



### ABSTRACT

WiLink8™ Wi-Fi® supports IEEE802.11 standards and proprietary enhancements. This document is intended to serve as a user's guide for integrating R8.8 Wi-Fi driver release and associated software components into host Linux®-based platform. The document details driver architecture, core components, configuration files, build procedures and testing of the R8.8 release. This document also serves as a user's guide for verifying the basic Wi-Fi functionalities and provides brief overview for debugging and FAQs. The examples provided are using the [PROCESSOR-SDK-LINUX-AM335X\\_06\\_00\\_00\\_07](#). But the same methodology is applicable to other SDKs as well that are based on Linux Kernel version 4.19.38 (2019 LTS).

### Table of Contents

<b>1 Driver Supported Features</b> .....	2
<b>2 WL18xx Linux Driver Architecture Overview</b> .....	2
<b>3 Platform Integration</b> .....	3
3.1 Configuration required for Board Device Tree (DTS/DTB).....	4
3.2 Configuring the Kernel for TI WLAN Drivers.....	4
3.3 Configuration required for Board Device Tree (DTS/DTB).....	4
3.4 Building R8.8 Release Using Build Utilities.....	5
3.5 Building WiLink8 Driver Release Binaries Individually.....	8
<b>4 Booting and WLAN Bring-Up</b> .....	9
4.1 Configuring the WiLink8 Target.....	9
<b>5 Testing Basic WLAN Functionality</b> .....	11
5.1 STA Mode.....	11
5.2 AP Mode.....	12
5.3 Multirole (AP +STA mode).....	13
5.4 IEEE802.11s Mesh Mode.....	14
<b>6 References</b> .....	14
<b>A FAQ and Debug Hints</b> .....	15

### Trademarks

WiLink8™ is a trademark of Texas Instruments.

Wi-Fi® is a registered trademark of Wi-Fi Alliance.

Linux® is a registered trademark of Linus Torvalds in the U.S. and other countries.

All trademarks are the property of their respective owners.

## 1 Driver Supported Features

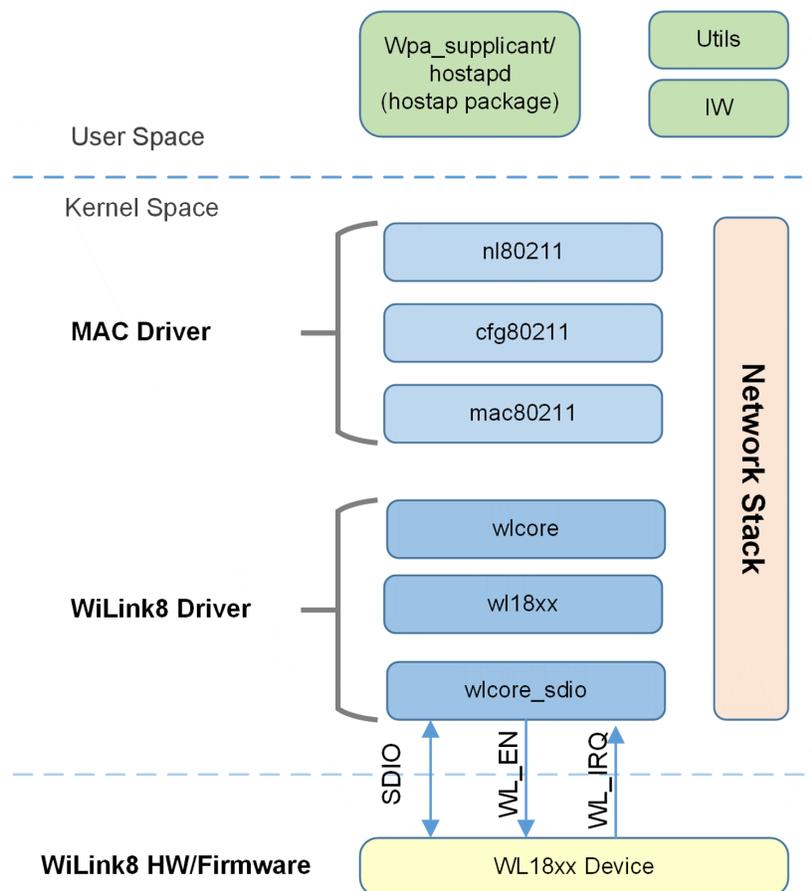
The following is the list of features supported by WiLink8 driver and device.

- Linux open-source Wi-Fi package.
- TI NLCP releases are Wi-Fi Alliance pre-certified.
- IEEE: 802.11 a,b,g,n, 2X2 MIMO @ 2.4 GHz and antenna diversity @ 5 GHz
- Supported Modes: STA, AP, P2P, Wi-Fi Direct, Wi-Fi Mesh
- Up to 100 Mbps UDP throughput
- Security: WPA3, WMM-PS, WMM-AC, WPA/2PSK, Ent,WPS,WPSv2
- Low power support: Station WoW & Suspend/Resume , AP ELP (800  $\mu$ A idle connect)
- Co-existence with other 2.4 GHz protocol: BT/BLE and TI ZigBee at 2.4 GHz
- AP DFS, radar detection at 5 GHz
- Multi Role Multi Channel: concurrent operation of 2 WLAN roles on a single device.
- Wi-Fi over mesh support: open 802.11s

For specific features and bug fixes updated in R8.8, see the [WiLink8 R8.8 Release Notes](#) . Additionally, [WiLink™ 8 WLAN Features User's Guide](#) provides complete details of the WiLink8 supported features.

