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-whcpy/* qsdk`

WAPid:	`$ git clone <Wapid Chipcode link>`
	`$ cd <Wapid Chipcode directory>`
	`$ git checkout r00008.1`
	`$ cd ..`
	`$ cp -rf <Wapid Chipcode directory>/NHSS.QSDK.5.0/apss_proc/out/proprietary/Wapid/qsdk-wapid/* qsdk`

WiGig:	`$ git clone <Wigig Chipcode Link>`
	`$ cd <Wigig Chipcode directory>`
	`$ git checkout r00008.1`
	`$ cd ..`
	`$ cp -rf <Wigig Chipcode directory>/NHSS.QSDK.5.0/apss_proc/out/proprietary/Wigig/qsdk-qca-wigig/* qsdk`
	`$ cp -rf <Wigig Chipcode directory>/NHSS.QSDK.5.0/apss_proc/out/proprietary/Wigig/qsdk-wigig-utils/* qsdk`
	`$ rm -rf qsdk/qca/feeds/wigig-utils/tools`
	`$ rm -rf qsdk/qca/feeds/wigig-utils/libwigigaoa`

2.4.3.2 Create the QSDK build

The QSDK framework has been developed using Ubuntu (from version 12.04 to version 16.04), and Debian. However, QSDK framework regenerates critical tools required to compile firmware at build-time. In that sense, the framework is independent from the host environment; although it is developed using the distributions above, it is expected to work on others such as RedHat, Mint, or Fedora.

This command is for Debian/Ubuntu; it must be customized for other distributions:

```
$ sudo apt-get install gcc g++ binutils patch bzip2 flex make gettext \
  pkg-config unzip zlib1g-dev libc6-dev subversion libncurses5-dev gawk \
  sharutils curl libxml-parser-perl ocaml-nox ocaml-nox ocaml ocaml-findlib \
  libpcre3-dev binutils-gold python-yaml
```

Because the framework automatically downloads the open source components, make sure an internet connection is active on the build host while creating the build.

To create the QSDK build, enter the following commands:

1. Install the different feeds in the build framework:
   ```
   $ cd qsdk
   $ ./scripts/feeds update -a
   $ ./scripts/feeds install -a -f
   ```
2. Copy the base configuration to use for the build. Choose either the standard or enterprise profile.

Premium (and Wi-Fi Base)	`$ cp qca/configs/qsdk/ipq806x_premium.config .config`
Enterprise	`$ cp qca/configs/qsdk/ipq806x_enterprise.config .config`

4. Regenerate a complete configuration file and start the build:

Non Wigig Customers Only	`$ make defconfig`
	`$ sed -i -e '/wigig-firmware/d' .config`
	`$ sed -i -e '/kmod-wil6210/d' .config`
	`$ sed -i -e "/CONFIG_PACKAGE_qca-wifi-fw-hw5-10.4-asic/d" .config`
	`$ make V=s`

Wigig Customers Only	`$ make defconfig`
	`$ sed 's/CONFIG_PACKAGE_iwinfo=m/CONFIG_PACKAGE_iwinfo=y/g' -i .config`
	`$ sed 's/CONFIG_PACKAGE_kmod-wil6210=m/CONFIG_PACKAGE_kmod-wil6210=y/g' -i .config`
	`$ sed 's/CONFIG_PACKAGE_wigig-firmware=m/CONFIG_PACKAGE_wigig-firmware=y/g' -i .config`
	`$ sed 's/CONFIG_PACKAGE_kmod-cfg80211=m/CONFIG_PACKAGE_kmod-cfg80211=y/g' -i .config`
	`$ sed 's/CONFIG_PACKAGE_iw=m/CONFIG_PACKAGE_iw=y/g' -i .config`
	`$ sed -i -e "/CONFIG_PACKAGE_qca-wifi-fw-hw5-10.4-asic/d" .config`
	`$ sed 's/CONFIG_PACKAGE_qca-wigig-debug-tools=m/CONFIG_PACKAGE_qca-wigig-debug-tools=y/g' -i .config`
	`$ $ sed 's/CONFIG_PACKAGE_qca-fst-manager=m/CONFIG_PACKAGE_qca-fst-manager=y/g' -i .config`
	`$ make V=s`

These instructions download the packages required for the corresponding profile and create the image. Once the build is complete, these files must be available in the **qsdk/bin/ipq806x** directory:

- openwrt-ipq806x-u-boot.elf (Bootloader)
- openwrt-ipq806x-qcom-ipq8064-${board_name}-fit-uImage.itb (Kernel + dtb)
- openwrt-ipq806x-squashfs-root.img (SquashFS)
- openwrt-ipq806x-qcom-ipq8064-${board_name}-ubi-root.img (UBIFS)

  Here, the board_name in the preceding image names can be any board (for example, ap148) from ipq806x family.

2.4.3.3 Generate a complete firmware image

IPQ806x requires multiple images to be flashed for Bootup, including SBL1, SBL2, SBL3, RPM, TZ, CDT, MIBIB, NSS Images, Kernel, Filesystem, etc. To simplify the loading of the device using the usage from flash memory, the images are combined into a single Flattened Image Tree (FIT) image, which can

be flashed into the respective partition based on user configuration. Additional tools required on the Ubuntu 12.04 64-bit machine are:

1. Install mkimage:
   ```
   sudo apt-get install uboot-mkimage
   ```
2. Install DTC:
   ```
   sudo apt-get install device-tree-compiler
   ```
3. Install Python 2.7
4. Switch to the Qualcomm ChipCode directory:
   ```
   $ cd <chipcode directory>
   ```
5. Copy the flash config files to **common/build/ipq**:

Premium and Enterprise

```
$ cp meta-scripts/ipq806x_standard/* IPQ8064.ILQ.5.0/common/build/ipq
```

6. Copy pack.py to the NHSS.QSDK.5.0/apss_proc/out/meta-scripts/ directory:
   ```
   $ mkdir -p NHSS.QSDK.5.0/apss_proc/out/meta-scripts
   $ cp qsdk/qca/src/u-boot/tools/pack.py NHSS.QSDK.5.0/apss_proc/out/meta-scripts/
   ```
7. Modify the SI path in contents.xml
   ```
   $ sed -i 's#</linux_root_path>#/</linux_root_path>#' IPQ8064.ILQ.5.0/contents.xml
   $ sed -i 's#</windows_root_path>#\\</windows_root_path>#'
      IPQ8064.ILQ.5.0/contents.xml
   ```
8. Copy the **openwrt*** images built to the **IPQ8064.ILQ.5.0/common/build/ipq** folder and run these commands to create a single image:
   ```
   $ cp qsdk/bin/ipq806x/openwrt* IPQ8064.ILQ.5.0/common/build/ipq
   $ mkdir QDART.WIGIG_DmTools.1.0
   $ cd IPQ8064.ILQ.5.0/common/build
   $ sed '/debug/d' -i update_common_info.py
   $ sed '/gcc/d' -i update_common_info.py
   $ sed '/allconf/d' -i update_common_info.py
   $ python update_common_info.py
   ```

The commands create **nand-ipq806x-single.img, nornand-ipq806x-single.img**, **nor-ipq806x-single.img**, and **emmc-ipq806x-single.img** as single images in the bin folder. The binary images are copied to the **ipq** directory either by the user or by the update_common_info.py command to create the FIT image:

```
cdt.mbn
nor-system-partition.bin
nand-flash.conf
nand_sbl1.mbn
nand_sbl2.mbn
nand_sbl3.mbn
nand-system-partition.bin
nor-flash.conf
nor_sbl1.mbn
nor_sbl2.mbn
norplusnand-flash.conf
norplusnand-system-partition.bin
nand-apps-flash.conf
nor-apps-flash.conf
nor_sbl3.mbn
emmc-flash.conf
emmc-apps-flash.conf
sdcc_sbl1.mbn
sdcc_sbl2.mbn
sdcc_sbl3.mbn
gpt_main0.bin
gpt_backup0.bin
openwrt-ipq806x-3.4-uImage
openwrt-ipq806x-u-boot.mbn
openwrt-ipq806x-squashfs-root.img
openwrt-ipq806x-ubi-root.img
rpm.mbn
ssd.mbn
tz.mbn
```

2.4.4 Load the flash image and boot the platform

Changes to the Wi-Fi transceivers ae necessary for images that are flashed on certain boards (RDPs) that were previously running QCA_Networking_2016.SPF.3.0 release or earlier.

These modifications involve either changing the order of module insertion as needed, or changing the wifi scripts to write the calibration data filenames appropriately. Such changes are needed only if the RDP has both the following restrictions:

- A combination of offload (OL) and direct-attach (DA) radios
- At least one OL radio is required to come up before the DA radio.

To set up the flash environment, do the following:

1. As a preliminary step, ensure that the board console port is connected to the PC using these RS232 parameters:
   - 115200bps
   - 8N1
2. Confirm that the PC is connected to the board using one of the Ethernet ports. The PC must have a TFTP server launched and listening on the interface to which the board is connected. At this stage power up the board and, after a few seconds, press any key during the countdown.

Flashing commands

The xxxx-ipq806x-single.img is already a packed image and does not need any further packing. Start by copying the xxxx-ipq806x-single.img to the TFTP server root directory.

1. Commands for the upgrade process:

```
set ipaddr 192.168.1.1
set serverip 192.168.1.xx      (This must be the address of the TFTP server)
set ethaddr 00:aa:bb:cc:dd:ee
set bootargs console=ttyMSM0,115200n8

saveenv
ping ${serverip}
tftpboot 0x42000000 xxxx-ipq806x-single.img
```

2. If using a NOR flash, execute the following command:

```
sf probe
```

3. Flash the image with this command:

```
imgaddr=0x42000000 && source $imgaddr:script
```

4. Reset the board after the loading of the image from flash memory is successful.
5. Once the single image is loaded, tftp the following package to the AP's /tmp dir from qsdk/prebuild/ipq806x/:
   - bluetopia_4.2.1.c1_9-1_ipq806x.ipk
6. Install the following package from the /tmp dir:
   - opkg install bluetopia_4.2.1.c1_9-1_ipq806x.ipk

2.4.4.1 Upgrade the firmware

This release has a feature to upgrade images from the OpenWrt web interface without the need for a TFTP server. After loading the first image from flash memory and booting the device, any future upgrades can be done from the web interface.

2.5 Related documentation for IPQ8064

For more details on IPQ8064, refer to *QCA_NETWORKING_2017-SPF.5.x Related Documents for Reference* (80-YA760-2).

3 IPQ4019.ILQ.5.0

This chapter describes details regarding the IPQ4019.ILQ.5.0 SP that is part of this SPF package.

3.1 Supported hardware

This section describes the hardware boards that are compatible with the SP.

Hardware board	Comments
AP.DK01.1	IPQ4018-based, Retail
AP.DK01.2	IPQ4028-based, Enterprise
AP.DK03.1	IPQ4018-based, Retail
AP.DK03.2	IPQ4028-based, Enterprise
AP.DK04.1	IPQ4019-based, Retail
AP.DK04.2	IPQ4029-based, Enterprise
AP.DK05.1	IPQ4018-based, Retail
AP.DK06.x	IPQ4019-based LTE gateway, Retail
AP.DK07.1	IPQ4019-based, Retail with CSR8811 (BT module)
AP.DK07.2	IPQ4029-based, Enterprise with CSR8811 (BT module)
AP.DK07.5	IPQ4019-SBS, Retail
AP.DK07.6	IPQ4019, Retail
RDP0321	IPQ4019.AP.DK07.1.QCA9888.XB.BSR02.1.QCA0000.SiLabs.QCA0000.MFI.1.RDP

For more information, see the hardware reference guides listed in Section 3.7.

3.2 Restrictions

Table 3-1 lists the restrictions on the software while using it for testing.

