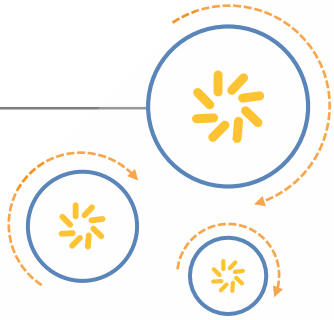
Qualcomm Atheros, Inc.



IPQ40x8

QSDK Setup and User Guide

80-Y6399-3 Rev. B

September 25, 2015

For additional information or to submit technical questions go to https://createpoint.qti.qualcomm.com/

Qualcomm Atheros, Inc.
1700 Technology Drive
San Jose, CA 95110
U.S.A.

Revision history

Revision	Date	Description
A	July 2015	Initial release
B	September 2015	Updated the setting up QSDK development environment section

Contents

Figures

Tables

1 Overview

Qualcomm Atheros hardware designs are shipped with a bootloader; customers can use TFTP to download the QSDK image.

1.1 Supported hardware and software

The following hardware designs and software profiles are being released in QSDK 3.0.

Hardware	Profiles
DK01	Standard, Premium
DK04	Premium

2 Customize and Build Firmware

Once the development environment has been set up (see Setting up QSDK Development Environment), the reference firmware can be customized to suit specific requirements. The simplest customization approach is to start with a predefined profile that best matches the type of product to be created, add and subtract packages, and change default configurations. For more complex customization, it is best to bundle the enhancements into 'packages' that can be easily added to any of the Qualcomm Atheros provided baseline profiles.

Figure 2-1 shows the quickest path for customers to take advantage of building products from the QSDK baseline profiles, by adding their own value-added packages.

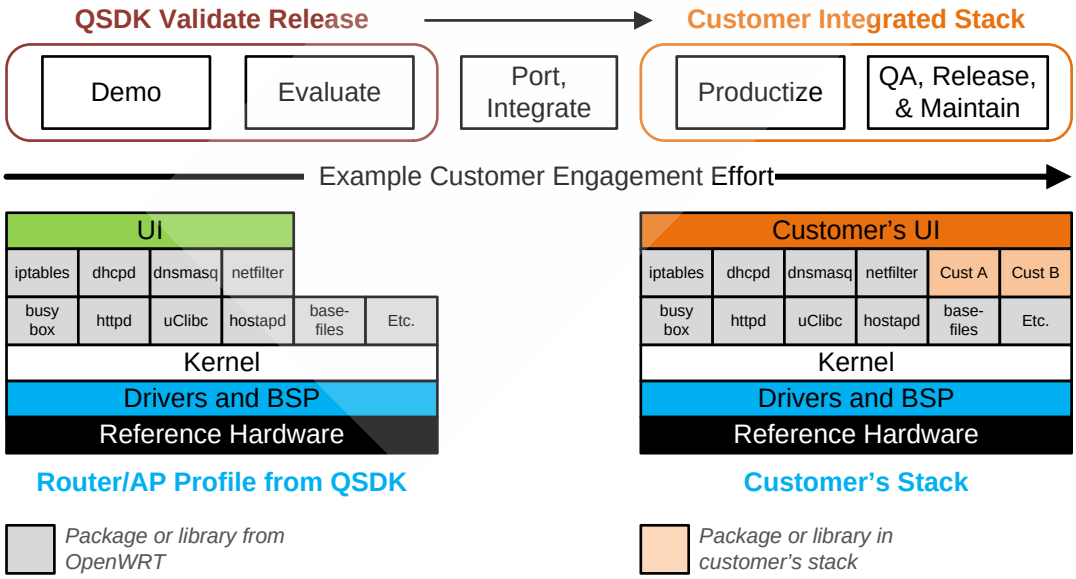


Figure 2-1 Path to build products from QSDK baseline evaluation profiles

2.1 Add customer software into QSDK build

The information on adding customer software into QSDK build can be found at:

http://wiki.openwrt.org/doc/devel/packages

2.2 Change Linux kernel configuration

The Linux kernel configuration can be changed using the following command:

```
$ make kernel_menuconfig
```

3 Baseline Firmware Profiles

Baseline profiles have functionality and default configurations akin to the needs of various product segments. The Qualcomm® Internet Processor (IPQ) platform releases are planned to eventually include two different profiles, as shown in Table 3-1. For early software releases, the profiles may be the same or similar.

Table 3-1 QSDK baseline profiles

Profile	Description
Standard	Wi-Fi router using Qualcomm Atheros proprietary driver for 16 MB flash (reduced set of packages)
Premium	Wi-Fi router using Qualcomm Atheros proprietary driver for 32 MB flash

3.1 Default configuration

By default, the router starts up with the standard OpenWrt configuration. It is common to all profiles, as listed here:

- WAN
  - One Ethernet port
  - DHCP client
  - NAT enabled
  - NTP client
- LAN/WLAN (bridge)
  - Includes all the other Ethernet ports
  - Includes all the Wi-Fi radios (one VAP per radio)
  - DHCP server enabled
- WebUI accessible from the LAN
- Firewall enabled on the WAN

3.2 Profile definitions

QSDK offers multiple profiles (build images) to represent different product definitions running on the IPQ4018, IPQ4019, IPQ4028, and IPQ4029 chips. They differ in the included set of packages, and therefore in the features set they provide.

Table 3-2 summarizes the features available on each profile.

