

WE865-DUAL SW User Guide

1v0300788 Rev. 0 08/08/08



Wireless Tools project site: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html

1.4.2 Wpa Supplicant

WPA Supplicant is free software; it can be redistributed and/or modified under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

For further information about GNU License please have a look at <http://www.gnu.org/copyleft/gpl.html>. Alternatively, this software may be distributed, used, and modified under the terms of BSD license.

Jouni Malinen's wpa_supplicant official project site: http://hostap.epitest.fi/wpa_supplicant/.

1.4.3 CSR Linux WiFi Driver

CSR Linux WiFi driver is licensed as follows:

SOFTWARE LICENCE AGREEMENT FOR UNIFI LINUX DRIVER SOURCE CODE

By receiving this software, the customer (YOU) accepts the terms and conditions herein.

GRANT OF LICENCE

Cambridge Silicon Radio Limited, hereafter referred to as CSR, grants YOU a worldwide royalty-free nonexclusive licence to use and distribute this software including source code under the following conditions:

- 1) The source will only be used in conjunction with projects that use CSR UniFi chips.
- 2) YOU will provide the source code of any bug fixes to the software back to CSR under the same terms as to which CSR provides the original software to YOU.
- 3) CSR does not accept liability for any bugs in the software.

LIMITATION OF LIABILITY

CSR makes no warranties as to the fitness for purpose, merchantability or function of this software. CSR accepts no responsibility for the use of the software. CSR accepts no liability for consequential loss. CSR does not warrant or provide any indemnification with respect to intellectual property infringement claims for third party claims.

EXTENTS

Where there are other agreements between YOU and CSR, the restrictions imposed by the other agreements shall be additive, and where there is conflict between agreements, any restrictions in agreements shall take precedence over grants made by this agreement, with the specific exception of other agreements granting distribution rights over this software.

GOVERNING LAW

These Terms and the supply of the Products by CSR are governed by English law, and YOU agree to resolve all disputes exclusively in the English Courts, but without prejudice to our right to seek injunctive or other relief in any court of competent jurisdiction world-wide.



1.7 Text Conventions

This section lists the paragraph and font styles used for the various types of information presented in this user guide.

Format	Content
Courier	Linux shell commands at command prompt.

1.8 Related Documents

The following documents are related to this user guide:

- [1] TelitGE863-PRO³ Hardware User Guide 1vv0300773a
- [2] TelitGE863PRO³ EVK User Guide 1VV0300776
- [3] TelitGE863PRO³ Linux SW User Guide 1vv0300781
- [4] TelitGE863PRO3Linux Development Environment User Guide1VV0300780
- [5] TelitWE865-DUAL Product Description
- [6] TelitWE865-DUAL Hardware User Guide

All documentation can be downloaded from Telit’s official web site www.telit.com if not otherwise indicated.

1.9 Document History

Revision	Date	Changes
ISSUE #0	08/08/08	First Release



2 WE865-DUAL architecture

2.1 Hardware

WE865-DUAL WiFi module is connected and communicates with GE863-PRO³ through an SDIO interface. For further hardware information please refer to [1] , [2] , [5] and [6]

2.2 Software

Studying Linux Operating System and Linux WiFi Software Framework is important to better understand how WE865-DUAL can be configured and controlled.

Below you can find a high level description of Linux OS Architecture and the different software layers involved in WE865-DUAL control.

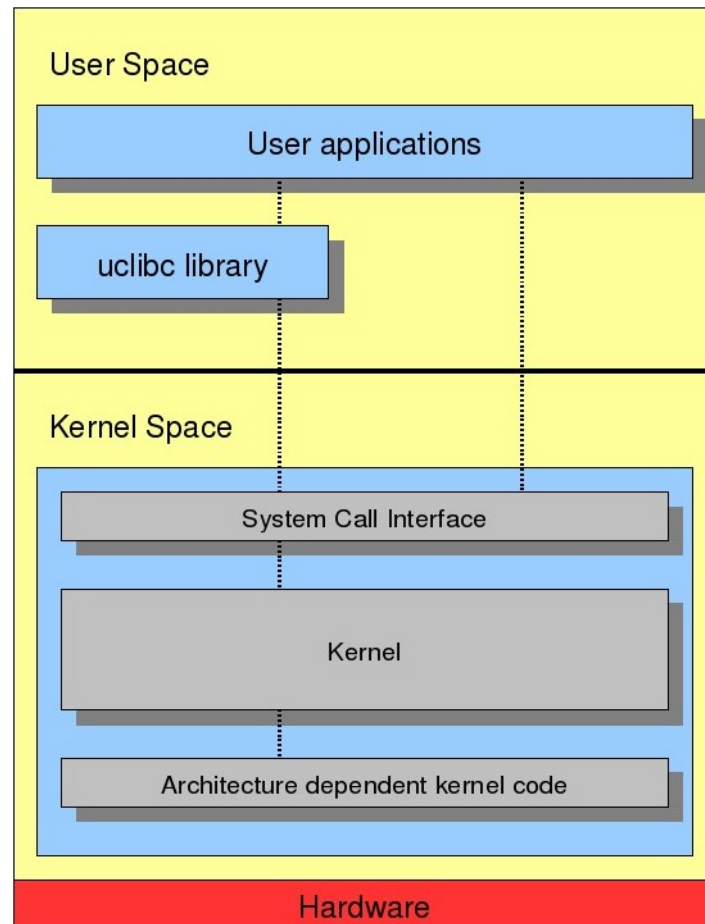
2.2.1 Linux OS overview

The kernel is the central part of the GNU/Linux operating system: its main task is to manage system's resources in order to make the hardware and the software to communicate. A kernel usually deals with process management (including inter-process communication), memory management and device management.

The Linux kernel belongs to the family of Unix-like operating system kernel; created in 1991, it has been developed in the years by a huge number of contributors worldwide, becoming one of the most common and versatile kernel for embedded systems.

Below there is a picture representing, from a high level perspective, the architecture of a GNU/Linux operating system.





Two regions can be identified:

- 1) User space: where the user applications are executed.
- 2) Kernel space: where the kernel (with all its components such as device drivers) works.

These two regions are separated and have different memory address spaces; there are several methods for user/kernel interaction:

- Using the System Call Interface that connects to the kernel and provides the mechanism to communicate between the user-space application and the kernel through the C library.
- Using kernel calls directly from application code leaping over the C library.
- Using the virtual filesystem /proc.

The ordinary C library in Linux system is the glibc. Uclibc is a C library mainly targeted for developing embedded Linux systems; despite being much smaller than the glibc it almost has all its features (including shared libraries and threading), making easy to port applications from glibc to uclibc.

The Linux kernel architecture-independent code stays on the top of platform specific code for the GE863-PRO³ board: this code allows exploiting all the hardware features of the GE863-PRO³.