Table 3-1 Restrictions on the software

<table>
<tr><td rowspan="7">Channel Conditions</td><td colspan="2">For peak performance results, the AP, STA, and channel conditions must be configured to:</td></tr>
<tr><td>Attribute</td><td>Value</td></tr>
<tr><td>Image</td><td>See Section 1.1 for the release version</td></tr>
<tr><td>AP</td><td>AP-DK04</td></tr>
<tr><td>RSSI of AP at STA</td><td>43 - 53</td></tr>
<tr><td>RSSI of STA at AP</td><td>43 - 53</td></tr>
<tr><td>Attenuation</td><td>32 dB per chain</td></tr>
<tr><td>ESS</td><td colspan="2">
▪ IPQ40xx Ethernet subsystem (ESS) LAN and WAN groups to be tagged with different VLAN IDs. Two VLAN IDs (1 and 2) are reserved for the LAN and WAN groups, respectively. Different VLAN IDs (such as X and Y) can be used through the following configuration:
<code>echo X > </code><code>/proc/sys/net/edma/edma_default_wtag</code> (for WAN)
<code>echo Y > </code><code>/proc/sys/net/edma/edma_default_ltag</code> (for LAN)
▪ Ethernet full-sized jumbo frames are not supported.
▪ Enhanced DMA (EDMA) driver supports S/G through paged array of fragments (nr_frags). It does not support fraglist.
▪ Auto scaling needs to be disabled and CPU operating at highest frequency for performance KPI measurements. In bidirectional KPI case, two flows must have two different RSS hash and different core mapping in order to run in multiple cores.
▪ GRO on GMAC interfaces needs to be disabled for performance KPI measurements. (It is done through a script automatically)
▪ IGMP snooper supports IGMPv3/MLDv2 but it has the report suppression issue when it works in MLDv1. And it does not support the IGMP server in the LAN side.
▪ Shortcut Forwarding Engine (SFE) can only accelerate UDP IPSEC downlink case.
</td></tr>
<tr><td>Wi-Fi</td><td colspan="2">
Qualcomm Wireless Repeater AP (QWRAP) design limitations
▪ Supported security modes: Open, WPA2-PSK, WEP, AES, TKIP
▫ Other modes such as WDS are not supported when QWRAP/Proxy mode is enabled.
▫ Dynamic switching of modes between Proxy STA and other modes is not supported.
▪ Enterprise authentication modes are not supported.
▪ Roaming supports STA roams among WRAPs, but not between WRAP and Root AP.
▪ For a list of protocols supported in payload when MAT is used, see the AP 10.4 Programmers Guide – Wireless LAN.
▪ Cascading QWRAP is supported only with a unique MAC address defined for each proxy station in each QWRAP AP level. More than one QWRAP level is not extensively deployed and tested.
▪ QWRAP + WAPI encryption mode is not supported.
DBDC repeater and TBTC
▪ Dynamically changing primary radio configuration and <code>alwaysprimary</code> configuration during runtime is not supported.
▪ If the platform is tri-radio and has NSS Wi-Fi offload support, for TBTC repeater feature to work, NSS Wi-Fi offload must be disabled using UCI commands. Enter the following UCI configuration to disable NSS Wi-Fi offload mode.
<code>uci set wireless.qcawifi=qcawifi</code>
<code>uci set wireless.qcawifi.nss_wifi_olcfg=0</code>
<code>uci commit</code>
▪ By default, wifi0 is set as primary radio. On tri-radio board, if wifi0 is disabled, and if wif1 and wifi2 are configured in DBDC Repeater mode, then user must explicitly set wifi1 or wifi2 as primary radio.
</td></tr>
</table>

PLC	▪ PLC SON and Ethernet backhaul/loop prevention cannot co-exist together. PLC SON and Ethernet backhaul are mutually exclusive. ▪ VLAN ID configuration for PLC interface is not supported in PLC SON. ▪ AVitar/EDM tool is not supported in PLC SON. ▪ Guest and private networks are not supported in PLC SON. ▪ When different PLC chipsets are used in the Wi-Fi SON network (for example, CAP is connected to QCA7500 and RE is connected to AR7420), the behavior will not be as expected. ▪ If QCA9531 AP152 + QCA7500 (or QCA7550 or QCA7420) platform runs with 90-100% CPU utilization in SON network, there might be topology discovery entry of PLC changing frequently due to CPU busy handling traffic. The topology discovery packet is a keep alive message between CAP and repeater. ▪ For PLC-SON, load balancing with daisy chain feature is not supported. ▪ When PLC-SON is used along with daisy chain feature, looping is observed. This is planned to be addressed in the post-CS release. ▪ REH172 as RootAP configuration (CAP) is not supported. REH172 can be used only as a Range Extender in the PLC-SON topology. RFC2544 throughput impact is observed for lower-size packets (64 bytes, 128 bytes) with EN50561 PIB. This reduced performance for smaller packet size is due to a CPU busy detection feature implemented in boosted PSD firmware, to address the dips in UDP SNR. The fix involved adding a new sequence to invoke the CPU busy detector since the tone map generation tasks did not get the CPU time to run. This is a system limitation.

3.3 Power management

ARM A7 CPU Frequency Scaling	CPU frequency can be scaled up and down based on CPU load. ▪ OnDemand governor is the default CPU frequency governor ▪ Following CPU frequencies are supported for this release: ▫ 716 MHz ▫ 500 MHz ▫ 200 MHz ▫ 48 MHz
Dynamic clock gating is enabled	Software detects the inactivity in parts of the system and shuts down the clock to these parts accordingly
DDR and NOC	DDR runs at 537 MHz or 672 MHz depending on IP40xx package SNOC at 200 MHz PCNOC at 100 MHz Frequency Scaling is not supported for DDR and NOC

For example, use these commands to switch governor and CPU frequency:

```
echo userspace > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
echo 716000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

3.4 Build and load the image for IPQ4019.ILQ.5.0

An SPF and its custom branches are created and managed in CreatePoint in the same manner as a single SP distribution. A single git repository is created as a publicly available open source software version control system. This git repository:

- Is used to record the release history for the entire codebase of a corresponding SPF
- Adds each SPF distribution with its own custom branch and manages each as a new snapshot of the SPF codebase

An SPF distribution contains the same SP and SI build command files contained in the individual SP distributions. The primary difference the SPF includes more than one instance of the same type of SI and can include more than one SP.

The file set for each software product (SP) is contained in a directory that has the same name as the product (such as IPQ4019.ILQ.5.0) and consists of the following files:

- Binary files to program into flash memory
- Build files that generate the binary files
- Command files that program the binary files into flash memory

Released SP images are available for download from the ChipCode portal for proprietary-based code. For open source code, obtain the files from the Linux Foundation at the Code Aurora Forum.

To build and load the SP image on the device, do the following:

1. Download the Qualcomm Technologies proprietary code from ChipCode (see section 3.4.1).
2. Download other components from external websites by QSDK while building the default configuration (see section 3.4.2).
3. Generate the firmware by doing the following:
   a. Reassemble the code (see section 3.4.3.1).
   b. Create the QSDK build (see section 3.4.3.2).
   c. Build a complete firmware image (see section 3.4.3.3).
4. Install the image in the flash memory of the device and boot using the image from the flash (see section 3.4.4).

It is recommended that you be familiar with the structure of directories that contain the SP images for the different subsystems before you download the code and build the images for loading. For more information, refer to the *QCA_Networking_2017.SPF.5.0 Product Family Overview* (80-YA935-1). For each SP included in an SPF, SP binary files are generated from the SI binary files of only a subset of the included SIs. In an SPF, some SIs may support multiple SPs while others may only support one SP.

Starting with the QCA_Networking_2016.SPF.4.0 CS release, all WiGig components are not a part of the APPS image and are distributed as loadable installable packages (ipks). Install them manually after the image is loaded from flash memory. In addition, wilserver and LogCollector are integrated to the Dandelion distribution.

3.4.1 Download packages available through ChipCode

Qualcomm Technologies proprietary code is available from ChipCode.

A web/GUI interface and a secure git server both allow access to this code. Browse available packages and obtain the download URL at https://chipcode.qti.qualcomm.com/ (see https://chipcode.qti.qualcomm.com/helpki/cloning-code-from-a-repository for more information).

Customers are recommended to obtain the proprietary code with ‘git clone’.

See https://chipcode.qti.qualcomm.com/helpki for more information on installation and configuration of the correct version of git and OpenSSL on both Windows and Linux platforms that is required to support the authentication methods used by ChipCode.

3.4.2 Download packages from external websites

For details on downloading packages from external sites, see Section 2.4.2.

3.4.3 Generate the firmware for IPQ4019.ILQ.5.0

To generate a firmware image, reassemble the code, create the QSDK build, and create a complete firmware image. This section describes the procedure to generate the firmware.

3.4.3.1 Reassemble the code

The first step is to reassemble the code from ChipCode and the Linux Foundation and generate the QSDK framework. The example illustrated in this section assumes that all packages listed in sections 3.4.1 and 3.4.2 are obtained using the **git clone** command and placed in the top-level directory:

1. Enter the following commands to reassemble the code and generate the QSDK framework:

```
$ git clone <chipcode-distro>
$ cd <chipcode directory>
$ git checkout r00008.1
```

2. After the copy of the existing Git repository is completed, the directories in which the files are present must be changed as described in the following table before the repo command is run.

All Packages:

Use git to obtain the following files from ChipCode and copy them to the working QSDK top-level directory of the device:	Local directory path to files fetched by git from ChipCode:
qsdk-qca-wifi qsdk-qca-wlan qsdk-ieee1905-security qsdk-qca-athdiag	NHSS.QSDK.5.0\apss_proc\out\proprietary\Wifi
qca-lib qca-mcs-apps	NHSS.QSDK.5.0\apss_proc\out\proprietary\QSDK-Base
qca-bluetopia.tar.bz2	NHSS.QSDK.5.0\apss_proc\out\proprietary\BLUETOPIA
qca-wifi-fw-IPQ4019_hw_1-WLAN.BL.3.5-00010-S-1.tar.bz2 qca-wifi-fw-QCA9888_hw_2-WLAN.BL.3.5-00010-S-1.tar.bz2 qca-wifi-fw-QCA9984_hw_1-WLAN.BL.3.5-00010-S-1.tar.bz2	WLAN.BL.3.5\cnss_proc\bin\IPQ4019\hw.1 WLAN.BL.3.5\cnss_proc\bin\QCA9888\hw.2 WLAN.BL.3.5\cnss_proc\bin\QCA9884\hw.1
qca-wifi-fw-src-component-cmn-WLAN.BL.3.5-00010-S-1.tgz qca-wifi-fw-src-component-halphy_tools-WLAN.BL.3.5-00010-S-1.tgz	WLAN.BL.3.5\cnss_proc\src\components
qca-wifi-fw-AR9887_hw_1-CNSS.PS.2.5-00006-S-1.tar.bz2 qca-wifi-fw-AR9888_hw_2-CNSS.PS.2.5-00006-S-1.tar.bz2	CNSS.PS.2.5

3. After copying the necessary files to the appropriate directories, enter the following commands to continue with the process of generating the QSDK framework:

```
$ repo init -u git://codeaurora.org/quic/qsdk/releases/manifest/qstak -b release -
   m caf_AU_LINUX_QSDK_RELEASE_ENDIVE_CC_TARGET_ALL.0.1.719.031.xml --repo-
   url=git://codeaurora.org/tools/repo.git --repo-branch=caf-stable
$ repo sync
$ mkdir -p qsdk/dl
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-qca-wifi/* qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-qca-wlan/* qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-ieee1905-security/*
   qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-qca-athdiag/* qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/QSDK-Base/qca-lib/* qsdk
$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/QSDK-Base/qca-mcs-apps/*
   qsdk
$ tar -xzvf WLAN.BL.3.5/cnss_proc/src/components/qca-wifi-fw-src-component-cmn-
   WLAN.BL.3.5-00010-S-1.tgz
$ mv include qsdk/qca/src/qca-wifi-10.4/fwcommon
$ tar -xzvf WLAN.BL.3.5/cnss_proc/src/components/qca-wifi-fw-src-component-
   halphy_tools-WLAN.BL.3.5-00010-S-1.tgz
$ mv wlan/halphy_tools qsdk/qca/src/qca-wifi-10.4
```