Table 3-2 Profile definitions

Area	Feature	Standard/Premium
WAN mode	Static IP	Yes
	DHCP client	Yes
	PPtP	Yes
	L2TP	Yes
	PPPoE	Yes
Firewall	NAT	Yes
	Connection filtering	Yes
	Port forwarding	Yes
	DMZ	Yes
Bridging/Routing	IPv4/IPv6	Yes
	RIP	Yes
	IGMP proxy	Yes
	VLAN configuration	Yes
	Dynamic DNS	Yes
	UPnP	Yes
Network share	Windows file sharing (Samba)	Yes
	Printers	Yes
Mass storage support	USB - MsDos FS	Yes
	USB - VFAT	Yes
	USB - NTFS	Yes
	USB - Ext2/3/4	Yes
System	Firmware upgrade	Yes
	WebUI (LuCI)	Yes
Hardware related features	Crypto acceleration	Yes

4 Setting up QSDK Development Environment

4.1 Set up Linux workstation

1. Make sure these packages are installed on the system to be able to build QSDK. Refer to the system distribution documentation for information on installing these packages.
   - gcc, g++, binutils, libc headers
   - gawk, patch, bzip2, flex, gettext, pkg-config, unzip, libz-dev, sharutils, libncurses-dev, curl
   - make, subversion, git

   On an Advanced Packaging Tool (APT) based system such as Ubuntu, KUbuntu, or Debian, this can be done using these commands:

```
$ sudo apt-get install gcc g++ binutils patch autoconf libcurl4-openssl-
dev bzip2 flex make \
gettext pkg-config unzip zlib1g-dev libc6-dev subversion libncurses5-
dev \
gawk sharutils curl libxml-parser-perl python-yaml git \
ocaml-nox ocaml ocaml-findlib
```

NOTE: Qualcomm Atheros builds have been tested on these Linux distributions: Ubuntu 12.04, Ubuntu 12.10, Ubuntu 13.04, Debian Squeeze, Debian, and Wheezy.

2. Configure git with OpenSSL

```
git clone https://github.com/git/git
cd git
git checkout v1.8.2
make configure
./configure --with-openssl --prefix=/usr
make all
sudo make install
cd ..
rm -fr git
```

3. Install repo

```
mkdir -p ~/bin
export PATH=~/bin:$PATH
curl https://commondatastorage.googleapis.com/git-repo-downloads/repo
~/bin/repo
chmod a+x ~/bin/repo
```

4.2 Download, unpack, and build QSDK

Obtain the QSDK reference software package. Place it in the working directory, and unpack the source code.

4.2.1 Download and reassemble the code

As Qualcomm Atheros does not distribute the complete open source package anymore, the first step is to generate the QSDK framework by reassembling the code from openwrt.org and Qualcomm Atheros. Refer to the 'reassemble the code' section in the corresponding release notes for the complete set of commands.

4.2.2 Customize the firmware

The release notes explains how to build the baseline profiles in section 3. The build may be fully customized for changing default components, target platform, and other options, by running this command:

```
$ make menuconfig
```

The profile can be selected in "Target Profile". The baseline profiles build the supported Qualcomm Atheros based designs (standard profile supports DK01 and premium profile supports DK01 as well as DK04).

During the build process, the framework automatically downloads all the packages from the web. To speed-up the build process, copy the downloaded packages to a local mirror, and set that mirror as the first place to look for packages that are used to build the image.

The example below points to an internal mirror, accessible to Qualcomm Atheros employees on the internal network only.

```
[*] Advanced configuration options (for developers) --->
     → Local mirror for source packages
         → ''http://<YOUR_MIRROR_HERE>''
```

Figure 4-1 through Figure 4-3 show sample screen shots from the make menuconfig display.