## 2 WL18xx Linux Driver Architecture Overview

WL18xx Linux driver uses the open source components along with interface driver for the device to realize Wi-Fi functionalities. [Figure 2-1](#) shows the high level driver partitioning and architecture.



**Figure 2-1. WiLink8 Driver Architecture**

The section below briefs the high-level components in the driver layers and their functionality.

- WiLink8 FW – The FW runs on the device HW to provide the PHY and MAC functionality of the Wi-Fi. The host communicates via SDIO to the WLAN device. On the device side, the WLAN MAC is responsible for the 802.11 MAC functions, and conveys WLAN packets from/to the external host to/from the FW. The MAC is responsible for the timing and the time critical decisions only. The PHY performs the 802.11 PHY functions of encoding/decoding and modulation/ demodulation, and is responsible for the RF functions of up/down modulation to carrier frequency, filtering and amplification.
- WiLink Driver is an abstraction layer to the device HW and FW. Implements low level operations required to support the MAC driver.
  - wlc core: Implements the low level driver for WiLink devices, supporting mac80211 operations. Contains the common functions for all supported WiLink chipsets.
  - Wl18xx: Implement chip specific functions and services. Supports the wlc core by implementing HW-specific functions.
  - wlc core\_sdio: Adaptation layer between the SDIO driver and the WiLink driver.
- MAC Driver implements layer-2 Wi-Fi protocol requirements (data and control path). This is a generic component, not platform/device specific. This layer consists of the following components.
  - nl80211: Implements a net-link interface between user-space and kernel space components of the Linux Wireless solution.
  - cfg80211: The Linux wireless configuration API. (This is the lowest layer that is common for both soft-MAC and hard-MAC).
  - mac80211: The Linux kernel module implementing MAC-layer functions for Wi-Fi Soft-MAC solution.
- Hostap package: Contains open-source user-space package. Provides the upper-management layers for all WLAN roles (STA, AP, P2P and Mesh). Generates 2 daemons: wpa\_supplicant (STA, P2P, Mesh), and hostapd (AP).
- Utilities provide initialization and configuration services. Implement debug and statistics capabilities.

### 3 Platform Integration

The following section details the integration of the driver to the Linux SDK platform. The references and instructions provided are applicable to any platform using Linux operating system. However specific instructions mentioned below are based on [PROCESSOR-SDK-LINUX-AM335X 06\\_00\\_00\\_07](#). For WiLink8 hardware integration, see the [WiLink™ Module Hardware Integration Guide](#).

The generic steps to integrate the WiLink8 R8.8 driver release manually to any kernel are provided below. The same method can be used for upgrading the WiLink8 driver version of an existing SDK to R8.8 release. However if TI SDK is used (with pre-built kernel) along with “build utilities” scripts can be used. Note that the following steps assume Linux host environment up and running. For more information on setting up your Linux host PC, follow the instructions provided in [Processor SDK Getting Started Guide](#).

1. Download the Kernel (4.19+) and platform SDK.
2. Install the [SDK image to SD card](#) as per the SDK installation instructions.
3. Configure Kernel using verify\_kernel.sh utility or manually.
4. Apply kernel patches – this needs to be done if building the kernel for the first time.
5. Build WLAN modules, kernel zImage (optional), kernel modules (build-utilities) and BeagleBone Black DTB.
6. Compile the device tree files for specific boards (dts → dtb) and update.
7. Copy the build outputs to SD card.

Steps 1 and 2 are manual. This is the starting step irrespective of the SDK used. These steps will ensure that default file system provided by the SDK is installed into the SD card. Since as part of the R8.8 WiLink8 WLAN driver build only a subset of components related to the WLAN and the kernel and modules are built, it is important to have the complete default file system installed.

Steps 3, 4 and 6 (except for step -5, DTS/B file) can be performed using the “build\_scripts” utility. The DTS/B file is hardware specific and needs to be created based on the hardware design.

### 3.1 Configuration required for Board Device Tree (DTS/DTB)

The device tree is a set of files in the Linux source that describe the hardware platform. In order to integrate WiLink8 the file needs to have the right configuration to integrate with required hardware interface. For WiLink8 WLAN functionality these include

- SDIO configuration
- WL\_EN GPIO configuration
- WL\_IRQ GPIO configuration
- Power control configuration
- MMC configuration

The default TI processor SDK dts/b file has already enabled the required configurations. The files are automatically chooses at boot time based on the HW platform used. For details of required configurations, see [AM335x\\_evm.dts](#). Note that these settings may need to be modified based on specific hardware platform used.

### 3.2 Configuring the Kernel for TI WLAN Drivers

Most of the kernel configurations are default and already configured as part of the TI SDK releases. The following steps are to ensure that the right kernel configuration selections are made in .config file and are applied while building the kernel.

The following command will open the makemenu config to enable the required configurations manually. Note that while using the “build utilities” script the required configurations are done automatically.

```
export PATH=<sdk path>/linux-devkit/sysroots/x86_64-arago-linux/usr/bin:$PATH
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- distclean
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- <config file>
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

The following kernel configuration flags/switches need to be enabled for WiLink8 driver.

