Technical Information Manual

WLAN Client Utility

Broadcom WLAN Client Utility Command Set

80211-TI305-R

5300 California Avenue • Irvine, CA 92617 • Phone: 949-926-5000 • Fax: 949-926-5203

October 16, 2014

Revision History

Revision	***Date***	***Change Description***
80211-TI305-R	10/16/14	**Updated:** • General changes based on software updates.
80211-TI304-R	08/30/12	**Updated:** • General changes based on software updates.
80211-TI303-R	12/17/10	**Updated:** • General changes based on software updates.
80211-TI302-R	2/11/09	**Updated:** • General changes based on software updates.
80211-TI301-R	11/20/08	**Updated:** • General changes based on software updates.
80211-TI300-R	7/13/07	Initial release

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Table of Contents

About This Document

Purpose and Audience

This document is intended for use by Broadcom® and customer engineers using the Broadcom WL tool to evaluate and test BCM43XX combo and embedded Wi-Fi chip solutions.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

Document Conventions

The following conventions may be used in this document:

Convention	***Description***
`Monospace`	Command syntax: `wl [-l] <command>`
< >	Placeholders for *required* elements: `wl <command>`
[ ]	Indicates *optional* command-line parameters: `wl [-l]`
|	Separates two mutually exclusive choices
…	Indicates that the user can type multiple arguments of the same type.

Technical Support

Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal (https://support.broadcom.com). For a CSP account, contact your Sales or Engineering support representative.

In addition, Broadcom provides other product support through its Downloads and Support site (http://www.broadcom.com/support/).

Introduction

This document describes the various commands that are available in the Broadcom WLAN Client Utility. To use the utility to start a build, the following tools must be available/installed on the build machine:

- Broadcom WLAN driver (version 5.20.18 or later).
- The latest Broadcom BCM43xx WLAN adapter.
- Microsoft® Windows® or Linux® operating system.

Command Structure

Syntax

```
wl [-a|-i <adapter>] [-h] [-d|-u|-x] <command> [arguments]
```

- `h` This message and command descriptions
- `h [cmd]` Command description for `cmd`
- `a, i` Adapter name or number
- `d` Output format signed integer
- `u` Output format unsigned integer
- `x` Output format hexadecimal

Options

To view the list of WLAN Client Utility options, run the `wl` command.

Command Descriptions

This section provides a description of the commands and supported options available in the Broadcom WLAN Client Utility. Commands are grouped as follows:

- "Version Information and List of Commands" on page 21.
- "Association Control" on page 22.
- "Initializing/Restarting" on page 24.
- "Radio Control" on page 25.
- "Debugging/Status/Configuration" on page 26.
- "Transmission Retry Control" on page 43.
- "Rate Parameters" on page 43.
- "Antenna Controls" on page 51.
- "Security and Encryption Controls" on page 52.
- "Scan Controls" on page 58.
- "Association and Status" on page 65.
- "Channel and Band Control Parameters" on page 67.
- "Roam Controls" on page 70.
- "Regulatory Test and Measurements" on page 73.
- "Country and Locale" on page 86.
- "Wireless Distribution System" on page 87.
- "Mode Controls" on page 88.
- "gmode Protection Controls" on page 91.
- "Radar Controls" on page 93.
- "WME Controls" on page 94.
- "Information Element Controls" on page 98.
- "NVRAM/SROM Write Controls" on page 99.
- "NDIS Related Commands" on page 100.
- "MIMO-Specific Commands" on page 101.
- "Packet Filter Related Commands" on page 112.
- "Statistics Related Commands" on page 114.
- "CIS Related Commands" on page 122.
- "Power Related Commands" on page 123.
- "Manufacturing Test Commands" on page 127.
- "External Log Commands" on page 131.
- "Address Resolution Protocol Commands" on page 131.
- "Duration Information Commands" on page 132.
- "TPC Commands" on page 133.
- "Link Quality Commands" on page 134.
- "Offload Commands" on page 134.

Version Information and List of Commands

ver

Gets the version information for the Broadcom WLAN Client Utility.

```
wl ver
```

Example return

```
   7.14 RC43.22
   wl0: Sep 29 2014 16:29:30 version 7.14.43.22
```

cmds

Generates a short list of available commands.

```
wl cmds
```

Association Control

join

Joins a specified network.

```
wl join <name|ssid> [key xxxxx] [imode bss|ibss] [amode
open|shared|auto|wpa|wpapsk|wpanone|wpa2|wpa2psk]
```

Example return

join Broadcom imode infra amode open

If the AP is not configured with Wired Equivalent Privacy (WEP) security, no WEP key is required. Otherwise, specify `wep xxxx` or `wepkey xxxx`.

Authentication mode (`amode`) choices:

- Open
- Shared

Infrastructure mode choices:

- bss
- managed
- infra

Ad hoc mode choices:

- ibss
- ad hoc

IEEE Std 802.11 supports open system and shared key network authentication services subtypes. Under open system authentication, any wireless station can request authentication. The station that attempts to authenticate with another wireless station sends an authentication management frame that contains the identity of the sending station. The receiving station then sends back a frame that indicates whether it recognizes the identity of the sending station. Under shared key authentication, each wireless station is assumed to have received a secret shared key over a secure channel that is independent from the 802.11 wireless network communications channel.

up

Reinitializes and marks the adapter as being up (operational). This command makes the interface operational. It does all the necessary initialization tasks to bring up the interface.

```
wl up
```

The following tasks are associated with this command:

- Configuring PCI/PCMCIA here to allow manufacturer hot-swap: down, hot-swap (chip power cycle), up.
- Reading the PHY revision.
- Setting the soft interrupt mask.
- Bringing the interface up in each frequency band.
- Initializing the default rate, channel, and type-dependent information.
- Initializing the basic rate look-up.
- Saving, suspending, disabling interrupts, and turning off the radio.
- Starting a one-second watchdog.
- Starting the activity LED timer.

down

Resets and marks the adapter as being down (disabled). This command disables the interface.

```
wl down
```

The following tasks are associated with this command:

- Disassociate
- Turns the radio off.
- Cancels the watchdog timer.
- Cancels the activity timer.
- Cancels any active scan.
- Cancels any IBSS timer.
- Cancels any association timer.
- Flushes the TX control queue.
- Reclaims the SCBS.
- If an AP, flushes the PS-POLL response (MSDU) packet queues and also flushes PSPOLL.
- Response (MPDU) packet queues.
- Restores to a known good default state.

out

Marks the adapter as being down but does not reset the hardware (disabled). On dual-band cards, the card must be band-locked before use.

```
wl out
```

reassoc

Initiates a reassociation request.

```
wl reassoc <bssid> [options]
```

Options:

- `-c CL --chanspecs` CL chanspecs (comma- or space-separated list)

Initializing/Restarting

reinit

Reinitializes the device.

```
wl reinit
```

reboot

Reboots the platform.

```
wl reboot
```

restart

Restarts the driver. The driver must already be down.

```
wl restart
```

Radio Control

radio

Turns the radio on or off using a software switch. Running `wl radio` returns the current state of the radio. For example, 0x0004 when ON or 0x0005 when OFF.

```
wl radio on | off
```

rrm

Enables or disables the radio resource management (RRM) feature.

```
wl rrm [0/1]
```

- 0: Disables the RRM feature.
- 1: Enables the RRM feature.

rrm_stat_rpt

Reads 11k stat measurement report from the STA.

```
wl rrm_stat_rpt [mac]
```

rrm_nbr_list

Gets 11k neighbor report list.

```
wl rrm_nbr_list
```

rrm_nbr_del_nbr

Deletes a node from the 11k neighbor report list.

```
wl rrm_nbr_del_nbr [bssid]
```

rrm_nbr_add_nbr

Adds a node to the 11k neighbor report list.

```
wl rrm_nbr_add_nbr [bssid] [bssid info] [regulatory] [channel] [phytype]
```

Debugging/Status/Configuration

ap

Sets the AP mode.

```
wl ap <value>
```

Options

- `0` = STA
- `1` = AP

apname

Gets the current associated AP name. If the client is not associated with an AP, a stale AP name may be returned. The maximum AP name length is 15 bytes.

```
wl apname
```

bi

Gets/sets the beacon interval.

```
wl bi
```

Returns the set beacon interval (BI). If the beacon interval is set for 100 ms, 100 is returned.

bssid

Gets the BSSID value. Returns an error if the STA is not associated.

The basic service set identifier is the 48-bit MAC (hardware) address of the WLAN interface in the AP that serves the stations in a basic service set (BSS).

```
wl bssid
```

cap

Returns the driver capabilities.

```
wl cap
```

counters

Returns driver counter values.

```
wl counters
```

Example return

```
txframe 92289 txbyte 7637260 txretrans 830936 txerror 0 rxframe 90957 rxbyte 665
3890 rxerror 17
txprshort 4398 txdmawar 0 txnobuf 0 txnoassoc 0 txchit 116 txcmiss 92173
reset 14926 txserr 0 txphyerr 1 txphycrs 0 txfail 689
d11_txfrag 509883 d11_txmulti 12 d11_txretry 267653 d11_txretrie 266858
d11_txrts 0 d11_txnocts 0 d11_txnoack 829661 d11_txfrmsnt 284515
rxcrc 3824120 rxnobuf 0 rxnondata 0 rxbadds 0 rxbadcm 0 rxdup 1017 rxfragerr 0
rxrunt 14 rxgiant 0 rxnoscb 0 rxbadproto 0 rxbadsrcmac 3
d11_rxfrag 2280968 d11_rxmulti 1591082 d11_rxundec 0
rxctl 2140668 rxbadda 0 rxfilter 0
rxuflo: 0 0 0 0 0 0
txallfrm 1407603 txrtsfrm 0 txctsfrm 17113 txackfrm 217150
txdnlfrm 0 txbcnfrm 0 txtplunfl 0 txphyerr 1
txfunfl: 0 0 0 0 0 0
tkipmicfaill 0 tkipicverr 0 tkipcntrmsr 0
tkipreplay 0 ccmpfmterr 0 ccmpreplay 0
ccmpundec 0 fourwayfail 0 wepundec 0
wepicverr 0 decsuccess 0 rxundec 0

rxfrmtoolong 828457 rxfrmtooshrt 8166 rxinvmachdr 1374201 rxbadfcs 3824120
rxbadplcp 5332054 rxcrsglitch 36254836 rxstrt 16714715 rxdfrmucastmbss 92241
rxmfrmucastmbss 124909 rxcfrmucast 284475 rxrtsucast 0 rxctsucast 0
rxackucast 284475 rxdfrmocast 82398 rxmfrmocast 1280135 rxcfrmocast 450757
rxrtsocast 19235 rxctsocast 137269 rxdfrmmcast 486848 rxmfrmmcast 9259123
rxcfrmmcast 4 rxbeaconmbss 1310371 rxdfrmucastobss 0 rxbeaconobss 7698574
rxrsptmout 827723 bcntxcancl 0 rxf0ovfl 0 rxf1ovfl 0
rxf2ovfl 0 txsfovfl 0 pmqovfl 0
rxcgprqfrm 0 rxcgprsqovfl 0 txcgprsfail 0 txcgprssuc 0
prs_timeout 0 rxnack 0 frmscons 0 txnack 0 txglitch_nack 0
txburst 0 txphyerror 0
txchanrej 237
```

cur_etheraddr

Gets/sets the current hardware address.

```
wl cur_etheraddr
```

customvar1

Prints the value of `customvar1` in hexadecimal format.

```
wl customvar1
```

dfs_status

Gets the DFS status.

```
wl dfs_status
```

Example return

```
state IDLE time elapsed 0ms radar channel cleared by dfs none
```

malloc_dump

Deprecated. Folded under wl dump malloc.

dump

Provides a list of various suboptions.

```
wl dump
```

Example return

```
wl0: June 26 2012 14:24:34 version 6.30.39.31 (r341185)

resets 190
perm_etheraddr 78:e4:00:61:ce:6d cur_etheraddr 78:e4:00:61:ce:6d
board 0x46d, board rev P406
rate_override: A 0, B 0
antrx_ovr 3 txant 3

BSS Config 0: ""
enable 0 up 0 wlif 0x00000000 "Primary"
wsec 0x7 auth 0 wsec_index -1 wep_algo 0

current_bss->BSSID 00:00:00:00:00:00
current_bss->SSID ""
bsscfg 0 assoc_state 0
```

fasttimer

Deprecated. Use fast_timer.

fast_timer

```
wl fast_timer
```

slowtimer

Deprecated. Use slow_timer.

slow_timer

```
wl slow_timer
```

glacialtimer

Deprecated. Use glacial_timer.

glacial_timer

```
wl glacial_timer
```

mfp_assoc

Sends association.

```
wl mfp_assoc
```

mfp_auth

Sends authorization.

```
wl mfp_auth
```

mfp_config

Configures PMF capability.

```
wl mfp_config
```

mfp_deauth

```
wl mfp_deauth
```

mfp_disassoc

```
wl mfp_disassoc
```

mfp_reassoc

```
wl mfp_reassoc
```

mfp_sa_query

```
wl mfp_sa_query
```

mfp_sha256

```
wl mfp_sha256
```

frag

Deprecated. Use fragthresh.

fragthresh

```
wl fragthresh
```

rts

Deprecated. Use rtsthresh.

rtsthresh

```
wl rtsthresh
```

event_msgs

Sets/gets the 128-bit hexadecimal filter bit mask for MAC event reporting (through packet indications). Takes a 128-bit vector, which selectively enables or disables the reporting of MAC events through the packet data path. For example, setting bit locations 0 and 3 enables reporting of WLC_E_SET_SSID and WLC_E_AUTH event messages.

```
wl event_msgs
```

- Default 0

List of bit vector event messages:

`WLC_E_SET_SSID`	`0`	`/* indicates status of set SSID */`
`WLC_E_JOIN`	`1`	`/* differentiates join IBSS from found (WLC_E_START) IBSS */`
`WLC_E_START`	`2`	`/* STA founded an IBSS or AP started a BSS */`
`WLC_E_AUTH`	`3`	`/* 802.11 AUTH request */`
`WLC_E_AUTH_IND`	`4`	`/* 802.11 AUTH indication */`
`WLC_E_DEAUTH`	`5`	`/* 802.11 DEAUTH request */`
`WLC_E_DEAUTH_IND`	`6`	`/* 802.11 DEAUTH indication */`

WLC_E_ASSOC	7	/* 802.11 ASSOC request */
WLC_E_ASSOC_IND	8	/* 802.11 ASSOC indication */
WLC_E_REASSOC	9	/* 802.11 REASSOC request */
WLC_E_REASSOC_IND	10	/* 802.11 REASSOC indication */
WLC_E_DISASSOC	11	/* 802.11 DISASSOC request */
WLC_E_DISASSOC_IND	12	/* 802.11 DISASSOC indication */
WLC_E_QUIET_START	13	/* 802.11h Quiet period started */
WLC_E_QUIET_END	14	/* 802.11h Quiet period ended */
WLC_E_BEACON_RX	15	/* BEACONS received/lost indication */
WLC_E_LINK	16	/* generic link indication */
WLC_E_MIC_ERROR	17	/* TKIP MIC error occurred */
WLC_E_NDIS_LINK	18	/* NDIS style link indication */
WLC_E_ROAM	19	/* roam attempt occurred: indicate status & reason */
WLC_E_TXFAIL	20	/* change in dot11FailedCount (txfail) */
WLC_E_PMKID_CACHE	21	/* WPA2 pmkid cache indication */
WLC_E_RETROGRADE_TSF	22	/* current AP's TSF value went backward */
WLC_E_PRUNE	23	/* AP was pruned from join list for reason */
WLC_E_AUTOAUTH	24	/* report AutoAuth table entry match for join attempt */
WLC_E_EAPOL_MSG	25	/* Event encapsulating an EAPOL message */
WLC_E_SCAN_COMPLETE	26	/* Scan results are ready or scan was aborted */
WLC_E_ADDTS_IND	27	/* indicate to host addts fail/success */
WLC_E_DELTS_IND	28	/* indicate to host delts fail/success */
WLC_E_BCNSENT_IND	29	/* indicate to host of beacon transmit */
WLC_E_BCNRX_MSG	30	/* Send the received beacon up to the host */
WLC_E_BCNLOST_MSG	31	/* indicate to host loss of beacon */
WLC_E_ROAM_PREP	32	/* before attempting to roam */
WLC_E_PFN_NET_FOUND	33	/* PFN network found event */
WLC_E_PFN_NET_LOST	34	/* PFN network lost event */

infra

Sets the infrastructure mode.

```
wl infra [0] [1]
```

Options

- `0` IBSS
- `1` Infrastructure BSS

macreg

Gets/sets any MAC register (include IHR and SB).

```
wl macreg offset size [2,4] [value] [band]
```

monitor

Sets the adapter to monitor mode. When set, the address filter accepts all received frames (when cleared, the address filter accepts only those frames that match the BSSID or local MAC address), accepts all received control frames that are accepted by the address filter, and accepts all beacon and probe response frames without regard to the source address.

```
wl monitor <value>
```

Options

- `0` Disable
- `1` Enable active monitor mode (interface still operates)

msglevel

Sets the driver console debugging message-bit vector.

```
wl msglevel N
```

N can be a list of numbers such as the following (Use a + or – prefix to make an incremental change.):

```
   0x0001 error, err
   0x0002 trace
   0x0004 prhdrs
   0x0008 prpkt
   0x0010 inform, info, inf
   0x0020 tmp
   0x0040 oid
   0x0080 rate
   0x0100 assoc, as
   0x0200 prusr
   0x0400 ps
   0x0800 txpwr, pwr
   0x1000 port
   0x2000 dual
   0x4000 wsec
   0x8000 wsec_dump
   0x10000 log
   0x20000 nrssi
   0x40000 loft
   0x80000 regulatory
   0x100000 taf
   0x200000 radar
   0x400000 mpc
   0x800000 apsta
   0x1000000 dfs
   0x4000000 mbss
   0x8000000 cac
   0x10000000 amsdu
   0x20000000 ampdu
   0x40000000 ffpld
   0x100000000 dpt
   0x200000000 scan
   0x400000000 wowl
   0x800000000 coex
   0x1000000000 rtdc
   0x2000000000 proto
   0x8000000000 chanim
   0x10000000000 wmf
   0x40000000000 itfr
   0x800000000000 psta
   0x200000000000 mcnx
   0x400000000000 prot
   0x4000000000000 tbtt
   0x8000000000000000 time
   0x1000000000000000 lpc
   0x10000000000000 txbf
```

```
    0x1000000000000 tso
    0x10000000000000 mq
    0x8000000000000000 chanlog
    0x400000000000000 wnm
```

msglevel chanim

When set, the following information is dumped on the console every second.

```
bgnoise: -92 dBm
**intf: 0 glitch cnt: 13 badplcp: 0 noise: -92 chanspec: 0x100b
***cca stats: txdur: 3, inbss: 0, obss: 3,nocat: 0, nopkt: 1, doze: 0
```