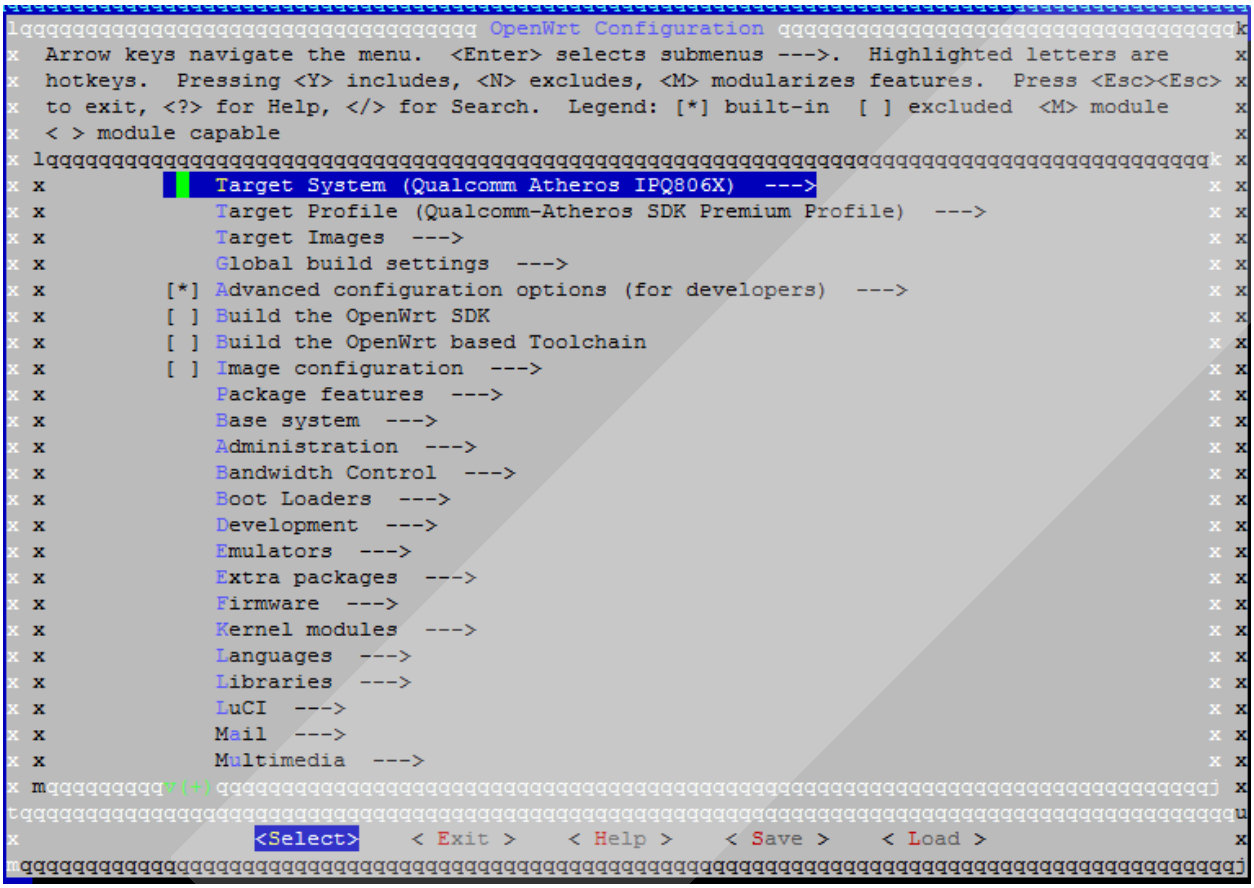


Figure 4-1 Configuring OpenWrt options

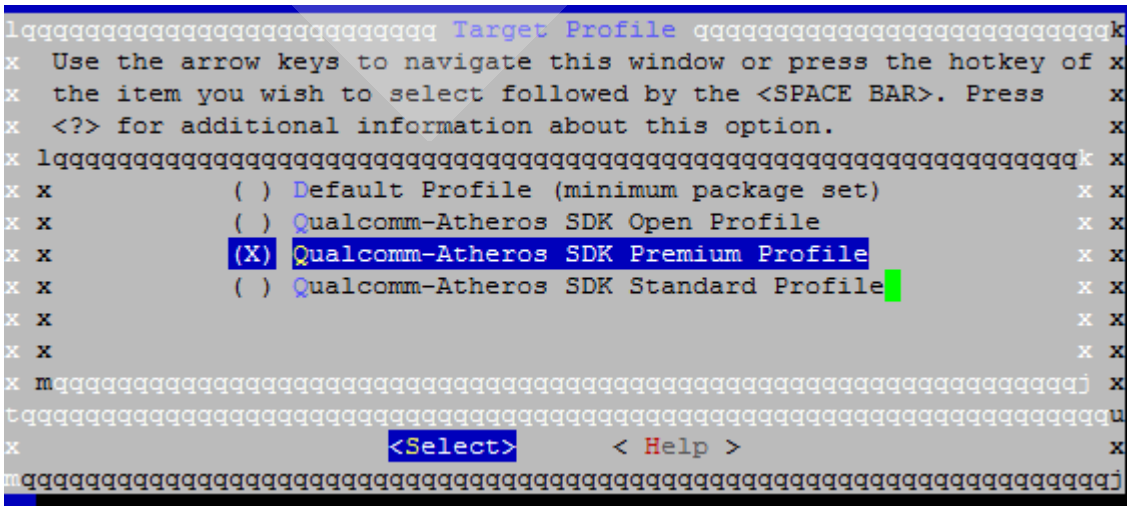


Figure 4-2 Selecting target profile

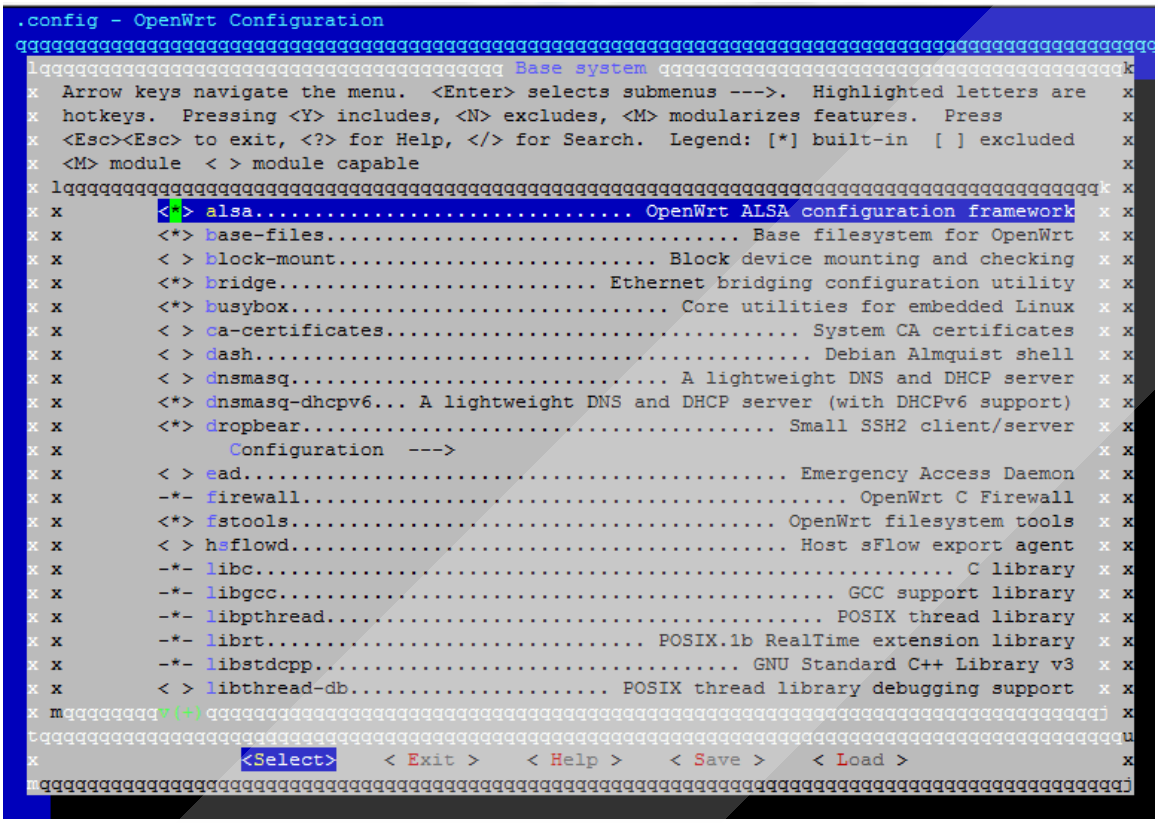


Figure 4-3 Configuring base system

After all changes to the default profile are completed and saved, exit the menuconfig, and run "make" to start the build process.

```
$ make V=s
```