```
CONFIG_CFG80211=m
CONFIG_MAC80211=m
CONFIG_WLCORE=m
CONFIG_WLCORE_SDIO=m
CONFIG_WL18XX=m
CONFIG_NL80211_TESTMODE=y
CONFIG_MAC80211_MESH=y
```

Additional configurations needed are mentioned are included as part of the *verify\_kernel\_config.sh* (detailed in later sections) can be used for updating the kernel configurations.

### 3.3 Configuration required for Board Device Tree (DTS/DTB)

The device tree is a set of files in the Linux source that describe the hardware platform. In order to integrate WiLink8 the file needs to have the right configuration to integrate with required hardware interface. For WiLink8 WLAN functionality these include

- SDIO configuration
- WL\_EN GPIO configuration
- WL\_IRQ GPIO configuration
- Power control configuration
- MMC configuration

The default TI processor SDK dts/b file has already enabled the required configurations. The files are automatically chooses at boot time based on the HW platform used. For details of required configurations, see [AM335x\\_evm.dts](#). Note that these settings may need to be modified based on specific hardware platform used.

### 3.4 Building R8.8 Release Using Build Utilities

The following section details the steps to build the R8.8 release using build utilities. Build utilities provide a unified method to build, update and integrate WL18xx WLAN driver and modules. This step needs to be done after configuring the kernel and updating the necessary changes for the DTS/B files for the target platform. **The WiLink8 R8.8 release is based on the Linux Kernel version 4.19 and does not support backports for previous kernel versions.**

The build utilities integrate the WiLink8 WLAN modules, drivers required to operate WL18xx devices. It also includes additional build packages for WPA-suplicant and hostapd based on the open source, but customized for WL18xx devices. The utility also integrates the tools for testing, example scripts to demonstrate Wi-Fi operation and device FW. The scripts can be used to build the entire modules and kernel or has ability to build individual modules.

The general process included in build-utility script is as follows:

1. Download built utilities package
2. Configure the setup
3. Clone source code and driver components for specific release (R8.8)
4. Add right kernel configurations using `verify_kernel.sh`
5. Apply kernel patches (required one time)
6. Build all release binaries or individual components
7. Install built binaries to the target filesystem

The script will download the following source files for the components. For exact version details of each of the components, see the [WiLink8 R8.8 Release Notes](#). The downloaded source files are placed in `./build-utilities/src` directory.

Directory	Contents
<code>fw_download</code>	Contains WiLink8 device FW accompanying the release
<code>hostap</code>	Source code for <code>wpa_supplicant</code> and <code>hostapd</code> . These have build dependencies on <code>openssl</code> and <code>libnl</code> , which are also downloaded under <code>./build-utilities/src</code>
<code>iw</code>	Source code for <code>iw</code> tool.
<code>openssl</code>	Contains the cloned source for <code>openssl</code>
<code>scripts_download</code>	An assortment of scripts to operate the WL18xx
<code>ti_utils</code>	Different utilities supplied by TI.
<code>wireless-regdb</code>	Wireless regulatory database

#### 1. Download the Build Scripts and Switch to R8.8 Branch.

Clone build-utilities script from [git://git.ti.com/wilink8-wlan/build-utilities.git](https://git.ti.com/wilink8-wlan/build-utilities.git). Below is an example:

```
user@ubuntu:~/ti-sdk-am335x-evm-07.00.00.00$ cd ~/wl8-build/
user@ubuntu:~/wl8-build$ git clone git://git.ti.com/wilink8-wlan/build-utilities.git
Cloning into 'build-utilities'...
remote: Counting objects: 888, done.
remote: Compressing objects: 100% (412/412), done.
Recreate: Total 888 (delta 490), reused 761 (delta 456)
Receiving objects: 100% (888/888), 12.82 MiB | 5.10 MiB/s, done.
Resolving deltas: 100% (490/490), done.
user@ubuntu:~/wl8-build$ cd build-utilities/
user@ubuntu:~/wl8-build/build-utilities$ ls
build_wl18xx.sh  configuration/  configuration.sh  patches/  README  setup-env.sample
sudo_build_wl18xx.sh  verify_kernel_config.sh
```

Switch to R8.8 branch using the following command:

```
user@ubuntu:~/wl8-build/'build-utilities'$ git checkout r8.8
```

After cloning, build-utilities following scripts are available in the `./build-utilities` folder. The details of the scripts are provided below:

<code>setup-env.sample</code>	Sample environment setup file. Should be copied or renamed to <code>setup-env</code>
<code>configuration.sh</code>	Contains the configuration details of git repository address and git tags that are used for downloading the source files.
<code>build_wl18xx.sh</code>	This is the main script that uses <code>setup-env</code> and <code>configuration.sh</code> along with user parameters to download, clean, update or build specific components selected by the user.
<code>sudo_build_wl18xx.sh</code>	Same as <code>build_wl18xx.sh</code> with <code>sudo</code> option. Note that using the "sudo" version of the script for "init" option will end up with directories owned by root .
<code>verify_kernel_config.sh</code>	A script used for verifying kernel configuration.

The following sections will use the `build-wl18xx.sh` script for clean, build, and install all of the components or specific components. Building specific components are discussed later. The `-h` parameter will display the available command options.

```
./build_wl18xx.sh -h
```

The available options are displayed as follows.

```
user@ubuntu:~/R8.8/build-utilities$ ./build_wl18xx.sh -h
This script builds all/one of the relevant wl18xx software packages.