Where:

- tx/Transmission of packets: All frame transmissions fall under this category.
- inbss/Reception of *In BSS* packets: Only frames that are received from AP/STAs within the BSS of the receiving terminal come under this category.
- obss/Reception of *Other BSS* packets: All frames received from AP/STAs that are outside the BSS of the receiving terminal come under this category.
- nocat/Reception of *No Category* packets: All 802.11 frames that are corrupted at the receiver (bad FCS) come under this category. Some control frames are also included in this category.
- nopkt/Reception of *No Packets*: Received frames with invalid PLCPs and CRS glitches come under this category. All receptions in this category are considered to be noise.
- doze: PM mode for the MAC. If the MAC core is in power management mode, it is in *doze* state.
- txop: Is the percentage of time that *free slot* is available for transmit. TXOP is calculated in percentage, ranging from 0 to 100.
- goodtx/Good transmissions: Only transmissions that are followed by a good expected response from the receiving terminal come under this category.
- badtx/Bad transmissions: Transmissions that receive a bad response (unexpected or bad FCS) or do not receive any response from the receiving terminal come under this category.
- glitch: is the sum of both OFDM and BPHY glitch counts and expressed as glitch_cnt * 1000 / aci_chan_idle_dur.
- badplcp: is the sum of both OFDM and BPHY packets with badp PLCP headers.
- knoise: Addresses improved noise measurement. Knoise is a modified version of the regular noise feature that provides improved dynamic measurement range. It achieves this by performing minimal gain control before running the *iqest* phy engine. It evaluates two fixed gain values and selects the measurement that results in no clipping. If both gain values result in clipping, it returns the measurement with the lowest gain.
- idle/Channel Idle time: This corresponds to the time when there is no transmit or receive activity at the terminal.

nvget

Gets the value of an NVRAM variable.

```
wl nvget <name>
```

nvram_source

Displays the NVRAM source.

```
wl nvram_source
```

nvram_dump

Prints NVRAM variables to STDOUT.

```
wl nvram_dump
```

nvset

Sets an NVRAM variable.

```
wl nvset <name=value>
```

nvram_get

Gets the value of an NVRAM variable (available only in manufacturing test builds).

```
wl nvram_get <name>
```

perm_etheraddr

Gets the permanent address from NVRAM.

```
wl perm_etheraddr
```

phylist

Returns the list of available PHY types.

```
wl phylist
```

Example return

```
g
```

phymsglevel

Sets the PHY debugging message-bit vector.

```
wl phymsglevel N
```

N can be a list of numbers such as the following:

```
0x0001 = error
0x0002 = trace
0x0004 = inform
0x0008 = tmp
0x0010 = txpwr
0x0020 = cal
0x0080 = radar
0x0100 = thermal
```

phyreg

Gets/sets a PHY register.

```
wl phyreg offset [value] [band]
```

phytable

Gets/sets the table element of a table with the given ID at the given offset. The supplied table width should be 8, 16, or 32. The table ID offset cannot be negative.

```
wl phytable <table_id offset> <width_of_table_element> <table_element>
```

phytype

Gets the PHY type.

```
wl phytype
```

Returns

- `0` PHY Type A
- `1` PHY Type B
- `2` PHY Type G
- `4` PHY Type N
- `15` Unknown PHY

lcnphy_papdepstbl

Prints papd eps table.

```
wl lcnphy_papdepstbl
```

phy_force_fdiqi

Enables/disables FDIQI Cal/Comp.

```
wl phy_force_fdiqi
```

Options

- `0` Disable
- `1` Enable

pktcnt

Gets the summary of good and bad packets.

```
wl pktcnt
```

- Receive: Good packet 0, bad packet 0.
- Transmit: Good packet 0, bad packet 0.

plcphdr

Gets/sets the PLCP header.

```
wl plcphdr <option>
```

Options

```
long
auto
debug
```

pmset

Sets driver power management mode.

```
wl pmset <value>
```

Values

- `0` Constantly awake mode (CAM)
- `1` Power-save mode (PS)
- `2` Fast power-save mode

promisc

Sets promiscuous mode Ethernet address reception. When set, the address filter accepts all received frames. When cleared, the address filter accepts only those frames that match the BSSID or local MAC address.

```
wl promisc <value>
```

Values

- `0` Disable
- `1` Enable

radioreg

Gets/sets a radio register.

```
wl radioreg offset [value] [band/core]
```

For the 802.11ac PHY, use the command as follows:

```
wl radioreg [ offset ] [ cr0/cr1/cr2/pll ]
wl radioreg [ offset ] [ value ] [ cr0/cr1/cr2/pll/all ]
```

revinfo

Gets hardware revision information.

```
wl revinfo
```

Example return

```
vendorid 0x14e4
deviceid 0x4329
radiorev 0x42055000
chipnum 0x4321
chiprev 0x1
corerev 0xb
boardid 0x46d
boardvendor 0x14e4
boardrev 0x4b
driverrev 0x4960c00
ucoderev 0x19a00d8
bus 0x1
```

scb_timeout

Inactivity time-out value for authenticated STAs.

```
wl scb_timeout <value>
```

shmem

Gets/sets a shared memory location.

```
wl shmem offset [value] [band]
```

shortslot

Gets the current IEEE 802.11g short slot timing mode.

```
wl shortslot
```

Options

- `0` Long
- `1` Short

shortslot_override

Gets/sets the IEEE 802.11g short slot timing mode override.

```
wl shortslot_override
```

Options

- `-1` Auto
- `0` Long
- `1` Short

shortslot_restrict

Gets/sets the AP restriction on association for the 802.11g short slot.

```
wl shortslot_restrict
```

For timing capable STAs

- `0` Does not restrict association based on short-slot capability.
- `1` Restricts association to STAs with short-slot capability.

staname

Gets/sets the station name. Returns the machine name. If the STA name has not been set by the operating system, the staname command returns a NULL string. The maximum STA name length (set/get) is 15 bytes.

```
wl staname
```

srclear

Clears the first <len> bytes of the SROM; len is in decimal or hexadecimal.

```
wl srclear <len>
```

srdump

Prints the contents of SPROM to STDOUT (dumps 64 16-bit words of the SROM present on-board). For details of the individual locations, check the Broadcom SROM memory map for that specific design. Memory maps are different, depending on the type of the design (for example, Mini PCI, Cardbus, PCMCIA, and so on).

```
wl srdump
```

Example return

```
               0x3001  0x0000  0x046d  0x14e4  0x4329  0x8000  0x0002  0x0000
    srom[008]: 0x1000  0x1800  0x0000  0x0000  0xffff  0xffff  0xffff  0xffff
    srom[016]: 0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[024]: 0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[032]: 0x5372  0x004b  0x0200  0x0000  0x0003  0x0000  0x0090  0x4c99
    srom[040]: 0x0235  0x0000  0x0000  0xffff  0xffff  0xffff  0x0007  0x0202
    srom[048]: 0xff02  0x4a4a  0x5b5b  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[056]: 0xffff  0xff3c  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[064]:  0x0000  0x0000  0x0000  0x0000  0x0000  0xffff  0xffff  0xffff
    srom[072]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[080]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0x0000
    srom[088]:  0x0000  0x0000  0x0000  0x0000  0xffff  0xffff  0xffff  0xffff
    srom[096]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[104]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[112]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[120]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[128]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[136]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[144]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[152]:  0xffff  0xffff  0xffff  0xffff  0x0000  0x0000  0x0000  0xffff
    srom[160]:  0xffff  0xffff  0xffff  0xffff  0xffff  0x0000  0x0000  0x0000
    srom[168]:  0x0000  0x0000  0x0000  0x0000  0x0000  0xffff  0xffff  0xffff
    srom[176]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[184]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[192]:  0xffff  0xffff  0xffff  0xffff  0xffff  0x0000  0x0000  0x0000
    srom[200]:  0x0000  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[208]:  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff  0xffff
    srom[216]:  0xffff  0xffff  0xffff  0xaf04
```

sta_info

Gets STA information. This command is AP-specific and gets the rateset of each associated wireless station.

```
wl sta_infor <MAC address of the WLAN intrface of the AP>
```

Example return

```
wl sta_info 00:90:4B:7A:7A:AC
STA 00:90:4B:7A:7A:AC:
rateset [ 1 2 5.5 6 9 11 12 18 24 36 48 54 ]
idle 0 seconds
in network 38 seconds
state: AUTHENTICATED ASSOCIATED
flags 0x1b: BRCM WME
```

ucantdiv

Enables/disables ucode antenna diversity.

```
wl ucantdiv <option>
```

Options

- `0` OFF
- `1` ON

upgrade

Upgrades the firmware on an embedded device.

```
wl upgrade
```

ucflags

Gets/sets ucode flags.

```
wl ucflags offset [value] [band]
```

Value

1, 2, or 3 (16-bits each)

Transmission Retry Control

lrl

Sets the long retry limit, where `limit` is an integer [1, 255]. This command indicates the number of retransmission attempts for frames longer than the RTS threshold. If this number is reduced, frames are discarded more quickly, so the buffer space requirement is lower. If this number is increased, retransmitting up to the limit takes longer and might cause the TCP to throttle back on the data rate.

```
wl lrl <limit>
```

srl

Sets the short retry limit, where `limit` is an integer [1, 255].

This command indicates the number of retransmission attempts for frames shorter than the RTS threshold. If this number is reduced, frames are discarded more quickly, so the buffer space requirement is lower. If this number is increased, retransmitting up to the limit takes longer and might cause the TCP to throttle back on the data rate.

```
wl srl <limit>
```

Rate Parameters

rate

Forces a fixed rate.

```
wl rate <rate>
```

Valid rate values

- IEEE 802.11a operation 6, 9, 12, 18, 24, 36, 48, 54
- IEEE 802.11b operation 1, 2, 5.5, 11
- IEEE 802.11g operation 1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48, 54
- -1 (default) Automatically determines the best rate.

default_rateset

Returns supported rateset of given phy. The following arguments (args) must be set:

- Arg 1. Phy Type: Must be one of the following: [a, b, g, n, lp, ssn, ht, lcn, lcn40, ac].
- Arg 2. Band Type: 2 for 2.4G or 5 for 5G.
- Arg 3. CCK Only: 1 for CCK Only or 0 for CCK and OFDM rates.
- Arg 4. Basic Rates: 1 for all rates WITH basic rates or 0 for all rates WITHOUT basic rates.
- Arg 5. MCS Rates: 1 for all rates WITH MCS rates or 0 for all rates WITHOUT MCS rates.
- Arg 6. Bandwidth: have to be one of the following: [10, 20, 40, 80, 160].
- Arg 7. TX/RX Stream: tx for TX streams or rx for RX streams.

  Example: PHY: AC, Band 2.4G, CCK rates only, With Basec rates, WithOut MCS rates, BW: 40 and TX streams Input: default_rateset ac 2 0 1 0 40 tx.

2g_rate

Forces a fixed rate for data frames in the 2.4G band: RATE_2G_USAGE.

```
wl 2g_rate
```

RATE_2G_USAGE

Either auto, or a simple CCK/DSSS/OFDM rate value of 1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54, or options to specify legacy, HT, or VHT rate.

- `-r R, --rate=R` Legacy rate (CCK, DSSS, OFDM)
- `-h M, --ht=M` HT MCS index [0–23]
- `-v M[xS], --vht=M[xS]` VHT MCS index M [0–9] and optionally, Nss S [1–8]. e.g., 5×2 is MCS=5, Nss=2
- `-c cM[sS]` VHT (c notation) MCS index M [0-9], and optionally Nss S [1–8]. e.g., c5s2 is MCS=5, Nss=2.
- `-s S, --ss=S` VHT Nss [1-8], number of spatial streams, default is 1. Only used with -v/--vht when M×S format is not used.
- `-x T, --exp=T` Tx Expansion, number of Tx chains (NTx) beyond the minimum required for the space-time-streams, exp = NTx – Nsts.
- `--stbc` Use STBC expansion; otherwise, no STBC.
- `-l, --ldpc` Use LDPC encoding; otherwise, no LDPC.
- `-g, --sgi` SGI, Short Guard Interval; otherwise, standard GI.
- `-b, --bandwidth` Transmit bandwidth in MHz: 20, 40, 80

2g_mrate

Forces a fixed rate for multicast/broadcast data frames in the 2.4G band: RATE_2G_USAGE.

```
wl 2g_mrate
```

RATE_2G_USAGE

Either auto, or a simple CCK/DSSS/OFDM rate value of 1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54, or options to specify legacy, HT, or VHT rate.

- `-r R, --rate=R` Legacy rate (CCK, DSSS, OFDM)
- `-h M, --ht=M` HT MCS index [0–23]
- `-v M[xS], --vht=M[xS]` VHT MCS index M [0–9] and optionally, Nss S [1–8]. e.g., 5×2 is MCS=5, Nss=2
- `-c cM[sS]` VHT (c notation) MCS index M [0-9], and optionally Nss S [1–8]. e.g., c5s2 is MCS=5, Nss=2.
- `-s S, --ss=S` VHT Nss [1-8], number of spatial streams, default is 1. Only used with -v/--vht when M×S format is not used.
- `-x T, --exp=T` Tx Expansion, number of Tx chains (NTx) beyond the minimum required for the space-time-streams, exp = NTx – Nsts.
- `--stbc` Use STBC expansion; otherwise, no STBC.
- `-l, --ldpc` Use LDPC encoding; otherwise, no LDPC.
- `-g, --sgi` SGI, Short Guard Interval; otherwise, standard GI.
- `-b, --bandwidth` Transmit bandwidth in MHz: 20, 40, 80

5g_rate

Forces a fixed rate for data frames in the 5G band: RATE_5G_USAGE

```
wl 5g_rate
```

RATE_5G_USAGE

Either auto, or a simple OFDM rate value of 6, 9, 12, 18, 24, 36, 48, 54, or options to specify legacy OFDM, HT, or VHT rate.

- `-r R, --rate=R` Legacy OFDM rate
- `-h M, --ht=M` HT MCS index [0-23]
- `-v M[xS], --vht=M[xS]` HT MCS index M [0-9] and optionally, Nss S [1–8]. e.g., 5×2 is MCS=5, Nss=2.
- `-c cM[sS]` VHT (c notation) MCS index M [0-9] and optionally, Nss S [1–8]. e.g., c5s2 is MCS=5, Nss=2.
- `-s S, --ss=S` VHT Nss [1-8], number of spatial streams; default is 1.
- Only used with -v/--vht when M×S format is not used.
- `-x T, --exp=T` Tx Expansion, number of Tx chains (NTx) beyond the minimum required for the space-time-streams, exp = NTx – Nsts.
- `--stbc` Use STBC expansion; otherwise, no STBC.
- `-l, --ldpc` Use LDPC encoding; otherwise, no LDPC.
- `-g, --sgi` SGI, Short Guard Interval; otherwise, standard GI.
- `-b, --bandwidth` Transmit bandwidth in MHz: 20, 40, 80

5g_mrate

Forces a fixed rate for multicast/broadcast data frames in the 5G band: RATE_5G_USAGE.

```
wl 5g_mrate
```

RATE_5G_USAGE

Either auto, or a simple OFDM rate value of 6, 9, 12, 18, 24, 36, 48, 54, or options to specify legacy OFDM, HT, or VHT rate.

- `-r R, --rate=R` Legacy OFDM rate
- `-h M, --ht=M` HT MCS index [0-23]
- `-v M[xS], --vht=M[xS]` HT MCS index M [0-9] and optionally, Nss S [1–8]. e.g., 5×2 is MCS=5, Nss=2.
- `-c cM[sS]` VHT (c notation) MCS index M [0-9] and optionally, Nss S [1–8]. e.g., c5s2 is MCS=5, Nss=2.
- `-s S, --ss=S` HT Nss [1-8], number of spatial streams; default is 1.
- Only used with -v/--vht when M×S format is not used.
- `-x T, --exp=T` Tx Expansion, number of Tx chains (NTx) beyond the minimum required for the space-time-streams, exp = NTx – Nsts.
- `--stbc` Use STBC expansion; otherwise, no STBC.
- `-l, --ldpc` Use LDPC encoding; otherwise, no LDPC.
- `-g, --sgi` SGI, Short Guard Interval; otherwise, standard GI.
- `-b, --bandwidth` Transmit bandwidth in MHz: 20, 40, 80

bg_mrate/a_mrate

Forces a fixed multicast rate.

```
wl bg_mrate/a_mrate <rate>
```

Valid rate values

- `IEEE 802.11a operation` 6, 9, 12, 18, 24, 36, 48, 54
- `IEEE 802.11b operation` 1, 2, 5.5, 11
- `IEEE 802.11g operation` 1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48, 54
- `-1` (default) Automatically determines the best rate.

a_rate

Forces a fixed rate for the A PHY. The rate option without any arguments returns the current negotiated rate between a STA and an AP in infrastructure mode, or the weighted average of the last 32 frames sent in ad hoc mode between two stations. The best choice is to keep the value at –1, which is the auto-negotiate mode. To override the current rate, add the required rate argument. Then, every packet sent out has this new rate stamped in it.

```
wl a_rate <rate>
```

Valid rate values

- `IEEE 802.11a operation` 6, 9, 12, 18, 24, 36, 48, 54
- `-1` (default) Automatically determines the best rate.

mrate

Forces a fixed multicast rate.

```
wl mrate <rate>
```

Valid values

- `IEEE 802.11a operation` 6, 9, 12, 18, 24, 36, 48, 54
- `IEEE 802.11b operation` 1, 2, 5.5, 11
- `IEEE 802.11g operation` 1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48, 54
- `-1` (default) Automatically determines the best rate.

bg_rate

Forces a fixed rate for the IEEE 802.11b/g PHY.

```
wl bg_rate <rate>
```

Valid values

- `IEEE 802.11b operation` 1, 2, 5.5, 11
- `IEEE 802.11g operation` 1, 2, 5.5, 6, 9, 11, 12, 18, 24, 36, 48, 54
- `-1` (default) Automatically determines the best rate.

nrate

When set, this command applies to a band-specific rate override; when used to query, it gets the moving average or band-specific rate override (if it is on).

```
wl nrate [-r] [-m] [-s] [-w]
```

Use `auto` to clear a rate override, or use the following:

- `r` Legacy rate (CCK, OFDM)
- `m` HT MCS index
- `s` STF mode (0=SISO, 1=CDD, 2=STBC, 3=SDM)
- `w` Override MCS only to support STAs with/without STBC capability.

bands

Returns the list of available IEEE 802.11 bands.

```
wl bands
```

Bands options

- `0` Auto-Select
- `1` 5 GHz
- `2` 2.4 GHz
- `3` All bands

rateparam

Sets driver rate selection tunables.

```
wl rateparam <arg1> <arg2>
```

- `<arg 1>` Tunable ID
- `<arg 2>` Tunable value

rateset

Returns or sets the supported and basic ratesets. With no arguments, returns the rateset.

```
wl rateset "default" | "all" | <arbitrary rateset>
```

Example return

```
[ 1(b) 2(b) 5.5(b) 6 9 11(b) 12 18 24 36 48 54 ]
```

Arguments

- `default` Driver defaults
- `all` All rates are basic rates.
- `<arbitrary rateset>` List of rates

Listed rates are in Mbps. Each rate is optionally followed by "`(b)`" or "`b`" for a basic rate. For example, `1(b) 2b 5.5 11`.