NOTE: The *V=s* option is not mandatory; it is used to display the build information during the build process. This option can be omitted if there is no interest in seeing the build log, but it is quite useful to be sure that the build process is running, since it may take a few hours to complete, depending on the processor speed and memory of the build PC.

4.2.3 Build results

After the build completes, the binaries should be available in the following directory:

```
$ ls bin/ipq/
md5sums
openwrt-ipq40xx-u-boot.elf
openwrt-ipq40xx-u-boot.img
openwrt-ipq40xx-u-boot-stripped.elf
openwrt-ipq806x-qcom-ipq40xx-ap.dk01.1-c1.dtb
openwrt-ipq806x-qcom-ipq40xx-ap.dk01.1-c1-fit-uImage-initramfs.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dk01.1-c1-fit-uImage.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dk01.1-c2.dtb
openwrt-ipq806x-qcom-ipq40xx-ap.dk01.1-c2-fit-uImage-initramfs.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dk01.1-c2-fit-uImage.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c1.dtb
openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c1-fit-uImage-initramfs.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c1-fit-uImage.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c2.dtb
openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c2-fit-uImage-initramfs.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c2-fit-uImage.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c3.dtb
openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c3-fit-uImage-initramfs.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dk04.1-c3-fit-uImage.itb
openwrt-ipq806x-qcom-ipq40xx-fit-uImage.itb
openwrt-ipq806x-qcom-ipq40xx-ap.dkxx-fit-uImage.itb
openwrt-ipq806x-squashfs-root.img
openwrt-ipq806x-vmlinux.bin
openwrt-ipq806x-vmlinux.elf
openwrt-ipq806x-vmlinux-initramfs.bin
openwrt-ipq806x-vmlinux-initramfs.elf
packages
```

The following files are required to generate a new image for DK01:

- openwrt-ipq40xx-u-boot-stripped.elf
- openwrt-ipq806x-squashfs-root.img
- openwrt-ipq806x-qcom-ipq40xx-ap.dkxx-fit-uImage.itb

4.2.4 Firmware reflash

Refer to the ‘Flash the firmware’ section in the IPQ4019.ILQ.1.0 Release Notes for the latest instructions.

5 Installing Post-load Applications

To download post-load applications, setup a HTTP server which makes the '**bin/ipq806x/packages**' folder accessible from the network.

Follow these steps for downloading:

- Copy the **bin/ipq806x/packages** folder to the root of the web server.
- Point the target to this new server by modifying the opkg.conf file. The first line should be as follows:

  ```
  src/gz packages http://<YOUR_SERVER_IP@>/packages
  ```

- Once it is done, packages are installable using the following commands:

  ```
  $ opkg update
  $ opkg install <package_name>
  ```

If a web server is not available, copy the respective *.ipk to the device manually, and install them using the command:

```
$ opkg install /path/to/file/<package_name>
```

6 Unified Configuration Interface (UCI)

6.1 Overview

The OpenWrt configuration is done using a decentralized scripting mechanism. Each package is responsible for providing the configuration script to configure itself and manage its own section of the configuration database.

To facilitate the process, OpenWrt provides a simple and efficient database mechanism. UCI stores the data on the file system (in /etc/config) and provides a database-like interface to it. The following commands are used when configuring packages through UCI:

```
$ uci show
$ uci get <variable_name>
$ uci set <variable_name>=<new_value>
$ uci commit
```

NOTE: UCI is only the software which allows to store and access the data.

After modifying some parameters, it is usually necessary to call a script to take care of accessing the UCI data and configure itself accordingly.

More uci commands can be displayed by running: **root@OpenWrt:/lib/config# uci**

Usage: uci [<options>] <command> [<arguments>]

```
Commands:
   batch
   export     [<config>]
   import     [<config>]
   changes    [<config>]
   commit     [<config>]
   add        <config> <section-type>
   add_list   <config>.<section>.<option>=<string>
   show       [<config>[.<section>[.<option>]]]
   get        <config>.<section>[.<option>]
   set        <config>.<section>[.<option>]=<value>
   delete     <config>[.<section[.<option>]]
   rename     <config>.<section>[.<option>]=<name>
   revert     <config>[.<section>[.<option>]]
```

More information about UCI can be found at: http://wiki.openwrt.org/doc/uci

Table 6-1 Example UCI commands

Command	Description
`uci show wireless`	Displays the current wireless persistent configuration (essid, encryption…)
`uci show network`	Displays the network configuration (IP address, bridges…)

Command	Description
`uci set network.lan.ipaddr=192.168.2.1 /etc/init.d/network restart`	Sets the LAN IP address to 192.168.2.1 Restarts the network daemon to apply the change
`uci set wireless.@wifi-iface[0].ssid=”MySSID” wifi`	Sets the SSID of VAP0 to “MySSID” Restarts the Wi-Fi configuration to apply the change

UCI is a decentralized configuration interface with extensive configuration capabilities. This section describes the most commonly configured packages. UCI is organized in sections and sub-sections. Each section usually represents a package, or a sub-system on the target.