2.2.2 Linux WiFi software framework

WE865-DUAL Linux WiFi package is made up of different components:

- WiFi Driver – WE865-DUAL Linux WiFi device driver
- Wireless Tools – Set of tools for configuring and managing WE865-DUAL
- Wpa Supplicant – Tool for configuring and managing WPA/WPA2 security

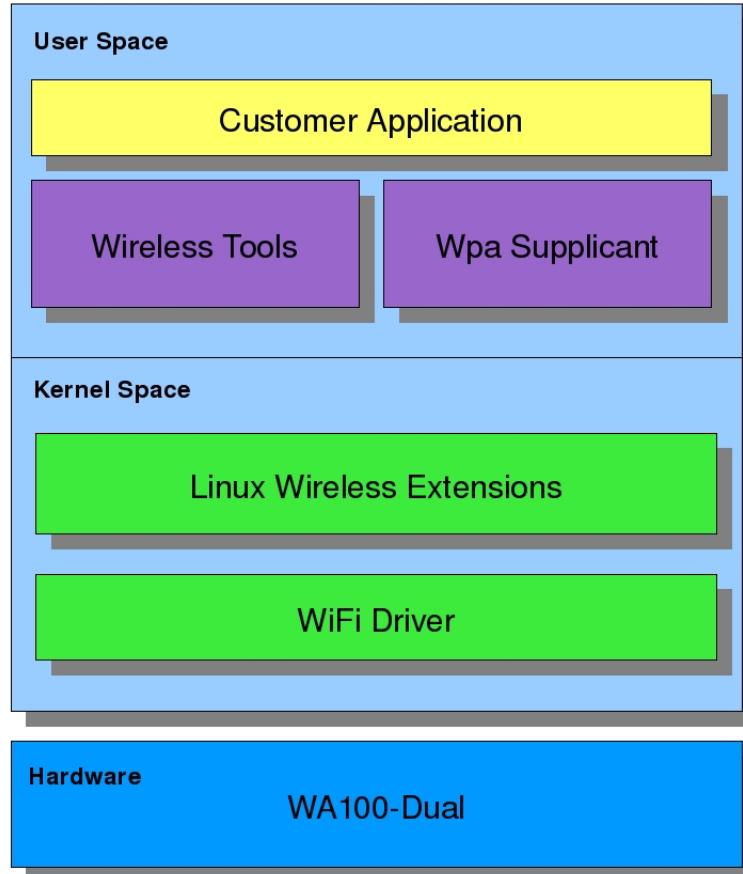
WE865-DUAL WiFi module is controlled, under Linux OS, by the means of a WiFi device driver loaded into Kernel Space.

WE865-DUAL functionalities are made available to User Space applications through Linux Wireless Extensions (WE), kernel space generic APIs allowing a driver to expose to the user space configuration and statistics specific to common Wireless LANs.

Customer applications can control/configure WE865-DUAL through simple system calls to shell commands such as Wireless Tools and Wpa Supplicant.

The image below shows the software framework used to configure and control WE865-DUAL.

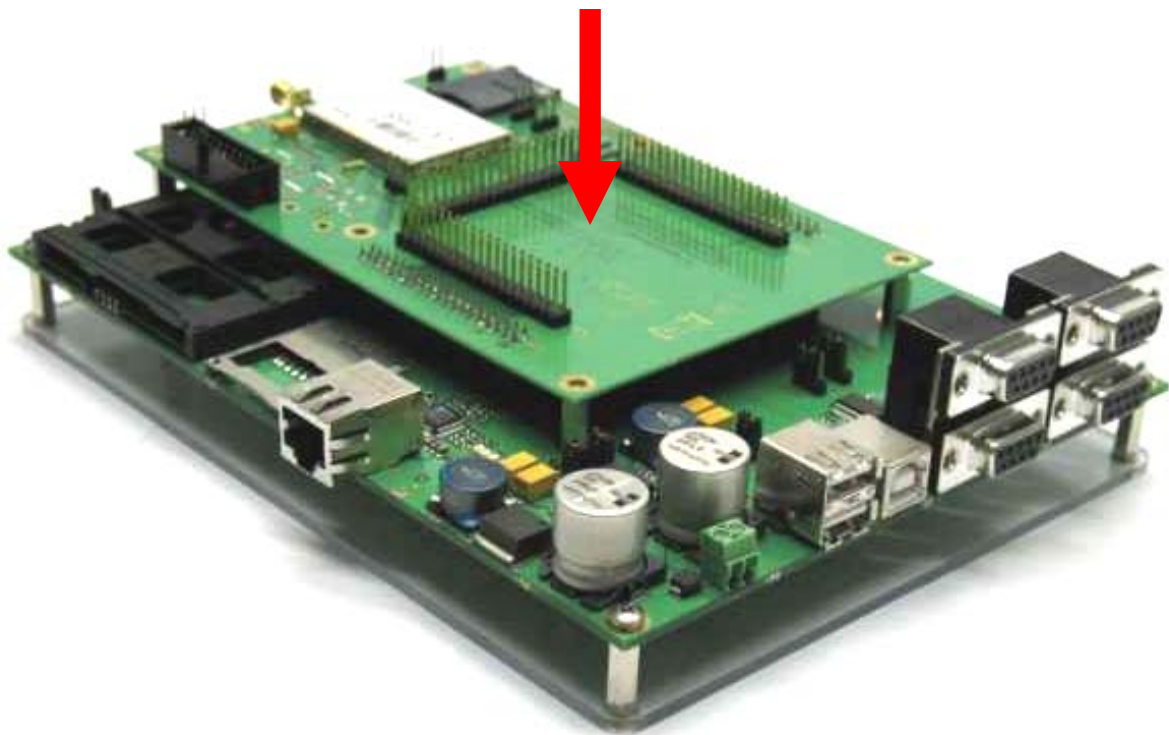




3 Connecting WE865-DUAL to PRO3

Prior to any use WE865-DUAL interface board must be correctly connected to GE863-PRO³ as shown below.

Please Note: to disable WE865-DUAL internal voltage regulators PL101 and PL102 jumpers must be closed (see [6] for further information).

