# Application Note for RTL819X System




No. 2, Innovation Road II, Hsinchu Science Park,
Hsinchu 300, Taiwan
www.realtek.com

# Change History

| Version | Date | Remarks |
|---|---|---|
| 1.0 | 2009/11/17 | Initial Release |
| 1.1 | 2010/1/14 | Add comments |
| 1.2 | 2010/1/29 | Add mib of WAPI |
| 1.3 | 2010/2/24 | Add new configuration API support |
| 1.4 | 2010/4/7 | Add mib<br>Add configuration file support |
| 1.5 | 2010/5/4 | Correct explanation of mib |
| 1.6 | 2010/5/31 | Add mib of manual WMM |
| 1.7 | 2011/3/14 | Add dual-band configuration & DFS |
| 1.8 | 2011/7/27 | Add multiple AP profile support |
| 1.9 | 2011/9/14 | Add comments for proc/stats |
| 1.10 | 2011/9/29 | Add LED type |
| 1.11 | 2011/10/19 | Add mib for band priority and special channel plan |
| 1.12 | 2012/3/1 | Add support for 8188RE |
| 1.13 | 2012/5/2 | Correct aggregation mib |
| 1.14 | 2012/5/10 | Delete the description for band priority and special channel plan. Add description for multiple profile. |
| 1.15 | 2012/5/30 | Modify comments for mibs “led_type” and “wifi_specific” |
| 1.16 | 2012/7/31 | Add 2.4G channel plan table. Add new regdomain value (14,15) for global and world-wide. |
| 1.17 | 2012/9/10 | Add mib for 802.11ac |
| 1.18 | 2012/9/26 | Add mib for VHT rate classify |
| 1.19 | 2012/11/21 | Modify the usage of deny_legacy |
| 1.20 | 2012/12/19 | Add mib of disable_ch1213 |
| 1.21 | 2012/12/28 | Add mib of pa type |
| 1.22 | 2013/1/8 | Add LED type<br>Add limitation for 8812E |
| 1.23 | 2013/4/26 | Add adaptivity test related mib |
| 1.24 | 2013/5/10 | Modify LED type for 8812E and 8192ER |
| 1.25 | 2013/9/27 | Modify LED setting for 8812E |
| 1.26 | 2013/11/21 | Correct default values for WMM mibs |
| 2.0 | 2014/02/17 | Initial Release |
| 2.1 | 2014/03/04 | Add mib for disable 92E ldpc and degrade 3dB in 1T rates when both tx2path and tx power limit enable. |

| | | |
|---|---|---|
| 2.2 | 2014/03/31 | Add missing mibs<br>Update description of mibs pa_type, ther, and xcap<br>Update Russian 5G channel |
| 2.3 | 2014/06/03 | Update channel table. Deprecate region domain SPAIN, FRANCE, MKK1, MKK2, and MKK3. |
| 2.4 | 2014/06/24 | Add section 5.3 to describe mix security feature for client mode<br>Remove MIB “phyBandSelect” |
| 2.5 | 2014/09/24 | Modify comment of mib “ldpc_92e” |

# Table of Contents

# 1. Introduction

This document introduces the usage and system architecture of Realtek Wi-Fi driver and all related software. This section includes the current supported Wi-Fi features, the 2.4GHz and 5GHz channel plan. The goal of this document is to help software developers to port Realtek WiFi to their target platform more effectively.

## 1.1. Feature Lists

- 802.11 a/b/g/n/ac compatible
- AP mode and client mode support
- Security support 64/128 bits WEP, WPA, and WPA2 (TKIP and AES-CCMP)
- Auto rate adaptive
- Wireless MAC address filter
- Broadcast SSID control
- IAPP (802.11f) support
- Auto channel selection
- Driver based MP functions
- WDS function support
- Universal repeater mode support
- WMM supported for AP mode
- Support WLAN ASIC of 8192CE, 8188RE, 8192DE, 8188ER, 8812E, and 8192ER
- WPS function support
- WAPI function support
- Set WMM parameters manually

# 1.2. Channel Plan

## 1.2.1. 2.4G Channel Plan

| regulation domain (mib *regdomain* value) | supported channels |
|---|---|
| FCC (1) | 1,2,3,4,5,6,7,8,9,10,11 |
| IC (2) | 1,2,3,4,5,6,7,8,9,10,11 |
| ETSI (3) | 1,2,3,4,5,6,7,8,9,10,11,12,13 |
| SPAIN (4) (deprecated) | 1,2,3,4,5,6,7,8,9,10,11,12,13 |
| FRANCE (5) (deprecated) | 10,11,12,13 |
| MKK (6) | 1,2,3,4,5,6,7,8,9,10,11,12,13,14 |
| ISREAL (7) | 3,4,5,6,7,8,9,10,11,12,13 |
| MKK1 (8) (deprecated) | 1,2,3,4,5,6,7,8,9,10,11,12,13,14 |
| MKK2 (9) (deprecated) | 1,2,3,4,5,6,7,8,9,10,11,12,13,14 |
| MKK3 (10) (deprecated) | 1,2,3,4,5,6,7,8,9,10,11,12,13,14 |
| NCC (11) | 1,2,3,4,5,6,7,8,9,10,11 |
| RUSSIAN (12) | 1,2,3,4,5,6,7,8,9,10,11,12,13 |
| CN (13) | 1,2,3,4,5,6,7,8,9,10,11,12,13 |
| GLOBAL (14) | 1,2,3,4,5,6,7,8,9,10,11 passive scan: 12,13,14 |
| WORLD-WIDE (15) | 1,2,3,4,5,6,7,8,9,10,11 passive scan: 12,13 |

**Note:** When wifi is used as client mode, it will only listen AP Beacon during scanning in passive channel except it found hidden AP existed. When wifi is used as AP mode and configure to “auto” channel, it will not select the passive channel.