```
$ tar xjvf NHSS.QSDK.5.0/apss_proc/out/proprietary/BLUETOPIA/qca-bluetopia.tar.bz2
   -C qsdk
$ cp WLAN.BL.3.5/cnss_proc/bin/IPQ4019/hw.1/* qsdk/dl
$ cp WLAN.BL.3.5/cnss_proc/bin/QCA9888/hw.2/* qsdk/dl
$ cp WLAN.BL.3.5/cnss_proc/bin/QCA9984/hw.1/* qsdk/dl
$ cp -rf WLAN.BL.3.5/cnss_proc/src/components/* qsdk/dl
$ cp CNSS.PS.2.5/* qsdk/dl
```

4. (Optional) This step applies only for customers with HY-FI, WHC, PLC or WAPid packages.

	Use git to obtain the following files from ChipCode and copy them to the working QSDK top-level directory of the device:	**Local directory path to files fetched by git from ChipCode:**
HY-FI Customers:	hyfi-ipq	NHSS.QSDK.5.0\apss_proc\out\proprietary\Hyfi
WHC Customers:	qsdk-whc qsdk-whcpy	NHSS.QSDK.5.0\apss_proc\out\proprietary\Wifi
WAPid Customers:	qsdk-wapid	NHSS.QSDK.5.0\apss_proc\out\proprietary\Wapid
PLC Customers	qca_plc	NHSS.QSDK.5.0/apss_proc/out/proprietary/qca_plc

	HY-FI , WHC, WAPId, or PLC customers: Run the additional code:
HY-FI:	`$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Hyfi/hyfi-ipq/* qsdk`
WHC:	`$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-whc/* qsdk` `$ cp -rf NHSS.QSDK.5.0/apss_proc/out/proprietary/Wifi/qsdk-whcpy/* qsdk`

WAPid:	`$ git clone <Wapid Chipcode link>`
	`$ cd <Wapid chipcode directory>`
	`$ git checkout r00008.1`
	`$ cd ..`
	`$ cp -rf <Wapid Chipcode directory>/NHSS.QSDK.5.0/apss_proc/out/proprietary/Wapid/qsdk-wapid/* qsdk`

PLC:	`$ git clone <PLC Chipcode link>`
	`$ cd <PLC chipcode directory>`
	`$ git checkout r00008.1`
	`$ cd ..`
	`$ cp –rf <PLC chipcode directory>/NHSS.QSDK.5.0/apss_proc/out/proprietary/qca_plc/* qsdk/prebuilt/ipq806x/`

5. (Optional) This step applies only for CSRMesh customers. Enter the following commands during QSDK installation if the CSRmesh application is to be run on the reference design. Obtain the CSRMeshGatewaySouthBoundv2.1_DK07 tarball from the CSRsupport site at https://www.csrsupport.com/Qwin/csrmesh.

   Contact your Qualcomm technical support team for information on obtaining these additional files:

```
$ unzip CSRMeshGatewaySouthBoundv2.1CS_DK07

$ tar xjvf CSRMeshGatewaySouthBoundv2.1CS_DK07/Binaries/qca-csrmesh-0.1.453.245.tar.bz2
-C <chipcode directory>/qsdk/
```

The local directory **qsdk** is created by these repo steps as a sub-directory of the current working directory, from which repo was executed. This is the working QSDK top level directory.

If the CONFIG_DOWNLOAD_FOLDER="" item in the default .config file is changed, the files qca-wifi-fw-*.tgz in cnss_proc/src/components must be copied to both the qsdk/dl folder as well as the newly configured download folder, in order for the builds on third party machines to complete. The file qca-wifi-fw-IPQ4019_hw_1-WLAN.BL.3.5-00010-S-1.tar.bz2 must be located in the newly configured download folder.

3.4.3.2 Create the QSDK build

QSDK supports two profiles: **Premium** and **Standard.** The premium profile is targeted for 32 MBytes flash and 256 MBytes DDR configuration, whereas the standard profile is a subset of the premium profile features to support 16 MBytes flash and 128 MBytes DDR configuration.

To fit the image in 16 MBytes, the following features have been disabled compared to the premium profile:

- CD Router certification
- Bluetooth
- Audio
- Video
- IPSec

You will be receiving the Premium distro for this release. Although this Premium distro can build either Premium or Standard profiles, DK01 boards are configured to boot from the 32 Mbyte-flash; therefore board changes are needed before a standard profile image can be flashed and booted.

QSDK framework has been developed using Ubuntu (from version 12.04 to version 16.04), and Debian. However, QSDK framework regenerates critical tools required to compile firmware at build-time. In that sense, the framework is independent from the host environment; although it is developed using the distributions above, it is expected to work on others such as RedHat, Mint, or Fedora. This command is for Debian/Ubuntu; it must be customized for other distributions:

```
$ sudo apt-get install gcc g++ binutils patch bzip2 flex make gettext \
  pkg-config unzip zlib1g-dev libc6-dev subversion libncurses5-dev gawk \
  sharutils curl libxml-parser-perl ocaml-nox ocaml-nox ocaml ocaml-findlib \
  libpcre3-dev binutils-gold python-yaml
```

Since the framework automatically downloads the open source components, make sure an internet connection is active on the build host while creating the build.

To create the QSDK build, do the following:

1. Move to the qsdk dir

   ```
   $ cd qsdk
   ```

2. Install the different feeds in the build framework:

   ```
   $./scripts/feeds update -a
   $./scripts/feeds install -a -f
   ```

3. Copy the base configuration to use for the build. Choose either the standard or premium profile.

Premium	`$ cp qca/configs/qsdk/ipq806x_premium.config .config`
Standard	`$ cp qca/configs/qsdk/ipq806x_standard.config .config`

4. Regenerate a complete configuration file.

```
$ make defconfig
$ for pkg_num in 2 9;do sed 's/CONFIG_PACKAGE_qca-wifi-fw-
hw'${pkg_num}'-10.4-asic=y/# CONFIG_PACKAGE_qca-wifi-fw-hw'${pkg_num}'-
10.4-asic is not set/g' -i .config;done
$ sed 's/CONFIG_PACKAGE_kmod-wil6210=m/# CONFIG_PACKAGE_kmod-wil6210 is
not set/g' -i .config
$ sed 's/CONFIG_PACKAGE_wigig-firmware=m/# CONFIG_PACKAGE_wigig-firmware
is not set/g' -i .config
```

5. (Optional) Use the following code to build GCC v5.2:

```
echo "CONFIG_TOOLCHAINOPTS=y" >> .config
echo '# CONFIG_GCC_USE_VERSION_4_8_LINARO is not set' >> .config
echo "CONFIG_GCC_USE_VERSION_5=y" >> .config
echo 'CONFIG_GCC_VERSION="5.2.0"' >> .config
echo "CONFIG_GCC_VERSION_5=y" >> .config
```

6. Start the build:

```
$ make V=s
```

The preceding instructions download the packages required for the corresponding profile and create the image. After the build is complete, these files are available in the **qsdk/bin/ipq806x** directory:

- openwrt-ipq40xx-u-boot-stripped.elf (Bootloader)
- openwrt-ipq806x-qcom-ipq40xx-ap.dkxx-fit-uImage.itb (Kernel + dtb)
- openwrt-ipq806x-squashfs-root.img (SquashFS)
- openwrt-ipq806x-ipq40xx-ubi-root.img (UBIFS)

3.4.3.3 Generate a complete firmware image

The IPQ40xx flash image includes multiple components. These include DTB, SBL1, TZ, CDT, DDR, MIBIB, Kernel, Filesystem, etc. To simplify the loading of images from flash memory and booting of devices, they have been combined into a single Flattened Image Tree (FIT) image. The FIT image can be flashed into the respective partition based on user configuration information. Additional tools are required on the Ubuntu 12.04 64-bit machine.

To build a complete firmware image:

1. Install mkimage: **sudo apt-get install uboot-mkimage**
2. Install DTC: **sudo apt-get install device-tree-compiler**
3. Install Python 2.7
4. Switch to the Qualcomm ChipCode directory:

```
$ cd <chipcode directory>
```

5. Some systems may require installing/updating the U-Boot tools: **sudo apt-get install u-boot-tools**. Ubuntu 14.04 build hosts also require the mtd-utils package, sudo apt-get install mtd-utils
6. Copy the flash config files to **IPQ4019.ILQ.5.0/common/build/ipq**; choose the either standard or premium profile:

Premium	`$ cp meta-scripts/ipq40xx_premium/* IPQ4019.ILQ.5.0/common/build/ipq`
Standard	`$ cp meta-scripts/ipq40xx_standard/* IPQ4019.ILQ.5.0/common/build/ipq`

7. Copy **pack.py** to the **NHSS.QSDK.5.0/apss_proc/out/** directory:

```
$ cp -rf qsdk/qca/src/uboot-1.0/tools/pack.py  NHSS.QSDK.5.0/apss_proc/out/
```

8. Copy trustzone files to common/build/ipq

```
$ cp -rf TZ.BF.2.7/trustzone_images/build/ms/bin/MAZAANAA/*
   IPQ4019.ILQ.5.0/common/build/ipq
```

9. Copy the **openwrt*** images built to the **IPQ4019.ILQ.5.0/common/build/ipq** folder:

```
$ cp -rf qsdk/bin/ipq806x/openwrt* IPQ4019.ILQ.5.0/common/build/ipq
```

10. Copy the **boardconfig*** files to the **IPQ4019.ILQ.5.0/common/build/ipq** folder; choose either premium or standard profile depending on your requirement:

Standard

```
$ cp
BOOT.BF.3.1.1/boot_images/build/ms/bin/40xx/misc/tools/config/boardconfig_
standard IPQ4019.ILQ.5.0/common/build/ipq
$ cp
BOOT.BF.3.1.1/boot_images/build/ms/bin/40xx/misc/tools/config/appsboardcon
fig_standard IPQ4019.ILQ.5.0/common/build/ipq
```

Premium

```
$ cp
BOOT.BF.3.1.1/boot_images/build/ms/bin/40xx/misc/tools/config/boardconfig_
premium IPQ4019.ILQ.5.0/common/build/ipq
$ cp
BOOT.BF.3.1.1/boot_images/build/ms/bin/40xx/misc/tools/config/appsboardcon
fig_premium IPQ4019.ILQ.5.0/common/build/ipq
```

11. Modify the SI path in contents.xml

```
$ sed -i 's#</linux_root_path>#/</linux_root_path>#'
   IPQ4019.ILQ.5.0/contents.xml
$ sed -i 's#</windows_root_path>#\\</windows_root_path>#'
   IPQ4019.ILQ.5.0/contents.xml
```

12. Run these commands to create a single image; choose either premium or standard profile depending on your requirement:

```
$ cd IPQ4019.ILQ.5.0/common/build
```

Standard

```
$ sed '/debug/d' -i update_common_info_standard.py
$ sed '/s-gcc5/d' -i update_common_info_standard.py
$ python update_common_info_standard.py
```

Premium

```
$ sed '/debug/d' -i update_common_info.py
$ python update_common_info.py
```

The commands create **nor-ipq40xx-single.img, nand-ipq40xx-single.img, nornand-ipq40xx-single.img, emmc-ipq40xx-single.img, norplusemmc-single.img, and nor-ipq40xx-standard-single.img** single images in the bin folder. The binary images are copied to the **ipq** directory either by the user or by the update_common_info.py command to create the FIT image:

```
cdt-AP.DK01.1-C1.bin
cdt-AP.DK01.1-C2.bin
cdt-AP.DK01.1-S1.bin
cdt-AP.DK04.1-C1.bin
cdt-AP.DK04.1-C2.bin
cdt-AP.DK04.1-C3.bin
cdt-AP.DK04.1-C5.bin
```