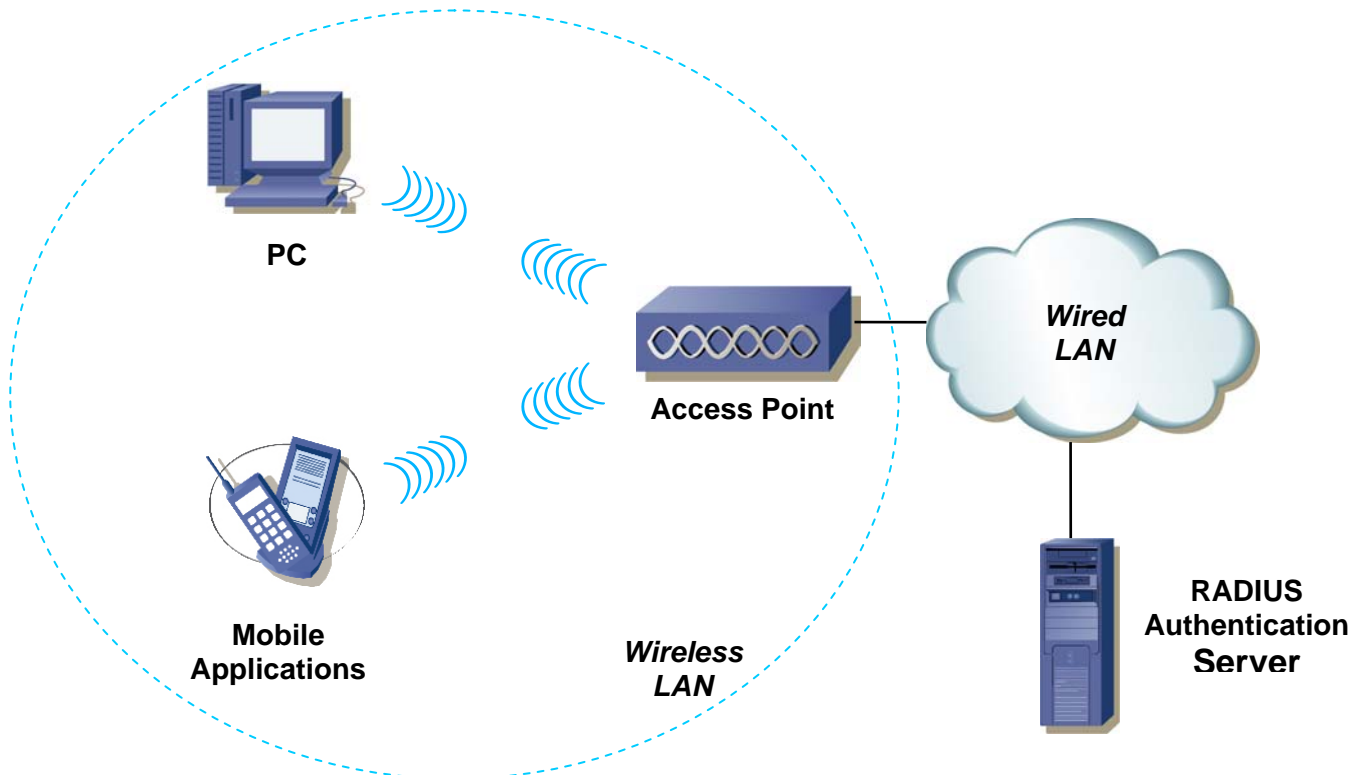


4 Configuring WE865-DUAL

In order to better understand how to configure WE865-DUAL wifi module it is very important to read the introduction in typical wireless networks configurations and devices involved.

4.1 Typical IEEE 802.11 network scenario

A typical wireless network is mainly made up of clients and Access Points (AP). Clients (such as PCs, PDAs, laptops, VOIP phones, etc.) connect to an AP. The AP usually connects to a wired network and can relay data between wireless devices and wired devices. Connection between clients and the AP can be secured enabling different encryption modes like WEP, WPA and WPA2 (IEEE 802.11i). Enterprise wireless LANs usually use RADIUS authentication servers along with encryption in order to have as strong as possible WiFi connections. When WPA/WPA2 encryption is used, we talk about WPA-Personal in non Enterprise environments, and WPA-Enterprise otherwise. For further information about WPA and WPA2 (IEEE 802.11i) see 8.1 paragraph.



When clients connect to each other through an AP, as shown above, they operate in Managed/Infrastructure mode.

Another WiFi network topology that allows clients to directly connect to each other without APs forming a peer-to-peer link is called Ad-Hoc.

4.2 WE865-DUAL Setup

4.2.1 WiFi Package Downloading

Before setting up WE865-DUAL, the components of the WiFi Package must be downloaded onto GE863-PRO³ filesystem.

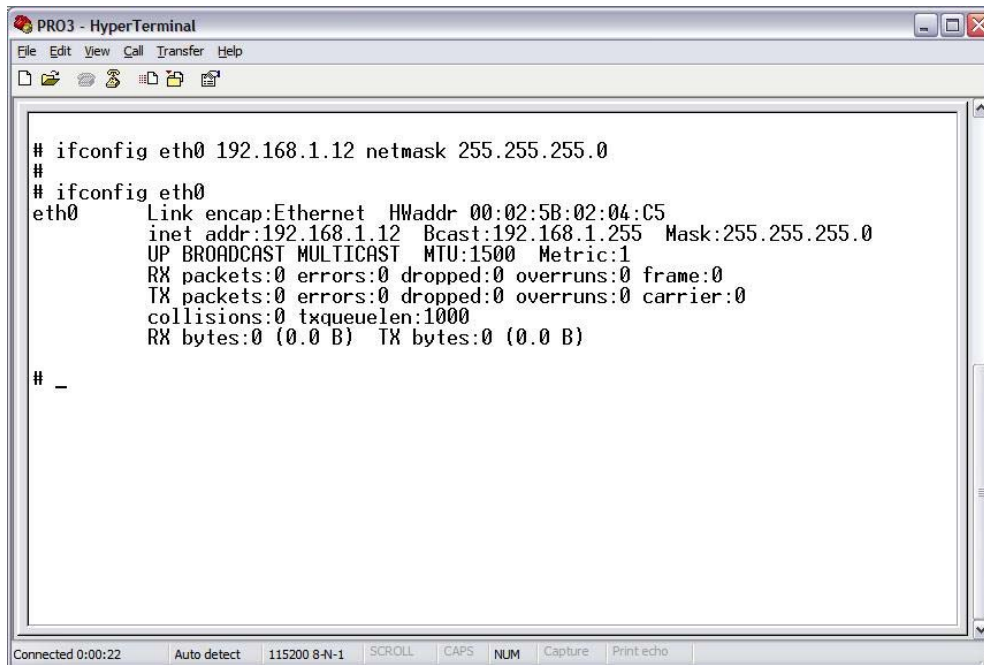
If you don't have WE865-DUAL WiFi Package yet, you can download it from Telit's official web site Download Zone <http://www.telit.com/en/products/download-zone.php>.

Connect the GE863-PRO³ to your host system via serial cable (use Debug port of the EVK, for further details refer to document [2]). Open a terminal program (such as Hyperterminal) on your host system and use for the connection the following parameters:

Bits per second: 115200
Data bits: 8
Parity: None
Stop bits: 1
Flow Control: None

Turn the GE863-PRO³ on. Once the system startup has finished, the terminal will display the shell prompt as shown below.





```
PRO3 - HyperTerminal
File Edit View Call Transfer Help
# ifconfig eth0 192.168.1.12 netmask 255.255.255.0
#
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:02:5B:02:04:C5
          inet addr:192.168.1.12  Bcast:192.168.1.255  Mask:255.255.255.0
          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)

# _
Connected 0:00:22  Auto detect  115200 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

Once the eth0 network interface has been activated it is possible, for example, to perform a scanning of wifi networks as shown below:

```
# iwlist eth0 scan
```



```

PRO3 - HyperTerminal
File Edit View Call Transfer Help
# iwlist eth0 scan
unifi:Scanning for wireless networks
eth0      Scan completed :
          Cell 01 - Address: 00:17:9A:B7:9F:E2
                   ESSID:"test"
                   Mode:Master
                   Channel:13
                   Quality=21/12  Signal level=-74 dBm  Noise level=-95 dBm
                   Encryption key:off
                   Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                               12 Mb/s; 24 Mb/s; 36 Mb/s; 9 Mb/s; 18 Mb/s
                               48 Mb/s; 54 Mb/s

# _

```

Then it is also possible to associate with a specific SSID:

```
# iwconfig eth0 essid "test"
```

```

PRO3 - HyperTerminal
File Edit View Call Transfer Help
# iwconfig eth0 essid "test"
unifi:Scanning for wireless networks
unifi:eth0: link is up
unifi:Associated with "test"
# iwconfig eth0
eth0      IEEE 802.11-b/g  ESSID:"test"
          Mode:Auto  Channel:13  Access Point: 00:17:9A:B7:9F:E2
          Bit Rate=1 Mb/s
          RTS thr:off  Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=25/12  Signal level=-75 dBm  Noise level=-100 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0  Missed beacon:0

#

```



For further information about wifi specific parameters configuration and WE865-DUAL management please refer to paragraphs 5, 6 and 7.

4.2.4 Auto-Setup at system startup

It is possible to automatically perform all the steps shown above at system startup. A startup script as the one shown below has to be written:

```
insmod /lib/unifi_sdio.ko  
sleep 15  
ifconfig eth0 192.168.1.12 netmask 255.255.255.0  
ifconfig eth0 up
```

The first line loads the wifi driver, the second adds a 15 seconds delay to let the driver to be loaded. The last two lines set an IP address and activate the wifi interface. The startup script has to be saved as "S03" into /etc/init.d system folder.



5 Commands summary

All the specific wifi parameters for WE865-DUAL can be configured and controlled through Wireless Tools and Wpa Supplicant.

Wireless Tools (iwconfig, iwlist, ifrename, iwevent and iwgetid) and WPA Supplicant, simple linux shell commands, can be used for example to perform scanning and association to a wireless network and/or to set the desired operating mode and to manage WPA security modes and related settings.

The table below shows examples of the most commonly used shell commands.

Functionality		Shell Commands (Wireless Tools, WPA Supplicant)
WiFi Network Info	WiFi network Scanning	iwlist eth0 scan
	AP Statistics Collecting	iwconfig eth0
	WiFi Interface Settings	iwconfig eth0
	WiFi MAC Protocol Used	iwconfig eth0
	WiFi Channel Used	iwconfig eth0 or iwlist eth0 frequency
	Associated AP MAC Address	iwconfig eth0
	WiFi Operating Mode Used	iwconfig eth0
	Network Interfaces Names	ifconfig
WiFi Interface Config	WiFi Network Interfaces Names	iwconfig iwconfig
	WiFi Interface IP Setting	ifconfig eth0 192.168.1.12 netmask 255.255.255.0
	Managed/Infrastructure Mode Setting	iwconfig eth0 mode Managed
	Ad-Hoc Mode Setting	iwconfig eth0 mode Ad-Hoc
	WiFi Channel Setting	iwconfig eth0 channel 3
	WiFi Bitrate Setting	iwconfig eth0 rate 11M
WiFi Security Config	ESSID Associating	iwconfig eth0 essid "test"
	WEP Key Setting	iwconfig eth0 key s:password
	WEP Encryption Setting	iwconfig eth0 key restricted
	WPA/WPA2 Encryption Setting	wpa_supplicant -Dunifi -ieth0 -c/etc/wpa_supplicant.conf -d

All the shell commands seen above can be used from source code performing system calls.



6 Wireless Tools

Linux Wireless Tools (WT) is a set of tools allowing to configure and manage WE865-DUAL by linux command shell.

Wireless Tools package includes the following executables:

- **iwconfig** manipulates the basic wireless parameters
- **iwlist** allows to initiate scanning and list frequencies, bit-rates, encryption keys...
- **ifrename** allows to name interfaces based on various static criteria
- **iwevent** allows to display wireless events
- **iwgetid** shows the ESSID or NWID of the specified device

Please have a look to paragraph 1.4.1 for information about Wireless Tools License.

The following paragraphs describe WT commands as shown in man pages.

6.1 iwconfig

iwconfig is similar to ifconfig, but is dedicated to the wireless interfaces. It is used to set the parameters of the network interface which are specific to the wireless operation (for example: the frequency). Iwconfig can also be used to display those parameters, and the wireless statistics (extracted from /proc/net/wireless).

All these parameters and statistics are device dependent. Each driver will provide only some of them depending on hardware support, and the range of values may change. Please refer to the man page of each device for details.

Synopsis

```
iwconfig [interface]
iwconfig interface [ssid X] [nwid N] [mode M] [freq F]
                  [channel C][sens S ][ap A ][nick NN ]
                  [rate R] [rts RT] [frag FT] [txpower T]
                  [enc E] [key K] [power P] [retry R]
                  [commit]

iwconfig -help
iwconfig -version
```



Parameters

ssid

Set the ESSID (or Network Name - in some products it may also be called Domain ID). The ESSID is used to identify cells which are part of the same virtual network.

As opposed to the AP Address or NWID which define a single cell, the ESSID defines a group of cells connected via repeaters or infrastructure, where the user may roam transparently. With some cards, you may disable the ESSID checking (ESSID promiscuous) with off or any (and on to reenale it).

If the ESSID of your network is one of the special keywords (off, on or any), you should use -- to escape it.

Examples:

```
iwconfig eth0 ssid any
iwconfig eth0 ssid "My Network"
iwconfig eth0 ssid -- "ANY"
```

nwid/domain

Set the Network ID (in some products it may also be called Domain ID). As all adjacent wireless networks share the same medium, this parameter is used to differentiate them (create logical collocated networks) and identify nodes belonging to the same cell.

This parameter is only used for pre-802.11 hardware, the 802.11 protocol uses the ESSID and AP Address for this function.

With some cards, you may disable the Network ID checking (NWID promiscuous) with off (and on to enable it again).

Examples:

```
iwconfig eth0 nwid AB34
iwconfig eth0 nwid off
```

nick[name]

Set the nickname, or the station name. Some 802.11 products do define it, but this is not used as far as the protocols (MAC, IP, TCP) are concerned and completely useless as far as configuration goes. Only some wireless diagnostic tools may use it.

Example:

```
iwconfig eth0 nickname "My Linux Node"
```

mode

Set the operating mode of the device, which depends on the network topology.

The mode can be:

- o Ad-Hoc (network composed of only one cell and without Access Point)
- o Managed (node connects to a network composed of many Access Points, with roaming)
- o Master (the node is the synchronization master or acts as an Access Point)
- o Repeater (the node forwards packets between other wireless nodes)
- o Secondary (the node acts as a backup master/repeater)



- Monitor (the node is not associated with any cell and passively monitor all packets on the frequency)
- Auto.

Examples:

```
iwconfig eth0 mode Managed
iwconfig eth0 mode Ad-Hoc
```

freq/channel

Set the operating frequency or channel in the device. A value below 1000 indicates a channel number, a value greater than 1000 is a frequency in Hz. You may append the suffix k, M or G to the value (for example, "2.46G" for 2.46 GHz frequency), or add enough '0'.

Channels are usually numbered starting at 1, and you may use iwlist to get the total number of channels, list the available frequencies, and display the current frequency as a channel. Depending on regulations, some frequencies/channels may not be available.

When using Managed mode, most often the Access Point dictates the channel and the driver may refuse the setting of the frequency. In Ad-Hoc mode, the frequency setting may only be used at initial cell creation, and may be ignored when joining an existing cell.

You may also use off or auto to let the card pick up the best channel (when supported).

Examples:

```
iwconfig eth0 freq 2422000000
iwconfig eth0 freq 2.422G
iwconfig eth0 channel 3
iwconfig eth0 channel auto
```

ap

Force the card to register to the Access Point given by the address, if it is possible. This address is the cell identity of the Access Point, as reported by wireless scanning, which may be different from its network MAC address. If the wireless link is point to point, set the address of the other end of the link. If the link is ad-hoc, set the cell identity of the ad-hoc network.

When the quality of the connection goes too low, the driver may revert back to automatic mode (the card selects the best Access Point in range).

You may also use off to re-enable automatic mode without changing the current Access Point, or you may use any or auto to force the card to associate again with the currently best Access Point.

Examples:

```
iwconfig eth0 ap 00:60:1D:01:23:45
iwconfig eth0 ap any
iwconfig eth0 ap off
```

rate/bit[rate]

For cards supporting multiple bit rates, set the bit-rate in b/s. The bit-rate is the speed at which bits are transmitted over the medium, the user speed of the link is lower due to medium sharing and various overhead.



You may append the suffix k, M or G to the value (decimal multiplier: 10³, 10⁶ and 10⁹ b/s), or add enough '0'. Values below 1000 are card specific, usually an index in the bit-rate list. Use auto to select automatic bit-rate mode (fallback to lower rate on noisy channels), which is the default for most cards, and fixed to revert back to fixed setting. If you specify a bit-rate value and append auto, the driver will use all bit-rates lower and equal than this value.

Examples:

```
iwconfig eth0 rate 11M
iwconfig eth0 rate auto
iwconfig eth0 rate 5.5M auto
```

txpower

For cards supporting multiple transmit powers, sets the transmit power in dBm. If W is the power in Watt, the power in dBm is $P = 30 + 10 \cdot \log(W)$. If the value is postfixed by mW, it will be automatically converted to dBm.

In addition, on and off enable and disable the radio, and auto and fixed enable and disable power control (if those features are available).

Examples:

```
iwconfig eth0 txpower 15
iwconfig eth0 txpower 30mW
iwconfig eth0 txpower auto
iwconfig eth0 txpower off
```

sens

Set the sensitivity threshold. This define how sensitive is the card to poor operating conditions (low signal, interference).

Positive values are assumed to be the raw value used by the hardware or a percentage, negative values are assumed to be dBm.

Depending on the hardware implementation, this parameter may control various functions.

On modern cards, this parameter usually control handover/roaming threshold, the lowest signal level for which the hardware remains associated with the current Access Point. When the signal level goes below this threshold the card starts looking for a new/better Access Point. Some cards may use the number of missed beacons to trigger this. For high density of Access Points, a higher threshold make sure the card is always associated with the best AP, for low density of APs, a lower threshold minimize the number of failed handoffs.

On more ancient card this parameter usually controls the defer threshold, the lowest signal level for which the hardware considers the channel busy. Signal levels above this threshold make the hardware inhibits its own transmission whereas signals weaker than this are ignored and the hardware is free to transmit. This is usually strongly linked to the receive threshold, the lowest signal level for which the hardware attempts packet reception. Proper setting of these thresholds prevents the card to waste time on background noise while still receiving weak transmissions. A modern design seems to control those thresholds automatically.

Examples:

```
iwconfig eth0 sens -80
```



```
iwconfig eth0 sens 2
```

retry

Most cards have MAC retransmissions, and some allow setting the behaviour of the retry mechanism.

To set the maximum number of retries, enter limit `value'. This is an absolute value (without unit). To set the maximum length of time the MAC should retry, enter lifetime `value'. By defaults, this value in seconds, append the suffix m or u to specify values in milliseconds or microseconds.

You can also add the min and max modifiers. If the card supports automatic mode, they define the bounds of the limit or life-time. Some other cards define different values depending on packet size, for example in 802.11 min limit is the short retry limit (non RTS/CTS packets).

Examples:

```
iwconfig eth0 retry 16  
iwconfig eth0 retry lifetime 300m  
iwconfig eth0 retry min limit 8
```

rts[_threshold]

RTS/CTS adds a handshake before each packet transmission to make sure that the channel is clear. This adds overhead, but increases performance in case of hidden nodes or a large number of active nodes. This parameter sets the size of the smallest packet for which the node sends RTS; a value equal to the maximum packet size disables the mechanism. You may also set this parameter to auto, fixed or off.

Examples:

```
iwconfig eth0 rts 250  
iwconfig eth0 rts off
```

frag[mentation_threshold]

Fragmentation allows splitting an IP packet in a burst of smaller fragments transmitted on the medium. In most cases this adds overhead, but in a very noisy environment this reduces the error penalty and allows packets to get through interference bursts.

This parameter sets the maximum fragment size which is always lower than the maximum packet size.

This parameter may also control Frame Bursting available on some cards, the ability to send multiple IP packets together. This mechanism would be enabled if the fragment size is larger than the maximum packet size.

You may also set this parameter to auto, fixed or off.

Examples:

```
iwconfig eth0 frag 512  
iwconfig eth0 frag off
```

key/enc[ryption]

Used to manipulate encryption or scrambling keys and security mode.



To set the current encryption key, just enter the key in hex digits as XXXX-XXXX-XXXX-XXXX or XXXXXXXX. To set a key other than the current key, prepend or append [index] to the key itself (this won't change which is the active key). You can also enter the key as an ASCII string by using the s: prefix. Passphrase is currently not supported.

To change which key is the currently active key just enter [index] (without entering any key value).

OFF and ON disables and enables again encryption.

The security mode may be open or restricted, and its meaning depends on the card used. With most cards, in open mode no authentication is used and the card may also accept non-encrypted sessions, whereas in restricted mode only encrypted sessions are accepted and the card will use authentication if available.

If you need to set multiple keys, or set a key and change the active key, you need to use multiple key directives. Arguments can be put in any order; the last one will take precedence.

Examples:

```
iwconfig eth0 key 0123-4567-89
iwconfig eth0 key [3] 0123-4567-89
iwconfig eth0 key s:password [2]
iwconfig eth0 key [2]
iwconfig eth0 key open
iwconfig eth0 key off
iwconfig eth0 key restricted [3] 0123456789
iwconfig eth0 key 01-23 key 45-67 [4] key [4]
```

power

Used to manipulate power management scheme parameters and mode.

To set the period between wake ups, enter period 'value'. To set the timeout before going back to sleep enter timeout 'value'. You can also add the min and max modifiers. By default, those values are in seconds, append the suffix m or u to specify values in milliseconds or microseconds. Sometimes, those values are without units (number of beacon periods, dwell or similar).

OFF and ON disables and enables again power management. Finally, you may set the power management mode to all (receive all packets), unicast (receive unicast packets only, discard multicast and broadcast) and multicast (receive multicast and broadcast only, discard unicast packets).

Examples:

```
iwconfig eth0 power period 2
iwconfig eth0 power 500m unicast
iwconfig eth0 power timeout 300u all
iwconfig eth0 power off
iwconfig eth0 power min period 2 power max period 4
```



commit

Some cards may not apply changes done through Wireless Extensions immediately (they may wait to aggregate the changes or apply it only when the card is brought up via ifconfig). This command (when available) forces the card to apply all pending changes. This is normally not needed, because the card will eventually apply the changes, but can be useful for debugging.

Display

For each device which supports wireless extensions, iwconfig will display the name of the MAC protocol used (name of device for proprietary protocols), the ESSID (Network Name), the NWID, the frequency (or channel), the sensitivity, the mode of operation, the Access Point address, the bit-rate, the RTS threshold, the fragmentation threshold, the encryption key and the power management settings (depending on availability).

The parameters displayed have the same meaning and values as the parameters you can set, please refer to the previous part for a detailed explanation of them.

Some parameters are only displayed in short/abbreviated form (such as encryption). You may use iwlist to get all the details.

Some parameters have two modes (such as bitrate). If the value is prefixed by '=', it means that the parameter is fixed and forced to that value, if it is prefixed by ':', the parameter is in automatic mode and the current value is shown (and may change).

Access Point/Cell

An address equal to 00:00:00:00:00:00 means that the card failed to associate with an Access Point (most likely a configuration issue). The Access Point parameter will be shown as Cell in ad-hoc mode (for obvious reasons), but otherwise works the same.

If /proc/net/wireless exists, iwconfig will also display its content. Note that those values will depend on the driver and the hardware specifics, so you need to refer to your driver documentation for proper interpretation of those values.

Link quality

Overall quality of the link. May be based on the level of contention or interference, the bit or frame error rate, how good the received signal is, some timing synchronisation, or other hardware metric. This is an aggregate value, and depends totally on the driver and hardware.

Signal level

Received signal strength (RSSI - strength of received signal). May be arbitrary units or dBm, iwconfig uses driver meta information to interpret the raw value given by /proc/net/wireless and display the proper unit or maximum value (using 8 bit arithmetic). In Ad-Hoc mode, this may be undefined and you should use iwspy.



6.2 iwlist

iwlist is used to display some additional information from a wireless network interface that is not displayed by iwconfig. The main argument is used to select a category of information, iwlist displays in detailed form all information related to this category, including information already shown by iwconfig.

Synopsis

iwlist interface scanning
iwlist interface frequency
iwlist interface rate
iwlist interface key
iwlist interface power
iwlist interface txpower
iwlist interface retry
iwlist interface event
iwlist --help
iwlist --version

Parameters

scan[ning]

Give the list of Access Points and Ad-Hoc cells in range, and optionally a whole bunch of information about them (ESSID, Quality, Frequency, Mode...). The type of information returned depends on what the card supports.

Triggering scanning is a privileged operation (root only) and normal users can only read left-over scan results. By default, the way scanning is done (the scope of the scan) will be impacted by the current setting of the driver. Also, this command is supposed to take extra arguments to control the scanning behaviour, but this is currently not implemented.

freq[ueency]/channel

Give the list of available frequencies in the device and the number of defined channels. Please note that usually the driver returns the total number of channels and only the frequencies available in the present locale, so there is no one-to-one mapping between frequencies displayed and channel numbers.

rate/bit[rate]

List the bit-rates supported by the device.

key/enc[ryption]

List the encryption key sizes supported and display all the encryption keys available in the device.

power

List the various Power Management attributes and modes of the device.



txpower

List the various Transmit Powers available on the device.

retry

List the transmit retry limits and retry lifetime on the device.

ap/accesspoint/peers

Give the list of Access Points in range, and optionally the quality of link to them. This feature is obsolete and now deprecated in favour of scanning support (above), and most drivers don't support it.

Some drivers may use this command to return a specific list of Peers or Access Points, such as the list of Peers associated/registered with the card. See your driver documentation for details.

event

List the wireless events supported by the device.

--version

Display the version of the tools, as well as the recommended and current Wireless Extensions version for the tool and the various wireless interfaces.



6.3 ifrename

ifrename is a tool allowing you to assign a consistent name to each of your network interface.

By default, interface names are dynamic, and each network interface is assigned the first available name (eth0, eth1...). The order network interfaces are created may vary. For built-in interfaces, the kernel boot time enumeration may vary. For removable interface, the user may plug them in any order.

Ifrename allow the user to decide what name a network interface will have. Ifrename can use a variety of selectors to specify how interface names match the network interfaces on the system, the most common selector is the interface MAC address.

Ifrename must be run before interfaces are brought up, which is why it's mostly useful in various scripts (init, hotplug) but is seldom used directly by the user. By default, ifrename renames all present system interfaces using mappings defined in /etc/iftab.

Synopsis

```
ifrename [-c configfile] [-p] [-d] [-u] [-v] [-V] [-D]  
ifrename [-c configfile] [-i interface] [-n newname]
```

Parameters

-c configfile

Set the configuration file to be used (by default /etc/iftab). The configuration file defines the mapping between selectors and interface names, and is described in iftab. If configfile is "-", the configuration is read from stdin.

-p

Probe (load) kernel modules before renaming interfaces. By default ifrename only check interfaces already loaded, and doesn't auto-load the required kernel modules. This option enables smooth integration with system not loading modules before calling ifrename.

-d

Enable various Debian specific hacks. Combined with -p, only modules for interfaces specified in /etc/network/interface are loaded.

-i interface

Only rename the specified interface as opposed to all interfaces on the system. The new interface name is printed.

-n newname



When used with `-i`, specify the new name of the interface. The list of mappings from the configuration file is bypassed, the interface specified with `-i` is renamed directly to `newname`. The new name may be a wildcard containing a single `*`.

When used without `-i`, rename interfaces by using only mappings that would rename them to `newname`. The new name may not be a wildcard. This use of `ifrename` is discouraged, because inefficient (`-n` without `-i`). All the interfaces of the system need to be processed at each invocation, therefore in most case it is not faster than just letting `ifrename` renaming all of them (without both `-n` and `-i`).

-t

Enable name takeover support. This allow interface name swapping between two or more interfaces.

Takeover enables an interface to 'steal' the name of another interface. This works only with kernel 2.6.X and if the other interface is down. Consequently, this is not compatible with Hotplug. The other interface is assigned a random name, but may be renamed later with 'ifrename'.

The number of takeovers is limited to avoid circular loops, and therefore some complex multi-way name swapping situations may not be fully processed.

In any case, name swapping and the use of this feature is discouraged, and you are invited to choose unique and unambiguous names for your interfaces...

-u

Enable udev output mode. This enables proper integration of `ifrename` in the udev framework, `udev` will use `ifrename` to assign interface names present in `/etc/iftab`. In this mode the output of `ifrename` can be parsed directly by `udev` as an `IMPORT` action. This requires `udev` version 107 or later.

-D

Dry-run mode. `ifrename` won't change any interface, it will only print new interface name, if applicable, and return.

In dry-run mode, interface name wildcards are not resolved. New interface name is printed, even if it is the same as the old name.

Be also aware that some selectors can only be read by root, for example those based on `ethtool`), and will fail silently if run by a normal user. In other words, dry-run mode under a standard user may not give the expected result.

-v

Verbose mode. `ifrename` will display internal results of parsing its configuration file and querying the interfaces selectors.

Combined with the dry-run option, this is a good way to debug complex configurations or trivial problems.



6.4 iwevent

iwevent displays Wireless Events received through the RTNetlink socket. Each line displays the specific Wireless Event which describes what has happened on the specified wireless interface. This command doesn't take any arguments.

Synopsis

iwevent

Display

There are two classes of Wireless Events.

The first class is events related to a change of wireless settings on the interface (typically done through iwconfig or a script calling iwconfig). Only settings that could result in a disruption of connectivity are reported. The events currently reported are changing one of the following settings:

- Network ID
- SSID
- Frequency
- Mode
- Encryption

All those events will be generated on all wireless interfaces by the kernel wireless subsystem (but only if the driver has been converted to the new driver API).

The second class of events are events generated by the hardware, when something happens or a task has been finished. Those events include:

New Access Point/Cell address

The interface has joined a new Access Point or Ad-Hoc Cell, or lost its association with it. This is the same address that is reported by iwconfig.

Scan request completed

A scanning request has been completed, results of the scan are available (see iwlist).

Tx packet dropped

A packet directed at this address has been dropped because the interface believes this node doesn't answer anymore (usually maximum of MAC level retry exceeded). This is usually an early indication that the node may have left the cell or gone out of range, but it may be due to fading or excessive contention.

Custom driver event



Event specific to the driver. Please check the driver documentation.

Registered node

The interface has successfully registered a new wireless client/peer. Will be generated mostly when the interface acts as an Access Point (mode Master).

Expired node

The registration of the client/peer on this interface has expired. Will be generated mostly when the interface acts as an Access Point (mode Master).

Spy threshold crossed

The signal strength for one of the addresses in the spy list went under the low threshold or went above the high threshold.

Most wireless drivers generate only a subset of those events, not all of them, the exact list depends on the specific hardware/driver combination. Please refer to driver documentation for details on when they are generated, and use iwlist(8) to check what the driver supports.



6.5 iwgetid

`iwgetid` is used to find out the NWID, ESSID or AP/Cell Address of the wireless network that is currently used. The information reported is the same as the one shown by `iwconfig`, but `iwgetid` is easier to integrate in various scripts.

By default, `iwgetid` will print the ESSID of the device, and if the device doesn't have any ESSID it will print its NWID.

The default formatting output is pretty-print.

Synopsis

