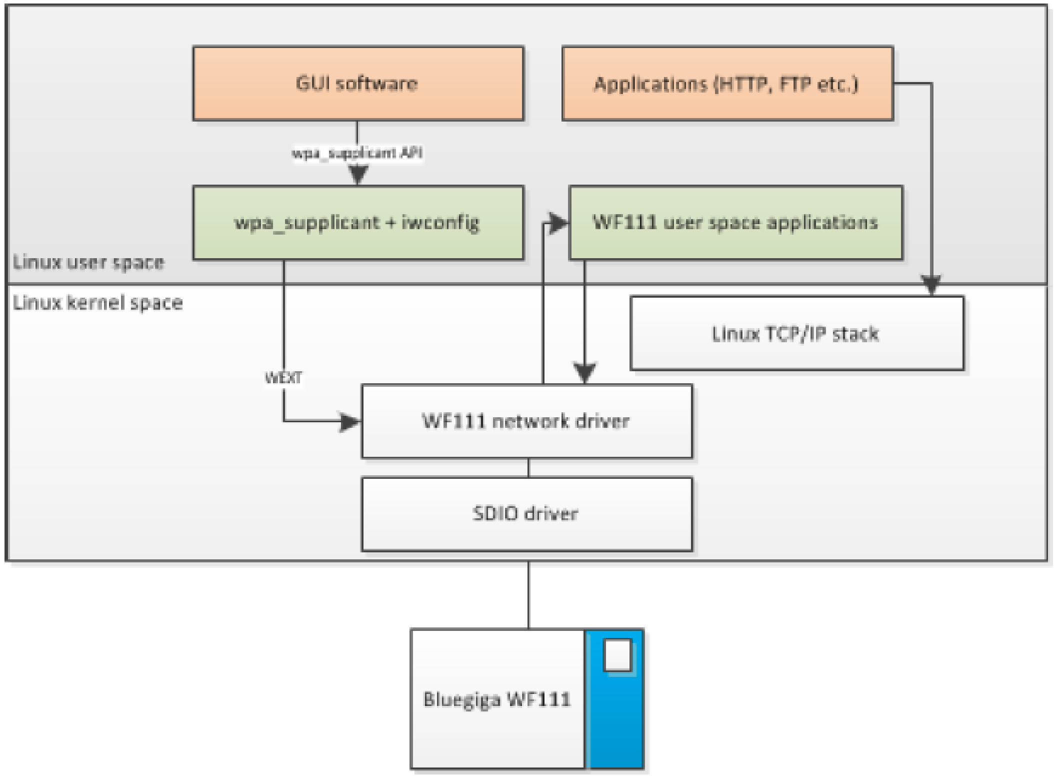
WF111 Troubleshooting

Linux driver architecture: overview



Wireless Extensions (WEXT) are based on ioctl() which provides the standard transport for communication between user <–> kernelspace

SILICON LABS

Linux driver architecture: WEXT

In the Linux kernel configuration you must have the following options enabled:

```
CONFIG_WIRELESS_EXT
CONFIG_MODULES
CONFIG_FW_LOADER
```

Please note that these options have to be enabled from the kernel of target machine.

In case during compilation you get the message

```
[...] netdev.c:614: error: 'struct net_device' has no member named 'wireless_handlers'
```

it means that Wireless-Extensions are not enabled.

In syslog/dmesg the following line should be found:

```
CSR SME with WEXT support
```

It might not appear if removing then re-inserting the card, as it is loaded only the first time it needs to be used, but it should always appear when loading the kernel driver with “modprobe unifi sdio….”

Linux driver files

```
/lib/modules/KERNEL_VERSION/extra/unifi_sdio.ko      ← copy to target machine after compilation
/lib/firmware/unifi-sdio-1 -> unifi-sdio-0            ← create softlink if using other than the first SDIO slot
/lib/firmware/unifi-sdio-0/staonly.xbv               ← just copy the rest of the files from the archives
/lib/firmware/unifi-sdio-0/ap.xbv
/lib/firmware/unifi-sdio-0/mib111_drv_coex.dat
/lib/firmware/unifi-sdio-0/mib111_drv_led.dat
/lib/firmware/unifi-sdio-0/mib111_drv_coex_led_etsi.dat
/lib/firmware/unifi-sdio-0/ufmib.dat -> mib111_drv_led.dat
/lib/firmware/unifi-sdio-0/mib111_drv_coex_etsi.dat
/lib/firmware/unifi-sdio-0/mib111_drv.dat
/lib/firmware/unifi-sdio-0/mib111_drv_coex_led.dat
/usr/sbin/unififw
/usr/sbin/unifi_helper
/usr/sbin/unifi_config
```

Third SDIO slot:
https://bluegiga.zendesk.com/entries/23428622-How-to-enable-a-third-SDIO-slot-for-the-WF111

SILICON LABS

Step 1

Once the kernel module is compiled and copied into the system, together with the other driver and firmware files, as soon as the WF111 is recognized by the system the following steps are taken until the module is fully operational.