Usage :

Building full package : Build all components except kernel, dtb
./build_wl18xx.sh init          [ Download and Update w/o
build ]

          update      R8.8          [ Update to specific TAG & Build ]
          clean       [ Clean & Build ]
          check_updates [ Check for build script updates ]

Building specific component :
hostapd          [ Clean & Build hostapd ]
wpa_supplicant  [ Clean & Build wpa_supplicant ]
modules         [ Clean & Build driver modules ]
firmware       [ Install firmware binary ]
scripts        [ Install scripts ]
utils          [ Clean & Build scripts ]
iw             [ Clean & Build iw ]
openssl        [ Clean & Build openssl ]
libnl          [ Clean & Build libnl ]
wireless-regdb [ Install wireless regdb ]
patch_kernel   [ Apply provided kernel patches ]
kernel <defconfig filename> [ Clean & Build Kernel ]
kernel_noclean <defconfig filename> [ Build Kernel w/o clean ]
patch_bbbel4_dts [Patch bbb black dts file to add e14 cape support]
```

## 2. Create setup-env file.

"`setup-env.sample`" file is used as base of "`setup-env`" that includes user specific environment variables. User should copy the `setup-env.sample` to `setup-env` and edit the variables according to user specific environment. The user shall edit the `setup-env` file to point to the correct directories where the kernel and toolchain are located. Below is an example file:

```
#                \\\//
#                -(o o)-
#=====oOO==( )==OOo=====
# This file contains the exports needed for automating the
# build process of WLAN components.
# Place this file in the same directory with wl18xx_build.sh
# build scripts. No need to run 'source setup-env', the build
# scripts will perform it internally.
#=====
# User specific environment settings - use full PATH

# TOOLCHAIN_PATH setting is mandatory. ex: TOOLCHAIN_PATH=/opt/ti-processor-sdk-linux-am335x-
# evm-06.00.00.07/linux-devkit/sysroots/x86_64-arago-linux/usr/bin
export TOOLCHAIN_PATH=

# ./fs folder will be created if ROOTFS is set to DEFAULT
export ROOTFS=DEFAULT
```

```
# KERNEL_PATH setting is mandatory. ex: KERNEL_PATH=/opt/ti-processor-sdk-linux-am335x-
evm-06.00.00.07/board-support/linux-4.19.38+gitAUTOINC+4dae378bbe-g4dae378bbe
export KERNEL_PATH=

# CROSS_COMPILE setting is mandatory
export CROSS_COMPILE=arm-linux-gnueabi-

# ARCH setting is mandatory
export ARCH=arm
[ "$TOOLCHAIN_PATH" != "" ] && export PATH=$TOOLCHAIN_PATH:$PATH
```

setup-env file should be placed in the same directory together with build scripts (build\_wl18xx.sh, and so forth.).

### 3. Download the Source (a.k.a Initialize) repo.

The following step downloads the entire source required for the build. This may take longer time for the first time installation. Subsequent updates will take shorter time.

```
user@ubuntu:~/wl8-build$ cd build-utilities
user@ubuntu:~/wl8-build/build-utilities$ ./build_wl18xx.sh init
```

### 4. Verify and set required kernel configurations.

WiLink8 WLAN functionality requires certain settings to be enabled in the kernel configuration. This can be set/verified using the *verify\_kernel\_config.sh* utility provided in the build-utility package as shown below:

```
user@ubuntu:~/wl8-build/build-utilities$ ./verify_kernel_config.sh <def_config file>
```

Example: user@ubuntu:~/wl8-build/build-utilities\$ ./verify\_kernel\_config.sh /opt/ti-processor-sdk-linux-am335x-evm-06.00.00.07/board-support/linux-4.19.38+gitAUTOINC+4dae378bbe-g4dae378bbe/arch/arm/configs/tisdk\_am335x-evm\_defconfig

### 5. Applying required kernel patches.

WiLink8 driver package includes a set of patches that needs to be applied to enable complete functionality. These patches have feature enhancements and bug fixes. This step is needed while building the kernel image for the first time only to enable WiLink8 WLAN.

```
user@ubuntu:~/wl8-build/build-utilities$ ./build_wl18xx.sh patch_kernel
```

### 6. Build Kernel.

The following commands will build the kernel. The kernel needs to be rebuilt to include the TI WiLink8 specific patches. **User can build the kernel from SDK directly or use build script to build the kernel.** Kernel defconfig filename is passed as an argument.

```
user@ubuntu:~/wl8-build/build-utilities$ ./build_wl18xx.sh kernel <defconfig file>
Ex: user@ubuntu:~/wl8-build/build-utilities$ ./build_wl18xx.sh kernel tisdk_am335x-evm_defconfig
```

The following steps are needed in case using BeagleBone Black Cape and BeagleBone for development.

---

#### Note

The kernel image. zImage is also placed along with the install file system.

---

### 7. Patch and Build BeagleBone DTB file (Optional).

DTS/B file needed for using BeagleBone Black with Element-14 Wireless cape can also be generated using the build\_utilities. Patch BeagleBone Black dts file to add support for Element-14 wireless cape with WL1837MOD using the following command

```
user@ubuntu:~/wl8-build/build-utilities$ ./build_wl18xx.sh patch_bbbe14_dts
```

Build Beaglebone black dts with element 14 wireless cape support to generate the required dtb file.