The most common sections are:

- Network section
- Wireless section
- Samba section

Each sub-section can be either named or unnamed, but they are assigned a type that the scripts use; sub-sections of the same type are usually iterated over by the scripts. The sub-sections usually define multiple instances of the same item, such as multiple interfaces, multiple VLANs, or multiple WLAN VAPs. This section describes the sub-sections with their keys and the values used.

6.2 Network section

The Network UCI section documents how to configure the interfaces (L3) in the system.

More documentation on the network section can be found at:
http://wiki.openwrt.org/doc/uci/network

Once the network section has been modified, this command can be used to restart the networking daemon and apply the modifications:

```
# /etc/init.d/network restart
```

The network section includes the below sub-section types:

- switch
- switch_vlan
- interface

6.2.1 switch

This sub-section represents the global switch configuration parameters.

Name	Type	Required	Default	Description
Reset	bool	No	0	Resets the switch configuration while starting the switch.
enable_vlan	bool	No	0	Enables VLANs on this switch.

6.2.2 switch_vlan

Switch VLANs are created to configure the VLANs inside the switch.

Name	Type	Required	Default	Description
device	string	Yes	N/A	Interface name connected to this switch VLAN.
vlan	int	Yes	N/A	VLAN ID for this VLAN.
ports	list	Yes	N/A	Space separated list of ports to include in this VLAN. If “t” is specified in front of the port, the frames is automatically tagged/untagged while going through this port. If “t” is not specified, the port-based VLAN is used.

6.2.3 interface

The interface sub-section configures the router. Interfaces in OpenWrt represent a logical network that serves as a container for IP address settings, aliases, routes etc. They should not be confused with Linux interfaces.

The interface protocol type defines the kind of protocol to use on a particular interface; it has the values defined here. The type conditions how the configuration framework uses the rest of the values inside this sub-section.

Table 6-2 Interface sub-section protocol types

Protocol	Description	See Table
static	Static configuration with fixed address and netmask	Table 6-3
dhcp	Address and netmask are assigned by DHCP	Table 6-4
dhcpv6	Address and netmask are assigned by DHCPv6	Table 6-5
ppp	PPP protocol-dialup modem connections	Table 6-6
pppoe	PPP over Ethernet-DSL broadband connection	Table 6-7
pptp	Connection via PPtP VPN	Table 6-8
none	Unspecified protocol	—

Table 6-3 Static protocol

Name	Type	Required	Default	Description
ipaddr	ip address	Yes, if ip6addr is not set	(None)	IP address
netmask	netmask	Yes, if ip6addr is not set	(None)	Netmask
gateway	ip address	No	(None)	Default gateway
broadcast	ip address	No	(None)	Broadcast address (auto generated if not set)
ip6addr	ipv6 address	Yes, if ipaddr is not set	(None)	Assigns the given IPv6 address to this interface (CIDR notation).
ip6gw	ipv6 address	No	(None)	Assigns the given IPv6 default gateway to this interface.

Name	Type	Required	Default	Description
ip6assign	prefix length	No	(None)	Delegates a prefix of given length to this interface (Barrier breaker and later only).
ip6hint	prefix hint (hex)	No	(None)	Hints the subprefix ID that should be delegated as hexadecimal number (Barrier breaker and later only)
ip6prefix	ipv6 prefix	No	(None)	IPv6 prefix routed for use on other interfaces (Barrier Breaker and later only)
ip6class	list of strings	No	(None)	Defines the IPv6 prefix-classes this interface accepts.
dns	list of ip addresses	No	(None)	DNS server(s).
metric	integer	No	0	Specifies the default route metric to use.

Table 6-4 dhcp protocol

Name	Type	Required	Default	Description
broadcast	boolean	No	0	Enables the broadcast flag in DHCP requests, required for certain ISPs, e.g., Charter with DOCSIS 3.
hostname	string	No	(None)	Hostname to include in DHCP requests.
clientid	string	No	system default	Overrides the client identifier in DHCP requests.
reqopts	list of strings	No	(None)	Specifies a list of additional DHCP options to request.
iface6rd	logical interface	No	(None)	Logical interface template for auto-configuration of iface6rd.

Table 6-5 dhcpv6 protocol

Name	Type	Required	Default	Description
reqaddress	[try,force,none]	No	try	Behavior for requesting addresses.
reqprefix	[auto,no,0-64]	No	auto	Behavior for requesting prefixes (numbers denote hinted prefix length). Use 'no' if it requires only a single IPv6 address for the AP itself without a subnet for routing.
clientid	string	No	system default	Overrides the client identifier in DHCP requests.
ifaceid	ipv6 addr	No	link-local identifier	Overrides the interface identifier for addresses received via RA.
reqopts	list of numbers	No	(None)	Specifies a list of additional DHCP options to request.
noslaaconly	boolean	No	0	Do not allow configuration via SLAAC (RAs) only (implied by reqprefix != no).
norelease	boolean	No	0	Do not send a RELEASE when the interface is brought down.
ip6prefix	ipv6 prefix	No	(None)	Use an (additional) user-provided IPv6 prefix for distribution to clients.
iface_dslite	logical interface	No	(None)	Logical interface template for auto-configuration of DS-Lite.

Table 6-6 ppp protocol

Name	Type	Required	Default	Description
username	string	No	(None)	Username for PAP/CHAP authentication.
password	string	No	(None)	Password for PAP/CHAP authentication.
connect	file path	No	(None)	Path to custom PPP connect script.
disconnect	file path	No	(None)	Path to custom PPP disconnect script.
keepalive	number	No	(None)	Number of connection failures before reconnect.
demand	number	No	(None)	Number of seconds to wait before closing the connection due to inactivity.
ipv6	boolean	No	0	Enables IPv6 on the PPP link.
pppd_options	string	No	(None)	Additional command line arguments to pass to the pppd daemon.