At least one rate must be basic for a legal rateset.

Note: To change the rateset, run `wl down` to take down the driver, change the rateset, and then run `wl up` to bring up the driver.

dump_rateset

Prints a per-scb rate set chosen by the rate selection stdout based on remote station MAC address[xx:xx:xx:xx:xx:xx].

Example of usage

```
# wl dump_rateset 88:9F:FA:0B:82:D7AC[0] ---
state 0xc4f9982c active_rates_num 11 bwcap 1 spmode 0 use_sgi 0 antsel OFF
rate index : rate   rspec 500Kbps fbr/dn/up
rate_id  0 : x02 0x10002    2   0  0  1
rate_id  1 : x04 0x10004    4   0  0  2
rate_id  2 : x0b 0x1000b   11   0  1  3
rate_id  3 : mx10 0x2010010   13   1  2  4
rate_id  4 : mx11 0x2010011   26   3  3  5
rate_id  5 : mx12 0x2010012   39   3  4  6
rate_id  6 : mx13 0x2010013   52   4  5  7
rate_id  7 : mx14 0x2010014   78   5  6  8
rate_id  8 : mx15 0x2010015  104   6  7  9
rate_id  9 : mx16 0x2010016  117   7  8 10
rate_id 10 : mx17 0x2010017  130   7  9 10
```

suprates

Gets/sets the IEEE 802.11g override for the supported rateset. With no arguments, returns the rateset.

```
wl suprates
```

Arguments

A list of rates `<rate1 rate2 rate3...>`

`0` or `-1` to specify an empty rateset to clear the override

Listed rates are in Mbps.

Example return

```
1 2 5.5 11
```

Antenna Controls

antdiv_bcnloss

```
wl antdiv_bcnloss <beaconloss_count>
```

- 0: Disables Rx antenna flip feature based on consecutive beacon loss.
- X: Beacon loss count after which Rx antenna will be flipped.

antdiv

Sets the antenna diversity property for RX. An antenna diversity receiver could be receiving the wireless/RF signal on one or more antennas having distinct characteristics (for example, location, radiation pattern and/or polarization). In such situations, when one antenna is experiencing bad reception conditions (deep or flat fading), the other should probably not. Therefore, when the right antenna is set at the right moment, the diversity receiver behaves as if it is receiving a continuously perfect signal.

```
wl antdiv <value>
```

- `0` Forces the use of antenna 0.
- `1` Forces the use of antenna 1.
- `3` Automatic selection of antenna diversity.

phy_rssi_ant

Gets RSSI per antenna (only gives RSSI of current antenna for SISO PHY).

```
wl phy_rssi_ant
```

txant

Sets the transmit antenna.

```
wl txant <value>
```

- `0` Forces the use of antenna 0.
- `1` Forces the use of antenna 1.
- `3` Uses the RX antenna selection that was in force during the most recently received good PLCP header.

txchain

Sets the transmit chain.

```
wl txchain <txchain bitmap>
```

- `1` Default bitmap of txchain for single stream PHY (legacy)
- `2 or 3 (10, 11)` Default bitmap of txchains for NPHY
- `7 (111)` Default bitmap for HTPHY (4331)

Example

- `wl txchain 0x1` Limit TX to antenna/chain 0.
- `wl txchain 0x2` Limit TX to antenna/chain 1.

Security and Encryption Controls

wepstatus

Sets or gets the WEP status. (This command is deprecated. Use “wsec” on page 55.)

primary_key

Sets/gets the index of the primary key.

```
wl primary_key
```

addwep

Sets an encryption key. The OID_802_11_ADD_WEP OID requests the miniport driver to set an IEEE 802.11 Wired Equivalent Privacy (WEP) key to a specified value. The key must be 5, 13, or 16 bytes long or 10, 26, 32, or 64 hexadecimal digits long. The encryption algorithm is automatically selected based on the key size. The typed key is accepted only when the key length is 16 bytes/32 hexadecimal digits and specifies whether AES-OCB or AES-CCM encryption is used. Default is CCM.

```
wl addwep <keyindex> <keydata> [ocb | ccm] [notx] [xx:xx:xx:xx:xx:xx]
```

rmwep

Removes the encryption key at the specified key index.

```
wl rmwep
```

keys

Prints a list of the current WEP keys.

```
wl keys
```

tsc

Prints the TX SEQ counter for the key at a specified key index.

```
wl tsc
```

wsec_test

Generates wsec errors.

```
wl wsec_test <test_type> <keyindex|xx:xx:xx:xx:xx:xx>
```

Example return

```
(wsec test_type may be a number or name from the following set):
   0x0001 mic_error
   0x0002 replay
```

tkip_countermeasures

Enables or disables TKIP countermeasures (TKIP-enabled AP only).

```
wl tkip_countermeasures
```

- `0` Disable
- `1` Enable

wsec_restrict

Drops unencrypted packets if wsec is enabled.

```
wl wsec_restrict
```

- `0` Disable
- `1` Enable

eap

Restricts traffic to IEEE 802.1X packets (until IEEE 802.1X authorization succeeds).

```
wl eap
```

- `0` Disable
- `1` Enable

eap_restrict

Gets/sets EAP restriction.

```
wl eap_restrict
```

authorize

Restricts traffic to IEEE 802.1X packets until the 802.1X authorization succeeds.

```
wl authorize
```

deauthorize

Does not restrict traffic to IEEE 802.1X packets until the 802.1X authorization succeeds.

```
wl deauthorize
```

deauthenticate

Deauthenticates a STA from the AP with optional reason code (AP only).

```
wl deauthenticate
```

wsec

Returns/sets a value containing a bit vector representing which wireless security modes are currently enabled.

```
wl wsec
```

- `1` WEP enabled
- `2` TKIP enabled
- `4` AES enabled
- `8` WSEC in software
- `0x80` FIPS enabled
- `0x100` WAPI enabled

auth

Gets/sets the IEEE 802.11 authentication type.

```
wl auth
```

- `0` Open system
- `1` Shared key
- `2` Open/shared

wpa_auth

Bit vector of WPA authorization modes.

```
wpa_auth
```

- `1` WPA–NONE
- `2` WPA–802.1X/WPA-Enterprise
- `4` WPA–PSK/WPA-Personal
- `8` CCKM (WPA)
- `16` CCKM (WPA2)
- `64` WPA2-802.1X/WPA2-Professional
- `128` WPA2-PSK/WPA2-Personal
- `0` Disable WPA

wpa_cap

Sets/gets IEEE 802.11i RSN capabilities.

```
wl wpa_cap
```

set_pmk

Sets the passphrase for PMK (in-driver resident supplicant).

```
wl set_pmk
```

mac

Sets or gets the list of source MAC address matches. Multiple access points having the same SSID could cause undesirable results during tests because the client and the server could intermittently roam to different access points. To gain control over such behavior, the utility supports the **mac** and **deny** commands.

For the AP driver, the list controls which stations are allowed to authenticate. For the STA driver, the list allows/restricts association to a particular set of BSSIDs. You can add multiple MAC addresses to the list and then allow/deny access of these based on your deny status.

By default, association works normally, and all valid targets are considered. If you specify a list of BSSIDs, then all other BSSIDs will be ignored before attempting a join. If you turn the list into a deny list by specifying `wl deny 1,` BSSIDs specified in the list will be ignored before a join is attempted. To clear the list and return to normal functionality, enter `wl mac none.`

```
wl mac xx:xx:xx:xx:xx:xx [xx:xx:xx:xx:xx:xx ...]
```

Values

- `0` Allows association to stations on the MAC list.
- `1` Denies association to stations on the MAC list.

macmode

Sets the mode of the MAC list.

```
wl macmode
```

Values

- `0` Disables MAC address matching.
- `1` Denies association to stations on the MAC list.
- `2` Allows association to stations on the MAC list.

encryptstrength

Gets the current WEP key length.

```
wl encryptstrength
```

decryptstatus

Gets the status of WEP decryption.

```
wl deencryptstatus
```

Returns

One of:
- `1` Success
- `2` Failure
- `3` Unknown

addkey

Sets an IEEE 802.11 Wired Equivalent Privacy (WEP) key. The OID_802_11_ADD_KEY OID requests the miniport driver to set an IEEE 802.11 Wired Equivalent Privacy (WEP) key to a specified value.

```
wl addkey <keyindex> <keydata> [notx] [xx:xx:xx:xx:xx:xx]
```

wepdefault

Resets the WEP keys to their power-on defaults.

```
wl wepdefault
```

pmkid_info

Returns the PMKID table.

```
wl pmkid_info
```

Options

- -s S, -ssid SSSID to scan
- -t ST, -scan_type ST [active|passive] scan type
- -t bss_type BT [bss/infra|ibss/adhoc] bss type to scan
- -b MAC, -bssid MAC Particular BSSID MAC address to scan, xx:xx:xx:xx:xx:xx
- -n N, -nprobes N Number of probes per scanned channel
- -a N, -activeN Dwell time per channel for active scanning
- -p N, -passiveN Dwell time per channel for passive scanning
- -h N, -homeN Dwell time for the home channel between channel scans
- -c L, -channelsL Comma- or space-separated list of channels to scan

Scan Controls

csscantimer

Auto channel scan timer, in minutes (0 to disable).

```
wl csscantimer
```

escan

Starts an escan. Defaults to an active scan across all channels for any SSID. Optional arg: SSIDs, list of [up to 10] SSIDs to scan (comma or space separated).

```
wl escan [option]
```

Options:

- -s S, --ssid=S SSIDs to scan
- -t ST, --scan_type=ST [active|passive|prohibit|offchan|hotspot] scan type
- --bss_type=BT [bss/infra|ibss/adhoc] bss type to scan
- -b MAC, --bssid=MAC particular BSSID MAC address to scan, xx:xx:xx:xx:xx:xx
- -n N, --nprobes=N number of probes per scanned channel
- -a N, --active=N dwell time per channel for active scanning
- -p N, --passive=N dwell time per channel for passive scanning
- -h N, --home=N dwell time for the home channel between channel scans

escanabort

Aborts an escan.Defaults to an active scan across all channels for any SSID. Optional arg: SSIDs, list of [up to 10] SSIDs to scan (comma or space separated).

```
wl escanabort [option]
```

Options:

- -s S, --ssid=S SSIDs to scan
- -t ST, --scan_type=ST [active|passive|prohibit|offchan|hotspot] scan type
- --bss_type=BT [bss/infra|ibss/adhoc] bss type to scan
- -b MAC, --bssid=MAC particular BSSID MAC address to scan, xx:xx:xx:xx:xx:xx
- -n N, --nprobes=N number of probes per scanned channel
- -a N, --active=N dwell time per channel for active scanning
- -p N, --passive=N dwell time per channel for passive scanning
- -h N, --home=N dwell time for the home channel between channel scans

iscan_s

Initiates an incremental scan. Defaults to an active scan across all channels for any SSID. Optional arg: SSIDs, list of [up to 10] SSIDs to scan (comma or space separated).

```
wl iscan_s [option]
```

Options:

- -s S, --ssid=S SSIDs to scan
- -t ST, --scan_type=ST [active|passive|prohibit|offchan|hotspot] scan type
- --bss_type=BT [bss/infra|ibss/adhoc] bss type to scan
- -b MAC, --bssid=MAC particular BSSID MAC address to scan, xx:xx:xx:xx:xx:xx
- -n N, --nprobes=N number of probes per scanned channel
- -a N, --active=N dwell time per channel for active scanning
- -p N, --passive=N dwell time per channel for passive scanning
- -h N, --home=N dwell time for the home channel between channel scans

iscan_c

Continues an incremental scan. Defaults to an active scan across all channels for any SSID. Optional arg: SSIDs, list of [up to 10] SSIDs to scan (comma or space separated).

```
wl iscan_c [option]
```

Options:

- -s S, --ssid=S SSIDs to scan
- -t ST, --scan_type=ST [active|passive|prohibit|offchan|hotspot] scan type
- --bss_type=BT [bss/infra|ibss/adhoc] bss type to scan
- -b MAC, --bssid=MAC particular BSSID MAC address to scan, xx:xx:xx:xx:xx:xx
- -n N, --nprobes=N number of probes per scanned channel
- -a N, --active=N dwell time per channel for active scanning
- -p N, --passive=N dwell time per channel for passive scanning
- -h N, --home=N dwell time for the home channel between channel scans

passive

Puts the scan engine into passive mode.

```
wl passive
```

prescanned

Uses channel and bssid list from scanresults. Gets prescanned channels and bssids.

```
wl prescanned
```

scan

Initiates a scan.Defaults to an active scan across all channels for any SSID. Optional arg: SSIDs, list of [up to 10] SSIDs to scan (comma or space separated).

```
wl scan
```

Options

- `-s S`, --ssid=S SSIDs to scan
- `-t ST`, --scan_type=ST [active|passive|prohibit|offchan|hotspot] scan type--bss_type=BT [bss/infra |ibss/adhoc] bss type to scan
- `-b` MAC, --bssid=MAC particular BSSID MAC address to scan, xx:xx:xx:xx:xx:xx
- `-n` N, --nprobes=N number of probes per scanned channel
- `-a` N, --active=N dwell time per channel for active scanning
- `-p` N, --passive=N dwell time per channel for passive scanning
- `-h` N, --home=N dwell time for the home channel between channel scans
- `-c` L, --channels=L comma or space separated list of channels to scan

Option

- `1` Suppress scans.

scan_ps

Gets/Sets scan power optimization enable/disable.

```
wl scan_ps
```

scanabort

Aborts a scan.

```
wl scanabort
```

scanresults

Returns the results from the last scan.

```
wl scanresults
```

scan_channel_time

Gets/sets the scan channel time.

```
wl scan_channel_time
```

scan_unassoc_time

Gets/sets the unassociated scan channel dwell time.

```
wl scan_unaccoc_time
```

scan_home_time

Gets/sets the scan home channel dwell time.

```
wl scan_home_time
```

scan_passive_time

Gets/sets the passive scan channel dwell time.

```
wl scan_passive_time
```

scan_nprobes

Gets/sets the scan parameter for number of probes to use (per channel scanned).

```
wl scan_nprobes
```

scansuppress

Suppresses all scans for testing.

```
wl scansuppress
```

- `0` Allow scans.

extdscan

Initiates an extended scan. Defaults to an active scan across all channels for any SSID.

```
wl extdscan
```

Options

- `-s S1 S2 S3-ssid S1 S2 S3` SSIDs to scan, comma- or space- separated
- `-x x -split_scanST [split_scan]` Scan type
- `-t ST-scan_typeST [background: 0/forced background: 1/foreground: 2]` Scan type
- `-n N-nprobesN` Number of probes per scanned channel per SSID
- `-c L-channelsL` Comma- or space-separated list of channels to scan

scan

Initiates a scan. The default scan is an active scan across all channels for any SSID. Optional argument to scan a specific SSID: `SSID`.

```
wl scan
```

iscan_s

Initiates an incremental scan. Defaults to an active scan across all channels for any SSID. Optional arguments: SSID (SSID to scan).

```
wl iscan_s
```

Options

- `-s S,-ssid` SSSID to scan
- `-t ST,-scan_typeST`[active|passive] scan type
- `-bss_typeBT`[bss/infra|ibss/adhoc] bss type to scan
- `-b MAC,-bssidMAC` Particular BSSID MAC address to scan, xx:xx:xx:xx:xx:xx
- `-n N,-nprobesN` Number of probes per scanned channel
- `-a N,-ActiveN` Dwell time per channel for active scanning
- `-p N,-PassiveN` Dwell time per channel for passive scanning
- `-h N,-HomeN` Dwell time for the home channel between channel scans
- `-c L,-ChannelsL` Comma- or space-separated list of channels to scan

iscan_c

Continues an incremental scan. Defaults to an active scan across all channels for any SSID.