```
user@ubuntu:~/wl8-build/build-utilities$ ./build_wl18xx.sh bbbe14_dtb
```

## 8. Build Release Binaries.

Each WiLink8 driver release is given a specific tag. For WiLink8 Driver R8.8 driver the tag is “**R8.8**”. To checkout R8.8 release, build it, and install to the target file system you would use the following command (assuming root privileges required for file system access):

```
user@ubuntu:~/wl8-build/build-utilities$ ./build_wl18xx.sh update R8.8
```

## 9. Install the WiLink8 Release Binaries.

At this stage all components of WiLink8 driver and kernel should be built. The output libraries, binaries, example scripts, firmware and TI utilities etc. are placed in the folder specified in setup\_env file (Default is *./build-utilities/fs*) . The same is ready for installation. Note that the following steps assume that the default SDK image is already installed to the SD card. Copy the fs folder to the target using the below example command from *./build-utilities* directory.

```
sudo cp -p ./fs/* <rootfs path on SD card>/
```

## 3.5 Building WiLink8 Driver Release Binaries Individually

The build-utilities also supports clean, build, and install one specific component rather than everything. Various options available as part of the script can be viewed with the following command.

```
./sudo_build_wl18xx.sh -h
```

The available options are displayed as follows:

openssl/libnl	Clean & build openssl and libnl. These libraries are needed to build user space components like hostapd/wpa_supplicant/iw
hostapd/wpa_supplicant/iw	Clean and build hostapd or wpa_supplicant or iw user space utilities. Build openssl and libnl (order matters) before this build
modules	Clean & build WiLink8 driver modules
firmware	Install the firmware binary to <i>./build-utilities/fs</i>
scripts	Install TI example scripts to run STA, AP, MESH
utils	Clean & Build scripts like wlconf
patch_kernel	Apply provided kernel patches that are not up streamed
kernel	Clean build Kernel
kernel_noclean	Build kernel without cleaning. Useful for incremental builds
patch_bbbe14_dts	Patches needed for Beaglebone Black to use Element 14 Wireless cape

Command syntax to build driver specific components.

```
./sudo_build_wl18xx.sh <module>
```

The above assumes that you need root permission to install into the file system.

## 4 Booting and WLAN Bring-Up

The following steps use HW setup using [AM335x EVM](#) along with [WL1837MODCOM8I](#) module. The SD card can now be mounted to SDRAM1 slot on AM335x EVM. Connect UART cable from J12 UART connector to PC. Run any terminal program (ex: TerraTerm) with baud rate configured to 115200, 8 bits, no parity. [BeagleBone Black](#) along with [Element 14 Wireless Cape](#) can also be used as an alternative to the below setup.



**Figure 4-1. Basic HW Setup Using AM335x and WiLink8 for Driver Bring-Up**

The following message should appear during the kernel boot, which indicates that the driver is up and WiLink8 FW is downloaded.

```
[ 28.358451] wlcore: wl18xx HW: 183x or 180x, PG 2.2 (ROM 0x11)
[ 28.478778] wlcore: loaded
[ 29.515823] wlcore: PHY firmware version: Rev 8.2.0.0.244
[ 29.662595] wlcore: firmware booted (Rev 8.9.0.0.84)
```

Also, the default wlan0 interface should be up once login is complete.

```
root@am335x-evm:~# ifconfig wlan0
wlan0  Link encap:Ethernet  HWaddr 0C:1C:57:BB:60:5E
        UP BROADCAST MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
root@am335x-evm:~#
```

The above steps confirms that the driver is up and running.

### 4.1 Configuring the WiLink8 Target

WiLink8 driver is provided with default *wl18xx-conf.bin* binary file. */lib/firmware/ti-connectivity/wl18xx-conf.bin* configures the require RF parameters, number of antennas used, desired band of operation etc. This file needs to be altered based on the flavor of the WL18xx device or module used in the setup. To simplify the process of generating the *wl18xx-conf.bin* (located at */lib/firmware/ti-connectivity*), the R8.8 release includes a configuration script that can be used for making the right selections based on the HW configuration. The following section details the usage of the “*configure-device.sh*” script. The script will update the existing *wlconf* file with selected options. Note the values programmed affects RF performance and accurate values are needed for passing certification.

*configure-device.sh* is a menu driven script. The script will ask hardware dependent questions that will be used to correctly configure the target for use with the WiLink 8 device. This script uses ‘*wlconf*’ utility to create WiLink8 configuration binary. [WiLink™ 8 Solutions WiLink8 – wlconf](#) contains more details on ‘*wlconf*’ utility and how to modify this configuration.

To begin complete bring up procedure listed above and make sure WiLink8 device is booted. Establish a serial connection with the target device and log in as *root*.

Navigate to the location of the script */usr/sbin/wlconf* and run *configure-device.sh*:

```
cd /usr/sbin/wlconf
./configure-device.sh
```

The script will take you through a series of hardware-dependent questions, and configure *wlconf* for your system. This script can be used for WiLink8 TI modules, non-TI modules or chip on board designs. Examples are shown below.

When using the **WiLink8 TI module**, the script will automatically pick the right INI file from /usr/sbin/wlconf/official\_inis:

Example for WL1837 without Japanese Certification:

```
root@am335x-evm:/usr/sbin/wlconf# ./configure-device.sh
```

Please provide the following information.

```
Are you using a TI module? [y/n] : y
What is the chip flavor? [1801/1805/1807/1831/1835/1837 or 0 for unknown] : 1837
Should Japanese standards be applied? [y/n] : n
How many 2.4GHz antennas are fitted? [1/2] : 2
How many 5GHz antennas are fitted (using 2 antennas requires a proper switch)? [0/1/2] : 1
[ 106.068069] wlcore: down
-----
```

The device has been successfully configured.