STEP 1 - Plug in the WF111 into the SDIO port

Explanation:

- This is simply the step when the WF111 is physically attached to the MCU via the SDIO interface

Step 2

STEP 2 - Linux SDIO subsystem notices the WF111

Explanation:

- Upon attaching the WF111 to the MCU over the SDIO interface, the system recognizes the new device.

How to verify that this step was successful:

- An entry similar to the one below should appear in the syslog/dmesg:

```
mmc0: new high speed SDIO card at address 0001
```

Possible errors and solutions:

- SDIO driver missing: the WF111 is not recognized, thus the above entry is not printed
- Verify that the mmc/sd/sdio card support is compiled into the kernel.
- Check the physical connections and power supply.

Step 3

STEP 3 - Kernel loads WF111 driver

Explanation:

- When the WF111 is recognized the kernel starts loading the CSR's unifi driver.

How to verify that this step was successful:

- The following entries should appear in the syslog/dmesg:

```
UniFi SDIO Driver: 5.1.0 May 28 2013 13:22:49
CSR SME with WEXT support
Split patch support
Kernel 3.2.36
UniFi: Using native Linux MMC driver for SDIO.
sdio bus_id: mmc0:0000:1 - UniFi card 0x0 inserted
```

SILICON LABS

Step 3

Possible errors and solutions:

- If you see the following entry in the syslog/dmesg:

```
Unifi: Using CSR embedded SDIO driver
[sdioemb: registered unifi driver
```

then the problem is that the native sdio linux drivers are not used and the solution is not to remove the mmc parameter during compilation

- Missing udev (daemon) or busybox's mdev (program):

  - They are supposed to run in the user space.
  - They are normally in charge of automatically launching modprobe.
  - If missing, manual loading should be performed, using

“depmod –a” + “modprobe unifi_sdio” or

“insmod /path/to/unifi_sdio.ko”

Step 4

STEP 4 - UnifiFW (user space script) is launched

Explanation:

- When loaded, the driver itself will launch the script called unififw
- Tasks of the scripts are described in steps 4.1 to 4.3

How to verify that this step was successful:

- At the shell, the ps axu command should reveal the running script with a never expiring line as below:

```
900 root     {unififw} /bin/sh /usr/sbin/unififw 0 2
```

- In the syslog the following would appear:

```
unifi0: starting /usr/sbin/unififw
unifi0: running /usr/sbin/unififw 0 2
```

(bottom line is only seen when debug level 3 is used)

SILICON LABS

Step 4

Possible errors and solutions:

- unififw script file must be executable (check rights)
- unififw script file must exist in the filesytem, in the correct directory

Step 4.1

STEP 4.1 - Unifi_helper application started

Explanation:

- At first, the unififw script will launch the unifi_helper, which is the 802.11 MAC stack running in the user space.

How to verify that this step was successful:

- At the shell, the ps axu command should reveal the running stack with a never expiring line as below:

```
907 root     /usr/sbin/unifi_helper --wifi-char-device /dev/unifi1 --wifi-on --wifi-restart-on-error
--wifi-exit-on-error --wifi-exit-on-unplug --wifi-address FF:FF:FF:FF:FF:FF --wifi-mib /lib/firmware/unifi-
sdio-1/ufmib.dat
```

- In the syslog the following would appear:

```
unifi0: Initialising UniFi, attempt 1
unifi0: Resetting UniFi
unifi0: Chip ID 0x07  Function 1  Block Size 512  Name UniFi-4(UF60xx)
unifi0: Block mode SDIO
unifi0: Chip Version 0x3A22
```

SILICON LABS

Step 4.1

Possible errors and solutions:

- unifi_helper file must be executable (check rights)
- unifi_helper file must exist in the filesytem , in the correct directory

  The syslog might then report the following warning:

```
Jan  22 16:29:13 tegra-ubuntu unififw: /usr/sbin/unififw: 94: /usr/
sbin/unififw: /usr/sbin/unifi_helper: not found
```

- When using dynamically linked application, make sure that all needed libraries are available in the OS.