```
iwgetid [interface] [--raw] [--scheme] [--ap] [--freq]
        [--mode] [--protocol] [--channel]
```

Parameters

--raw

This option disables pretty-printing of the information. This option is orthogonal to the other options (except `--scheme`), so with the appropriate combination of options you can print the raw ESSID, AP Address or Mode.

This format is ideal when storing the result of `iwgetid` as a variable in Shell or Perl scripts or to pass the result as an argument on the command line of `iwconfig`.

--scheme

This option is similar to the previous one, it disables pretty-printing of the information and removes all characters that are not alphanumeric (like space, punctuation and control characters).

The resulting output is a valid Pcmcia scheme identifier (that may be used as an argument of the command `cardctl scheme`). This format is also ideal when using the result of `iwgetid` as a selector in Shell or Perl scripts, or as a file name.

--ap

Display the MAC address of the Wireless Access Point or the Cell.

--freq

Display the current frequency or channel used by the interface.

--channel

Display the current channel used by the interface. The channel is determined using the current frequency and the frequency list provided by the interface.

--mode

Display the current mode of the interface.

--protocol



Display the protocol name of the interface. This allows identifying all the cards that are compatible with each other and accept the same type of configuration.
This can also be used to check Wireless Extension support on the interface, as this is the only attribute that all drivers supporting Wireless Extension are mandated to support.



7 WPA/WPA2 Security

wpa_supplicant is a user space program used to set and manage wireless connections secured through WPA/WPA2 both Personal and Enterprise. All the parameters used to secure the wifi connection can be set into /etc/wpa_supplicant.conf configuration file.

wpa_supplicant can be simply run through a command shell.

wpa_supplicant implements a control interface that can be used by external programs such as wpa_cli to control the operations of the wpa_supplicant itself and to get status information and event notifications.

For further information about wpa_supplicant supported WPA/IEEE 802.11i features and Enterprise modes please have a look to paragraph 8.2.

Please have a look to paragraph 1.4.2 for information about wpa_supplicant/wpa_cli License.

7.1 Configuring wpa_supplicant

wpa_supplicant is configured using a text file that lists all accepted networks and security policies, including pre-shared keys. See example configuration file, [wpa_supplicant.conf](#), for detailed information about the configuration format and supported fields. wpa_supplicant.conf configuration file should be saved into /etc/ folder.

wpa_supplicant configuration can also be changed by the means of wpa_cli as shown in 7.3.

Simple example configurations files:

- For WPA Personal:

```
ctrl_interface=/var/run/wpa_supplicant
```

```
network={
    ssid="example wpa network"
    key_mgmt=WPA-PSK
    proto=WPA
    pairwise=TKIP
    group=TKIP
    psk="secret passphrase"
}
```

- For WPA2 Personal:



```
ctrl_interface=/var/run/wpa_supplicant
```

```
network={
    ssid="example wpa2 network"
    key_mgmt=WPA-PSK
    proto=WPA2
    pairwise=CCMP
    group=CCMP
    psk="secret passphrase"
}
```

- For WPA+WPA2 Personal:

```
ctrl_interface=/var/run/wpa_supplicant
```

```
network={
    ssid="example wpa_wpa2 network"
    key_mgmt=WPA-PSK
    proto=WPA WPA2
    pairwise=TKIP CCMP
    group=TKIP CCMP
    psk="secret passphrase"
}
```

- For WPA/WPA2 Enterprise using EAP-TLS:

```
ctrl_interface=/var/run/wpa_supplicant
```

```
network={
    ssid="example wpa2-eap network"
    key_mgmt=WPA-EAP
    proto=WPA WPA2
    pairwise=TKIP CCMP
    group=TKIP CCMP
    eap=TLS
    ca_cert="/etc/cert/ca.pem"
    private_key="/etc/cert/user.p12"
    private_key_passwd="PKCS#12 passphrase"
}
```



7.2 Running wpa_supplicant

wpa_supplicant can be run simply typing:

```
# wpa_supplicant -Dunifi -ieth0 -c/etc/wpa_supplicant.conf -d
```

7.3 wpa_cli

wpa_cli is a tool that can be used to control the operations of the wpa_supplicant and to get status information and event notifications.

Synopsis:

```
wpa_cli [-p<path to ctrl sockets>] [-i<iface>] [-hvB] [-a<action file>]
        [-P<pid file>] [-g<global ctrl>] [command..]
```

- h = help (show this usage text)
- v = shown version information
- a = run in daemon mode executing the action file based on events from wpa_supplicant
- B = run a daemon in the background

Parameters:

- status [verbose]** = get current WPA/EAPOL/EAP status
- mib** = get MIB variables (dot1x, dot11)
- help** = show this usage help
- interface [iface]** = show interfaces/select interface
- level <debug level>** = change debug level
- license** = show full wpa_cli license
- logoff** = IEEE 802.1X EAPOL state machine logoff
- logon** = IEEE 802.1X EAPOL state machine logon
- set** = set variables (shows list of variables when run without arguments)
- pmksa** = show PMKSA cache
- reassociate** = force reassociation
- reconfigure** = force wpa_supplicant to re-read its configuration file
- preauthenticate <BSSID>** = force preauthentication
- identity <network id> <identity>** = configure identity for an SSID
- password <network id> <password>** = configure password for an SSID
- new_password <network id> <password>** = change password for an SSID
- pin <network id> <pin>** = configure pin for an SSID
- otp <network id> <password>** = configure one-time-password for an SSID
- passphrase <network id> <passphrase>** = configure private key passphrase for an SSID



bssid <network id> <BSSID> = set preferred BSSID for an SSID
list_networks = list configured networks
select_network <network id> = select a network (disable others)
enable_network <network id> = enable a network
disable_network <network id> = disable a network
add_network = add a network
remove_network <network id> = remove a network
set_network <network id> <variable> <value> = set network variables (shows list of variables when run without arguments)
get_network <network id> <variable> = get network variables
save_config = save the current configuration
disconnect = disconnect and wait for reassociate/reconnect command before connecting
reconnect = like reassociate, but only takes effect if already disconnected
scan = request new BSS scan
scan_results = get latest scan results
get_capability <eap/pairwise/group/key_mgmt/proto/auth_alg> = get capabilities
ap_scan <value> = set ap_scan parameter
stkstart <addr> = request STK negotiation with <addr>
terminate = terminate wpa_supplicant



8 Appendix

8.1 WPA/WPA2 (IEEE 802.11i)

Wi-Fi Protected Access (WPA and WPA2) is a class of systems to secure Wi-Fi networks. It was created in response to several serious weaknesses researchers had found in the previous system, Wired Equivalent Privacy (WEP). WPA implements the majority of the IEEE 802.11i standard, and was intended as an intermediate measure to take the place of WEP while 802.11i was prepared. WPA2 implements the mandatory elements of 802.11i. In particular, it introduces a new AES-based algorithm, CCMP, that is considered fully secure.

8.1.1 Personal Mode (PSK)

Personal mode (also known as Pre-shared key mode, PSK) is designed for home and small office networks that don't require the complexity of an IEEE 802.1X authentication server. Each user must enter a passphrase to access the network. The passphrase may be from 8 to 63 printable ASCII characters or 64 hexadecimal digits (256 bits). If you choose to use the ASCII characters, a hash function reduces it from 504 bits (63 characters * 8 bits/character) to 256 bits (using also the SSID). The passphrase may be stored on the user's computer at their discretion under most operating systems to avoid re-entry. The passphrase must remain stored in the Wi-Fi access point.

8.1.2 Enterprise Mode

Enterprise networks may use WPA/WPA2 along with 802.1X, an IEEE standard for port-based Network Access Control, to make WiFi security stronger. IEEE 802.1X provides authentication to devices wanting to join a wireless network, establishing a point-to-point connection or preventing access if authentication fails. Enterprise wireless LANs usually use RADIUS authentication servers to perform IEEE 802.1X authentication using EAP (Extensible Authentication Protocol) authentication frameworks.

8.2 WPA Supplicant

wpa_supplicant is the IEEE 802.1X/WPA component that is used in the client stations and is designed to be a "daemon" program that runs in the background and acts as the backend component controlling the wireless connection. It implements key negotiation with a WPA Authenticator and it controls the roaming and IEEE 802.11.



Following steps are used when associating with an AP using WPA:

- wpa_supplicant requests the kernel driver to scan neighbouring BSSes
- wpa_supplicant selects a BSS based on its configuration
- wpa_supplicant requests the kernel driver to associate with the chosen BSS
- if WPA-EAP: integrated IEEE 802.1X Supplicant completes EAP authentication with the authentication server (proxy by the Authenticator in the AP)
- If WPA-EAP: master key is received from the IEEE 802.1X Supplicant
- If WPA-PSK: wpa_supplicant uses PSK as the master session key
- wpa_supplicant completes WPA 4-Way Handshake and Group Key Handshake with the Authenticator (AP). WPA2 has integrated the initial Group Key Handshake into the 4-Way Handshake.
- wpa_supplicant configures encryption keys for unicast and broadcast
- normal data packets can be transmitted and received

8.2.1 Supported features

Both WPA-Personal and WPA-Enterprise are supported.

▪ Supported WPA/IEEE 802.11i features

- WPA-PSK ("WPA-Personal")
- WPA with EAP (e.g., with RADIUS authentication server) ("WPA-Enterprise")
- key management for CCMP, TKIP, WEP104, WEP40
- WPA and full IEEE 802.11i/RSN/WPA2
- RSN: PMKSA caching, pre-authentication

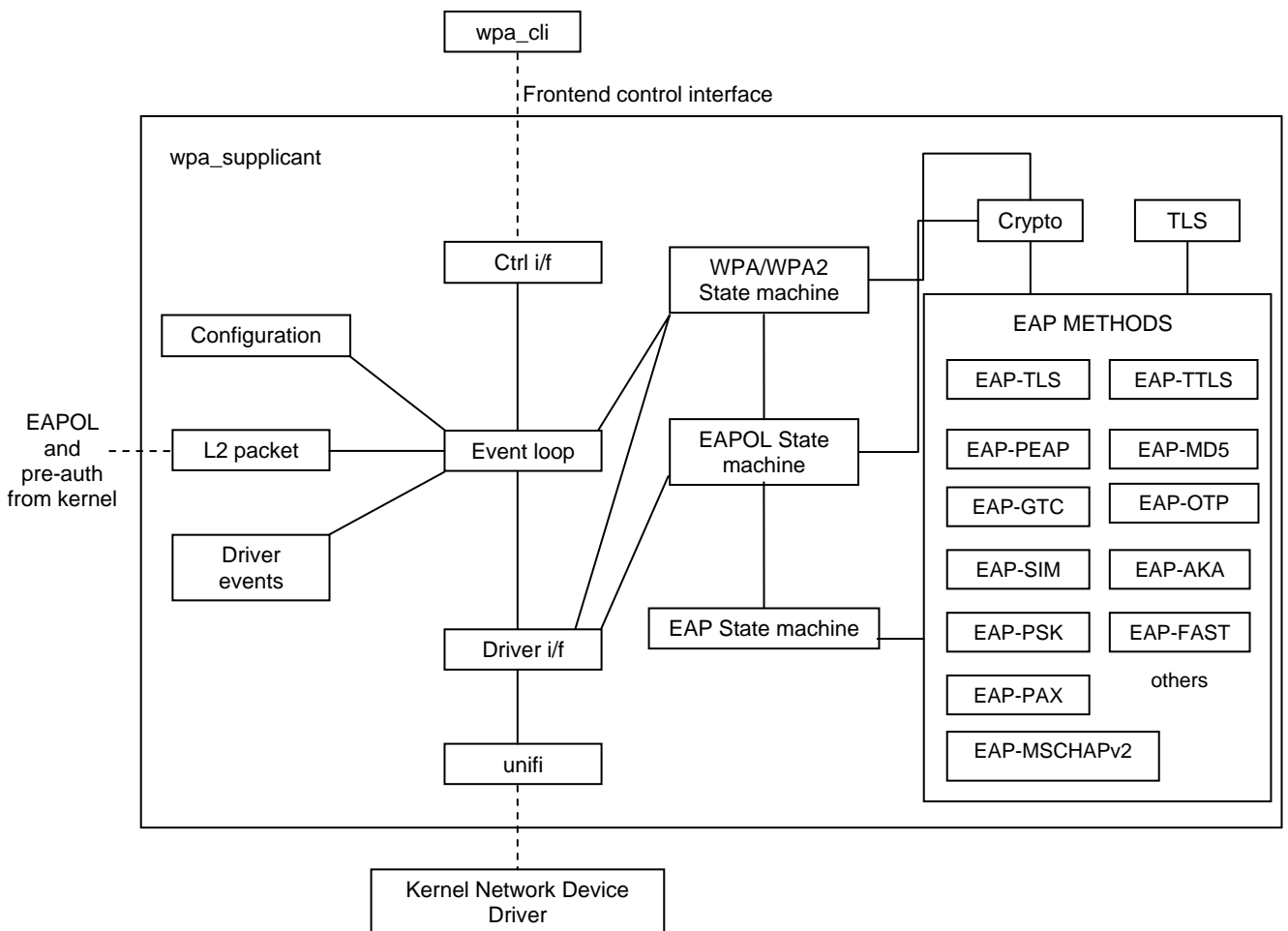
▪ Supported EAP methods (IEEE 802.1X Supplicant)

- EAP-TLS
- EAP-PEAP/MSCHAPv2 (both PEAPv0 and PEAPv1)
- EAP-PEAP/TLS (both PEAPv0 and PEAPv1)
- EAP-PEAP/GTC (both PEAPv0 and PEAPv1)
- EAP-PEAP/OTP (both PEAPv0 and PEAPv1)
- EAP-PEAP/MD5-Challenge (both PEAPv0 and PEAPv1)
- EAP-TTLS/EAP-MD5-Challenge
- EAP-TTLS/EAP-GTC
- EAP-TTLS/EAP-OTP
- EAP-TTLS/EAP-MSCHAPv2
- EAP-TTLS/EAP-TLS
- EAP-TTLS/MSCHAPv2
- EAP-TTLS/MSCHAP
- EAP-TTLS/PAP
- EAP-TTLS/CHAP
- EAP-SIM



- EAP-AKA
- EAP-PSK
- EAP-FAST
- EAP-PAX
- EAP-SAKE
- EAP-IKEv2
- EAP-GPSK (experimental)

8.2.2 Source code architecture



9 Acronyms and Abbreviations

AES	Advanced Encryption Standard
AP	Access Point
CCMP	Counter Mode with Cipher Block Chaining Message Authentication Code Protocol
EAP	Extensible Authentication Protocol
LAN	Local Area Network
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistant
PSK	Pre-Shared Key
RADIUS	Remote Authentication Dial In User Service
RISC	Reduced Instruction Set Computer
SDIO	Secure Digital Input Output
VOIP	Voice Over IP
WEP	Wired Equivalent Privacy
WPA/WPA2	Wi-Fi Protected Access