```
cdt-AP.DK04.1-S1.bin
cdt-AP.DK05.1-C1.bin
cdt-AP.DK06.1-C1.bin
cdt-AP.DK07.1-C1.bin
gpt_backup0.bin
gpt_main0.bin
nor-system-partition-ipq40xx-s.bin
nand-system-partition-ipq40xx.bin
nor-system-partition-ipq40xx.bin
norplusnand-system-partition-ipq40xx.bin
norplusemmc-system-partition-ipq40xx.bin
openwrt-ipq40xx-u-boot-stripped.elf
openwrt-ipq806x-ipq40xx-ubi-root.img
openwrt-ipq806x-ipq40xx-ubi-root-512MB.img
openwrt-ipq806x-qcom-ipq40xx-ap.dkxx-fit-uImage.itb
openwrt-ipq806x-squashfs-root.img
sbl1_emmc.mbn
sbl1_nand.mbn
sbl1_nor.mbn
tz.mbn
nand-flash.conf
nor-flash.conf
emmc-flash.conf
norplusnand-flash.conf
norplusemmc-flash.conf
packages
```


Flash Type	Single Image
SPI NOR (32 MBytes)	nor-ipq40xx-single.img
SPI NOR + NAND	nornand-ipq40xx-single.img
SPI NOR (16 MBytes) and DDR 128 MBytes	nor-ipq40xx-standard-single.img
ONFI NAND	nand-ipq40xx-single.img
eMMC	emmc-ipq40xx-single.img
SPI NOR (16 MBytes) + eMMC	norplusemmc-ipq40xx-single.img

3.4.4 Load the flash image and boot the platform

Changes to the Wi-Fi transceivers might be necessary for images that are flashed on certain boards (RDPs) that were previously running QCA_Networking_2016.SPF.3.0 release or earlier.

These modifications involve either changing the order of module insertion as needed, or changing the wifi scripts to write the calibration data filenames appropriately. Such changes are needed only if the RDP has both the following restrictions:

- A combination of offload (OL) and direct-attach (DA) radios
- At least one OL radio is required to come up before the DA radio.

To set up the flash memory environment, do the following:

1. As a preliminary step, ensure that the board console port is connected to the PC using these RS232 parameters:
   - 115200bps
   - 8N1
2. Confirm that the PC is connected to the board using one of the Ethernet ports. The PC must have a TFTP server launched and listening on the interface to which the board is connected. At this stage power up the board and, after a few seconds, press any key during the countdown.

The xxxx-ipq40xx-single.img is already a packed image and does not need any further packing.

Procedure for standard board configuration programming

To load the image in flash and boot the platform using the image from flash, do the following:

1. Copy the xxxx-ipq40xx-single.img to the TFTP server root directory.
2. Check hardware jumper Configuration according to reference board and configuration. For more information on jumper configuration, see the appropriate Setup Guide document as listed in section 3.7 for your board.
3. Confirm the machine ID, Meta version, profile and single image.

Board/ Configuration	Machine ID	Flash	Meta/ Profile Support	Image Name
AP.DK01.1-C1	8010000	NOR 32 MBytes	Premium profile	nor-ipq40xx-single.img
AP.DK01.1-C2	8010100	NOR (SO8) + SPI NAND (128 MBytes)	Premium profile	nornand-ipq40xx-single.img
AP.DK01.1-S1	8010200	NOR 16 MBytes	Standard profile	nor-ipq40xx-standard-single.img
AP.DK04.1-C1	8010001	NOR 16 MBytes	Standard profile	nor-ipq40xx-standard-single.img
AP.DK04.1-C1	8010001	NOR 32 MBytes	Premium profile	nor-ipq40xx-single.img
AP.DK04.1-C1	8010001	QPIC NAND (128 MBytes)	Premium profile	nand-ipq40xx-single.img
AP.DK04.1-C1	8010001	eMMC	Premium profile	emmc-ipq40xx-single.img
AP.DK04.1-C1	8010001	NOR (SO8) + QPIC NAND (128 MBytes)	Premium profile	nornand-ipq40xx-single.img
AP.DK04.1-C2	8010101	NOR 32 MBytes – Audio	Premium profile	nor-ipq40xx-single.img
AP.DK04.1-C3	8010201	NOR (SO8) + eMMC	Premium profile	norplusemmc-ipq40xx--single.img
AP.DK04.1-C5	8010401	NOR (SO8) + SPI NAND (128 MBytes)	Premium profile	nornand-ipq40xx-single.img
AP.DK05.1-C1	8010007	NOR (SO8) + SPI NAND (128 MBytes)	Premium profile	nornand-ipq40xx-single.img
AP.DK06.1-C1	8010005	QPIC NAND (128 MBytes)	Premium profile	nand-ipq40xx-single.img
AP.DK07.1-C1	8010006	QPIC NAND (128 MBytes)	Premium profile	nand-ipq40xx-single.img

4. Commands for the TFTP process:

```
set ipaddr 192.168.1.11
set serverip 192.168.1.xx (This must be the address of the TFTP server)
ping ${serverip}
tftpboot 0x84000000 xxxx-ipq40xx-single.img
```

5. Flash the image with this command:

```
imgaddr=0x84000000 && source $imgaddr:script
```

If CSRmesh infrastructure, LTE support, or both are included according to the procedure described in section 3.4.3.2, complete the following steps to ensure the extra packages are transferred to and installed on the target system.

6. After the single image is loaded, tftp these packages to the AP's /tmp dir from qsdk/prebuild/ipq806x/:

```
bluetopia_4.2.1.c1_9-1_ipq806x.ipk
qca-hyd_ge474d2e-1_ipq806x.ipk
```

7. Install these packages from the /tmp dir:

```
opkg install bluetopia_4.2.1.c1_9-1_ipq806x.ipk
opkg install qca-hyd_ge474d2e-1_ipq806x.ipk
```

8. For LTE, install Sierra-cm:

```
$ opkg install sierra-cm_SLQS03.03.10.bin-1_ipq806x.ipk
```

Procedure for non-standard board configuration programming

The procedures for changing machine ID in u-boot is complex which requires more steps. For example: changing AP.DK01.1-C1 to AP.DK01.1-C2 on a DK01 board. AP.DK01 includes a 32 MBytes NOR flash and a 2 MBytes NOR flash. For a new AP.DK01 board, the 2 MBytes NOR flash is empty.

1. To load the SPI NOR+NAND singe image from flash and boot the device:
   a. Check hardware jumper configuration. Because the 2M NOR flash is empty, the jumper settings for AP.DK01.1-C2 cannot be used to bring up the board. Use the jumper settings for AP.DK01.1-C1 to bring up to u-boot shell. Ensure the machine ID is 0x8010000.
   b. Confirm machine ID, meta version, profile, and single image name of image to be flashed into the 2M NOR + NAND flash.
   c. Commands for setting machine ID

```
set machid ABCDEYZ
```

   Example for AP.DK01.1-C1

```
set machid 8010000
```

   d. Set the machine ID to 0x8010100 (AP.DK01.1-C2).
   e. Upload nornand-ipq40xx-single.img by TFTP.
   f. Flash the image.
      i Make sure the jumper settings for AP.DK01.1-C2 is used. Otherwise the 2 MBytes NOR and NAND cannot be accessed.
      ii imgaddr=0x84000000 && source $imgaddr:script
      iii An error occurs with the log:

```
"NAND erase: Size exceeds partition or device limit"
```

      Root cause of failure: If the running u-boot uses machine ID 0x801000, it cannot access NAND. In this case, reset the board and redo Step 4 and Step 5 to write the second part of the image to NAND.

      iv After board reset, new software is read and booted from the 2 MBytes NOR flash. The running u-boot uses machine ID 0x8010100. No need to redo Step 3.
      v Once done, Step 5 will be successful.

2. To load the image to the NOR flash and boot, enter the following command:

```
sf probe && imgaddr=0x84000000 && source $imgaddr:script
```

3. If the source version is standard profile, use the low 128-MByte DDR address space.

   For example, while changing machine ID to 8010000(AP.DK01.1-C1) from 8010200(AP.DK01.1-S1), do not use 0x88000000 DDR address because the AP.DK01.1-S1 version supports only 128 MByte DDR.

   Incorrect command

   ```
   tftpboot 0x88000000 nor-ipq40xx-single.img&& sf probe
   &&imgaddr=0x88000000&&source $imgaddr:script
   ```

   Correct command

   ```
   tftpboot 0x84000000 nor-ipq40xx-single.img&& sf probe
   &&imgaddr=0x84000000&&source $imgaddr:script
   ```

3.4.4.1 Upgrade the firmware

This release has a feature to upgrade images from the OpenWrt web interface without the need for a TFTP server. After loading the first image from flash memory and booting the device, any future upgrades can be done from the web interface. Use single image to upgrade the image.

3.4.4.2 Procedure to load Ramdisk image

1. Halt the auto-boot process at u-boot prompt by typing during the auto-boot count-down phase.
2. Execute set fdt_high 0x87000000.
3. Download ramdisk image to address 0x88000000 based on board type.

   ```
   tftpb 0x88000000 openwrt-ipq806x-qcom-ipq40xx-<xxx>-fit-uImage.itb
   ```

4. Execute bootm 0x88000000

Board/Configuration	Machine ID	Image Name
AP.DK01.1-C1	8010000	openwrt-ipq806x-qcom-ipq40xx-ap.dk01.1-c1-fit-uImage-initramfs.itb
AP.DK01.1-C2	8010100	openwrt-ipq806x-qcom-ipq40xx-ap.dk01.1-c2-fit-uImage-initramfs.itb
AP.DK04.1-C1	8010001	openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c1-fit-uImage-initramfs.itb
AP.DK04.1-C2	8010101	openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c2-fit-uImage-initramfs.itb
AP.DK04.1-C3	8010201	openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c3-fit-uImage-initramfs.itb
AP.DK04.1-C5	8010401	openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c5-fit-uImage-initramfs.itb
AP.DK05.1-C1	8010007	openwrt-ipq806x-qcom-ipq40xx-ap.dk05.1-c1-fit-uImage-initramfs.itb
AP.DK06.1-C1	8010005	openwrt-ipq806x-qcom-ipq40xx-ap.dk06.1-c1-fit-uImage-initramfs.itb
AP.DK07.1-C1	8010006	openwrt-ipq806x-qcom-ipq40xx-ap.dk07.1-c1-fit-uImage-initramfs.itb

3.5 PLC SON-related updates

3.5.1 ESS port mapping in REH172

- If REH172 is used as a repeater in Wi-Fi SON enabled mode, the machine id of the device must be set to 8010100 on U-Boot at the time of flashing the image. For example, set machine id 8010100.

ESS port mappings in REH172 Machine Id 8010007		
Interface	**Connectivity**	**Ports connection**
Eth0	PLC	0, 4
Eth0	External LAN Port	0,5

ESS port mappings in DK01 Machine Id 8010000		
Interface	**Connectivity**	**Ports connection**
Eth1	LAN	0,1,2,3,4
Eth0	WAN	0,5

- If REH172 is already programmed with machine ID 8010007 in field, use the following procedure to configure the **Wi-Fi SON** mode; the commands can be added in startup script in the platform.

```
cd /proc/sys/net/edma/
echo 0x20 > default_group1_bmp
echo 0x1E > default_group2_bmp
echo 2 > default_group1_vlan_tag
echo 1 > default_group2_vlan_tag
cd /* go back to root dir,*/
```

  - Edit /etc/config/network and ensure that the ports are in the right group.

```
   <snip>
   config switch
           option name 'switch0'
           option reset '1'
           option enable_vlan '1'
   config switch_vlan
           option device 'switch0'
           option vlan '1'
           option ports '0t 1 2 3 4'
   config switch_vlan
           option device 'switch0'
           option vlan '2'
           option ports '0t 5'
<snip>
```

- eth0 can be added in the bridge by using brctl addif br-lan eth0, or by entering the uci set network.lan.ifname='eth1 eth0' command.
- Enter the uci commit command to save the configuration.
- Reboot the system after it is verified.

- If REH172 already programmed with the machine ID 8010007 in field, use the following procedure to configure the **non-Wi-Fi SON** mode; the commands can be added in startup script in the platform.

```
cd /proc/sys/net/edma/
echo 0x3e > default_group1_bmp
echo 0x0 > default_group2_bmp
echo 0 > default_group1_vlan_tag
echo 0 > default_group2_vlan_tag
cd /* go back to root dir,*/
```

  - Edit /etc/config/network and ensure that the ports are in the right group.