Table 6-7 pppoe protocol

Name	Type	Required	Default	Description
username	string	No	(None)	Username for PAP/CHAP authentication.
password	string	No	(None)	Password for PAP/CHAP authentication.
ac	string	No	(None)	Specifies the access concentrator to connect to. If unset, pppd uses the first discovered one.
service	string	No	(None)	Specifies the service name to connect to, If unset, pppd uses the first discovered one.
connect	file path	No	(None)	Path to custom PPP connect script.
disconnect	file path	No	(None)	Path to custom PPP disconnect script.
keepalive	number	No	(None)	Number of connection failures before reconnect.
demand	number	No	(None)	Number of seconds to wait before closing the connection due to inactivity.
ipv6	boolean	No	0	Enable IPv6 on the PPP link.
pppd_options	string	No	(None)	Additional command line arguments to pass to the pppd daemon.

Table 6-8 pptp protocol

Name	Type	Required	Default	Description
server	ip address	Yes	(None)	Remote PPtP server.
username	string	No	(None)	Username for PAP/CHAP authentication.
password	string	No	(None)	Password for PAP/CHAP authentication.
keepalive	integer	No		Number of attempts to reconnect.

6.3 Wireless section

The wireless section is generated dynamically during the first bootup. The script:

1. Probes the driver.
2. Uses it to fetch the number of radios and their characteristics.
3. Stores the result in UCI.

4. Creates one VAP per radio. These VAPs are created without encryption, with SSID ‘OpenWrt’, but are not enabled by default.

The wireless section can have three main sub-sections:

qcawifi	Allows to setup some global driver configuration parameters.
wifi-device[]	Unnamed section representing a Wi-Fi radio; there is one daughter card per WLAN connected to the system.
wifi-iface[]	Unnamed section representing a Wi-Fi VAP. By default, there is one per radio, but the user can add some additional sections if more VAPs are required. In each section, the device key points to the Wi-Fi device [] section representing the radio, to which this VAP belongs.

After modifying the wireless section, the network can be reconfigured in two possible ways:

- To reconfigure the entire networking stack, including Wi-Fi
  ```
  # /etc/init.d/network restart
  ```

- To reconfigure the Wi-Fi
  ```
  # wifi
  ```

The wireless section has these sub-sections:

- wifi-device
- wifi-iface
- WPA modes

6.3.1 wifi-device

Name	Type	Required	Default	Description
type	string	Yes	(auto detected)	Type is determined on first boot during the initial radio device detection. It is usually not required to change it. In this, it is always “qcawifi”.
macaddr	MAC address	Yes/No	(auto detected)	Specifies the radio adapter associated to this section; it is not used to change the device MAC but to identify the underlying interface. The value is auto detected at first boot or while using the PHY parameter. For a hardware independent config (to restore the config on many routers) use the PHY parameter instead of MAC address.
disabled	boolean	No	0	Disables the radio adapter if set to 1. Removing this option or setting it to 0 enables the adapter.
channel	integer or "auto"	Yes	auto	Specifies the wireless channel to use. In station mode, the value auto is allowed. In access point mode, an actual channel number must be given.

Name	Type	Required	Default	Description
hwmode	string	No	(driver default)	Selects the wireless protocol to use; possible values are: 11b, 11bg, 11g, 11a, 11ng (11N+11G, 2.4 GHz, mac80211 only), 11na (11N+11A, 5 GHz, mac80211 only), 11ac, or auto
htmode	string	No	(driver default)	Specifies the channel width in 11ng, 11na and 11ac modes. Possible values are: HT20 (single 20 MHz channel), HT40- (2x 20 MHz channels, primary/control channel is upper, secondary channel is below), HT40+ (2x 20 MHz channels, primary/control channel is lower, secondary channel is above) or HT80 – 802.11ac only – (4x20 MHz channels)
chanbw	integer	No	20	Specifies a narrow channel width, possible values are: 5 (5 MHz channel), 10 (10 MHz channel) or 20 (20 MHz channel).
txpower	integer	No	(driver default)	Specifies the transmission power in dBm
Country	hex or string	No	(driver default)	Sets the AP to the regulatory requirements of the country. It could be set either to a country ID or to a country code. The format is automatically detected and the script calls the appropriate command.
AMPDU	bool	No	(driver default)	Enables (1) or disables (0) Tx AMPDU aggregation for the entire interface. Receiving aggregate frames is still performed, but no aggregate frames transmit if this is disabled. The get parameter returns the current value.
bcnburst	bool	No	(driver default)	Set the beaconing scheme to burst or staggered mode. The default is staggered mode.
dcs_enable	bool	No	(driver default)	Enable or disable dynamic channel selection for interference mitigation.
dcs_errth	int	No	(driver default)	Configures transmission failure rate threshold, used to indicate presence of interference. Default value of transmission failure rate threshold is 30%.
dcs_phyerrth	int	No	(driver default)	Configures channel time wasted due to each PHY error (PHY error penalty). Default value of PHY error penalty is set as 500 µsec.
ani_enable	bool	No	(driver default)	Enables (1) or disables (0) ANI functionality. The default is 0.
txchainmask	hex	No	(driver default)	Sets the Tx and Rx chainmask values. For MIMO devices, indicates the number of Tx/Rx