Optional arguments: SSID (SSID to scan)

```
wl iscan_c
```

Options

- -s S, -ssid SSSID to scan
- -t ST, -scan_typeST [active|passive] scan type
- - bss_typeBT [bss/infra|ibss/adhoc] bss type to scan
- -b MAC, -bssidMAC Particular BSSID MAC address to scan, xx:xx:xx:xx:xx:xx
- -n N, -nprobesN Number of probes per scanned channel
- -a N, -activeN Dwell time per channel for active scanning
- -p N, -passiveN Dwell time per channel for passive scanning
- -h N, -homeN Dwell time for the home channel between channel scans
- -c L, -channelsL Comma- or space-separated list of channels to scan

iscanresults

Returns results of the last `iscan`. Specify a `buflen` (maximum of 8188) to artificially limit the size of the results buffer.

```
wl iscanresults [buflen]
```

obss_scan_params

Sets/gets overlapping BSS scan parameters.

```
wl obss_scan_params a b c d e f g
```

- `a` Passive Dwell, {5–1000TU}, default = 100
- `b` Active Dwell, {10–1000TU}, default = 20
- `c` Width Trigger Scan Interval, {10–900 sec}, default = 300
- `d` Passive Total per Channel, {200–1000 0TU}, default = 200
- `e` Active Total per Channel, {20–100 0TU}, default = 20
- `f` Channel Transition Delay Factor, {5–100}, default = 5
- `g` Activity Threshold, {0–100%}, default = 25

scancache_clear

Clears the scan cache.

```
wl scancache_clear
```

Association and Status

isup

Gets the operational state of the driver.

```
wl isup
```

- `0` Down
- `1` Up

assoclist

Gets the list of associated MAC addresses. AP only: Gets the list of associated MAC addresses.

```
wl assoclist /* AP Side
```

Example return

```
Linux */
```

assoc

Prints information about the current network association to the AP with the specified BSSID.

```
wl assoc
```

disassoc

Disassociate from the current BSS/IBSS.

```
wl disassoc
```

shownetworks

Pretty-prints the BSSID list.

```
wl shownetworks
```

authe_sta_list

Gets the authenticated STA MAC address list.

```
wl authe_sta_list
```

autho_sta_list

Gets the authorized STA MAC address list.

```
wl autho_sta_list
```

ssid

Sets or gets the SSID of a wireless network connection profile. This setting initiates an association attempt if in infrastructure mode, joins/creates an IBSS if in IBSS (ad hoc) mode, or creates a BSS if in AP mode.

```
wl ssid
```

closednet

Sets/gets the BSS closed network attribute. When this setting is enabled (= 1), the adapter does not send out a probe response as a result of having received a Broadcast Probe request.

```
wl closednet
```

assoc_info

Returns the association request and response information (STA only).

```
wl assoc_info
```

closed

Hides the network from active scans, 0 or 1.

```
wl closed
```

- `0` Open
- `1` Hide

bss

Sets/gets BSS Enabled status: Up/Down.

```
wl bss
```

join_pref

Sets/gets join target preferences.

```
wl join_pref
```

assoc_pref

Sets/gets the association preference.

```
wl assoc_pref [auto|a|b|g]
```

Channel and Band Control Parameters

auto

Switches between available frequency bands (default).

```
wl auto
```

- `a` Forces use of IEEE 802.11a band.
- `b` Forces use of IEEE 802.11b band.

autochannel

Automatic channel selection. Without an argument to show only the channel selected, `ssid` must be set to null before this process, and RF must be up.

```
wl autochannel
```

- `1` Issues a channel scanning.
- `2` Sets the channel based on the channel scanning result.

bs_data

Displays per station band steering data.

```
wl bs_data [options]
```

Options

- `comma`: Use commas to separate values rather than blanks.
- `tab`: Use <TAB> to separate values rather than blanks.
- `raw`: Displays raw values as received from the driver.
- `noidle`: Do not display idle stations.
- `noreset`: Do not reset counters after reading

chanspecs_defset

Gets default chanspecs for current driver settings.

```
wl chanspecs_defset
```

Example return in 2G band within the current locale and regulatory revision

```
1 (0x1001)
2 (0x1002)
3 (0x1003)
4 (0x1004)
5 (0x1005)
6 (0x1006)
7 (0x1007)
8 (0x1008)
9 (0x1009)
10 (0x100a)
11 (0x100b
```

chan_info

Returns channel information.

```
wl chan_info
```

channel

This command sets the default channel. As an AP, the default channel is used as the channel on which the AP found the network. For a STA, the default channel is used as the channel on which the STA creates an IBSS (ad hoc) network.

Valid channels for 802.11a (5 GHz band) are 36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116,120, 124, 128, 132, 136, 140, 149, 153, 157, 161, 184, 188, 192, 196, 200, 204, 208, 212, 216.

Valid channels for 802.11b/g (2.4 GHz band) are 1 through 14.

```
wl channel <channel #>
```

chanlist

Deprecated. Use channels.

channels

Returns valid channels for the current settings.

```
wl channels
```

Example return for US country code/locale

`1 2 3 4 5 6 7 8 9 10 11`, and so on

channels_in_country

Returns valid channels for the country specified.

```
wl channels_in_country
```

- Argument 1 is the country abbreviation.
- Argument 2 is the band (a or b).

force_vsdb_chans

Sets/gets channels for the forced virtual simultaneous dual-band (vsdb) mode. The parameter functions based on time division multiplexing (TDM). Within the context of single radio use, the device can be configured to operate only on a specified channel. To function simultaneously on two different channels, the radio should switch between these channels frequently so that the user is aware that both interfaces are always active.

```
wl force_vsdb_chans chan1 chan2
```

Note: Provides a channel in the same format as chanspec: eg force_vsdb_chans 1l 48u.

chq_event

Sets parameters associated with channel quality event notification.

```
wl chq_event <rate_limit> <cca_levels> <nf_levels> <nf_lte_levels>
```

rate_limit: Number of events posted to application will be limited to 1 per this rate limit. Set to 0 to disable rate limit.

csa/nf/nf_lte levels: Variable number of threshold levels (maximum 8) in pairs of hi-to-low/lo-to-hi, and in increasing order (such as -90, -85, -80). A 0 0 pair terminates level array for one metric. An event will be posted whenever a threshold is being crossed.

Roam Controls

roam_channels_in_cache

Gets a list of channels in the roam cache.

```
wl roam_channels_in_cache
```

roam_channels_in_hotlist

Gets a list of channels in the roam hot channel list.

```
wl roam_channels_in_hotlist
```

roam_trigger

Gets or sets the roam trigger RSSI threshold.

```
wl roam_trigger
```

Example

Get: `wl roam_trigger [a|b]`

Set: `wl roam_trigger <integer> [a|b|all]`

Integer

- `0` Default
- `1` Optimize bandwidth.
- `2` Optimize distance.
- `[-1, -99]` dBm trigger value.

roam_delta

Sets the roam candidate qualification delta.

```
wl roam_delta [integer [, a/b]]
```

Example return

```
roam_delta is 0x0014(20)
```

roamscan_parms

Sets/Gets roam scan parameters. Uses a standard scan parameter syntax shown below. Only active/passive/ home times, nprobes, and types are used. All other values are silently discarded. Defaults to an active scan across all channels for any SSID.

```
wl roamscan_parms a|b|p| [option]
```

Optional arg: SSIDs, list of [up to 10] SSIDs to scan (comma or space separated). Options:

-s S, --ssid=S	SSIDs to scan
-t ST, --scan_type=ST	[active|passive|prohibit|offchan|hotspot] scan type
--bss_type=BT	[bss/infra|ibss/adhoc] bss type to scan
-b MAC, --bssid=MAC	particular BSSID MAC address to scan, xx:xx:xx:xx:xx:xx
-n N, --nprobes=N	number of probes per scanned channel
-a N, --active=N	dwell time per channel for active scanning
-p N, --passive=N	dwell time per channel for passive scanning
-h N, --home=N	dwell time for the home channel between channel scans
-c L, --channels=L	comma or space separated list of channels to scan

roam_prof

Gets/sets roaming profiles (need to specify band).

```
wl roam_prof_2g a|b|2g|5g flags rssi_upper rssi_lower delta, boost_thresh boot_delta nfscan
fullperiod initperiod backoff maxperiod
```

roam_scan_period

Sets the roam candidate qualification delta.

```
wl roam_scan_period <integer>
```

prb_resp_timeout

Gets/sets the probe response timeout.

```
wl prb_resp_timeout
```

pfn_roam_alert_thresh

Gets/sets PFN and roam alert threshold.

```
wl pfn_roam_alert_thresh [pfn_alert_thresh] [roam_alert_thresh]
```

Regulatory Test and Measurements

band

Returns or sets the current band.

```
wl band
```

- `auto` Autoswitch between available bands (default).

clk

Sets the board clock state. Returns an error for a `set_clk` attempt if the driver is not down.

```
wl clk
```

- `0` Clock off
- `1` Clock on

channel_qa

Gets the last channel quality measurement.

```
wl channel_qa
```

channel_qa_start

Starts a channel quality measurement.

```
wl channel_qa_start
```

ccode_info

Get country code information.

```
wl ccode_info
```

chanspec_txpwr_max

Returns valid chanspecs with max tx power settings.

```
wl chanspec_txpwr_max [a, b] [w]
```

- -b band (5(a) or 2(b/g))
- -w bandwidth, 20, 40, 80, 160 or 8080

Example

```
# wl chanspec_txpwr_max
1       (0x1001)       16.00(dbm)
2       (0x1002)       16.00(dbm)
3       (0x1003)       16.00(dbm)
4       (0x1004)       16.00(dbm)
5       (0x1005)       16.00(dbm)
6       (0x1006)       16.00(dbm)
7       (0x1007)       16.00(dbm)
8       (0x1008)       16.00(dbm)
9       (0x1009)       16.00(dbm)
10      (0x100a)       16.00(dbm)
11      (0x100b)       16.00(dbm)
5u      (0x1903)        1.00(dbm)
6u      (0x1904)        1.00(dbm)
7u      (0x1905)        1.00(dbm)
8u      (0x1906)        1.00(dbm)
9u      (0x1907)        1.00(dbm)
10u     (0x1908)        1.00(dbm)
11u     (0x1909)        1.00(dbm)
1l      (0x1803)        1.00(dbm)
2l      (0x1804)        1.00(dbm)
3l      (0x1805)        1.00(dbm)
4l      (0x1806)        1.00(dbm)
5l      (0x1807)        1.00(dbm)
6l      (0x1808)        1.00(dbm)
7l      (0x1809)        1.00(dbm)
```

csa

Sends an IEEE 802.11h channel switch announcement with `chanspec`.

```
wl csa <mode> <count> <channel>[a,b][n][u,l]
```

- `mode` 0 or 1
- `count` 0–254
- `channel` 0–224

Band:
- `a` 5G band
- `b` 2G band

Bandwidth:
- `n` 10, non for 20 & 40

CTL sideband:
- `l` Lower
- `u` Upper, no CTL sideband (default)

constraint

Sends an IEEE 802.11h power constraint information element (IE).

```
wl constraint 1-255 db
```

curpower

Returns the current TX power settings.

```
wl curpower
```

- `-v, --verbose` Displays power settings for every rate even when every rate in group has identical power.

curppr

Returns the current TX power per rate offset.

```
wl curppr
```

diag

Diagnostic test index. Precede the `diag` command by the `wl down` command, and then follow with `wl up`.

- `1` Interrupt
- `2` Loopback
- `3` Memory
- `4` LED

dtim

Gets/sets the delivery traffic indication message (DTIM).

```
wl dtim
```

evm

Starts an EVM test on the given channel or stops an EVM test.

```
wl evm
```

- Argument 1 is the channel number 1–14, or off, or 0 to stop the test.
- Argument 2 is an optional rate (1, 2, 5.5, or 11).

To perform EVM tests using the wl:

1. Issue the command `wl down` to disable the wireless driver.
2. Turn on EVM mode on any channel. To do this, issue the command `wl evm 2`, where `2` is the channel on which you want to transmit.

When you are done with EVM, issue the command `wl evm 0` and then `wl up` to return the wireless driver back to normal operation.

noise

Immediately after Tx, gets the noise level (moving average) in dBm.

```
wl noise
```

PM

Sets the driver power management mode.

```
wl PM
```

- `0` CAM (constantly awake)
- `1` PS (power-save)
- `2` FAST PS mode

In the Broadcom WLAN driver, FAST PS mode consumes slightly more power than the standard PS mode, but can operate at higher rates (close to 54g® rates). The difference between FAST PS and PS is very dependent on the usage model. If there is no traffic, there is no difference. If the traffic is a steady stream (some packets every DTIM), there is no difference. If the traffic is a stream that happens less often than every DTIM, then the adapter consumes more power in FAST PS mode and less power but more latency in PS mode. The amount of power difference depends on the frequency of traffic.

txpwr_target_max

Returns current max tx target power settings.

```
wl txpwr_target_max
```

Example

```
wl txpwr_target_max
Maximum Tx Power Target (chanspec:0x100b):      14.50  14.50  14.50
```

wake

Sets the driver Power-save mode sleep state.

```
wl wake <value>
```

- `0` Core-managed
- `1` Awake

For STA only:

wake

- TRUE (1Awake)
- FALSE (0Core managed)

When this field is set to 1, the MAC exits the SLEEP state immediately. When the field is set to 0, the MAC enters the SLEEP state, as appropriate. The host must write 1 to this field when the MAC is functioning as an AP.

tssi

Gets the TSSI value from the radio. The need to measure signal levels in wireless infrastructure equipment is critical to adjusting transceiver automatic gain control (AGC) circuits, in the receiver (RX) to achieving maximum sensitivity to the varying inputs from the mobile users, and in the transmitter (TX) to maintaining the output power at its optimum level for performance mask, power amplifier (PA) efficiency and linearity, and government regulations.

Within an RX, the received signal strength indication (RSSI) is used to adjust the gain of the RX to extend the dynamic range to 100 dB. For the TX, accurately controlling the transmit signal power with a transmitted signal strength indication (TSSI) at RF frequencies at the higher power levels significantly eases the implementation of controls for PA operating level for maximum efficiency.

```
wl tssi
```

txpwr

Sets Tx power in milliwatts. Range [1, 84]. When Tx power is set, it overrides the current setting (SPROM, locale, and so on) and forces the transmitter to always use this new value for power output.

```
wl txpwr
```

Note: For dual-band cards, you must bandlock to your required band to make this command work.

Example

First run `wl band b` to lock in the 2.4 GHz band, and then use this command.

txpwr1

Sets Tx power in various units. Choose one of:

- `d` dBm (default)
- `q` quarter dBm
- `m` milliwatt

```
wl txpwr1
```

Can be combined with:

- Turns on override to disable regulatory and other limitations.

Use wl `txpwr -1` to restore defaults.

txpathpwr

On BCM2050 radios, turns the Tx path power on or off.

```
wl txpathpwr
```

txpwrlimit

Returns the current Tx power limit.

```
wl txpwrlimit
```

txchain_pwr_offset

Gets/sets the current offsets for each core in qdBm (quarter dBm).

```
wl txchain_pwr_offset [qdBm offsets]
```

powerindex

Sets the transmit power for A band (0–63).

```
wl powerindex
```

- `1` Default value

rssi

Gets the current RSSI value. For an AP, the MAC address of the STA must be specified. For an AP, the MAC address of the STA must be specified.

```
wl rssi /*STA side */
```

Note: The MAC layer operates with the physical layer (PHY) by sampling the transmitted energy over the medium transmitting the data. The PHY uses a clear channel assessment (CCA) algorithm to determine if the channel is clear. This function is accomplished by measuring the RF energy at the antenna and determining the strength of the received signal. This measured signal is commonly known as RSSI. If the received signal strength is below a specified threshold, the channel is declared clear, and the MAC layer is given the clear channel status for data transmission. If the RF energy is above the threshold, data transmissions are deferred in accordance with the protocol rules. The standard provides another option for CCA that can be alone or with the RSSI measurement.

rssi_event

Sets the parameters associated with an RSSI event notification.

```
wl rssi_event <rate_limit> <rssi_levels>
```

Options

- `rate_limit` The number of events posted to the application are limited to 1 according to this rate limit. Set to 0 to disable rate limit.
- `rssi_levels` Variable number of RSSI levels (maximum 8) in increasing order (such as –85, –70, –60). An event is posted each time the RSSI of received beacons/packets crosses a particular level.

pwr_percent

Gets/sets the power output percentage.

```
wl pwr_percent
```

To set a specified percentage, such as 60 percent of the full (100%) power:

- `wl pwr_percent 60` Sets the new value.
- `wl pwr_percent` Gets the new setting.

Example return

```
pwr_percent is 60(0x3c)
```

rand

Gets a 2-byte random number from the MAC PRNG.

```
wl rand
```

reset_d11cnts

Resets the IEEE 802.11 MIB counters.

```
wl reset_d11cnts
```

regulatory

Gets/sets the regulatory domain mode (IEEE 802.11d).

Note: The driver must be down.

```
wl regulatory
```

spect

Gets/sets 802.11h spectrum management mode.

```
wl spect <mode>
```

- `0` Off
- `1` Loose interpretation of 11h spec (May join non–11h APs.)
- `2` Strict interpretation of 11h spec (May not join non–11h APs.)
- `3` Disable 11h and enable 11d
- `4` Loose interpretation of 11h+d spec (May join non-11h APs.)

fqacurcy

A manufacturing test that sets the frequency accuracy mode. This command is used to measure the center frequency of the carrier. After this mode is set, the user can use a spectrum analyzer to measure the accuracy of the frequency for a specific channel.

```
wl fqacurcy <channel>
```

The argument is the channel number 1–14, or 0 to stop the test.

longtrain

A manufacturing test that sets the longtraining mode.

```
wl longtrain <channel>
```

The argument is a band channel number, or 0 to stop the test.

freqtrack

Sets the frequency tracking mode.

```
wl freqtrack
```

- `0` Auto
- `1` On
- `2` Off

measure_req

Sends an IEEE 802.11h measurement request.

```
wl measure_req <type> <target MAC addr>
```

Measurement types

- TPC
- Basic
- CCA
- RPI

The target MAC address format is: `xx:xx:xx:xx:xx:xx`

quiet

Sends an IEEE 802.11h quiet command.

```
wl quiet <TBTTs until start>, <duration (in TUs)>, <offset (in TUs)>
```

rm_req

Requests a radio measurement of type basic, CCA, or RPI. Defines a series of measurement types and options.

```
wl rm_req cca -c 1 -d 50 cca -c 6 cca -c 11
```

Options

- `t n` Numeric token ID for measurement set or measurement
- `c n` Channel
- `d n` Duration in TUs (1024 µs)
- `p` Parallel flag, measurement starts at the same time as previous

Each measurement specified uses the same channel and duration as the previous measurement unless a new channel or duration is specified.

rm_rep

Gets the current radio measurement report.

```
wl rm_rep
```

rssidump

Dumps RSSI values from ACI scans.

```
wl rssidump
```

interference

```
wl interference
```

NON-ACPHY. Get/Set interference mitigation mode. Choices are:

- 0 = none
- 1 = non WLAN
- 2 = WLAN manual
- 3 = WLAN automatic
- 4 = WLAN automatic with noise reduction

ACPHY. Get/Set interference mitigation mode. Bit-Mask:

- 0 = desense based on glitches
- 1 = limit pktgain based on hwaci (high pwr aci)
- 2 = limit pktgain based on w2/nb (high pwr aci)
- 3 = enable preemption

So a value of 15 would enable all four.

interference_override

```
wl interference_override
```

NON-ACPHY. Get/Set interference mitigation override. Choices are:

- 0 = no interference mitigation
- 1 = non WLAN
- 2 = WLAN manual
- 3 = WLAN automatic
- 4 = WLAN automatic with noise reduction
- -1 = remove override

ACPHY. Get/Set interference mitigation mode. Bit-Mask:

- -1 = remove override, override disabled 0 = desense based on glitches
- 1 = limit pktgain based on hwaci (high pwr aci)
- 2 = limit pktgain based on w2/nb (high pwr aci)
- 3 = enable preemption

So a value of 15 would enable all four.

itfr_get_stats

Gets interference source information.

```
wl itfr_get_stats
```

itfr_enab

Gets/sets STA interference detection mode (STA only).

```
wl itfr_enab
```

Options

- `0` Disable
- `1` Enables manual detection
- `2` Enables auto detection

itfr_detect

Issues an interference detection request.

```
wl itfr_detect
```

frameburst

Disables/enables frameburst mode.

```
wl frameburst <N>
```

With frame bursting enabled, multiple frames are sent with a minimum interframe gap. This enhances network efficiency and reduces overhead. Windows STA drivers can perform frame bursting which is disabled by default.