```
config switch
        option name 'switch0'
        option reset '1'
        option enable_vlan '0'
config switch_vlan
        option device 'switch0'
        option vlan '0'
        option ports '0 1 2 3 4 5'
#config switch_vlan
        #option device 'switch0'
        #option vlan '2'
        #option ports '0t 5'
```

  - Reboot the system after it is verified.

3.5.2 PLC SON interface mapping

- Before doing the PLC SON configuration, set PLC interface name on CAP and RE side:

```
uci set plc.config.PlcIfname='eth1'
uci commit
```

  eth1 is used for PLC Interface for PLC backhaul connectivity between CAP and RE.

- In the repeater, PLC and one Ethernet port can be added in the bridge networks along with STA, AP VAPs:

```
uci set network.lan.ifname ='eth1 eth0'
uci commit
```

- When only PLC is used as backhaul between CAP and RE, use the following command on RE for auto configuration cloning thru PLC interface:

```
uci set repacd.repacd.DefaultREMode='son'
uci commit
```

For more details on the interface mapping, refer to *Wireless LAN Access Point (Driver Version 10.4) Programmer Guide* (80-Y8053-1).

PIB information

In the REH172 platform, following PIBs are included for QCA7550/QCA7520 support:

- QCA7550-REH172_HomePlugAV_NorthAmerica.pib
- QCA7520-REH172_HomePlugAV_NorthAmerica.pib
- QCA7550-REH172_EN50561-1.pib
- QCA7520-REH172_EN50561-1.pib

To change the PIB:

1. Run the following commands:
   ```
   uci set plc.config.PibPath='/etc/plc/<QCA7500-
   REH172_HomePlugAV_NorthAmerica>.pib'
   uci commit
   ```
2. Remove all the files under **/etc/plc/** directory.
3. Reboot the system.

3.6 Testing GATT profile with sample applications with onboard CSR8811 Bluetooth on AP.DK07

This release contains a preloaded Bluetooth sample application. To test GATT profile containing sample applications:

1. Launch the Bluetooth sample application from the AP after the Premium profile is built. It is located in: **NHSS.QSDK.5.0/apss_proc/out/proprietary/BLUETOPIA/qca-bluetopia-REV.tar.bz2**
2. After the complete firmware image is built, launch the Bluetooth stack to test the GATT profile with these LinuxSPPLE sample applications:
   - The LinuxSPPLE application is intended to demonstrate the usage of the Qualcomm® Bluetopia™ Serial Port Profile Low Energy API and relevant Bluetopia Core APIs. The application supports issuing all the basic commands used by the Serial Port Profile Low Energy.
   - Launching the BT sample APP which will enable BT and download the PSKeys to the BT module.
3. Wifi/BT coex PSKeys are enabled once the sample APP is launched.
4. Export the environment variable before launching the sample application.

Platform/Module	Environment variable
IPQ4019 platform DK07 or QFN BT module	**BTHOST_8311_SOC_TYPE=onboard**
IPQ8064 platform or CSP module	**BTHOST_8311_SOC_TYPE=sdio**
custom SOC GPIO settings	**BTHOST_8311_SOC_TYPE=custom**

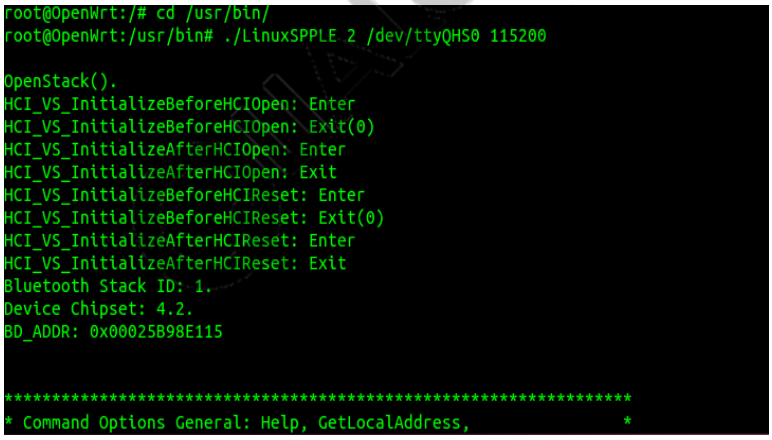
This reads PSKeys from the text file located in **/usr/bin as PS_KEY_CSR8811.txt.** Contents of the text file must be in this format:

```
# PSKEY_COEX_SCHEME
\x06\xC2\x02\x00\x09\x00\x34\x00\x03\x70\x00\x00\x80\x24\x01\x00\x00\x00
\x03\x00
# PSKEY_COEX_PIO_UNITY_3_BT_ACTIVE (BT_active)
```

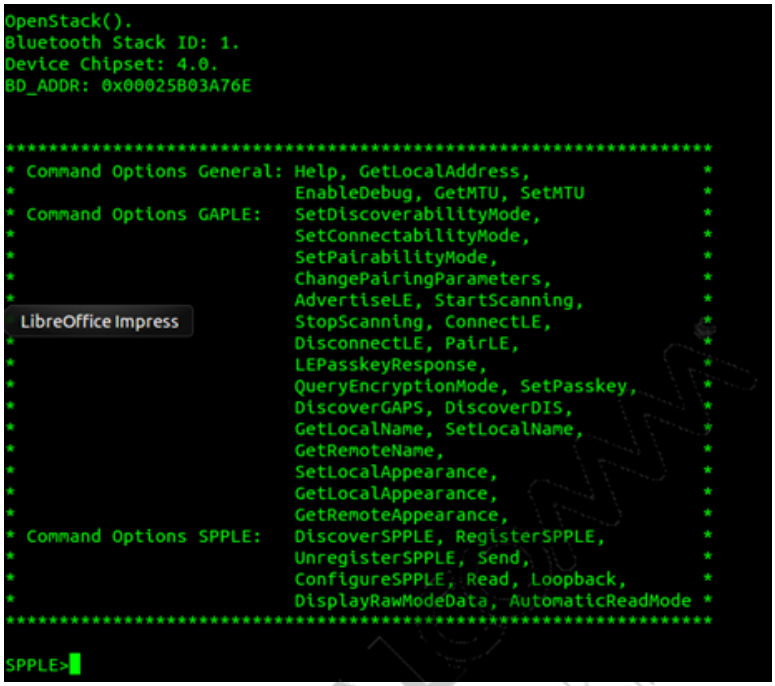
```
\x05\xC2\x02\x00\x0A\x00\x35\x00\x03\x70\x00\x00\x83\x24\x02\x00\x00\x00
\x05\x00\x01\x00
# PSKEY_COEX_PIO_UNITY_3_BT_STATUS (BT_STATUS)
\x05\xC2\x02\x00\x0A\x00\x36\x00\x03\x70\x00\x00\x84\x24\x02\x00\x00\x00
\xff\xff\x01\x00
# PSKEY_COEX_PIO_UNITY_3_BT_DENY (WLAN Deny)
\x05\xC2\x02\x00\x0A\x00\x37\x00\x03\x70\x00\x00\x85\x24\x02\x00\x00\x00
\x04\x00\x01\x00
# WARM RESET
\x00\xC2\x02\x00\x09\x00\x47\x00\x02\x40\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00
```

root@openwrt:cd /usr/bin
root@openwrt: ./LinuxSPPLE 2 /dev/ttyQHS0 115200

4. Note the Bluetooth address of the module on the AP.
   - Bluetooth Stack ID: 1.
   - Device Chipset: 4.2.
     BD_ADDR: 0x00025B98E115



5. Ignore the HCI_VS print statements; the Bluetooth stack application will launch with the prompt as shown:



6. Run these commands to ensure that the radio is attached to the BT connector is good and is able to scan surrounding devices:
   - StartScanning – Starts scanning all Bluetooth devices with their device ID and Bluetooth address
   - StopScanning – Stops scanning devices. This command can be run during scanning.
7. Use this command to register a SPPLE service.

   ```
   SPPLE>RegisterSPPLE
   Successfully registered SPPLE Service.
   SPPLE>
   ```

   This function will return zero on successful execution and a negative value on errors. This command takes no parameters.
8. Use this command to turn on LE advertising on the device.

   ```
   AdvertiseLE 1
   GAP_LE_Advertising_Enable success.
   SPPLE>
   ```

   The only parameter required is the advertising flag which is 0 for disable and 1 for enable. This registers the device as SPPLE and clients can connect to this one.

   - Bluetooth clients
     - Bluetooth clients can use a custom App or LightBlue App on an AP.DK07 device with Bluetooth or with the CSR USB dongle 8510. LinuxSPPLE must show up in the scanning. Connect to a Bluetooth server with the Bluetooth address and send data.
     - Connecting using Bluetopia stack from another DK07 with On-board CSR 8811

       ```
       SPPLE>ConnectLE 00025B98E115    0
       ```
10. This command is used for performing a SPPLE service discovery operation. This function returns zero on successful execution and a negative value on errors. This command takes no parameters.

Client side:

```
SPPLE>DiscoverSPPLE
GATT_Start_Service_Discovery_Handle_Range() success.

SPPLE>
Service 0x001E - 0x0028, UUID: 14839AC47D7E415C9A42167340CF2339.

SPPLE>
Service Discovery Operation Complete, Status 0x00.

Valid SPPLE Service Found.
```

11. This command is used to configure a SPPLE service on a remote device. This function returns zero on successful execution and a negative value on errors. This command takes no parameters.

    The following function enables notifications of the proper characteristics based on a specified handle. Depending on whether the device role for SPPLE is server or client, either a GATT_Handle_Value_Notification or a GATT_Write_Without_Response_Request API function is called. The called function notifies the receiving credit characteristic or sends a write without response packet to the transmission credit characteristic respectively.

```
SPPLE>ConfigureSPPLE

SPPLE Service found on remote device, attempting to read Transmit
Credits, and configured CCCDs.