```
TI Module: y
Chip Flavor: 1837
Base INI file used: /usr/sbin/wlconf/official_inis/WL1837MOD_INI_FCC_CE.ini
Number of 2.4GHz Antennas Fitted: 2
Number of 5GHz Antennas Fitted: 1
Diversity Support: y
SISO40 Support: y
Japanese Standards Applied: n
-----
```

Example for WL1801

```
root@am335x-evm:/usr/sbin/wlconf# ./configure-device.sh
```

Please provide the following information.

```
Are you using a TI module? [y/n] : y
What is the chip flavor? [1801/1805/1807/1831/1835/1837 or 0 for unknown] : 1801
Should SISO40 support be applied? [y/n] : y
[ 539.778261] wlcore: down
-----
```

The device has been successfully configured.

```
TI Module: y
Chip Flavor: 1801
Base INI file used: /usr/sbin/wlconf/official_inis/WL1835MOD_INI_C2PC.ini
Number of 2.4GHz Antennas Fitted: 1
Number of 5GHz Antennas Fitted: 0
Diversity Support: n
SISO40 Support: y
Japanese Standards Applied: n
-----
```

The **WiLink8 chip on board or non-TI Module** configuration the script will pick /usr/sbin/wlconf/official\_inis/WL8\_COB\_INI.ini. TI provides template of this file. **Customers will need to modify RF parameters in this file as per their design or use the values from respective module vendor.**

```
root@am335x-evm:/usr/sbin/wlconf# ./configure-device.sh

Please provide the following information.

Are you using a TI module? [y/n] : n
What is the chip flavor? [1801/1805/1807/1831/1835/1837 or 0 for unknown] : 1837
How many 2.4GHz antennas are fitted? [1/2] : 2
How many 5GHz antennas are fitted (using 2 antennas requires a proper switch)? [0/1/2] : 1
[ 148.158965] wlcore: down

-----

The device has been successfully configured.
TI Module: n
Chip Flavor: 1837
Base INI file used: /usr/sbin/wlconf/official_inis/WL8_COB_INI.ini
Number of 2.4GHz Antennas Fitted: 2
Number of 5GHz Antennas Fitted: 1
Diversity Support: n
SISO40 Support: y
Japanese Standards Applied: n

-----
```

## 5 Testing Basic WLAN Functionality

The following sections provide the steps to verify the build for STA, AP, STA+AP (multi-roll) and Mesh functionality. The required scripts to perform the tests are already installed as part of the previous build procedure. The setup is based on [AM335x EVM](#), along with [TI WL1837MODCOM8I evaluation module](#).

Before proceeding further, make sure that you have a valid image on a SD card with the build procedures mentioned above. The scripts are located at `/usr/share/wl18xx/`. This folder contains the following sample scripts for verifying functionality

```
root@am335x-evm:/usr/share/wl18xx# ls
ap_start.sh          mesh_start.sh          ps_lock.sh           udhcpd.conf
ap_stop.sh           mesh_stop.sh           set_cmd_silence.sh  udhcpd.leases
calibrate.sh         mesh_supPLICANT.conf  sta_connect-ex-dhcp.sh udhcpd2.conf
dynamic-debug.sh    mod_start.sh           sta_connect-ex.sh   udhcpd_mesh.conf
entropy.bin          mod_stop.sh           sta_start.sh         unload_wlcore.sh
hostapd.conf         p2p_cli.sh            sta_stop.sh          wlconf-toggle-set.sh
load_wlcore.sh       p2p_start.sh          testing-boot.sh      wlcore-print-fw-stat.sh
mesh_bridge.sh       p2p_stop.sh           testing.ini           wpa_supPLICANT.conf
mesh_join.sh         print_stat.sh          testing_set_wlcore.sh
root@am335x-evm:/usr/share/wl18xx#
```

Note that the configuration files needed for wpa\_supPLICANT and hostapd are also included in this folder. These files may need to be modified to enable WPA2, channel selection and mode of operation, and so forth.

### 5.1 STA Mode

The following section details how to get WiLink8 device to be a STA mode and connect to an access point and verify the connectivity between the station and AP. This uses the same hardware setup as detailed earlier.



**Figure 5-1. Wi-Fi Station Hardware Setup**

The general procedure of using the pre-built scripts for Station Mode is as follows:

- Navigate to the directory which contains the out-of-box scripts
- Start station mode
- Connect to an unsecured access point
- Request an IP address from the access point
- Ping the access point to verify the connection

### 5.1.1 Station Mode Procedure for Unsecured AP

```
root@am335x-evm:~# cd /usr/share/wl18xx/
root@am335x-evm:/usr/share/wl18xx# ./sta_start.sh
root@am335x-evm:/usr/share/wl18xx# Successfully initialized wpa_supplicant

root@am335x-evm:/usr/share/wl18xx# ./sta_connect-ex.sh <SSID Ex:abc>
root@am335x-evm:/usr/share/wl18xx# udhcpc -i wlan0
```

### 5.1.2 Station Mode Procedure for Secured AP