  Verify the needed libraries and versions with a command similar to the following one for Ubuntu: “arm-linux-gnueabi-objdump -x unifi_helper”

  Needed libraries are:

  - libpthread.so.0
  - librt.so.1
  - libgcc_s.so.1
  - libc.so.6

Step 4.2

STEP 4.2 - Firmware patch loading to WF111

Explanation:

- .xbv firmware patch files are delivered to the Wi-Fi chip via the user space hotplug handlers (udev or mdev)

How to verify that this step was successful:

- In the syslog the following would appear:

```
unifi1: unifi_dl_patch c3a75e44 0100060e
unifi1: SDIO block size 64 requires 8 padding chunks
unifi1: UniFi f/w protocol version 9.1 (driver 9.1)
unifi1: Firmware build 1089: 2010-10-05 14:50 cindr03_core_softmac_rom_sdio_
        gcc 1089  bfsw@eagle@630492
```

- In the syslog the following would appear, if changing to AP mode with iwpriv (debug level 3 only):

```
unifi1: Resetting UniFi with AP patch
unifi mmc1:0002:1: firmware: requesting unifi-sdio-1/ap.xbv
```

Step 4.2

- If udev is used, at the shell the ps axu command should show udev running with a never expiring entry as in the example below:

```
root       305  0.0  0.0   2848   848 ?        S<   06:23   0:00 udevd –daemon
```

- If Busybox’s mdev is used instead, you can verify that path to mdev is in “cat /proc/sys/kernel/hotplug”

Possible errors and solutions:

- Version mismatch: firmware patch is not sent to module, so default module’s firmware is used which does not match with driver version.

  In the syslog/dmesg the following would appear:

  ```
  unifi0: UniFi f/w protocol version 8.0 (driver 9.1)
  ```

- Check that .xbv file exists in the filesystem in the correct directory (check softlink if using other than the first SDIO slot, for example "unifi-sdio-2 -> unifi-sdio-0/“ if using third SDIO slot)
- Make sure that udev or mdev are operational

Step 4.3

STEP 4.3 - MIB settings are entered

Explanation:

- MIB parameters from .dat files are delivered to the module's chipset by the unifi_helper application
- A .dat file might contain a MAC address used to overwrite the unique module's factory MAC address. Alternatively, such MAC address can exist in a separate .txt file as follows: /lib/firmware/unifi-sdio-0/mac.txt

How to verify that this step was successful:

- MAC address can for example be verified with command : "ifconfig wlan0"

Possible errors and solutions:

- See application note

Ready-to-use

In syslog/dmesg the following line should be found:

```
unifi0: unifi0 is wlan0
unifi0: UniFi ready
```

Use Linux programs to test that all is fine:

Ifconfig

iwlist scan

iwconfig

    only for testing with non-secured networks

wpa_supplicant

    .conf

```
/usr/local/sbin/wpa_supplicant -B -Dwext -iwlan0 -c/etc/wpa_supplicant.conf
```

The unififw script keeps running even after all tasks are executes (seen using ps)

SILICON LABS

SDIO Problems (most common)

- How are the SDIO data lines wired? (Short wires and proper grounding should be used: remember that signal frequency is high)

- Are SDIO interrupts and SDIO multiblock tranfer supported by the SDIO host driver?

- Does decreasing the bus frequency to 400 kHz make the module initialize appropriately? (modprobe unifi_sdio sdio_clock=400)

- Does switching the bus width into 1 bit mode make the module initialize appropriately? (modprobe unifi_sdio buswidth=1)

- Does testing the latest mainline kernel make the module initialize appropriately (It might have improvements in SDIO drivers)

- Use atmel_mci driver instead of at91_mci with Atmel processors (at91_mci is known to have issues in 2.6.36 and earlier)

SILICON LABS

SDIO Problems

Typical syslog/dmesg examples revealing SDIO problems

```
unifi0: UniFi ready
unifi0: Block read failed after 3 retries
unifi0: Failed: CopyToHost hlen=1088, slen=1088, handle 94 - slot=13 bfa8a024+0x0
unifi0: Failed to process bulk cmd
unifi0: Error occured processing to-host signals
unifi0: unifi_bh: state=0 A, clock=50000kHz, interrupt=0 host=0, power_save=enabled
unifi0: handle_bh_error: fatal error is reported to the SME.
unifi0: SME client close (unifi0)
unifi0: Message for the SME dropped, SME has gone away
unifi0: Initialising UniFi, attempt 1

unifi0: SME SEND: Signal 0x0208
unifi0: bh_thread calls unifi_bh().
unifi0: Sending signal 0x0208
unifi0: incomplete signal or command: has size zero
unifi0: Error occured processing to-host signals
unifi0: unifi_bh: state=0 A, clock=12500kHz, interrupt=0 host=0, power_save=disabled
unifi0: Mini-coredump requested after reset
unifi0: handle_bh_error: fatal error is reported to the SME.
unifi0: SME client close (unifi0)

unifi0: Block read failed after 3 retries
unifi0: Failed to read ToHost signal
unifi0: Error occured reading to-host signals
```

Thank You

www.silabs.com