SPPLE>
Write Response.
Connection ID:   1.
```

3.7 Related documentation for IPQ4019

For more details on IPQ4019, refer to *QCA_NETWORKING_2017-SPF.5.x Related Documents for Reference* (80-YA760-2).

4 QCA9531.ILQ.5.0

This chapter describes details regarding the QCA9531 SP that is part of this SPF package. The QCA9531.ILQ.5.0 SP applies for QCA9531, QCA9558, and QCA9563 chipsets.

4.1 Supported hardware

This section describes the hardware boards that are compatible with the SP.

Hardware Board	Comments
XB241.1	QCA9887-based 802.11ac 5 GHz scan radio, iPA/iLNA
XB243.1	QCA9887-based 802.11ac 2.4 GHz scan radio; xFEM
XB143.1	QCA9882-based 802.11ac 5 GHz, 2x2 11AC
CUS239.5	QCA9994-based 802.11ac 5 GHz , 4x4 configuration + Qorvo 5 GHz (Connected SmartHome)
CUS238.5	QCA9994+QFE1952-based 802.11ac 5 GHz (Retail)
CUS223.5	QCA9880-based 802.11ac 5 GHz, 3X3 802.11AC/A/N High Power (SKY) 4.2V XPA
XB:BSR01.2	QCA9896-based 802.11ac 5 GHz, 2x2 80M configuration
XB:BSR01.3	QCA9888 1.0 1x1 802.11ac 160MHz xPA Retail
XB:BSR01.4	QCA9888 1.0 1x1 802.11ac 160MHz xPA Enterprise
AP135.1	QCA9558 802.11n based AP
SCP01.1	QCA9558 802.11n based AP with PCIe Support for QCA98xx
AP147.1	QCA9531 802.11n based AP; 64MB/16MB
AP147.2	QCA9531 802.11n based AP; 128MB/16MB
HB03.1	QCA9531 802.11n based AP + on board QCA9886 2x2 80M
HB03.2	QCA9531 802.11n based AP + on board QCA9888 1x1 160M
HB04.1	QCA9531 802.11n based AP + on board QCA9888 1x1 160M
DF03.1	QCA9561 802.11n based AP + on board QCA9888 2x2 80M
DF03.2	QCA9561 802.11n based AP + on board QCA9898 1x1 160M
AP152.3	QCA9563 802.11n based AP
RDP0319	QCA9531.AP.HB04.1.RDP QCA9531 HB04 (80 MHz iPA 2L PCB) + on board QCA9886 BSR01 (1x1 11ac 160MHz Retail). 2L PCB 8MB Flash, 64MB

4.2 Build and load the image for QCA9531.ILQ.5.0

The QCA9531.ILQ.5.0 SP applies for QCA9531, QCA9558, and QCA9563 chipsets. An SPF and its custom branches are created and managed in CreatePoint in the same manner as a single SP distribution. A single git repository is created as a publicly available open source software version control system. This git repository:

- Is used to record the release history for the entire codebase of a corresponding SPF
- Adds each SPF distribution with its own custom branch and manages each as a new snapshot of the SPF codebase

An SPF distribution contains the same SP and SI build command files contained in the individual SP distributions. The primary difference the SPF includes more than one instance of the same type of SI and can include more than one SP.

The file set for each software product (SP) is contained in a directory that has the same name as the product (such as QCA9531/QCA9558/QCA9563.ILQ.5.0) and consists of the following files:

- Binary files to program into flash memory
- Build files that generate the binary files
- Command files that program the binary files into flash memory

Released SP images are available for download from the ChipCode portal for proprietary-based code. For open source code, obtain the files from the Linux Foundation at the Code Aurora Forum.

To build and load the SP image on the device, do the following:

9. Download the Qualcomm Technologies proprietary code from ChipCode (see section 4.2.1).
10. Download other components from external websites by QSDK while building the default configuration (see section 4.2.2).
11. Generate the firmware by doing the following:
    a. Reassemble the code (see section 4.2.3.1).
    b. Create the QSDK build (see section 4.2.3.2).
12. Install the image in the flash memory of the device and boot using the image from the flash (see section 4.2.4).

It is recommended that you be familiar with the structure of directories that contain the SP images for the different subsystems before you download the code and build the images for loading. For more information, refer to the *QCA_Networking_2017.SPF.5.0 Product Family Overview* (80-YA935-1). For each SP included in an SPF, SP binary files are generated from the SI binary files of only a subset of the included SIs. In an SPF, some SIs may support multiple SPs while others may only support one SP.

4.2.1 Download packages available through ChipCode

Qualcomm Technologies proprietary code is available from ChipCode.

A web/GUI interface and a secure git server both allow access to this code. Browse available packages and obtain the download URL at https://chipcode.qti.qualcomm.com/ (see https://chipcode.qti.qualcomm.com/helpki/cloning-code-from-a-repository for more information).

See https://chipcode.qti.qualcomm.com/helpki for more information on installation and configuration of the correct version of git and OpenSSL on both Windows and Linux platforms that is required to support the authentication methods used by ChipCode.

4.2.2 Download packages from external websites

These components are downloaded by QSDK as needed while building the various profile configurations. QSDK may be further customized to download additional components; this list only contains the components that are necessary for at least one of the QSDK 2.0 default profiles.

Most packages listed are downloaded from external web sites, but several Qualcomm Technologies proprietary packages must be downloaded from a private access customer support account.

Packages	
LinuxART2CS10.2v4.9.423.tar.bz2	automake-1.11.3.tar.xz
ipkg-utils-1.7.tar.gz	quilt-0.48.tar.gz
xz-5.0.4.tar.bz2	gmp-5.0.5.tar.xz
m4-1.4.16.tar.gz	libelf-0.8.13.tar.gz
genext2fs-1.4.1.tar.gz	e2fsprogs-1.42.4.tar.gz
mklibs_0.1.34.tar.gz	mm-common-0.9.5.tar.bz2
sed-4.2.1.tar.bz2	util-macros-1.11.0.tar.bz2
yaffs2_android-2008-12-18.tar.bz2	xfce4-dev-tools-4.8.0.tar.bz2
cmake-2.8.9.tar.gz	mtd-utils-1.4.5.tar.gz
lzma-4.32.tar.bz2	mpfr-3.0.0.tar.bz2
scons-2.1.0.tar.gz	mpc-0.9.tar.gz
lzma-4.65.tar.bz2	uClibc-0.9.33.2.tar.bz2
occinelle-1.0.0-rc24.tar.gz	gdb-linaro-7.2-2011.03-0.tar.bz2
u-boot-2012.04.01.tar.bz2	gcc-linaro-4.6-2012.02.tar.bz2
squashfs4.2.tar.gz	linux-3.3.8.tar.bz2
libtool-2.4.tar.gz	binutils-2.22.tar.bz2
pkg-config-0.25.tar.gz	opkg-618.tar.gz
flex-2.5.35.tar.bz2	lua-5.1.4.tar.gz
squashfs3.0.tar.gz	zd1201-0.14-fw.tar.gz
bison-2.5.tar.bz2	hotplug2-201.tar.gz
autoconf-2.68.tar.bz2	1.0.4.3.arm
u-boot-2013.10.tar.bz2	util-linux-2.21.2.tar.xz
bridge-utils-1.5.tar.gz	readline-5.2.tar.gz
busybox-1.19.4.tar.bz2	miniupnpd-1.8.20130426.tar.gz
dnsmasq-2.72.tar.gz	angular-mocks-0.1-gdeea515.tar.gz
dropbear-2011.54.tar.bz2	angular-route-0.1-ge548b5b.tar.gz
ncurses-5.7.tar.gz	angular-translate-0.1-g5969f29.tar.gz
wireless_tools.29.tar.gz	coreutils-8.16.tar.xz
iproute2-3.3.0.tar.bz2	openssl-1.0.1e.tar.gz
iptables-1.4.10.tar.bz2	elfutils-0.155.tar.bz2
libnfnetlink-1.0.0.tar.bz2	iperf-2.0.5.tar.gz
zlib-1.2.7.tar.bz2	iputils-s20101006.tar.bz2
bzip2-1.0.6.tar.gz	jquery-contextmenu-1.01-gecb2ce1.tar.gz
gmp-4.3.1.tar.bz2	flot-0.8.0.zip
dosfstools-3.0.12.tar.gz	2.0.0beta10.tar.gz
argp-standalone-1.3.tar.gz	jquery.sparkline.min.js
ethtool-3.4.1.tar.xz	jquery-swapsies.js
fcgi-2.4.0.tar.gz	jquery-ui-1.8.21-gde5bb86.tar.gz
iozone3_420.tar	logrotate_3.8.1.orig.tar.gz
uClibc++-0.2.4.tar.bz2	libgcrypt-1.5.0.tar.bz2
angular-0.1-gffab5c4.tar.gz	p0f-3.06b.tgz
sysfsutils-2.1.0.tar.gz	quagga-0.99.21.tar.gz

jansson-2.4.tar.gz	ppp-2.4.5.tar.gz
jquery-1.7.2.min.js	tftp-hpa-0.48.tar.gz
libdaemon-0.14.tar.gz	libubox-2013-07-04-11e8afea0f7eb34f8c23a8e589ee659c46f3f8aa.tar.gz
libnetfilter_conntrack-0.9.1.tar.bz2	gdb-6.8a.tar.bz2
popt-1.7.tar.gz	samba-3.6.25.tar.gz
mcproxy-1.1.0.y.tar.bz2	curl-7.29.0.tar.bz2
libgpg-error-1.9.tar.bz2	jquery-flot-axislabels-0.1-ga0d11e5.tar.gz
libpcap-1.1.1.tar.gz	jquery-flot-gant-0.1-g90ec5b8.tar.gz
procps-3.2.8.tar.gz	libevent-2.0.19-stable.tar.gz
radvd-1.9.1.tar.gz	libxml2-2.7.8.tar.gz
v2.1.0.tar.gz	ntfs-3g_ntfsprogs-2011.4.12.tgz
redis-2.6.13.tar.gz	pure-ftpd-1.0.32.tar.bz2
linux-atm-2.5.2.tar.gz	rp-pppoe-3.10.tar.gz
strace-4.5.20.tar.bz2	ubus-2013-01-13-bf566871bd6a633e4504c60c6fc55b2a97305a50.tar.gz
sysstat-10.1.7.tar.bz2	uci-g424292.1.tar.gz
urijs-0.1-g2bdf950.tar.gz	uhttpd-2012-10-30-99f729378f69b2985c559bc8639b2edd06d75233.tar.gz
tcpdump-4.2.1.tar.gz	netifd-2013-05-13-bc4a4bb127622c76085ecec7fd20448aad7bafaf.tar.gz
wide-dhcpv6-20080615.tar.gz	luci-0.11.1.tar.gz
xtables-addons-1.42.tar.xz	util-linux-2.21.2.tar.xz
libnl-3.2.21.tar.gz	readline-5.2.tar.gz
yaml-0.1.4.tar.gz	miniupnpd-1.8.20130426.tar.gz
json-c-0.9.tar.gz	angular-mocks-0.1-gdeea515.tar.gz

4.2.3 Generate the firmware for QCA9531/QCA9558/QCA9563

To generate a firmware image, reassemble the code, create the QSDK build, and create a complete firmware image. This section describes the procedure to generate the firmware.

4.2.3.1 Reassemble the code

The first step is to reassemble the code from ChipCode and the Linux Foundation and generate the QSDK framework. The example given in this section assumes that all packages listed in section 4.2.1and 4.2.2 are downloaded and placed in the top-level directory:

1. Enter the following commands to reassemble the code and generate the QSDK framework:

```
$ git clone <chipcode-distro>
$ cd <chipcode directory>
$ git checkout r00008.1
```

2. After the copy of the existing Git repository is completed, the directories in which the files are present must be changed as described in the following table before the repo command is run.

All Packages:

Use git to obtain the following files from ChipCode and copy them to the working QSDK top-level directory of the device:	Local directory path to files fetched by git from ChipCode:
qsdk-qca-wifi qsdk-qca-wlan qsdk-ieee1905-security qsdk-qca-art qsdk-whc	NHSS.QSDK_MIPS.5.0\apss_proc\out\proprietary\Wifi
qca-lib	NHSS.QSDK_MIPS.5.0\apss_proc\out\proprietary\QSDK-Base
qca-wifi-fw-QCA9888_hw_2-WLAN.BL.3.5-00010-S-1.tar.bz2	WLAN.BL.3.5\cnss_proc\bin\hw.2
qca-wifi-fw-QCA9984_hw_1-WLAN.BL.3.5-00010-S-1.tar.bz2	WLAN.BL.3.5\cnss_proc\bin\hw.1
qca-wifi-fw-src-component-cmn-WLAN.BL.3.5-00010-S-1.tgz qca-wifi-fw-src-component-halphy_tools-WLAN.BL.3.5-00010-S-1.tgz	WLAN.BL.3.5\cnss_proc\src\components
qca-wifi-fw-AR9887_hw_1-CNSS.PS.2.5-00006-S-1.tar.bz2 qca-wifi-fw-AR9888_hw_2-CNSS.PS.2.5-00006-S-1.tar.bz2	CNSS.PS.2.5

3. After copying the necessary files to the appropriate directories, enter the following commands to continue with the process of generating the QSDK framework:

```
$ repo init -u git://codeaurora.org/quic/qsdk/releases/manifest/qstak -b release
   -m caf_AU_LINUX_QSDK_RELEASE_ENDIVE_MIPS_AA_TARGET_ALL.0.2.3173.029.xml --
   repo-url=git://codeaurora.org/tools/repo.git --repo-branch=caf-stable
$ repo sync
$ mkdir -p qsdk/dl
$ cp -rf NHSS.QSDK_MIPS.5.0/apss_proc/out/proprietary/Wifi/qsdk-qca-wifi/* qsdk
$ cp -rf NHSS.QSDK_MIPS.5.0/apss_proc/out/proprietary/Wifi/qsdk-qca-wlan/* qsdk
$ cp -rf NHSS.QSDK_MIPS.5.0/apss_proc/out/proprietary/Wifi/qsdk-qca-art/* qsdk
$ cp -rf NHSS.QSDK_MIPS.5.0/apss_proc/out/proprietary/Wifi/qsdk-ieee1905-
   security/* qsdk
$ cp -rf NHSS.QSDK_MIPS.5.0/apss_proc/out/proprietary/QSDK-Base/qca-lib/* qsdk
$ cp WLAN.BL.3.5/cnss_proc/src/components/* qsdk/dl
```

4. (Optional) This step applies only for QCA9531.ILQ.5.0/QCA9563.ILQ.5.0.

```
$ cp WLAN.BL.3.5/cnss_proc/bin/QCA9888/hw.2/* qsdk/dl
```

5. (Optional) This step applies only for QCA9558.ILQ.5.0.

```
$ cp WLAN.BL.3.5/cnss_proc/bin/QCA9984/hw.1/* qsdk/dl
```

6. Run the following commands for all chipsets, regardless of whether they are QCA9558.ILQ.5.0, QCA9531.ILQ.5.0, or QCA9563.ILQ.5.0.

```
$ tar xzvf WLAN.BL.3.5/cnss_proc/src/components/qca-wifi-fw-src-component-
   cmn-WLAN.BL.3.5-00010-S-1.tgz -C qsdk/dl
$ tar xzvf WLAN.BL.3.5/cnss_proc/src/components/qca-wifi-fw-src-component-
   halphy_tools-WLAN.BL.3.5-00010-S-1.tgz -C qsdk/dl
$ cp CNSS.PS.2.5/* qsdk/dl/
```

7. (Optional) This step applies only to **WHC** customers.