```
root@am335x-evm:~# cd /usr/share/wl18xx/
root@am335x-evm:/usr/share/wl18xx# ./sta_start.sh
root@am335x-evm:/usr/share/wl18xx# Successfully initialized wpa_supplicant

root@am335x-evm:/usr/share/wl18xx# ./sta_connect-ex.sh <SSID Ex:abc> WPA-PSK <password Ex:12345678>
root@am335x-evm:/usr/share/wl18xx# udhcpc -i wlan0
```

### 5.1.3 Verifying Connectivity

In order to verify the connection, use the ping command. For example, if the Access Point IP address is 192.168.1.1, we will invoke the following command on the EVM:

```
ping 192.168.1.1
```

You should see the an output similar to the following:

```
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=1003.369 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=2.526 ms
```

You can also invoke the command. This command will display the AP settings and verify the connection in wlan0 link.

## 5.2 AP Mode

The following sections detail how to get WiLink8 device to be an AP and connect another Wi-Fi device and verify the connectivity between the connected device and AP. This uses the same hardware setup as detailed earlier with one addition of using another Wi-Fi device.



**Figure 5-2. Wi-Fi Access Point (AP) Hardware Setup**

### 5.2.1 AP Mode Procedure

Configure the AP by editing the `hostapd.conf` file, located at `/usr/share/wl18xx/hostapd.conf`. In this file, there are options to change the SSID, security and other advanced features shown in the file. Key parameters to consider are listed below:

```
# SSID to be used in IEEE 802.11 management frames
ssid=xyzabc

# Channel number (IEEE 802.11)
channel=6

# ieee80211n: Whether IEEE 802.11n (HT) is enabled
# 0 = disabled (default)
# 1 = enabled
# Note: You will also need to enable WMM for full HT functionality.
ieee80211n=1

# ht_capab: HT capabilities (list of flags)
ht_capab=[SHORT-GI-20][GF][HT]

##### WPA/IEEE 802.11i configuration #####
wpa=2
wpa_passphrase=wilink80

# Set of accepted key management algorithms (WPA-PSK, WPA-EAP, or both). The
# entries are separated with a space. WPA-PSK-SHA256 and WPA-EAP-SHA256 can be
# added to enable SHA256-based stronger algorithms.
# (dot11RSNAConfigAuthenticationSuitesTable)
wpa_key_mgmt=WPA-PSK

# Operation mode (a = IEEE 802.11a, b = IEEE 802.11b, g = IEEE 802.11g,
# Default: IEEE 802.11b
hw_mode=g

# Pairwise cipher for WPA (v1) (default: TKIP)
wpa_pairwise=TKIP CCMP

# Pairwise cipher for RSN/WPA2 (default: use wpa_pairwise value)
rsn_pairwise=CCMP
```

### 5.2.2 Starting the AP

Run the `ap_start.sh` script:

```
root@am335x-evm:/usr/share/wl18xx# ./ap_start.sh
adding wlan1 interface
Configuration file: /usr/share/wl18xx/hostapd.conf
[ 4398.173284] IPv6: ADDRCONF(NETDEV_UP): wlan1: link is not ready
wlan1: interface state UNINITIALIZED->COUNTRY_UPDATE

root@am335x-evm:/usr/share/wl18xx# Using interface wlan1 with hwaddr 0c:1c:57:bb:60:5f and ssid "kns"
[ 4403.305047] netlink: 'hostapd': attribute type 213 has an invalid length.
[ 4403.321775] IPv6: ADDRCONF(NETDEV_CHANGE): wlan1: link becomes ready
[ 4403.430771] cryptd: max_cpu_qlen set to 1000
wlan1: interface state COUNTRY_UPDATE->ENABLED
wlan1: AP-ENABLED
root@am335x-evm:/usr/share/wl18xx#
```

### 5.2.3 Verifying Connectivity

To verify the AP broadcasting, use any commercial Station (smart phone, laptop, and so forth) and connect. You should see a prompt (`AP_STA_CONNECTED`) when the station successfully connects to the WiLink AP.

## 5.3 Multirole (AP +STA mode)

This demo shows how to use the WiLink device as both a station and as an AP, using the out-of-box scripts available at `/usr/share/wl18xx`. An example use-case for a multirole Wi-Fi device might be to act as a wireless bridge. In multirole, the WiLink device can connect to an internet-enabled secure AP (STA mode) as a client, while simultaneously acting as a station to which other devices can then connect to for internet access. The same hardware used for AP mode testing can be used.

### 5.3.1 General Procedure for Multirole Connection

1. Start station role.
2. Connect to an access point.
3. Start AP role.

The following scripts need to be run to enable STA and AP role. `hostapd.conf` updates mentioned in AP role are applicable to this procedure as well.

```
./sta_start.sh
./sta_connect-ex.sh exampleSSID WPA-PSK examplepassword
udhpcp -i wlan0
./ap_start.sh
```

If the default `hostapd.conf` settings were not changed, the AP should now be broadcasting as SitaraAP while also being connected as a Station to the AP specified in the "Start Station Role" section. To verify the station connectivity, invoke:

```
iw wlan0 link
```

The details of the AP you are connected to should show up. To verify the AP broadcast, use any commercial Station (smart phone, laptop, etc.) and connect. You should see a prompt (`AP_STA_CONNECTED`) when the station successfully connects to the WiLink AP.

### 5.4 IEEE802.11s Mesh Mode

WiLink8 devices also support IEEE 802.11s mesh functionality. To verify this mode, two WiLink8 setups used earlier are needed at a minimum. More devices can be connected added to the mesh network as well.



**Figure 5-3. Wi-Fi Mesh Hardware Setup**

More details on the mesh network, supported topologies and usage of sample scripts to start the network are provided in [WiLink™ 8 WLAN Software - 802.11s Mesh](#).

## 6 References

- Texas Instruments: [WiLink™ 8 WLAN Features User's Guide](#)
- Texas Instruments: [WiLink™ Module Hardware Integration Guide](#)
- Texas Instruments: [WiLink™ 8 Solutions WiLink8 – wlconf](#)
- Texas Instruments: [WL18xx .ini File](#)
- Texas Instruments: [WL18x7MOD WiLink™ 8 Dual-Band Industrial Module – Wi-Fi®, Bluetooth®, and Bluetooth® Low Energy \(LE\) Data Sheet](#)

## A FAQ and Debug Hints

The following section is intended to provide generic issues faced while integrating and running the WiLink8 drivers on Linux platform. For more comprehensive FAQ list and for additional help, you can reach [E2E forum](#).

### Q: How do I know if the Wi-Fi is functional?

A. Bring up the WLAN interface and perform a scan using the “iw” utility:

```
ifconfig wlan0 up
```

The following message should be displayed below:

```
wlcore: PHY firmware version: Rev 8.2.0.0.244  
wlcore: firmware booted (Rev 8.9.0.0.84)
```

Next, perform a scan and look for scan results:

```
iw wlan0 scan | grep <SSID>  
    SSID: IOP_035  
    SSID: Demo_24  
    SSID: externalhotspot84
```

If any errors were encountered, follow the steps below:

1. Check your device configuration.

Did you remember to use the configure-device.sh script upon initialization?

---

#### Note

This script is located at /usr/share/wl18xx/configure-device.sh.

---

Make sure you are using the proper .ini file to match your needs.

For more information, see [WL18xx .ini File](#).

2. Try to reproduce on 1-2 other platforms.
3. Try to reproduce the issue with the latest firmware and software driver version possible (currently R8.8).
4. Try to reproduce with the Enhanced Low Power (ELP) mode disabled.

To disable ELP:

```
iw wlan0 set power_save off  
echo 0 > /sys/kernel/debug/ieee80211/phy0/wlcore/sleep_auth
```

5. Try to reproduce with a different peer vendor.

For STATION/CLIENT mode - try with a different Access Point vendor.

For Access Point (AP) mode - try with a different Station vendor.

For Peer to Peer (P2P) mode - try with a different P2P vendor.

If still not resolved, find your specific case below.

**Q: I am able to bring the interface up using *ifconfig* but when performing a scan I see a driver crash log.**

**A:** Ensure that you are receiving interrupts from the wl18xx device. This can be done using the following command:

```
cat /proc/interrupts | grep wl18xx
```

Following output, or something similar should be seen:

```
54:          15  44e07000.gpio  27 Edge          wl18xx
```

If the WL\_IRQ pin is configured correctly, you should see a number higher than "0" as shown above. If the value is zero, re-visit your board device tree file and make sure the WL\_IRQ GPIO is muxed correctly and no errors are seen when loading the "wlcore\_sdio" module.

**Q: I've checked that I have the right pins are connected and MUXed, but still the WLAN interface does not work.**

**A:** Make sure the power-on and reset sequences are followed based on the design guidelines in the data sheet. For more information, see the *Power-Up and Shut-Down States* section in the [WL18x7MOD WiLink™ 8 Dual-Band Industrial Module – Wi-Fi®, Bluetooth®, and Bluetooth® Low Energy \(LE\) Data Sheet](#). In other words, VBAT/VIO voltages and the slow clock (32 kHz) must be stable prior to WLAN\_EN being engaged. Once WL\_IRQ is reading as logic '0', the module is awake. When the IRQ is triggered for the first time, the host can start communication over the SDIO interface.

**Q: I've confirmed the power-up and reset sequences are being followed, but the WLAN interface is still not working.**

**A:** Make sure a WLAN card is detected during device enumeration. If the platform integration was done according to the hardware spec guide/platform integration guide, an SDIO device should be detected during kernel boot. Please review your kernel boot log and look for the following message:

```
[ 18.538564] mmc1: new high speed SDIO card at address 0001
```

**Q: I've confirmed that the WLAN device is being detected on the SDIO interface, but WLAN is still not working.**

**A:** Make sure that the WLAN drivers are loaded, using the WL8 software build process in the WL18xx processor wiki, or built inside the kernel (in case using a kernel version >= 4.1)

You should see the following modules loaded when using the "lsmod": command:

Module	Size	Used by
...		
wl18xx	83954	0
wlcore	186624	1 wl18xx
mac80211	479316	2 wl18xx,wlcore
cfg80211	397999	3 mac80211,wl18xx,wlcore
wlcore_sdio	7829	0

**Q: Yes, I do have the modules built but I still do not see the interface up when using *ifconfig*, instead I am seeing the following error message?**

```
SIOCGIFFLAGS: No such device
```

**A:** This error indicates that the modules are not loaded properly. Try to insert the modules manually and look for errors during the modules loading.

**Q: How do I check which versions of WiLink drivers and firmware I have?** **A:** To find out the version of WiLink™ firmware (usually named wl18xx-fw-x.bin), type the following command into the Sitara serial terminal once booted and logged in as root:

```
grep Rev /lib/firmware/ti-connectivity/wl18xx-fw-4.bin
```

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to [TI's Terms of Sale](#) or other applicable terms available either on [ti.com](http://ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2022, Texas Instruments Incorporated