Options:

- N: 0 disables frameburst
- 1: enables frameburst

without argument returns current FB state

rclass

Gets the regulatory operating class.

```
wl rclass
```

Country and Locale

country

Selects the country code for the region in which the driver will be operating.

For a simple country setting

```
wl country <country>
```

where `<country>` is either a long name or country code from ISO 3166. `Germany` or `DE, for example`.

For a specific built-in country definition

```
wl country <built-in> [<advertised-country>]
```

where `<built-in>` is a country code followed by a forward slash (/) and a regulatory revision number (`US/3`, for example), and where `<advertised-country>` is either a long name or country code from ISO 3166.

If `<advertised-country>` is omitted, the result is the same as the built-in country code.

Use `wl country list [band(a or b)]` for the list of supported countries.

country_ie_override

Sets/gets the country information element.

```
wl country_ie_override
```

autocountry_default

Selects the country code for use with auto country discovery.

```
wl autocountry_default
```

Wireless Distribution System

lazywds

Sets or gets lazy WDS mode (dynamically grants WDS membership to anyone).

```
wl lazywds
```

Returns/sets the value of the lazy WDS setting.

- `0` Lazy WDS is disabled. WDS partners must be set explicitly.
- `1` Lazy WDS is enabled and the AP will accept WDS partners from any MAC address.

wds

Sets or gets the list of WDS member MAC addresses.

```
wl wds
```

Set using a space-separated list of MAC addresses.

```
   wl wds xx:xx:xx:xx:xx:xx [xx:xx:xx:xx:xx:xx ...]
```

wds_type

Indicates whether the interface to which this IOVAR is sent is of a WDS or DWDS type.

```
wl wds_type -i <ifname>
```

ifname is the name of the interface to query the type.

Return values:

- 0: The interface type is neither WDS nor DWDS.
- 1: The interface is WDS type.
- 2: The interface is DWDS type.

wds_remote_mac

Gets the MAC address of the WDS link remote endpoint.

```
wl wds_remote_mac
```

wds_wpa_role_old

Gets the WPA role (old) of the WDS link local endpoint.

```
wl wds_wpa_role_old
```

wds_wpa_role

Gets/sets the WPA role of the WDS link local endpoint.

```
wl wds_spa_role>
```

Mode Controls

gmode

Sets the 54g mode.

```
wl gmode <mode>
```

```
   Mode = LegacyB | Auto | GOnly | BDeferred | Performance | LRS
```

LegacyB

- Rateset 1b, 2b, 5.5, 11
- Preamble Long
- Short slot Off

In LegacyB g-mode operation, only CCK rates are allowed in the network, and only 1 and 2 Mbps are basic, so that legacy IEEE 802.11 devices can join (older than IEEE 802.11b). In this mode, the IEEE 802.11g AP or IBSS will not include an ERP IE or an IE. This mode is intended to look as much like an early IEEE 802.11b network as possible to allow interoperability with devices that have trouble with any of the newer specification changes. The AP advertises and uses only IEEE 802.11b CCK rates. IEEE 802.11g clients can still associate but will only operate at IEEE 802.11b rates.

Summary:

- Uses IEEE 802.11 long slot timing only.
- Only 1 and 2 Mbps rates are basic rates, so that all legacy devices can join.
- For IEEE 802.11b networks only.

LRS

- Rateset: 1b, 2b, 5.5b, 11b (CCK only)
- Extended rateset: 6, 9, 12, 18, 24, 36, 48, 54
- Preamble: Long
- Shortslot: Auto

In LRS g-mode operation, all IEEE 802.11g rates are available, but only CCK rates are basic to allow IEEE 802.11 devices to join. The rateset is split, with only four rates in the supported rates IE to allow interoperability with IEEE 802.11 devices that have trouble with more than four rates in the IE (take the full 12 rates, throwing away none, and split the four CCK rates into the first rate element and put all the OFDM rates in the ESR).

Summary:

- Allows both short and long slot timings (11b and 11g).
- No IEEE 802.11 devices (legacy) can join.
- Supports devices that can handle *only* four CCK rates in the rateset.

Auto [default]

- Rateset: 1b, 2b, 5.5b, 11b, 18, 24, 36, 54
- Extended rateset: 6, 9, 12, 48
- Preamble: Long
- Short slot: Auto

This mode allows for maximum compatibility and is fully compliant with the IEEE 802.11g specification. All 12 IEEE 802.11g rates (1(b), 2(b), 5.5(b), 6, 9, 11(b), 12, 18, 24, 36, 48, and 54) will be advertised, but the basic rateset will include only 1, 2, 5.5, and 11, so that legacy IEEE 802.11b clients can associate (no-OFDM). This mode defaults to high-speed IEEE 802.11g operation by using short slot timing. The AP will switch to long slot timing if an IEEE 802.11b or an IEEE 802.11g client that does not support short slot timing enters the network.

When set to auto, the AP advertises this short slot timing in the beacon frames. As long as there are only 11g clients associated, short slot timing will be used, and throughput will be high. If a client that does not support short-slot timing (such as a legacy 11b client) joins the BSS, the AP will cease to advertise short slot. All STA devices in the BSS will then start to use the normal IEEE 802.11b interpacket timings. If all non-short-slot timing STAs leave the BSS, the AP and remaining STAs will revert to using short slot timing.

Summary:

Use of IEEE 802.11g short slot timing is automatic:

- If no IEEE 802.11b clients are associated, short slot timing will be used.
- If an IEEE 802.11b client associates, then the AP will use long slot timing.
- Because no OFDM rates are in the basic rateset, the maximum IEEE 802.11g performance cannot be achieved.

IEEE 802.11g mode performance

- Rate set: 1b, 2b, 5.5b, 6b, 9, 11b, 12b, 18, 24b, 36, 48, 54
- Preamble: Short required
- Short slot: On and required

In IEEE 802.11g mode operation, all 12 IEEE 802.11g rates are available, and the rates are *not* split. OFDM Basic rates are present, so CCK-only (IEEE 802.11b) devices cannot join.

The rates are not split, so that legacy 54g drivers can see all the rates, not just a good subset. Putting more than eight rates in the Supported Rates element is not included in the IEEE 802.11 specification or the IEEE 802.11(a, b, or g) specification, but the practice works with all Broadcom drivers.

As in the Auto setup, short preambles are required to join. In addition, short slot support is required to join, and short slot operation is always on in the network.

In the normal automatic setup, short slot operation attention is given to overlapping BSSs, but in this mode it is not.

Summary:

- Use of IEEE 802.11g short slot timing is mandated.
- Use of IEEE 802.11g short preamble is mandated.
- IEEE 802.11b clients cannot associate because of the above two mandates.
- Designed to use the maximum bandwidth only on an IEEE 802.11g network topology.

wet

Gets/sets the wireless Ethernet bridging mode.

```
wl wet
```

Default wet is OFF:

```
wl wet on
wl wet
```

wet is 1 (ON).

To change back to the default OFF state:

```
wl wet off
wl wet
```

wet is 0 (OFF).

Note: Older IEEE 802.11 devices (supporting only 1 and 2 Mbps) cannot join in this mode.

gmode Protection Controls

gmode_protection

Gets gmode protection.

```
wl gmode_protection <option>
```

- `0` Disabled
- `1` Enabled

This command is a query command used to determine whether protection mechanisms are currently being used.

- `0` Protection mechanisms are not currently being used.
- `1` Protection mechanisms are currently being used.

The IEEE 802.11g standard uses orthogonal frequency division multiplexing (OFDM) to attain its high data speed. To protect IEEE 802.11b users, IEEE 802.11g is required to also send a protection signal based on the longer complementary code keying (CCK). Omitting the protection signal ensures high data speeds for IEEE 802.11g users at the cost of locking out IEEE 802.11b users.

Without such protection, an IEEE 802.11b user would be blocked by an invisible flow of IEEE 802.11g data and would assume that the wireless network had crashed.

Features:

- As required for Wi-Fi compliance, protection mechanisms are enabled automatically whenever an IEEE 802.11b STA joins the BSS.
- If no IEEE 802.11b STA joins the BSS, then no protection mechanism will be used, and full IEEE 802.11g performance will be attained.
- The default protection mechanism is not CTS-to-self.
- Typing this command without any parameters displays the following string on the console:
  `gmode_protection is 0 (off)`
- Mixed environments (ERP and Legacy PHYs [IEEE 802.11, IEEE 802.11b, and IEEE 802.11g coexisting]) require a protection mechanism.
- ERP ONLY STAs use a short-slot time to improve performance.
- The IEEE 802.11g specification defines a gmode protection mechanism as one that involves prefixing each OFDM IEEE 802 data frame with an RTS/CTS CCK frame sequence. The duration fields of the RTS and CTS frames should allow the IEEE 802.11b node to correctly set its NAV and avoid collisions with the subsequent OFDM frames.
- In accordance with the specification, the data frames should be sent at one of the basic rates, but with a CCK-only basic rateset.
- STAs are expected to automatically honor the bit announced in BSS beacons and should require no configuration.

gmode_protection_control

Gets/sets the 11g protection mode control algorithm.

```
wl gmode_protection_control <option>
```

- `0` Always off.
- `1` Monitor local association.
- `2` Monitor overlapping BSS.

gmode_protection_override

Gets/sets 11g protection mode override.

```
wl gmode_protection_override <option>
```

- `-1` AUTO. Protection will automatically be used if either an IEEE 802.11b STA associates to the AP, or the AP detects another legacy IEEE 802.11b BSS.
- `0` OFF. Turns off protection on the 54g AP. Protection mechanisms will *never* be used.
- `1` ON. Turns on protection on the 54g AP. Protection mechanisms will *always* be used.

The driver default is `g_protection_override == AUTO`, but can be set to `0/1` to force protection off/on. Typing this command without any parameters displays the following string on the console.

```
gmode_protection_override is -1
```

This indicates that the override is set for auto.

There are three modes:

- No protection (default in 54g). This mode is configured with `wl gmode_protection_override 0`.
- RTS/CTS, when legacy IEEE 802.11b STA is associated. This mode is configured with:

```
wl ignore_bcns TRUE (default)
wl gmode_protection_override -1 (AUTO /default)
```

- RTS/CTS, when overlapping legacy IEEE 802.11b operation is detected (`ignore_bcns=FALSE` may be required for Wi-Fi compliance for IEEE 802.11g devices). This mode is configured with:

```
wl ignore_bcns FALSE
wl gmode_protection_override -1 (AUTO /default)
```

legacy_erp

Gets/sets 11g legacy ERP inclusion.

```
wl legacy_erp
```

- `0` Disable
- `1` Enable

The command gets/sets the legacy ERP inclusion flag for the driver for NonERP element advertisement. If set, include legacy ERP information element ID 47 along with IEEE 802.11g information element ID 42. The beacon sender shall set b0 (NonERP_present) and b1 (use_protection) for the use of this element. An ERP STA that is aware of a non-ERP STA shall set bit0 of NonERP information element true and transmit this information in a subsequent beacon frame.

Radar Controls

radar

Enables/disables radar.

```
wl radar
```

radarargs

Gets/sets radar parameters in order as blank, fmdemodcfg, npulses_lp, min_pw_lp, max_pw_lp, min_fm_lp, max_span_lp, min_deltat, max_deltat, autocorr, st_level_time, t2_min, fra_pulse_err, npulses_fra, npulses_stg2, npulses_stg3, percal_mask, quant, min_burst_intv_lp, max_burst_intv_lp, nskip_rst_lp, max_pw_tol, feature_mask.

```
wl radarargs
```

WME Controls

cac_addts

Adds TSPEC (an error is returned if the STA is not associated or WME (WMM) is not enabled).

```
wl cac_addts <arg>
```

- `arg` TSPEC parameter input list

cac_delts

Deletes TSPEC (an error is returned if the STA is not associated or WME (WMM) is not enabled).

```
wl cac_delts <arg>
```

- `arg` TSINFO for the target TSPEC

cac_delts_ea

Deletes TSPEC (an error is returned if the STA is not associated or WME (WMM) is not enabled).

```
wl cac_delts_ea <arg1> <arg2>
```

- `arg1` TSINFO for the target TSPEC
- `arg2` MAC address

cac_tslist

Gets the list of TSINFO in drivereg.

```
wl cac_tslist
```

cac_tslist_ea

Gets the list of TSINFO for given STA in drivereg.

```
wl cac_tslist_ea
```

cac_tspec

Gets a specific TSPEC with matching TSINFO.

```
wl cac_tspec <0xaa 0xbb 0xcc>
```

- `0xaa 0xbb 0xcc` TSINFO octets

cac_tspec_ea

Gets specific TSPEC for given STA with matching TSINFO.

```
wl cac_tspec_ea <0xaa 0xbb 0xcc> <xx:xx:xx:xx:xx:xx>
```

- `0xaa 0xbb 0xcc` TSINFO octets
- `xx:xx:xx:xx:xx:xx` MAC address

tclas_add

Adds a tclas frame classifier type entry.

```
wl tclas_add <user priority> <type> <mask>
```

- type 0 eth2: <src mac> <dst mac> <ether type>
- type 1/4 ipv4: <ver> <src> <dst> <s_port> <d_port> <dscp> <prot>
- type 2 802.1Q: <vlan tag>
- type 3 filter: <offset> <value> <mask>
- type 4 ipv6: <ver> <src> <dst> <s_port> <d_port> <dscp> <nxt_hdr> <flw_lbl>
- type 5 802.1D/Q: <802.1Q PCP> <802.1Q CFI> <802.1Q VID>

tclas_del

Deletes a tclas frame classifier type entry.

```
wl tclas_del [<idx> [<len>]]
```

tclas_list

Lists the added tclas frame classifier type entry.

```
wl tclas_list
```

wme

Sets Wireless Multimedia Extensions (WME) mode.

```
wl wme ap | sta [be|bk|vi|vo [ecwmax|ecwmin|txop|aifsn|acm <value>] ...]
```

- `0`:OFF
- `1`: ON
- `-1` AUTO

wme_apsd

Sets Automatic Power Save Delivery (APSD) mode on the AP.

```
wl wme_apsd
```

- `0`:OFF
- `1`: ON

wme_apsd_sta

Sets APSD parameters on the STA (The driver must be down.).

```
wl wme_apsd_sta <max_sp_len> <be> <bk> <vi> <vo>
```

Options

`<max_sp_len>` = number of frames per USP: `0` (all), `2`, `4`, or `6` `<xx>`

- `0`: Disable
- `1:` Enable U-APSD per AC

wme_apsd_trigger

Sets periodic APSD trigger frame timer timeout in ms (0 = OFF).

```
wl wme_apsd_trigger
```

wme_autotrigger

Enables/disables sending of APSD trigger frame when all ACs are delivery-enabled.

```
wl wme_autotrigger
```

wme_clear_counters

Clears WMM counters.

```
wl wme_clear_counters
```

wme_counters

Prints the WME statistics.

```
wl wme_counters
```

wme_dp

Sets AC queue discard policy.

```
wl wme_dp <be> <bk> <vi> <vo>
```

<xx>: value 0 for newest-first, 1 for oldest-first.

wme_maxbw_params

Sets the wme tx parameters.

```
wl wme_maxbw_params [be|bk|vi|vo <value> ....]
```

wme_tx_params

Sets the wme tx parameters.

```
wl wme_tx_params [be|bk|vi|vo [short|sfb|long|lfb|max_rate <value>] ...]
```

Information Element Controls

add_ie

Adds a vendor-proprietary information element (IE) to management packets.

```
wl add_ie <pktflag> length OUI hexdata <pktflag>
```

- Bit 0: Beacons
- Bit 1: Probe response
- Bit 2: Associate/reassociate response
- Bit 3: Authenticate response
- Bit 4: Probe request
- Bit 5: Associate/reassociate request

Example

`wl add_ie 3 10 00:90:4C 0101050c121a03` to add this IE to beacons and probe responses.

del_ie

Deletes a vendor-proprietary IE from management packets.

```
wl del_ie <pktflag> length OUI hexdata <pktflag>
```

- Bit 0: Beacons
- Bit 1: Probe response
- Bit 2: Associate/reassociate response
- Bit 3: Authenticate response
- Bit 4: Probe request
- Bit 5: Associate/reassociate request

Example

```
wl del_ie 3 10 00:90:4C 0101050c121a03
```

hs20_ie

Sets the Hotspot 2.0 IE.

```
wl hs20_ie <length> <hexdata>
```

list_ie

Dumps the list of vendor-proprietary IEs.

```
wl list_ie
```

NVRAM/SROM Write Controls

otpw

Writes an SROM image to the on-chip OTP.

```
wl otpw
```

nvotpw

Writes NVRAM to the on-chip OTP.

```
wl nvotpw file
```

legacylink

Sets the IBSS legacy link behavior.

```
wl legacy_link
```

Options

- `0` Disable
- `1` Enable

listen

Sets or queries the listen time in units of beacon interval.

```
wl listen
```

rdvar

Reads a named variable from the SROM (if cissource is present).

```
wl rdvar <variable name>
```

wrvar

Writes a named variable to the SROM (if cissource is present).

```
wl wrvar <variable name>
```

NDIS Related Commands

ndisscan

Initiates a broadcast SSID scan across all channels (no SSID argument).

```
wl ndisscan
```

ndis_frag

Gets/sets the fragmentation threshold.

```
wl ndis_frag
```

Range

[256–2346]

ndis_rts

Gets/sets the RTS threshold.

```
wl ndis_rts
```

MIMO-Specific Commands

actframe

It is assumed that the action frame is built by the application, so this command does not do any formatting. The only parameter besides the formatted action frame is the target MAC address.

```
wl actframe targetmacaddr data
```