```
$ cp -rf NHSS.QSDK_MIPS.5.0/apss_proc/out/proprietary/Wifi/qsdk-whc/* qsdk
$ cp -rf NHSS.QSDK_MIPS.5.0/apss_proc/out/proprietary/Wifi/qsdk-whcpy/* qsdk
```

8. (Optional) This step applies only to **Hy-Fi** customers

```
$ cp -rf NHSS.QSDK_MIPS.5.0/apss_proc/out/proprietary/Hyfi/hyfi-
ar71xx/* qsdk
```

9. Optional) This step applies only to **PLC** customers

```
$ git clone <PLC Chipcode link>
$ cd <PLC chipcode directory>
$ git checkout r00008.1
$ cd ..
$ cp <PLC chipcode
directory>/NHSS.QSDK_MIPS.5.0/apss_proc/out/proprietary/qca-plc/*
qsdk/prebuilt/ar71xx/
```

The local directory qsdk is created by these repo steps as a sub-directory of the current working directory, from which repo was executed. This is the working QSDK top level directory.

4.2.3.2 Create the QSDK build

The QSDK framework has been developed using Ubuntu (from version 12.04 to version 16.04), and Debian. However, QSDK framework regenerates critical tools required to compile firmware at build-time. In that sense, the framework is independent from the host environment; although it is developed using the distributions above, it is expected to work on others such as RedHat, Mint, or Fedora. This command is for Debian/Ubuntu; it must be customized for other distributions:

```
$ sudo apt-get install gcc g++ binutils patch bzip2 flex make gettext \
  pkg-config unzip zlib1g-dev libc6-dev subversion libncurses5-dev gawk \
  sharutils curl libxml-parser-perl ocaml-nox ocaml-nox ocaml ocaml-findlib \
libpcre3-dev binutils-gold python-yaml
```

Because the framework automatically downloads the open source components, make sure an internet connection is active on the build host while creating the build.

To create the QSDK build, enter the following commands:

1. Install the different feeds in the build framework.

```
$ cd qsdk
$ make package/symlinks
```

2. Copy the base configuration to use for the build.

Premium Beeliner	`$ cp qca/configs/qca955x.ln/ar71xx_premium_beeliner.config .config`
Target	`$ cp qca/configs/qca955x.ln/ar71xx_target.config .config`

3. Regenerate a complete configuration file and start the build:

```
$ make defconfig
```

4. (Optional) This step applies only for QCA9531.ILQ.5.0/QCA9563.ILQ.5.0.

```
$ sed -i -e '/CONFIG_PACKAGE_qca-wifi-fw-hw7-10.4-asic/d' .config
$ sed -i -e '/CONFIG_PACKAGE_qca-wifi-fw-hw9-10.4-asic/d' .config
$ make V=s
```

5. (Optional) This step applies only for QCA9558.ILQ.5.0.

```
$ sed -i -e '/CONFIG_PACKAGE_qca-wifi-fw-hw7-10.4-asic/d' .config
$ sed -i -e '/CONFIG_PACKAGE_qca-wifi-fw-hw10-10.4-asic/d' .config
$ make V=s
```

4.2.4 Load the flash image and boot the device

To set up the flash memory environment, do the following:

1. As a preliminary step, ensure that the board console port is connected to the PC using these RS232 parameters:
   - 115200bps
   - 8N1
2. Confirm that the PC is connected to the board using one of the Ethernet ports. The PC must have a TFTP server launched and listening on the interface to which the board is connected. At this stage power up the board and, after a few seconds, press any key during the countdown.

To load the image in flash and boot the platform using the image from flash, do the following:

All flashing commands start with the following U-Boot configuration. The IP address and the TFTP server IP address must reflect the current network topology. To ensure this, run these commands from U-Boot:

Premium_Beeliner 16 MBytes Flash

```
setenv bc <BOARD_NAME>
setenv ipaddr <YOUR_IP@>
setenv serverip <YOUR_TFTP_SERVER_IP@>
setenv bootcmd 'bootm 0x9fe80000'
tftp 0x80060000 openwrt-ar71xx-${bc}-qca-legacy-uboot.bin && erase
0x9f000000 +0x30000 && cp.b $fileaddr 0x9f000000 $filesize
setenv lok 'tftp 0x80060000 openwrt-ar71xx-generic-${bc}-kernel.bin &&
erase 0x9fe80000 +${filesize} && cp.b $fileaddr 0x9fe80000 0x160000'
setenv lof 'tftp 0x80060000 openwrt-ar71xx-generic-${bc}-rootfs-
squashfs.bin && erase 0x9f050000 +${filesize} && cp.b $fileaddr 0x9f050000
$filesize'
setenv lqsdk 'run lof && run lok'
saveenv
run lqsdk
```

The board name for the QCA9561.AP.DF03.1 is AP151-16M and for the AP152 is AP152-16M and for the QCA9531.AP.HB03.1 is AP147-16M. Reset the device after loading the device with the U-Boot binary from the flash memory.

4.2.4.1 Upgrade the firmware

This release has a feature to upgrade images from the OpenWrt web interface without the need for a TFTP server. After loading the device using the first image from flash memory, any future upgrades can be done from the web interface.

4.3 PLC SON interface mapping

- In AP152 + QCA9886 platform, a single GMAC has been shared between PLC backhaul and the legacy Ethernet port.

```
uci set plc.config.PlcIfname='eth0.1'
uci commit
```

- In RE, eth0.1 and eth0.2 can be attached to br-lan as follows:

```
uci set network.lan.ifname ='eth0.1 eth0.2'
uci commit
```

  In CAP, eth0.2 is identified as WAN interface for broadband connectivity.

- When only PLC is used as backhaul between CAP and RE, use the following command on RE for auto configuration cloning through the PLC interface.