Name	Type	Required	Default	Description
rxchainmask	hex	No	(driver default)	streams, and which chains are used. For some Qualcomm Atheros devices, up to 3 chains can be used, others are restricted to 2 or 1. Note the maximum number of chains available for the device. For dual chain devices, chain 2 is not available. Single chain devices support only chain 1. The chains are represented in the bit mask as: Chain 0 0x01 Chain 1 0x02 Chain 2 0x04 Chainmask selection can affect several performance factors. For a 3-chain device, an Rx chainmask of 0x05 (or 0x03) is used for 2x2 stream reception. For near range operations, a Tx chainmask of 0x05 (or 0x03) minimizes near range effects. For far range, a mask of 0x07 is used for Tx. The default chainmask values are stored in EEPROM. This iwpriv command overrides the chainmask settings. The get parameters returns the current values.
TXPowLim2G TXPowLim5G	int	No	(driver default)	Sets the maximum transmit power limit for the 2 GHz band or 5 GHz band. The maximum transmit power is also governed by country-specific regulatory requirements set by the iwpriv setCountry or setCountryID parameters. The `iwconfig txpower` command is similar but sets maximum transmit power for all frequencies. The TxPowLim2G/TxPowLim5G settings can be overridden by TxPwrOvr. The TxPowLim2G/TxPowLim5G values may also be updated by other portions of the code, so the effect of the value may be temporary. The limit is expressed as an integer that equals +0.5 dBm for each value of 1. For example, 0 = 0 dBm; 10 = 5 dBm; 100 = 50 dBm. The default is 100 for both parameters. The get parameters return the current values.

6.3.2 wifi-iface

Name	Type	Required	Default	Description
ampdu	boolean	No	No	Enable/disable Tx AMPDU aggregation for the entire interface.
amsdu	boolean	No	No	Sets maximum number of AMSDU sub frames
ani_enable	integer	No	No	Enables/disables ANI.
ant_ps_on	boolean	No	*(driver default)*	Enables (1) or disables (0) green AP power save logic. The default value is 1.
bintval	integer	No	No	Sets the AP beacon interval (in ms).
blockdfschan	integer	No	No	Blocks the DFS channel in the selection process.

Name	Type	Required	Default	Description
blockdfschan	integer	No	No	Blocks the DFS channel in the selection process.
bssid	BSSID address	No	*(driver default)*	Overrides the BSSID of the network, only applicable in adhoc or STA mode. In WDS mode specifies the BSSID of another AP to create WDS with.
chwidth	integer	No	*(driver default)*	Sets the channel width field in the AP beacon High Throughput Information Element (HTIE). If this command is not executed, then the channel width is taken from the device settings. The get parameter returns the current value. The default value is 0. Sets the current channel width setting. Not necessarily the value set by `cwmode`, because it can be automatically overridden.
dbgLVL	hex	No	*(driver default)*	Controls the debug level of the VAP-based debug print statements. It is normally set to zero, eliminating all prints.
device	string	Yes	*(first device id)*	Specifies the used wireless adapter, must refer to one of the defined wifi-device sections.
doth	boolean	No	0	Enables 802.11h support.
dtim_period	integer	No	*2 (hostapd default)*	Sets the Delivery Traffic information Message (DTIM) period. There is one DTIM per many beacon frames. This may be set between 1 and 255. This option only has an effect on AP wifi-ifaces.
extprotmode	bool	No	*(driver default)*	Sets the protection mode used on the extension (secondary) channel when using 40 MHz channels. The default is 0. The get parameter returns the current value.
hidden	boolean	No	0	Turns off SSID broadcasting if set to 1.
ieee80211w	integer	No	0	Enables MFP (802.11w) support (0 = disabled, 1 = optional, 2 = required).
ieee80211w_max_timeout	integer	No	*(hostapd default)*	Specifies the 802.11w association SA Query maximum timeout.
ieee80211w_retry_timeout	integer	No	*(hostapd default)*	Specifies the 802.11w association SA query retry timeout.
isolate	boolean	No	0	Isolate wireless clients from each other, only applicable in AP mode.
key	integer or string	No	*(None)*	In any WPA-PSK mode, this is a string that specifies the pre-shared passphrase from which the pre-shared key is derived. If a 64 character hexadecimal string is supplied, it is used directly as the pre-shared key instead. In WEP mode, this can be an integer specifying which key index to use (key1, key2, key3, orkey4.) Alternatively, it can be a string specifying a passphrase or key directly, as in key1. In any WPA-Enterprise AP mode, this option has a different interpretation.

Name	Type	Required	Default	Description
key1	string	No	*(None)*	WEP passphrase or key #1 (selected by the index in key). This string is treated as a passphrase from which the WEP key isderived. If a 10 or 26 character hexadecimal string is supplied, it is used directly as the WEP key instead.
key2	string	No	*(None)*	WEP passphrase or key #2 (selected by the index in key), as in key1.
key3	string	No	*(None)*	WEP passphrase or key #3 (selected by the index in key), as in key1.
key4	string	No	*(None)*	WEP passphrase or key #4 (selected by the index in key), as in key1.
macfilter	string	No	disable	Specifies the *mac filter policy*, disable to disable the filter, allow to treat it as whitelist or deny to treat it as blacklist.
maclist	list of MAC addresses	No	*(None)*	List of MAC addresses (divided by spaces) to put into the mac filter.
maxampdu	integer	No	*no*	Maximum AMPDU length exponent.
mcast_rate	integer	No	*(driver default)*	Sets the fixed multicast rate, in Kbps.
mcastenhance	int	No	*(driver default)*	Sets multicast enhancement mode. ▪ 0 Enabled: true multicast packet is sent if any interested member is present. ▪ 1 Enabled: tunneled unicast packet is sent to the interested members. ▪ 2 Enabled: translated unicast packet is sent to the interested members. ▪ 4-F Enabled: disabled (Set bit 2 = 1)
metimeout	int	No	*(driver default)*	Sets timeout in ms for STA to be removed from the snoop table if idle. The parameter value may be any unsigned integer value. The default is 120000 (2 minutes). The get parameter returns the current value.
metimer	int	No	*(driver default)*	Sets timer in ms to check snoop table status. The timer value may be any unsigned integer. The default is 30000 (30 sec). The get parameter returns the current value.
mode	string	Yes	ap	Selects operation mode of the wireless network interface controller (some are supported simultaneously by some drivers): ▪ ap for access point ▪ sta for managed (client) mode ▪ adhoc for ad hoc ▪ wds for static WDS ▪ monitor for monitor mode
network	string	Yes	lan	Specifies the network interface to attach the wireless.