Options

- `targetmacaddr` Target MAC address
- `data` Formatted action frame

Example

```
wl actframe 00:22:68:94:E4:D4 1AF456D32B
```

ampdu_activate_test

Activates AMPDU test.

```
wl ampdu_activate_test
```

ampdu_cfg_txaggr

Enables/disables tx aggregation per all category tid's and per category tid's for specified interface.

```
wl ampdu_cfg_txaggr <0/1>
wl ampdu_cfg_txaggr [<tid> <0/1>]
```

Example

Calling the command without any arguments gets the current values.

```
# wl ampdu_cfg_txaggr
tid:0 status:1
tid:1 status:1
tid:2 status:1
tid:3 status:1
tid:4 status:1
tid:6 status:1
tid:7 status:1
```

ampdu_tid

Enables/disables PER-TID AMPDU. Enables/disables AMPDU on a per-traffic identified (TID) basis.

```
wl ampdu_tid <tid> [0/1]
```

ampdu_retry_limit_tid

Sets PER-TID AMPDU retry limit.

```
wl ampdu_retry_limit_tid <tid> [0-31]
```

ampdu_rr_retry_limit_tid

Sets PER-TID AMPDU regular rate retry limit.

```
wl ampdu_rr_retry_limit_tid <tid> [0-31]
```

ampdu_send_addba

Sends Add Block Acknowledgement (ADDBA) to specified EA-TID.

```
wl ampdu_send_addba <tid> <ea>
```

ampdu_send_delba

Sends Delete Block Acknowledgement (DELBA) to specified EA-TID.

```
wl ampdu_send_delba <tid> <ea> [initiator]
```

ampdu_txq_prof_start

Starts sample transmit queue profiling data.

```
wl ampdu_txq_prof_start
```

ampdu_txq_prof_dump

Shows transmit queue histogram.

```
wl ampdu_txq_prof_dump
```

ampdu_txq_ss

Takes transmit queue snapshot.

```
wl ampdu_txq_ss
```

amsdu

Enables/disables AMSDU. Disable the driver with WL down, issue the command, and then enable the driver with WL up.

```
wl amsdu
```

- `0` Disable
- `1` Enable

bw_cap

Gets/sets the per-band bandwidth.

```
wl bw_cap <2g | 5g> [<cap>]
```

`2g|5g`Band: 2.4 GHz or 5 GHz, respectively

`cap`:

- `0x1` 20 MHz
- `0x3` 20/40 MHz
- `0x7` 20/40/80 MHz
- `0xff` Unrestricted

chanspec

Sets the channel.

```
wl chanspec <channel> [ab][n][ul]
```

Options

- `<channel>` Channel number (0–224)
- `a` 5G band
- `b` 2G band (default is 2G if channel ≤14)
- `n` Bandwidth 10 (none for 20 and 40)
- `l` Lower ctl sideband
- `u` Upper ctl sideband

The channel can also be set with the legacy format:

- c Channel number (0–224)
- b Band (5[a] or 2[b/g])
- w Bandwidth 10, 20, or 40
- s Ctl sideband: –1 = lower, 0 = none, 1 = upper

dfs_channel_forced

Sets the forced channel and sets the channel list.

```
Set <channel>[band][bandwidth][control sideband]
```

- `<channel>` Channel number (0–224)
- `a` 5G band
- `b` 2G band (default is 2G if channel ≤14)
- `n` Bandwidth 10 (none for 20 and 40)
- `l` Lower ctl sideband
- `u` Upper ctl sideband

Set the channel list using the -l option.

```
wl dfs_channel_forced -l <chanspec list> | 0
```

- wl dfs_channel_forced -l <chanspec list> | 0
- 20 MHz: <channel>[/20]
- 40 MHz: <channel>l|u|/40
- 80 MHz: <channel>/80

Channels specified using “-l” option should be separated by “,” or “/” and should be prefixed with +/-. The existing configuration is deleted when 0 is specified.

cur_mcsset

Gets the current modulation coding scheme set (MCSSET) set, if associated, else default MCSSET.

```
wl cur_mcsset
```

Example return

```
MCS SET: [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15]
```

The modulation coding scheme includes variables such as the number of spatial streams, modulation, and the data rate on each stream. Radios establishing and maintaining a link must automatically negotiate the optimum MCS based on channel conditions and then continuously adjust the selection of MCS as conditions change due to interference, motion, fading, and other events.

The following table shows an example of how MCSs are specified.

MCS Index	*Modulation*	*Code Rate*	*N_{BPSC}[a] $_b$[b] (iSS)*	*N_{SD}[c]*	*N_{SP}[d]*	*N_{CBPS}[e]*	*N_{DPSC}[f]*	*Data Rate (Mbps)*	
								800 ns GI[g]	*400 ns GI*
0	BPSK	$^1/_2$	1	108	6	108	54	13.5	15.0
1	QPSK	$^1/_2$	2	108	6	216	108	27.0	30.0
2	QPSK	$^3/_4$	2	108	6	216	162	40.5	45.0
3	16-QAM	$^1/_2$	4	108	6	432	216	54.0	60.0
4	16-QAM	$^3/_4$	4	108	6	432	324	81.0	90.0
5	64-QAM	$^2/_3$	6	108	6	648	432	108.0	120.0
6	64-QAM	$^3/_4$	6	108	6	648	486	121.5	135.0
7	64-QAM	$^5/_6$	6	108	6	648	540	135.0	150.0

a. Number of coded bits per single carrier
b. Number of coded bits per single carrier for each spatial stream, iSS
c. Number of data subcarriers
d. Number of pilot subcarriers
e. Number of coded bits per symbol
f. Number of data bits per symbol
g. Guard interval (GI) is the time delay used by the receiver to let the reflections in the channel settle before sampling data bits.

Note: MCS indexes 0–7 default to CDD, 8–15 default to SDM.

nrate

Forces MIMO (IEEE 802.11n) rates. When set, it applies to band-specific rate_override, and when a query, it gets the moving average or band-specific rate_override (if it is on).

```
wl nrate –r [legacy rate] –m [mcs index] –s [stf mode 0=SISO, 1=CDD, 2=STBC, 3=SDM]
```

- Default AUTO

For both legacy or MCS, the `stf` field shows the STF mode (and number of streams) in use.

- `0` SISO, 1 stream
- `1` CDD, 2 streams
- `2` STBC, 2 streams
- `3` SDM, 2 streams
- `Other` Invalid

Note: For 2×2 solutions, the valid `nrate` is 1–15 or 32.

nphy_antsel

Gets/sets the antenna configuration. This command controls the antenna selection feature in the driver. It can enable/disable the various antenna selection algorithms (explained later) and/or manually select/override antenna configurations. The default is AUTO, if SROM supports three antennas. This command is primarily for internal use. For external use, it should use simple syntax to set `-1` = Auto selection.

```
wl nphy_antsel (to get values of [utx urx dtx drx])
```

Setting the antennas:

- Basic `wl nphy_antsel cfg` (for example, `-1` or `0x01`)
- Advanced `wl nphy_antsel utx urx dtx drx`

where the argument `cfg` in the basic setting controls the antenna configuration for TX and RX of frames. A value of `-1` means auto selection, and a value of `0xAB` means fixed antennas (A and B are the antenna numbers used for RF chain 0 and 1, respectively).

The four arguments/results [`utx urx dtx drx`] are as follows:

- `utx` Antenna configuration for transmission of unicast data frames. Set values can be either `-1` (auto selection, Algorithm1) or `0xAB`. The get value is the current ANTCFG selected by Algorithm1 including a flag AUTO or the user-specified TX ANTCFG override 0xAB.
- `urx` Antenna configuration for reception of unicast data frames protected by RTS/CTS. Set values can be either `-1` (auto selection, Algorithm2) or `0xAB`. The get value is the current ANTCFG selected by Algorithm2 including a flag AUTO or the user-specified RX ANTCFG override 0xAB.
- `dtx` Antenna configuration for transmission of frames that are *not* unicast data frames (also known as default tx configuration). Set values can be either `-1` (auto selection, Algorithm3) or `0xAB`. The get value is the current ANTCFG selected by Algorithm3 including a flag "AUTO" or the user-specified TX ANTCFG override 0xAB.
- `dtx` Antenna configuration for reception of all frames, except RTS/CTS protected frames (also known as default rx configuration). Set values can be either `-1` (auto selection, Algorithm3) or `0xAB`. The get value is the current ANTCFG selected by Algorithm3 including a flag AUTO or the user-specified RX ANTCFG override 0xAB.

Note: The distinction between `dtx` and `drx` is mainly for backward compatibility with 2×3 CB superswitch.

The query will also have a flag *AUTO* if auto selection is ON.

dpt_deny

Adds/removes EA to direct packet transmission (DPT) deny list usage. DPT is a mechanism through which two devices (STA) can communicate directly to each other.

```
wl dpt_deny <add remove> <ea?
```

dpt_endpoint

Creates/updates/deletes DPT endpoint for EA.

```
wl dpt_endpoint <create update delete> <ea>
```

dpt_pmk

Sets DPT preshared key.

```
wl dpt_pmk
```

dpt_fname

Sets/gets DPT friendly name.

```
wl dpt_fname
```

dpt_list

Gets status of all DPT peers.

```
wl dpt_list
```

ampdu_clear_dump

Clears AMPDU counters. This command allows users to clear AMPDU-specific statistics (WLCNT or wireless related counters), examples of which are counters for WMM and AMSDU.

```
wl ampdu_clear_dump
```

bssmax

Gets the number of BSSs.

```
wl bssmax
```

radarargs40

Gets/sets radar parameters for 40 MHz channels. Order as version, npulses, ncontig, min_pw, max_pw, thresh0, thresh1, blank, fmdemodcfg, npulses_lp, min_pw_lp, max_pw_lp, min_fm_lp, max_deltat_lp, min_deltat, max_deltat, autocorr, st_level_time, t2_min, fra_pulse_err, npulses_fra, npulses_stg2, npulses_stg3, percal_mask, quant, min_burst_intv_lp, max_burst_intv_lp, nskip_rst_lp, max_pw_tol, feature_mask.

```
wl radarargs40
```

radarthrs

Sets the radar threshold for 20 MHz and 40 MHz bandwidths. Order as thresh0_20_lo, thresh1_20_lo, thresh0_40_lo, thresh1_40_lo, thresh0_20_hi, thresh1_20_hi, thresh0_40_hi, thresh1_40_hi.

```
wl radarthrs
```

nphy_test_tssi

Sets the NPHY TSSI value.

```
wl nphy_test_tssi val
```

nphy_rssiant

Set the NPHY RSSI antenna index.

```
wl nphy_rssiant antindex (o-3)
```

mimo_ss_stf

Gets/sets SS STF mode.

```
wl mimo_ss_stf <value> -b <a | b>
```

Options

- `0` SISO
- `1` CDD
- `b a` 5G band
- `b b` 2.4G band

mimo_txbw

Gets/sets the MIMO transmit bandwidth (txbw).

```
wl mimo_txbw [option number]
```

Options:

- 2: 20 MHz (lower)
- 3: 20 MHz (upper)
- 4: 40 MHz
- 5: 40 MHz (DUP)
- 6: 80 MHz (20LL) 80MHz has 4 subbands, Lower-Lower (20 MHz)
- 7: 80 MHz (20LU) Lower-Upper (20 MHz)
- 8: 80 MHz (20UL) Upper-Lower (20 MHz)
- 9: 80 MHz (20UU) Upper-Upper (20 MHz)
- 10: 80 MHz (40L) Lower 40 MHz
- 11: 80 MHz (40U) Upper 40 MHz
- 12: 80 MHz

obss_coex_action

Sends OBSS 20/40 coexistence management action frame. At least one option must be provided.

```
wl obss_coex_action -i <0 | 1> -w <0 | 1> -c <channel list>
```

- `i` 40 MHz intolerant bit
- `w` 20 MHz width req bit
- `c` Channel list 1–14

rifs

Sets/gets the relay interframe space (RIFS) status.

```
wl rifs <1 | 0>
```

- `1` ON
- `0` OFF

rifs_advert

Sets/gets the RIFS mode advertisement status.

```
wl rifs_advert <-1 | 0>
```

- `-1` AUTO
- `0` OFF

rxmcsset

Gets the Receive MCS rateset for 802.11n devices.

```
wl rxmcsset <mcs rateset>
```

txmcsset

Gets the Transmit MCS rateset for 802.11n devices.

```
wl txmcsset <mcs rateset>
```

spatial_policy

Gets/sets spatial policy (high-throughput PHY only).

```
wl spatial_policy <-1: auto / 0: turn off / 1: turn on>
```

To control individual band/subband use

`wl spatial_policy a b c d e`

where `a` is 2.4 GHz band setting

where `b` is 5 GHz lower band setting

where `c` is 5 GHz middle band setting

where `d` is 5 GHz high band setting

where `e` is 5 GHz upper band setting

Packet Filter Related Commands

pkt_filter_add

Installs a packet filter.

```
wl pkt_filter_add <id> <polarity> <type> <offset> <bitmask> <pattern>
```

- `<id>` User specified ID (integer)
- `<type>` `0` (Pattern matching filter)
- `<type>` `1` (Magic pattern match (variable offset)
- `<type>` `2` (Extended pattern list)
- `<offset>` (type 0): Integer offset in received packets to start matching.
- `<offset>` (type 1): Integer offset matches here and anywhere later.
- `<offset>` (type 2): [<base>:]<offset>. Symbolic packet loc plus relative offset, use wl_pkt_filter_add -l for a <base> list.
- `<polarity>` Set to `1` to negate match result. `0` is default.
- `<bitmask>` Hexadecimal bitmask that indicates which bits of the pattern to match. Must be same size as the pattern. Bit 0 of bitmask corresponds to bit 0 of pattern, and so on. If bit *n* of bitmask is 0, then do not match bit *n* of the pattern with the received payload. If bit *n* of bitmask is 1, perform the match.
- `<pattern>` Hexadecimal pattern to match. Must be same size as <bitmask>. Syntax: same as bitmask, but for type 2 (pattern list), a '!' may be used to negate that pattern match (such as 0xff03). For type 2: [<base>:]<offset> <bitmask> [!]<pattern> triple may be repeated; all sub-patterns must match for the filter to match.

pkt_filter_clear_stats

Clears packet filter statistic counter values.

```
wl pkt_filter_clear_stats <id>
```

pkt_filter_delete

Uninstalls a packet filter.

```
wl pkt_filter_clear_delete <id>
```

pkt_filter_enable

Enables/disables a packet filter.

```
wl pkt_filter_enable <id> <0 | 1>
```

pkt_filter_list

Lists installed packet filters.

```
wl pkt_filter_list [val]
```

Value

- `0` Disabled filters
- `1` Enabled filters

pkt_filter_mode

Sets the packet filter match action.

```
wl pkt_filter_mode <value>
```

Value

- `0` Forwards packet on match, discards on nonmatch (default).
- `1` Discards packet on match, forwards on nonmatch.

pkt_filter_ports

Sets up additional port filters for TCP and UDP packets.

```
wl pkt_filter_ports [<port-number>]
wl pkt_filter_ports none (to clear/disable)
```

pkt_filter_stats

Retrieves packet filter statistic counter values.

```
wl pkt_filter_stats <id>
```

Statistics Related Commands

bcnlenhist

```
wl bcnlenhist [0]
```

This command dumps recent 10 beacon lengths received.

Example: After associating with the AP or after issuing `wl scan`, if user issues `wl bcnlenhist`, the result will resemble: 144 198 197 212 198 203 194 130 243 293. To clear this list, type `wl bcnlenhist 0`.

beacon_info

Returns the 802.11 management frame beacon information.

```
wl beacon_info [-f file] [-r]
```

- `f` Write beacon data to file
- `r` Raw hex dump of beacon data

cca_get_stats

Gets CCA statistics.

```
wl cca_stats [-c channel] [-s num seconds][-n]
```

Options

- `c channel` Specifies a channel (optional). The default is the current channel, 0 is all channels.
- `s num_seconds` Number of seconds (optional). Default is 10, max is 60.
- `i` Lists individual measurements in addition to the averages.
- `curband` Only recommend channels on current band.

chanim_acs_record

Gets the auto channel scan record.

```
wl chanim_acs_record
```

chanim_mode

Gets/sets channel interference measure (chanim) mode.

```
wl chanim_mode
```

Options

- `0` Disabled
- `1` Detection only
- `2` Detection and avoidance

intfer_params

Sets/Gets intfer parameters.

```
wl intfer_params period (in sec) cnt(0~4) txfail_thresh tcptxfail_thresh period=0: disable Driver
monitor txfail
```

pktq_stats

Provides packet queue (pktq) statistics. Dumps packet queue log info for [C] common, [A] AMPDU, [N] NAR or [P] power save queues. A:, N: or P are used to prefix a MAC address (a colon: separator is necessary), or else C: is used alone. The '+' option after the colon provides more detail. Up to four parameters may be used. The common queue is the default when no parameters are supplied. Use '/<PREC>' as a suffix to restrict to certain prec indices; multiple /<PREC>/<PREC>/...can be used.

Also, '//' as a suffix to the MAC address or 'C://' will enable automatic logging of <> all prec as they are seen. Full automatic operation is also possible with the shorthand 'A:' (or 'A://'), 'P:' (or 'P://') which scans through all known addresses for those parameters that take a MAC address.

```
wl pktq_stats [C:[+]]|[A:[+]|P:[+]|N:[+]<xx:xx:xx:xx:xx:xx>][/<PREC>[/<PREC>]][//]
```

Example output for common queue

```
prec    rqstd   stored  dropped  retried   rtsfail   rtrydrop   psretry  acked  utlisatn  q       Data     Phy      %air     %effcy
                                                                                          length  Mbits/s  Mbits/s           (v5)

00      0       0       0        0         0         0          0        0      0         1368    0.00     0.00     0.0      0.0
```

where:

- precedence (prec): packet precedence.
- requested (rqstd): packets requested to be stored.
- stored: packets stored.
- dropped: packets dropped because pktq per that precedence is full.
- retried: packets resent because they were not received.
- rtsfail: count of (receive to send) rts attempts that failed to receive clear to send (cts) packets.
- rtrydrop: count of send attempts retried and then dropped.
- ps_retry: packets retried prior to moving to power save mode.
- acked: count of packets sent (acknowledged) successfully.
- utlisatn: queue utilisation factor. A *high water mark* that can be used to view how much headroom of the queue size is being used.