```
uci set repacd.repacd.DefaultREMode='son'
uci commit
```

4.4 Related documentation for QCA9531/QCA9558/QCA9563

For more details on QCA9531/QCA9558/QCA9563, refer to *QCA_NETWORKING_2017-SPF.5.x Related Documents for Reference* (80-YA760-2).

5 New features

This chapter describes the new features and enhancements in the QCA_Networking_2017.SPF.5.0 release. Also, the mapping of these features with the supported chipsets and corresponding SPs is presented here. A ✓ (tick mark) indicates that the feature is supported in a particular chipset and SP combination. A blank cell indicates that the feature is not supported in a particular chipset and SP combination.

For an overview of release contents, refer to *QCA_Networking_2017.SPF.5.0 Product Family Overview* (80-YA935-1).

	SP	**IPQ4019.ILQ.5.0**				**IPQ8064.ILQ.5.0**					**QCA9531.ILQ.5.0 (applies to QCA9531)**				**QCA9531.ILQ.5.0 (applies to QCA9558)**			**QCA9531.ILQ.5.0 (applies to QCA9563)**			
	Radio	**IPQ 4019**	**QCA 9886**	**QCA 9984**	**QCA 9889**	**QCA 9984**	**QCA 9980**	**AR 9380**	**QCA 9880**	**QCA 9889**	**QCA 9880**	**QCA 9889**	**QCA 9531**	**QCA 9886**	**QCA 9984**	**QCA 9558**	**QCA 9880**	**QCA 9889**	**QCA 9563**	**QCA 9880**	**QCA 9886**
	Kernel	**3.14**				**3.14**					**3.3.8**				**3.3.8**			**3.3.8**			
No.	**Feature**																				
1.	HNAT Support on QCA9531 Platform with QCA8337N															✓			✓		
2.	Enabling AES192 crypto operation using slow path					✓	✓	✓	✓	✓											
3.	Increase the number of crypto sessions					✓	✓	✓	✓	✓											
4.	QRFS enablement for VLAN Wireless Network Isolation	✓																			
5.	Support QT embedded framework and demo GUI	✓	✓	✓	✓																
6.	Upgrade to a newer busybox version	✓	✓	✓	✓	✓	✓	✓	✓	✓											
7.	LK bootloader support for sec.dat fusing	✓	✓	✓	✓																

No.	SP	IPQ4019.ILQ.5.0				IPQ8064.ILQ.5.0					QCA9531.ILQ.5.0 (applies to QCA9531)				QCA9531.ILQ.5.0 (applies to QCA9558)			QCA9531.ILQ.5.0 (applies to QCA9563)			
	Radio	IPQ 4019	QCA 9886	QCA 9984	QCA 9889	QCA 9984	QCA 9980	AR 9380	QCA 9880	QCA 9889	QCA 9880	QCA 9889	QCA 9531	QCA 9886	QCA 9984	QCA 9558	QCA 9880	QCA 9889	QCA 9563	QCA 9880	QCA 9886
	Kernel	3.14				3.14					3.3.8				3.3.8			3.3.8			
	Feature																				
8.	Support additional LK boot loader features	✓	✓	✓	✓																
9.	Support crypto operation in LK bootloader	✓	✓	✓	✓	✓	✓	✓	✓	✓											
10.	Band Steering with QWRAP	✓	✓	✓		✓	✓				✓	✓	✓	✓							
11.	Wi-Fi SON: Channel planning on distributed Wi-Fi systems	✓	✓	✓	✓	✓	✓	✓	✓	✓											
12.	Tx Mirror (Direct attach)							✓					✓			✓			✓		
13.	AMPDU per VAP (Direct Attach)							✓					✓			✓			✓		
14.	Wi-Fi SON: Support on MIPS platforms and PLC-enabled MIPS										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
15.	PLC SON uses cases 7/8 for all PLC platforms	✓	✓	✓	✓																
16.	MIPS PLC SON Support										✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
17.	Support for cloud based channel change	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
18.	Printk – code cleanup	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
19.	Add API or CLI to add/delete/modify 11k neighborhood report content	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
20.	Improved dynamic beacon	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

No.	SP	IPQ4019.ILQ.5.0				IPQ8064.ILQ.5.0					QCA9531.ILQ.5.0 (applies to QCA9531)				QCA9531.ILQ.5.0 (applies to QCA9558)			QCA9531.ILQ.5.0 (applies to QCA9563)			
	Radio	IPQ 4019	QCA 9886	QCA 9984	QCA 9889	QCA 9984	QCA 9980	AR 9380	QCA 9880	QCA 9889	QCA 9880	QCA 9889	QCA 9531	QCA 9886	QCA 9984	QCA 9558	QCA 9880	QCA 9889	QCA 9563	QCA 9880	QCA 9886
	Kernel	3.14				3.14					3.3.8				3.3.8			3.3.8			
No.	Feature																				
21.	WFA - Multi Band Operation (MBO)	✓	✓	✓		✓	✓							✓	✓						✓
22.	IEEE 802.11 update: Extended NSS signaling		✓	✓		✓								✓	✓						✓
23.	QCN: Channel/Band capability and preference	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
24.	AMPDU per VAP (Offload)	✓	✓	✓		✓	✓		✓		✓			✓	✓		✓			✓	✓
25.	Identify and clean dead code on MAC core and data path	✓	✓	✓		✓	✓							✓	✓						✓
26.	Optimize multiple STATS and MESH features	✓	✓	✓		✓	✓							✓	✓						✓
27.	Enabling AK+CAS/BL as the IoT Central platform					✓	✓	✓	✓	✓											
28.	Wi-Fi SON: ATF enhancement (Strict/fair) (QCA9980)	✓	✓	✓		✓								✓	✓						✓
29.	Macro, indirection, and test command optimization	✓	✓	✓		✓	✓							✓	✓						✓
30.	Support Dynamic WAN/LAN selection for single net device	✓	✓	✓	✓																
31.	Qualcomm IE support (WIN)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

No.	SP	IPQ4019.ILQ.5.0				IPQ8064.ILQ.5.0					QCA9531.ILQ.5.0 (applies to QCA9531)				QCA9531.ILQ.5.0 (applies to QCA9558)			QCA9531.ILQ.5.0 (applies to QCA9563)			
	Radio	IPQ 4019	QCA 9886	QCA 9984	QCA 9889	QCA 9984	QCA 9980	AR 9380	QCA 9880	QCA 9889	QCA 9880	QCA 9889	QCA 9531	QCA 9886	QCA 9984	QCA 9558	QCA 9880	QCA 9889	QCA 9563	QCA 9880	QCA 9886
	Kernel	3.14				3.14					3.3.8				3.3.8			3.3.8			
No.	Feature																				
32.	AMSDU_AMPDU support per VAP (Direct Attach)							✓					✓			✓			✓		
33.	Knob to control packet detection threshold (Adaptive Noise Immunity)	✓	✓	✓		✓	✓							✓	✓						✓
34.	IPQ4019 SW: Support dual band switchable feature	✓																			
35.	Spectral Analysis Debug Enhancements	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
36.	Runtime Core Dump for QCA9880/QCA9984	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓		✓	✓
37.	Connection State Logging		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
38.	MBO Neighbors Info Sharing (for WIN)	✓	✓	✓		✓	✓							✓	✓						✓
39.	Smart Logging Enhancement	✓	✓	✓		✓								✓	✓						✓
40.	10 ms TXOPs in Europe	✓	✓	✓		✓								✓	✓						✓
41.	FW config recovery in event of FW assert	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓		✓	✓		✓	✓
42.	Wi-Fi SON: Backhaul credential cloning for multi-SSID traffic separation	✓	✓	✓		✓	✓	✓	✓		✓		✓	✓	✓	✓	✓		✓	✓	✓
43.	CE Diagnostics Framework	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓		✓	✓		✓	✓

No.	SP	IPQ4019.ILQ.5.0				IPQ8064.ILQ.5.0					QCA9531.ILQ.5.0 (applies to QCA9531)				QCA9531.ILQ.5.0 (applies to QCA9558)			QCA9531.ILQ.5.0 (applies to QCA9563)			
	Radio	IPQ 4019	QCA 9886	QCA 9984	QCA 9889	QCA 9984	QCA 9980	AR 9380	QCA 9880	QCA 9889	QCA 9880	QCA 9889	QCA 9531	QCA 9886	QCA 9984	QCA 9558	QCA 9880	QCA 9889	QCA 9563	QCA 9880	QCA 9886
	Kernel	3.14				3.14					3.3.8				3.3.8			3.3.8			
No.	Feature																				
44.	TDMA support for Wave2 radios	✓	✓	✓		✓								✓	✓						✓
45.	Buffer tracking	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓
46.	Enhancement of ACS report (2G + 5G)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
47.	Support Offload based Radios to connect to iPhone Wi-Fi Hotspot	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓		✓	✓
48.	Client statistics	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
49.	WiFi SON: Multi-node WPS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
50.	QCN: Send Broadcast Probe Response when STA supports it	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
51.	QTF coverage for block ack and VDEV modules	✓	✓	✓		✓	✓							✓	✓						✓
52.	WLAN FW - QTF Coverage for Connection - MAC core module	✓	✓	✓		✓	✓							✓	✓						✓
53.	Tx Mirror (QCA9880)								✓		✓						✓			✓	
54.	AMSDU_AMPDU Support per VAP (Offload)	✓	✓	✓		✓								✓	✓						✓
55.	Provide HYD Version	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
56.	BLE Co-existence to be tested at a BLE interval of 100 ms/10 sec	✓				✓															

No.	SP	IPQ4019.ILQ.5.0				IPQ8064.ILQ.5.0					QCA9531.ILQ.5.0 (applies to QCA9531)				QCA9531.ILQ.5.0 (applies to QCA9558)			QCA9531.ILQ.5.0 (applies to QCA9563)			
	Radio	IPQ 4019	QCA 9886	QCA 9984	QCA 9889	QCA 9984	QCA 9980	AR 9380	QCA 9880	QCA 9889	QCA 9880	QCA 9889	QCA 9531	QCA 9886	QCA 9984	QCA 9558	QCA 9880	QCA 9889	QCA 9563	QCA 9880	QCA 9886
	Kernel	3.14				3.14					3.3.8				3.3.8			3.3.8			
No.	Feature																				
57.	Configurable GI in any MCS	✓	✓	✓		✓								✓	✓						✓
58.	AES-CTR mode in Crypto driver					✓	✓	✓	✓	✓											
59.	Tx chain-mask changing without reconnection	✓	✓	✓	✓	✓	✓	✓	✓	✓											
60.	Wi-Fi Qualcomm® FSM™ Support								✓												
61.	Wi-Fi SON: Bandsteering enable/disable on Per SSID pair basis	✓	✓	✓	.	✓	✓	✓	✓		✓		✓	✓	✓	✓	✓		✓	✓	✓
62.	Wi-Fi SON: Match[1] SSID to Support AP and Band steering	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
63.	Reduce false positives for DFS (supported for 5 GHz radios only)		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	✓		✓	✓
64.	IPQ8064 SW: Support newer version of devicetree parser	✓	✓	✓	✓	✓	✓	✓	✓	✓											
65.	Support Micron 1 GB flash	✓	✓	✓	✓																
66.	Support AP161 with Toshiba eMMC					✓	✓	✓	✓	✓											
67.	Support Winbond DDR W631GU6KS-12	✓	✓	✓	✓																
68.	Support Samsung DDR K4B2G1646F-BYMA	✓	✓	✓	✓																

No.	SP	IPQ4019.ILQ.5.0				IPQ8064.ILQ.5.0					QCA9531.ILQ.5.0 (applies to QCA9531)				QCA9531.ILQ.5.0 (applies to QCA9558)			QCA9531.ILQ.5.0 (applies to QCA9563)			
	Radio	IPQ 4019	QCA 9886	QCA 9984	QCA 9889	QCA 9984	QCA 9980	AR 9380	QCA 9880	QCA 9889	QCA 9880	QCA 9889	QCA 9531	QCA 9886	QCA 9984	QCA 9558	QCA 9880	QCA 9889	QCA 9563	QCA 9880	QCA 9886
	Kernel	3.14				3.14					3.3.8				3.3.8			3.3.8			
No.	Feature																				
69.	Support Samsung eMMC KLM8G1GEME-B031	✓	✓	✓	✓																
70.	Authorized OTP programming in the field	✓	✓	✓	✓																
71.	Wi-Fi SON: AP[1] Steering of legacy clients (monitor mode)	✓	✓	✓		✓	✓	✓	✓												
72.	QCN: Unsolicited BSS transition management	✓	✓	✓		✓	✓	✓	✓		✓		✓	✓	✓	✓	✓		✓	✓	✓
73.	Wi-Fi SON: Mechanism for knowing when peer joins/leaves	✓	✓	✓		✓	✓	✓	✓		✓		✓	✓	✓	✓	✓		✓	✓	✓
74.	FILS SK and FILS[1] HLS (QCN)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

1. Match SSID to support AP and band steering, AP steering of legacy clients (monitor mode), and FILS SK and HLS (QCN) features are of ED quality.

6 Known and resolved issues

This chapter lists the known and resolved issues in hardware and software in this release.

6.1 Known issues

No.	Title	Customer Impact
1.	Not able to meet expected KPI value while trying linearity testing when both AP and RTT bandwidths are in 2 GHz HT40/HT20/NG.	**Impact:** Low RTT KPIs are not met. Impact is low as there are no known customers for RTT.
2.	AP multicast video traffic to 8 clients results in huge RTP packets lost + MLR is high (greater than 40 ms).	**Impact:** High Packet loss is seen while running multicast video traffic.
3.	VoW AT&T U-verse DVR service is not working.	**Impact:** High U-verse DVR service does not work.
4.	Throughput degrades from ~891 Mbps to 541 Mbps for allocating 50% for each SIID in fair mode. UDP throughput drops up to ~200 Mbps when two SSIDs are in place.	**Impact:** High Likely to be a problem in DF.
5.	RE's front haul VAPs are not down when PLC connection is removed at the CAP side in req-7.	**Impact:** High Issue seen only when PLC link is pulled from the CAP side. With just PLC link between CAP and RE, STA will be connected without any backhaul.
6.	PLC entry is not removed from CAP's topology discovery after removing the RE's PLC connection.	**Impact:** Low No functionality impact. Td entries not updated on the remote device when the PLC connection is unplugged. When the connection is restored everything is back to normal.
7.	Load balancing brings down the performance up to 50 Mb in few cases.	**Impact:** Low Load balancing feature is hogging CPU to 100 % which brings down the performance in few cases. There are issues in ARM for load balancing.
8.	Legacy client steering is not triggered always even when RSSI criteria is met.	**Impact:** High AP Steering of legacy clients (monitor mode): This feature is of ED quality for this release; the issue will be fixed in the upcoming release.
9.	Guest network feature has steering issues.	**Impact:** Low Guest network is still under development and this issue is related to stability.

No.	Title	Customer Impact
10.	Matching SSID cannot participate in AP steering all the time.	**Impact:** Low Steering success rate was not 100 % from RE to CAP steering failure. Match SSID to support AP and band steering: This feature is of ED quality for this release; the issue will be fixed in the upcoming release.
11.	Guest hyd is not starting when the guest bridge interface is not attached.	**Impact:** Low Running two instances of hyd for QCA9531 takes more memory. For this release, guest network for QCA9531 alone is not supported.
12.	Regression SmartLogEvent: CE failure event − makes 5 GHz backhaul to disconnect in IPQ4019.	**Impact:** High Running peak performance with UDP uplink will make 5 GHz to disconnect and reconnect.
13.	RE 2 GHz backhaul does not connect if Ethernet is unplugged/replugged between CAP and RE.	**Impact:** Low If Ethernet backhaul is removed and reconnected multiple times, 2 GHz backhaul is not connecting in RE. 5 GHz backhaul is available for the regular functionality to work.
14.	Observing low throughput with iphone7, 7plus, and MBP as STA with secure mode enabled in competitive bench marking tests.	**Impact:** High
15.	False radar detection is seen while running traffic test in dense environment within 8 hours.	**Impact:** High Radar detection observed in dense environment. Experiments are being done on discussion with the algorithm team.
16.	Spectral scan continuously detects CW, Wi-Fi and FHSS in clean RF environment when CW is on.	**Impact:** Low SigGen does not yield this issue. Issue is seen when iGen (or) video bridge is used.
17.	Observed MU-MIMO low throughput [~681−504 Mbps]. This issue is not observed in AP148 for the same build.	**Impact:** Low Limited to Veriwave tests and not seen with real clients.

6.2 Resolved issues

No.	Title	Customer Impact
1.	Approximately 150 Mbps low throughput observed TCP_UL in 4x4 QCA9984. Expected TCP_UL: >1200 Mbps; Obtained: ~1075Mbps	**Impact:** Low Primarily due to supplicant running in open mode. This issue is seen in the supplicant version 2.6.
2.	2.4 GHz station is not associating with HT40+ mode; however, association is happening in HT20 mode.	**Impact:** High Works fine with AP135. This issue is to be fixed in AP161.

7 QDART_Connectivity

This chapter describes the QDART_ Connectivity 1.0.43 and QDART Windows PCIE Support Package 4.3. Refer *QCA99xx QDART Connectivity User Guide* (80-Y8050-1) for more information.

Contact Litepoint and NI for 160 MHz updated software package.

160-wide channel support requires using QTI WLAN SCPI interface.

The following WLAN test equipment is supported exclusively with the QTI WLAN SCPI interfaces. Non-SCPI interfaces for these test instruments are not supported with QDART-CONN 1.0.42.

- Anritsu 8870A
- LitePoint IQxel80/IQxel160
- NI PXI 5644/5645/5646

The following functionalities are supported in this build:

- QDART_Connectivity components running on a Windows 7 PC
- Windows 7 PC (with PCIE slot) STA configuration support
- Tx power calibration
- Instrument support: NI MIMO, LP MIMO
- Single instrument: LP IQxel 80 or 160
- FTM support with 1x1 LP
- Cal data saved in EEPROM on the applicable radio module
- Cal data saved into Flash (NOR and NAND), based on the platform support
- Able to load board data file
- All chain mask combinations including 4x4
- Tx single tone at carrier frequency
- Tx99 support
- Concurrent Tx of 2 GHz and 5 GHz cards
- Link test (DUT to DUT)
- Tx power accuracy test
- Frequency accuracy test
- Tx mask test
- Target power vs Tx EVM sweep
- Rx waterfall sweep
- Enterprise half and quarter rates
- Backup/restore of calibration data
- Rx gain and Noise Floor (NF) calibration
- CCA threshold setting are set during Rx gain/NF calibration

- Provide capability to get MAC from board data
- XTAL calibration
- Support for QCA99x chips with ATE information programmed into static section of OTP
- Resolved issue with Rx Gain and NF calibration results not properly saved into the board data information area
- Include several Microsoft re-distributable DLLs to eliminate installation difficulties that may be experienced on some PCs
- For the PC station configuration, the Windows system must be set up to run in test mode before the driver can be installed
- Instrument support: LP MIMO is for a configuration of two IQxel 160s