Name	Type	Required	Default	Description
protmode	bool	No	*(driver default)*	Enables or disables 802.11g protection mode. Causes RTS/CTS sequence (or CTS to self) to be sent when 802.11b devices are detected on the 802.11g network. Used to protect against Tx by devices that do not recognize OFDM modulated frames. The default is 0. The get parameter returns the current value.
ps_timeout	int	No	*(driver default)*	Sets the transition time in seconds between power save off to power save on mode. The default value is 20.
retrydur	int	No	*(driver default)*	Sets the retry threshold in 1 sec. Feature disabled if set at 0.
rsn_preauth	boolean	No	0	Allows pre-authentication for WPA2-EAP networks (and advertise it in WLAN beacons). Works only if the specified network interface is a bridge.
rx_stbc tx_stbc	bool	No	*(driver default)*	Enables (1) or disables (0) the Space Time Coding Block (STBC) feature, as described in 802.11n specification, in the transmit (txstbc) or receive (rxtsbc) direction. The default value is 1. This option has an effect only on chips supporting STBC. On other chips, this option has no effect.
ssid	string	Yes	OpenWrt	The broadcasted SSID of the wireless network (for managed mode the SSID of the network to connect to).
vhtmaxampdu	integer	No	*no*	VHT maximum AMPDU length exponent.
wds	boolean	No	*0*	Sets 4-address mode.
wmm	boolean	No	1	Enables WMM (802.11e) support; required for 802.11n support.
encryption	string	No	None	Wireless encryption method. None for an open network, wep for WEP, psk for WPA-PSK, or psk2for WPA2-PSK. See WPA modes in the following section for additional possible values. For an access point in WEP mode, the default is "open system" authentication. Use wep+sharedfor "shared key" authentication (less secure), wep+open to explicitly use "open system," orwep+mixed to allow either wep+mixed is only supported by hostapd.

6.3.3 WPA modes

In the encryption field, multiple values can be used for WPA. Table 6-9 shows the list of values.

Table 6-9 WPA modes

Value	WPA version	Cipher
psk2+tkip+ccmp psk2+tkip+aes	WPA2 Personal (PSK)	TKIP, CCMP
psk2+tkip	WPA2 Personal (PSK)	TKIP

Value	WPA version	Cipher
psk2+ccmp psk2+aes psk2	WPA2 Personal (PSK)	CCMP
psk+tkip+ccmp psk+tkip+aes	WPA Personal (PSK)	TKIP, CCMP
psk+tkip psk	WPA Personal (PSK)	TKIP
psk+ccmp psk+aes	WPA Personal (PSK)	CCMP
mixed-psk+tkip+ccmp mixed-psk+tkip+aes mixed-psk	WPA/WPA2 Personal (PSK) mixed mode	TKIP, CCMP
mixed-psk+tkip	WPA/WPA2 Personal (PSK) mixed mode	TKIP
mixed-psk+ccmp mixed-psk+aes	WPA/WPA2 Personal (PSK) mixed mode	CCMP
wpa2+tkip+ccmp wpa2+tkip+aes	WPA2 Enterprise	TKIP, CCMP
wpa2+ccmp wpa2+aes wpa2	WPA2 Enterprise	CCMP
wpa2+tkip	WPA2 Enterprise	TKIP
wpa+tkip+ccmp wpa+tkip+aes	WPA Enterprise	TKIP, CCMP
wpa+ccmp wpa+aes	WPA Enterprise	CCMP
wpa+tkip wpa	WPA Enterprise	TKIP
mixed-wpa+tkip+ccmp mixed-wpa+tkip+aes mixed-wpa	WPA/WPA2 Enterprise mixed mode	TKIP, CCMP
mixed-wpa+tkip	WPA/WPA2 Enterprise mixed mode	TKIP
mixed-wpa+ccmp mixed-wpa+aes	WPA/WPA2 Enterprise mixed mode	CCMP

6.4 Samba section

The samba section allows the configuration of the Windows share daemon. There are two main sub-sections that must be configured:

- samba
- sambashare

6.4.1 samba

This sub-section defines the generic samba option. It contains options valid for all shares on the system.

Name	Type	Required	Default	Description
workgroup	string	Yes	workgroup	Name of the workgroup.
name	string	Yes	openwrt	Name of the server.
homes	boolean	Yes	1	Shares the user directory.
description	string	Yes	openwrt	Description of the server.

6.4.2 sambashare

This sub-section contains options valid on one particular share.

Name	Type	Required	Default	Description
name	string	Yes	(None)	Name of the entry; it is shown in the file browser.
path	file path	Yes	(None)	The complete path of the directory.
users	string	No	(None)	The samba-users are allowed access to this entry; use smbpasswd to create a user-pwd combination. Several users can be specified, separated by a comma (for example: option 'users' 'root,nobody').
guest_ok	string	Yes	No	Specifies if a login requires the samba username and password to access this share.
create_mask	integer	Yes	0700	chmod mask for files created (need write access).
dir_mask	integer	Yes	0700	chmod mask for directories created (need write access).
read_only	boolean	Yes	No	No permission for read/write, else only read access is granted; (for rw, the mount fs rw is also required).