- q length: queue length in bytes.
- Data Mbits/s: Data rate in Mbps.
- PHY Mbits/s: PHY rate in Mbps.
- %air: percentage of air time used by the packet.
- %effcy (v5): air-time efficiency of the packets [pkt air-time/total time] * 100

Note: This feature is available with conditional compile w/ PKTQ_LOG.

probe_resp_info

Returns the 802.11 management frame probe response information.

```
wl probe_resp_info [-f file] [-r]
```

- `f` Write probe response data to file.
- `r` Raw hex dump of probe response data.

delta_stats_interval

Sets/gets the delta statistics interval, in seconds (0 to disable).

```
wl delta_stats_interval
```

delta_stats

Gets the delta statistics for the last interval.

```
wl delta_stats
```

mac_rate_histo

Returns the MAC address rate history.

```
wl mac_rate_histo <mac address> <access category> <num_pkts>
```

- MAC address example `00:11:20:11:33:33`
- Access Category (AC) `0x10` for entire MAC or `0x4` for video AC for this MAC
- `num_pkts` (optional) Number of packets to average. Max 64 for AC 0x10, max 32 for AC 0x4.

manfinfo

Shows the chip package information in the OTP.

```
wl manfinfo
```

mempool

Gets the memory pool statistics.

```
wl mempool
```

pktq_stats

Dumps packet queue log information for [C] common, [A] AMPDU, or [P] power-save queue.

A: or P: are used to prefix a MAC address (a colon separator is necessary), or else C: is used alone. Up to four parameters may be given. The common queue is the default when no parameters are supplied.

```
wl pktq_stats [C:] [| A: | P: <xx:xx:xx:xx:xx:xx>]...
```

sample_collect

Optional parameters ACPHY/HTPHY/(NPHY with NREV >= 7) are:

```
-f File name to dump the sample buffer (default "sample_collect.dat")
-t Trigger condition (default now) now, good_fcs, bad_fcs, bad_plcp, crs, crs_glitch, crs_deassert
-b PreTrigger duration in us (default 10)
-a PostTrigger duration in us (default 10)
-m Sample collect mode (default 1)
     SC_MODE_0_sd_adc
     SC_MODE_1_sd_adc_5bits            1
     SC_MODE_2_cic0                    2
     SC_MODE_3_cic1                    3
     SC_MODE_4s_rx_farrow_1core        4
     SC_MODE_4m_rx_farrow              5
     SC_MODE_5_iq_comp                 6
     SC_MODE_6_dc_filt                 7
     SC_MODE_7_rx_filt                 8
     SC_MODE_8_rssi                    9
     SC_MODE_9_rssi_all                10
     SC_MODE_10_tx_farrow              11
     SC_MODE_11_gpio                   12
     SC_MODE_12_gpio_trans             13
     SC_MODE_14_spect_ana              14
     SC_MODE_5s_iq_comp                15
     SC_MODE_6s_dc_filt                16
     SC_MODE_7s_rx_filt                17
```

smfstats

Gets/clears selected management frame (SMF) statistics.

```
wl smfstats [-C num]|[--cfg=num] [auth]|[assoc]|[reassoc]|[clear]
```

- `clear` Clears the statistics.

wnm_url

Sets/gets wnm (who near me/802.11v) session information URL.

```
wl wnm_url
```

To set

```
wl wnm_url length urlstring
```

To get

```
wl wnm_url
```

wnm_tfsreq_add

Adds one tfs request element and sends a tfs request frame.

```
wl wnm_tfsreq_add <tfs_id> <tfs_action_code> <tfs_subelem_id> <send>
```

- tfs_id: a non-zero value (1 ~ 255).
- tfs_action_code bitfield: 1: delete after match, 2: notify.
- tfs_subelem_id: TFS subelement (0 for none or 1 for previous tclas_add).
- send: 0: store element, 1: send all stored elements.

wnm_dms_set

Optionally adds pending DMS descriptors (after tclas_add) and optionally register all descriptors on the AP side to enable the service (with send=1).

```
wl wnm_dms_set <send> [<user_id> [<tc_pro>]]
```

- send: 0: store descriptor, 1: sends all stored descriptors and enables DMS on AP.
- user_id: Adds new ID to assign to the created descriptors (if TCLAS added) or existing ID to enable on AP (if no TCLAS added), 0 for all descriptors.
- tc_pro: TCLAS processing element (if several TCLAS added).

wnm_dms_status

Lists all DMS descriptors and provides their internal and AP status.

```
wl wnm_dms_status
```

wnm_dms_term

Disables registered DMS descriptors on the AP side and optionally discards them.

```
wl wnm_dms_term <del> [<user_id>]
```

- del: Discards descriptors after disabling the service on the AP side.
- user_id: descriptor to disable/delete, 0 for all descriptors.

wnm_service_term

Disables service. Checks specific wnm_XXX_term for more information.

```
wl wnm_service_term <srv> <service related params>
```

- srv 1 for DMS
- srv 2 for FMS
- srv 3 for TFS

wnm_timbc_offset

Gets/Sets TIM broadcast offset by –32768 period > offset (µs) > 32768. CAUTION. Due to resource limitations, only one radio can have one set of TIMBC offset settings. MBSS must share the same setting.

```
wl wnm_timbc_offset <offset> [<tsf_present> [<fix_interval> [<rate_override>]]]
```

- Rate overrides are in units of 500k (and are a given). The max setup is 108. If set, then the override high rate is used to transmit the TIM broadcast to a high rate frame.
- The offset is in units of µs.

wnm_timbc_set

Enables/disables TIM Broadcast. Station will send appropriate request if AP supports TIMBC.

```
wl wnm_timbc_set <interval> [<flags> [<min_rate> [<max_rate>]]]
```

- Interval: Use 0 to disable the interval.
- min_rate: Minimal rate requirement, in Mbps, for TIM high rate or TIM low rate frames.
- max_rate: Maximal rate requirement in Mbps.

wnm_timbc_status

Retrieves TIM Broadcast configuration set with current AP.

```
wl wnm_timbc_status
```

wnm_maxidle

Sets up WNM BSS Max Idle Period interval and option.

```
wl wnm_maxidle <Idle Period> <Option>
        Idle Period: in unit of 1000TU (1.024s)
        Option: protected keep alive required(0 ~ 1)
```

wnm_bsstrans_query

Sends 11v BSS transition management query.

```
wl wnm_bsstrans_query [ssid]
```

wnm_bsstrans_req

Sends a BSS transition management request frame with a BSS termination included in the bit set.

```
wl wnm_bsstrans_req <reqmode> <tbtt> <dur> [unicast]
```

- reqmode: Request mode of BSS transition request.
- tbtt: time of BSS to end of life, in unit of TBTT, max to 65535.
- dur: time of BSS to keep off, in unit of minute, max to 65535.
- unicast: [1|0] unicast or broadcast to notify STA in BSS. Default in unicast.

wnm_keepalives_max_idle

Sets/Gets the number of keepalives, mkeep-alive index and max_interval configured per BSS-Idle period.

```
wl wnm_keepalives_max_idle <keepalives_per_bss_max_idle> <mkeepalive_index> [<max_interval>]
```

bss_peer_info all

Gets BSS peer info of all the peer's in the individual interface. If a non-zero MAC address is specified, gets the peer info of the PEER alone.

```
wl bss_peer_info [MAC address]
```

drift_stats_reset

Reset drift statistics.

bssload_static

Gets or sets static BSS load.

```
wl bssload_static [off | <sta_count> <chan_util> <acc>]
```

CIS Related Commands

srwrite

Writes the SROM.

```
wl srwrite <byte offset> <value>
```

srcrc

Gets the CRC for the input binary file.

```
wl srcrc [binary_file_name]
```

ciswrite

Writes a specified <file> to the SDIO/PCIe CIS source (either SROM or OTP).

```
wl ciswrite [-p|--pciecis] <file> -p|--pciecis --
```

- `pciecis` Writes OTP for PCIe full-dongle

cisupdate

Writes a hexadecimal byte stream to a specified byte offset of the CIS source (either SROM or OTP).

```
wl cisupdate <byte offset> <hex byte stream> [--preview]
```

- `preview` Review the update without committing it.

cisdump

Displays the content of the SDIO CIS source.

```
wl cisdump -b <file> <len>
```

- `b <file>` Writes raw bytes to `<file>`.
- `<len>` Optional count of bytes to display (must be an even number).

cis_source

Displays which source is used for the SDIO CIS (SDIO mode only).

```
wl cis_source
```

cisconvert

Prints the CIS tuple for given name = value pair.

```
wl cisconvert
```

Power Related Commands

lpc_params

Sets/Gets Link Power Control parameters.

```
wl powersel_params <tp_ratio_thresh> <rate_stab_thresh> <pwr_stab_thresh> <pwr_sel_exp_time>
```

maxpower

Sets the temporary MAXP2G(5G)A0(A1) value.

```
wl maxpower
```

Options

- `maxp2ga0` 0x1
  `maxp2ga1` 0x2
  `maxp5ga0` 0xff
  `maxp5ga1` 0xff
  `maxp5gla0` 0x3
  `maxp5gla1` 0x4
  `maxp5gha0` 0x5
  `maxp5gha1` 0x6

mkeep_alive

Sends a periodic keepalive packet or null data frame at the specified interval.

```
wl mkeep_alive <index 0-3> <period> <packet>
```

- `index` 0–3.
- `period` Retransmission period in milliseconds. Use an index of 0 to disable packet transmits.
- `packet` Hex packet contents to transmit. The packet contents should include the entire Ethernet packet (Ethernet header, IP header, UDP header, and UDP payload) specified in network byte order. If no packet is specified, a null data frame will be sent instead.

To send a keepalive packet every 30 seconds using an index of 1

```
wl mkeep_alive 1 30000 0x0014a54b164f000f66f45b7e08004500001e000040004011c
52a0a8830700a88302513c413c4000a00000a0d
```

pavars

Sets/gets temp PA parameters. Overrides the PA parameters after driver attach (SROM read), before the driver is up. These override values will be propagated to the hardware when the driver goes up. PA parameters in one band range (2g, 5gl, 5g, 5gh) must all present if one of them is specified in the command. Otherwise, it will be filled with `0`.

```
wl pavars
```

Example

```
wl down
wl pavars pa2gw0a0=0x1
pa2gw1a0=0x2 pa2gw2a0=0x3 ...
wl pavars
wl up
```

pm_dur

Retrieves accumulated PM duration information (Get only).

```
wl pm_dur
```

pm2_sleep_ret_ext

Gets/sets the Dynamic Fast Return To Sleep parameters.

```
wl pm2_sleep_ret_ext
```

povars

Sets/gets temperature power offset. Overrides the power offset after driver attach (SROM read), before the driver is up. These override values will be propagated to the hardware when the driver goes up. Power offsets in one band range (2g, 5gl, 5g, 5gh) must all present if one of them is specified in the command. Otherwise, it will be filled with `0`.

```
wl povars
```

Example

```
wl down
wl pavars cck2gpo=0x1 ofdm2gpo=0x2 ...
wl pavars
wl up
```

pwrstats

Get power usage statistics.

```
wl pwrstats [<type>]
```

ratetbl_ppr

Sets/gets the PPR rate table.

```
wl ratetbl_ppr
```

To get

```
wl ratetbl_ppr
```

To set

```
wl ratetbl_ppr <rate> <ppr>
```

wowl_keepalive

Sends a specified keepalive packet periodically in WOWL mode. Use an index of 0 to disable packet transmits.

```
wl wowl_keepalive <index0-1> <period> <packet>
```

- `index` 0–1.
- `period` Retransmission period in milliseconds.
- `packet` Hex packet contents to transmit. The packet contents should include the entire Ethernet packet (Ethernet header, IP header, UDP header, and UDP payload) specified in network byte order.

To send a keepalive packet every 30 seconds using an index of 1

```
wl wowl_keepalive 1 30000
0x0014a54b164f000f66f45b7e08004500001e000040004011c52a0a8830700a88302513c413c4000a00000a0d
```

wowl_wakeup_reason

Returns the pattern ID and the associated wake-up reason.

```
wl wowl_wakeup_reason
```

wowl_status

Shows the last system wake-up setting.

```
wl wowl_status [clear]
```

wowl_wakeind

Shows the last system wake-up event indications from the PCI and D11 cores.

```
wl wowl_wakeind
```

- `clear` Clears the indications.

wowl_pkt

Sends a wakeup frame to wake up a sleeping STA (station) in WAKE mode.

```
wl wowl_pkt <len> <dst ea | bcast | ucast <STA ea>> [magic [<STA ea>] | net <offset> <pattern>
<reason code> ]
```

To send bcast magic frame

```
wl wowl_pkt 102 bcast magic 00:90:4c:AA:BB:CC
```

To send ucast magic frame

```
wl wowl_pkt 102 ucast 00:90:4c:aa:bb:cc magic
```

To send a frame with L2 unicast

```
wl wowl_pkt 102 00:90:4c:aa:bb:cc net 0 0x00904caabbcc 0x03
```

Note: The offset for a NET pattern frame starts from <Dest EA> of the Ethernet frame. Thus, the destination Ethernet address will be used only when the offset is ≥ 6.

To send a EAPOL identity frame with L2 unicast

```
wl wowl_pkt 102 00:90:4c:aa:bb:cc eapid id-string
```

Manufacturing Test Commands

crsuprs

A manufacturing test that sets the carrier suppression mode. This command is used to measure the lo-leakage/ carrier. The carrier is required to be 15 dB below the peak power spectrum. The transmitter transmits a repetitive `01` data sequence with the scrambler disabled using DQPSK modulation.

```
wl crsuprs <channel>
```

The argument is channel number `1-14`, or `0` to stop the test.

keep_alive

Periodically sends a specified keep-alive packet.

```
wl keep_alive <period> <packet>
```

- `Period` Retransmission period in milliseconds. `0` to disable packet transmits.
- `Packet` Hexadecimal packet contents to transmit. The packet contents should include the entire Ethernet packet (Ethernet header, IP header, UDP header, and UDP payload) specified in network byte order.

For example, to send a keep-alive packet every 30 seconds:

```
wl keep_alive 30000
0x0014a54b164f000f66f45b7e08004500001e000040004011c52a0a8830700a88302513c413c4000a00000a0d
```

lifetime

Sets the lifetime parameter (milliseconds) for each AC.

```
wl lifetime
wl lifetime be | bk | vi | vo [<value>]
```

ota_teststop

OTA stands for over-the-air.

```
ota_teststop
```

ota_loadtest

Picks up ota_test.txt if file name is not provided.

```
ota_loadtest [filename]
```

ota_stream

```
wl ota_stream [option]
```

Options:

- start: Starts a test.
- sync: Synchronizes an ota stream.
- test_setup: Sets up test for synchtimeout (seconds) synchbreak/loop synchmac txmac rx mac.
- ota_tx chan bandwidth: contrlchan rates stf txant rxant tx_ifs tx_lennum_pkt pwrctrl start:delta:end.
- wl ota_stream ota_rx chan bandwidth: contrlchan -1 stf txant rxant tx_ifstx_len num_pkt.
- stop: Stops a test.

ota_teststatus

```
wl otateststatus [option]
```

Options:

- Default: no entry. Displays current running test details.
- n: displays test arguments for the *nth* line.

patrim

Gets PA trim option.

```
wl patrim
```

rssi_cal_freq_grp_2g

Sets/gets RSSI calibration frequency grouping.

```
wl rssi_cal_freq_grp_2g [chan_1_2,chan_3_4,...,chan_13_14]
```

Each of the variables such as - chan_1_2 is a byte. Upper nibble of this byte is for chan1 and lower for chan2. MSB of the nibble tells if the channel is used for calibration. 3 LSBs tell which group the channel falls in Set/get rssi calibration frequency grouping.

rpcalvars

Sets/gets temp RPCAL parameters. Overrides the RPCAL parameters after a driver attach (srom read) and before a driver up. These override values will be propagated to the hardware when the driver goes up. Only the RPCAL parameter specified in the command is updated, while the other parameters are untouched.

```
wl down
       wl rpcalvars rpcal2g=0x1
       wl rpcalvars
       wl up
```

send_nulldata

Sends a null frame to the specified hardware address.

```
wl send_nulldata <mac_addr>
```

sendprb

Sends a probe request.

```
wl sendprb [option]
```

Options

- -s S, --ssid S SSIDs to scan
- -b MAC, --bssid MAC Particular BSSID MAC address, xx:xx:xx:xx:xx:xx
- -d MAC, --da MAC Destination address

txcal_gainsweep

Starts a gain sweep for TX Cal.

```
wl txcal_gainsweep <xx:xx:xx:xx:xx:xx> [ipg] [len] [nframes] [gidx_start:step:gidx_stop]
```

Options:

- ipg: inter packet gap in μs.
- len: Packet length.
- nframes: Number of frames; 0 indicates continuous tx test.
- gidx_start: Starting TX gain index.
- gidx_stop: Stopping TX gain index.
- step: Step size for tx gain index increment.

txcal_gainsweep_meas

Gets TSSI/PWR measurements from last TX Cal gain sweep. Sets PWR measurements for TX Cal gain sweep.

```
wl txcal_gainsweep_meas [core p0 p1 ... p127]
```

txcal_pwr_tssi_tbl

Gets the saved consolidated TSSI/PWR table.

```
wl txcal_pwr_tssi_tbl
```

Generates consolidated TSSI/PWR table from last TX Cal Gain Sweep.

```
wl txcal_pwr_tssi_tbl <core> <Ps> <N> <Ch>
```

- Ps: Starting power in 6.3 format.
  - N: Number of entries in table covering the power range (Ps: (Ps+N-1): si_tbl <core> <Ps> <N> <Ch>
- Ps: Starting power in 6.3 format.
  - N: Number of entries in the table covering the power range (Ps: (Ps+N-1))

wci2_config

Gets/sets LTE coexistence MWS signaling configuration.

```
wl wci2_config <rxassert_off> <rxassert_jit> <rxdeassert_off> <rxdeassert_jit> <txassert_off>
<txassert_jit> <txdeassert_off> <txdeassert_jit> <patassert_off> <patassert_jit> <inactassert_off>
<inactassert_jit> <scanfreqassert_off> <scanfreqassert_jit> <priassert_off_req>
```

mws_params

Gets/sets LTE coexistence MWS channel parameters.

```
wl mws_params <rx_center_freq> <tx_center_freq> <rx_channel_bw> <tx_channel_bw> <channel_en>
<channel_type>
```

mws_debug_msg

Get/sets LTE coexistence BT-SIG messages.

```
wl mws_debug_msg <Message> <Interval 20us-32000us> <Repeats>
```

Sets the consolidated TSSI/PWR table.

```
wl txcal_pwr_tssi_tbl <core> <Ps> <N> <Ch> <Tssi_Ps Tssi_Ps+1.. Tssi_Ps+N-1>
```

- Ps: Starting power in 6.3 format.
  - N: Number of entries in the table covering the power range (Ps: (Ps+N-1)).
- Ch: Channel number.
- Tssi_X: Adjusted TSSI corresponding to power X.

External Log Commands

assertlog

Gets external assert logs.

```
wl assertlog>
```

Address Resolution Protocol Commands

arpoe

Enables/disables Address Resolution Protocol (ARP) agent offload feature.

```
wl arpoe
```

arp_hostip

Adds or displays a host IP address.

```
wl arp_hostip
```

arp_hostip_clear

Clears all host IP addresses.

```
wl arp_hostip_clear
```

arp_ol

Gets/sets ARP offload components.

```
wl arp_ol
```

arp_peerage

Gets/sets age of the ARP entry, in minutes.

```
wl arp_peerage
```

arp_stats

Displays ARP off-load statistics.

```
wl arp_stats
```

arp_stats_clear

Clears ARP off-load statistics.

```
wl arp_stats_clear
```

arp_table_clear

Clears the ARP cache.

```
wl arp_table_clear
```

Duration Information Commands

mpc_dur

Retrieves the accumulated MPC duration information in ms (GET) or clear accumulator (SET).

```
wl mpc_dur <any-number-to-clear>
```

pm_dur

Retrieves the accumulated PM duration information (GET) or clear accumulator (SET).

```
wl pm_dur <any-number-to-clear>
```

TPC Commands

tpc_lm

Gets the current link margins.

```
wl tpc_lm
```

tpc_mode

Enables/disables the AP TPC.

```
wl tpc_mode
```

Options

- `0` Disable
- `1` BSS power control
- `2` AP power control
- `3` Both `1` and `2`

tpc_period

Sets the AP TPC periodicity, in seconds.

```
wl tpc_period
```

Link Quality Commands

monitor_lq

Starts/stops monitoring the link quality metrics (RSSI and SNR).

```
wl monitor_lq
```

Options

- `0` OFF
- `1` ON

monitor_lq_status

Returns averaged link quality metrics (RSSI and SNR values).

```
wl monitor_lq_status
```

Offload Commands

toe

Enables/disables TCPIP offload feature.

```
wl toe
```

toe_ol

Gets/sets TCPIP offload components.

```
wl toe_ol
```

toe_stats

Displays checksum offload statistics.

```
wl toe_stats
```

toe_stats_clear

Clears checksum offload statistics.

```
wl toe_stats_clear
```

OTP Commands

One time programmable (OTP) commands are described in this section.

otpdump

Dumps the raw OTP.

```
wl otpdump
```

otpstat

Dumps the OTP status.

```
wl otpstat
```

Batch Sequence Commands

seq_start

Initiates command batching sequence. Subsequent IOCTLs are queued until `seq_stop` is received.

```
wl seq_start
```

seq_stop

Defines the end of command batching sequence. Queued IOCTLs are executed.

```
wl seq_stop
```

seq_delay

Driver should spin for the indicated amount of time. Only valid within the context of batched commands.

```
wl seq_delay
```

seq_error_index

Retrieves the index (starting at 1) of the command that failed within a batch.

```
wl seq_error_index
```

BT Coexistence Commands

bt_regs_read

Reads the Bluetooth register usage.

```
wl bt_regs_read [option]
```

Options:

- start_addr: Defines the start address for the register read.
- size: Defines the register read size in bytes.

btc_params

Gets/sets the BT coexistence parameters.

```
wl btc_params
```

btc_flags

Gets/sets the BT coexistence flags.

```
wl btc_flags
```

btcx_clear_dump

Clears btcoex debug counters.

```
wl btcx_clear_dump
```

LED Commands

ledbh

Gets/sets LED behavior.

```
wl ledbh <0-3> <0-15>
```

led_blink_sync

Gets/sets LED blink sync behavior.

```
wl led_blink_sync <0-3> <0 | 1>
```

Miscellaneous Commands

aibss_bcn_force_config

Gets/sets AIBSS beacon force configuration.

```
wl aibss_bcn_force_config <initial_min_bcn_dur,min_bcn_dur,initial_bcn_flood_dur>
```

aibss_txfail_config

Set/Get txfail configuration for bcn_timeout and max tx retries.

```
wl aibss_txfail_config [bcn_timeout, max_retry]
```

antgain

Sets the temporary AG0/1 value.

```
wl antgain
```

Options

- `ag0` 0x1
- `ag1` 0x2

ap_isolate

Gets/sets AP isolation.

```
wl ap_isolate
```

assert_type

Gets/sets the assert_bypass flag.

```
wl assert_type
```

Options

- `0` OFF
- `1` ON

atten

Sets the transmit attenuation for B band.

```
wl atten <Args>
```

Argument: `bb radio txctl1`

`Auto` = Revert to automatic control.

`Manual` = Suspend automatic control.

bmac_reboot

Reboots BMAC.

```
wl bmac_reboot
```

bmon_bssid

Sets the monitored BSSID.

```
wl bmon_bssid xx:xx:xx:xx:xx:xx 0|1
```

chanim_state

Gets channel interference state.

```
wl chanim_state <channel>
```

- Valid channels 1–14
- Returns
  - 0: Acceptable
  - 1: Severe

devpath

Prints the device path.

```
wl devpath
```

eventing

Sets/gets hex filter bitmask for MAC event reporting up to the application layer.

```
wl eventing
```

event_msgs

Sets/gets hex filter bitmask for MAC event reporting via packet indications.

```
wl event_msgs
```

event_msgs_ext

Sets/gets bit arbitrary size hex filter bitmask for MAC.

```
wl event_msgs_ext
```

fem

Sets temp FEM 2G/5G value.

```
wl fem
```

Options

- `tssipos2g` 0x1
  `extpagain2g` 0x2
  `pdetrange2g` 0x1
  `triso2g` 0x1
  `antswctl2g` 0
  `tssipos5g` 0x1
  `extpagain5g` 0x2
  `pdetrange5g` 0x1
  `triso5g` 0x1
  `antswctl5g` 0

gpioout

Sets any GPIO pins to any value (use with caution, as GPIOs would normally be assigned to chipcommon).

```
wl gpiomask gpioval
```

ioctl_echo

Checks ioctl functionality.

```
wl ioctl_echo
```

iov

Gets information on driver IOVARs.

```
wl iov
```

iovars

Gets information on driver IOVARs.

```
wl iovars
```

monitor_promisc_level

Sets a bitmap of different MAC promiscuous levels in the monitor mode.

```
wl monitor_promisc_level [<bitmap> | <+|-name>]
```

The bitmap values and corresponding name are:

Args:

- bit:0: promisc: When set, address filter accepts all received frames. When cleared, the address filter accepts only those frames that match the BSSID or local MAC address.
- bit:1: ctrl: When set, the RX filter accepts all received control frames that are accepted by the address filter. When cleared, the RX filter rejects all control frames other than PS poll frames.
- bit:3: fcs: When set, the RX filter forwards received frames with FCS errors to the driver.When cleared, frames with FCS errors are discarded.

  Example: wl monitor_promisc_level +promisc

  Example: wl monitor_promisc_level 0x2

  Example: wl monitor_promisc_level 0

ns_hostip

Adds an ns-ip address or display.

If a host IP address is given, add it to the host-cache, such as:

- wl ns_hostip fe00:0:0:0:0:290:1fc0:18c0.

If no address is given, dump all the addresses.

```
wl ns_hostip
```

ns_hostip_clear

Clears all ns-ip addresses.

```
wl ns_hostip_clear
```

otpraw

Read/Write raw data to on-chip otp.

```
wl otpraw <offset> <bits> [<data>]
```

probresp_mac_filter

Sets/Gets MAC filter-based probe response mode.

```
wl probresp_mac_filter
```

probresp_sw

Enables/disables the MAC filter-based probe response. The MAC filter-based software probe response is an AP feature that uses the MAC filter list to determine the *send* destination MAC for the probe response.

```
wl probresp_sw
```

Note: The probresp_sw command must be enabled separately to use this feature. WLPROBRESP_SW=1 must be set in WL configuration file during build.

Options:

- 0: Disables MAC filter-based probe response mode.
- 1: Enables MAC filter-based probe response mode.
- No parameter: Returns the current setting.

rmc_ackmac

Sets/Gets ACK required multicast mac address.

```
wl rmc_ackmac -i [index] -t [multicast mac address]
```

rmc_ackreq

Sets/gets ACK rmc_mode.

0 disable, 1 enable transmitter, 2 enable initiator.

```
wl rmc_ackreq [mode]
```

rmc_actf_time

Sets/gets mcast action frame tx time period in ms.

```
wl rmc_actf_time [value]
```

rmc_ar_timeout

Sets/gets rmc active receiver timeout in ms.

```
wl rmc_ar_timeout [duration in ms]
```

rmc_rssi_thresh

Sets/gets the minimum RSSI required for a station to be an active receiver.

```
wl rmc_rssi_thresh [value]
```

rmc_stats

Displays/clears reliable multicast client statistical counters.

```
wl rmc_stats [arg]
```

rmc_rssi_delta

Displays/sets RSSI delta to switch receive leader.

```
wl rmc_rssi_delta [arg]
```

rmc_vsie

Displays/set vendor specific IE contents.

```
wl rmc_vsie [OUI] [Data]
```

send_frame

Sends the ethernet frame provided in hex.

```
wl send_frame
```

sr_verify

Performs a string sanity check.

```
wl sr_verify
```

staprio

Sets/Gets the STA priority.

```
wl staprio <xx:xx:xx:xx:xx:xx> <prio><prio>: 0~3
```

sta_monitor

```
wl sta_monitor [<add/del> <xx:xx:xx:xx:xx:xx>]
```

taf

Traffic airtime fairness.

```
wl taf <MAC> [<scheduler_id> [<priority>]] +-
wl taf <scheduler_id> [coeff [<coeff>]|dump|list]
wl taf enable [0|1]|order [0|1]|bypass [0|1]|high [<val>]|low [<val>]|force [<val>]|list
```

wlc_ver

Returns wlc interface version.

```
wl wlc_ver
```

obss_prot

Gets/sets OBSS protection.

```
wl obss_prot [option]
```

Options:

- -1: auto
- 0: disable

- 1: enable

dump_obss

```
wl dump_obss [-d num msecs] to begin measurement
wl dump_obss to query for the measurement results
```

ibss_route_tbl

Gets/set the ibss route table.

```
wl ibss_route_tbl num_entries [{ip_addr1, mac_addr1}, ...]
```

ip_route_table

Gets/Sets IP route table.

```
wl ip_route_tbl num_entries [{ip_addr1, mac_addr1}, ...]
```

uartparams

Sets UART baud rate for different UART interfaces.

Usage (Get):

```
wl uartparams [-i interface] -i interface: Optional, 0- UART0, 1- UART1 etc. Default = UART0
```

Usage (Set): 1

```
wl uartparams [baudrate]
```

Usage (Set): 2.

```
wl uartparams [-i interface][-b baudrate]
         -i interface: Optional, 0- UART0, 1- UART1 etc. Default = UART0
         -b baud rate
```

PHY Related Commands

phy_adj_tssi

Reads last adjusted tssi.

```
wl phy_adj_tssi core
```

phy_read_estpwrlut

Reads the estimated power lookup table (estpwrlut) for a specified core.

```
wl phy_read_estpwrlut core
```

phy_afeoverride

Gets/Sets AFE override

```
wl phy_afeoverride
```

phy_rxiqest

Gets the PHY RX IQ noise in dBm.

```
wl phy_rxiqest <args>
```

- s # of samples (2^n)
- a Antenna select, 0,1, or 3
- r Resolution select, 0 (coarse) or 1 (fine)
- f LPF HPC override select, 0 (HPC unchanged) or 1 (overridden to ltrn mode)
- w Digital LPF override select, 0 (LPF unchanged) or 1 (overridden to ltrn_lpf mode) or 2 (bypass)
- g Gain-correction select, 0 (disable) or 1 (enable)
- e Extra INIT gain in dB on top of default. Valid values = {0, 3, 6, .., 21, 24}

phy_txiqcc

Sets/gets the IQCC a b values.

```
wl phy_txiqcc <a b>
```

phy_txlocc

Sets/gets the LOCC values.

```
wl phy_txlocc <di dq ei eq fi fq>
```

phy_rssi_gain_delta_2gb0

Sets/gets RSSI gain delta values.

```
wl phy_rssi_gain_delta_2gb0 [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rssi_gain_delta_2gb1

Sets/gets RSSI gain delta values.

```
wl phy_rssi_gain_delta_2gb1 [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rssi_gain_delta_2gb2

Sets/gets RSSI gain delta values.

```
phy_rssi_gain_delta_2gb2 [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rssi_gain_delta_2gb3

Sets/gets RSSI gain delta values.

```
phy_rssi_gain_delta_2gb3 [val0 val1 ....]
```

The number of arguments can be:
- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rssi_gain_delta_2gb4

Sets/gets RSSI gain delta values.

```
phy_rssi_gain_delta_2gb4 [val0 val1 ....]
```

The number of arguments can be:
- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rssi_gain_delta_2g

Sets/gets RSSI gain delta values.

```
phy_rssi_gain_delta_2g [val0 val1 ....]
```

The number of arguments can be:
- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rssi_gain_delta_5gl

Sets/gets RSSI gain delta values.

```
phy_rssi_gain_delta_5gl [val0 val1 ....]
```

The number of arguments can be:
- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rssi_gain_delta_5gml

Sets/gets RSSI gain delta values.

```
phy_rssi_gain_delta_5gml [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rssi_gain_delta_5gmu

Sets/gets RSSI gain delta values.

```
phy_rssi_gain_delta_5gmu [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rssi_gain_delta_5gh

Sets/gets RSSI gain delta values.

```
phy_rssi_gain_delta_5gh [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rxgainerr_2g

Sets/gets RX gain error values.

```
phy_rxgainerr_2g [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rxgainerr_5gl

Sets/gets RX gain error values.

```
phy_rxgainerr_5gl [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rxgainerr_5gm

Sets/gets RX gain error values.

```
phy_rxgainerr_5gm [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rxgainerr_5gh

Sets/gets RX gain error values.

```
phy_rxgainerr_5gh [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_rxgainerr_5gu

Sets/gets RX gain error values.

```
phy_rxgainerr_5gu [val0 val1 ....]
```

The number of arguments can be:

- 8 for single core (BCM4345 and BCM4350).
- 9 by specifying core_num, followed by 8 arguments (BCM4345 and BCM4350).
- 16 for both cores (BCM4350).

phy_test_idletssi

Gets idletssi for the given core.

```
wl phy_test_idletssi corenum
```

phy_debug_cmd

General purpose command for PHY debugging. The PHY phy function call can be added to wlc_phy_cmn.c.

```
wl phy_debug_cmd <int32 var> [option]
```

Options:

- -s # of samples (2^n)
- -a antenna select, 0,1 or 3
- -r resolution select, 0 (coarse) or 1 (fine)
- -f lpf hpc override select, 0 (hpc unchanged) or 1 (overridden to ltrn mode)
- -w dig lpf override select, 0 (lpf unchanged) or 1 (overridden to ltrn_lpf mode) or 2 bypass
- -g gain-correction select, 0 (disable), 1 (enable full correction) 2 (enable temperature correction) or 3 (verify rssi_gain_delta)
- -e extra INITgain in dB on top of default. Valid values = {0, 3, 6, .., 21, 24}
- -i gain mode select, 0 (default gain), 1 (init gain) or 4 (clip LO gain)

protection_control

Gets/sets protection mode control algorithm.

```
wl protection_control <mode>
```

- `0` Always off.
- `1` Monitors local association.
- `2` Monitors overlapping BSS.

rrm_nbr_req

Sends an 11k neighbor report measurement request.

```
wl rrm_nbr_req [ssid]
```

srchmem

Gets/sets ucode search engine memory.

```
wl srchmem
```

tsf

Gets/sets the TSF register.

```
wl tsf [<high> <low>]
```

txcore

Gets the txcore.

```
wl txcore -k <CCK core mask> -o <OFDM core mask> -s <1–4> -c <core bitmap>
```

- `k` CCK core mask
- `o` OFDM core mask
- `s` Number of space-time-streams (1–4)
- `c` Active core (bitmask) to be used when transmitting frames

txcore_override

Gets the user override of txcore.

```
wl txcore_override
```

txfifo_sz

Sets/gets the Tx FIFO size.

```
wl txfifo_sz <fifonum> <size_in_bytes>
```

wowl_pkt

Sends a wake-up frame to wake up a sleeping STA in wake mode.

```
wl wowl_pkt <len> <dst ea | bcast | ucast <STA ea>>[ magic [<STA ea>] | net <offset> <pattern>]
```

Examples:

To send bcast magic frame:

```
wl wowl_pkt 102 bcast magic 00:90:4c:AA:BB:CC
```

To send ucast magic frame:

```
wl wowl_pkt 102 ucast 00:90:4c:aa:bb:cc magic
```

To send a frame with L2 unicast:

```
wl wowl_pkt 102 00:90:4c:aa:bb:cc net 0 0x00904caabbcc
```

Note: Because the offset for net pattern frame starts from dest EA of the Ethernet frame, dest EA is used only when the offset is ≥ 6.

To send an eapol identity frame with L2 unicast:

```
wl wowl_pkt 102 00:90:4c:aa:bb:cc eapid id-string
```

wowl_ext_magic

Sets the 6-byte extended magic pattern.

```
wl wowl_ext_magic 0x112233445566
```

Host Offload Commands

The following commands are part of Broadcom 802.11 host offload module.

ol_clr

Provides a list of various suboptions.

```
wl ol_clr
```

ol_cons

Displays the ARM console or issues a command to the ARM console.

```
wl ol_cons [<cmd>]
“?” Displays the list of active console commands
```

ol_eventlog

Provides a suboption list to list various suboptions.

```
wl ol_eventlog
```

ol_stats

Provides a suboption list to list various suboptions.

```
wl ol_stats
```

ol_wowl_cons

Provides a list of various suboptions.

```
wl ol_wowl_cons
```

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Phone: 949-926-5000
Fax: 949-926-5203
E-mail: info@broadcom.com
Web: www.broadcom.com

80211-TI305-R October 16, 2014