## 1.2.2. 5G Channel Plan

| regulation domain (mib *regdomain* value) | supported channels – DFS enabled | supported channels – DFS disabled |
|---|---|---|
| FCC (1) | 36,40,44,48,52,56,60,64,100,104,108, 112,116, 136,140,149,153,157,161,165 | 36,40,44,48,149,153,157, 161,165 |
| IC (2) | 36,40,44,48,52,56,60,64,149,153,157, 161 | 36,40,44,48,149,153,157, 161 |
| ETSI (3) | 36,40,44,48,52,56,60,64,100,104,108, 112,116,120,124,128,132,136,140 | 36,40,44,48 |
| SPAIN (4) (deprecated) | 36,40,44,48,52,56,60,64,100,104,108, 112,116,120,124,128,132,136,140 | 36,40,44,48 |

| | | |
|---|---|---|
| FRANCE (5) (deprecated) | 36,40,44,48,52,56,60,64,100,104,108, 112,116,120,124,128,132,136,140 | 36,40,44,48 |
| MKK (6) | 36,40,44,48,52,56,60,64,100,104,108, 112,116,120,124,128,132,136,140 | 36,40,44,48 |
| ISREAL (7) | 36,40,44,48,52,56,60,64,100,104,108, 112,116,120,124,128,132,136,140 | 36,40,44,48 |
| MKK1 (8) (deprecated) | 34,38,42,46 | 34,38,42,46 |
| MKK2 (9) (deprecated) | 36,40,44,48 | 36,40,44,48 |
| MKK3 (10) (deprecated) | 36,40,44,48,52,56,60,64 | 36,40,44,48 |
| NCC (11) | 56,60,64,100,104,108,112,116,136,140, 149,153,157,161,165 | 56,60,64,149,153,157,161, 165 |
| RUSSIAN (12) | 36,40,44,48,52,56,60,64,132,136,140,149 ,153,157,161,165 | 36,40,44,48,149,153,157, 161,165 |
| CN (13) | 36,40,44,48,52,56,60,64,149,153,157,161 ,165 | 36,40,44,48,149,153,157, 161,165 |
| GLOBAL(14) | 36,40,44,48,52,56,60,64,100,104,108,112 ,116,136,140,149,153,157,161,165 | 36,40,44,48,149,153,157, 161,165 |
| WORLD-WIDE(15) | 36,40,44,48,52,56,60,64,100,104,108,112 ,116,136,140,149,153,157,161,165 | 36,40,44,48,149,153,157, 161,165 |

# 2. Configuration

System configuration is used for store parameters for various features. RTL8192cd driver and software package provides three methods to configure parameters. One is through configuration files, one is through "iwpriv" command to set MIB directly, and the other one is via iwconfig/iwlist command.

Configuration files method creates configuration files to store parameters in file system in user space. Through scripts or software utility, these parameters will be read from the configuration files, maps value to related MIB value and then set to driver by "iwpriv" command.

Using "iwpriv" to set MIB directly and using iwconfig/iwlist command don't need to create configuration files. When using "iwpriv" command, the parameters map to MIB directly.

## 2.1. Configuration Files

The driver can be configured via a *configuration file* each time an interface is up. Before using configuration file method, be sure to turn on this function in kernel configuration:
Select *"Network device support ---> Wireless LAN (non-hamradio) ---> Config File support"*; then rebuild kernel image.

The configuration file is located at /etc/Wireless/RTL8192CD.dat with *Sytax:* <wlan_interface>_<mib_command> , e.g. wlan0_ssid=xxxx.
Bellows are the rules to compose the configuration file.

1. Add '#' in front of comment lines.
2. Space is NOT allowed between <wlan_interface> and <mib_command>.
3. If the user needs to configure MIB values with special characters, e.g. '#', the value of <mib_command> MUST be **quoted** E.g. wlan0_ssid="#XXXXX@##$$%%"
4. <wlan_interface>: wlan interface, e.g., wlan0, wlan0-va0. However, please **DO NOT** configure **WDS** interfaces because WDS is configured in wlan0 interface.
5. <mib_command>: MIB commands, e.g., ssid=xxxx, please refer to table "MIB command table" and following "Extended MIB command table"
6. MIB value should be also configured for each virtual interface separately.
7. Each time an interface is up, the configuration file will be loaded.

Extended MIB command table (available only if Config File support is turned on):

| Name | Meaning | Value | Default | Comment |
|---|---|---|---|---|
| hwaddr | MAC address of WLAN interface | 12 hex digits, e.g. 00e04c8192a1 | 0 | |

## 2.2. MIB

RTL8192cd driver provide MIB interface to get/set parameters by “iwpriv” command. Below lists “iwpriv” MIB commands format and all MIB parameter table.

1. Set_mib description:
   Usage: “iwpriv <iface> set_mib name=value1[,value2,value3…]”
   Iface: “wlan0”
   (1) Value can be a single field or multiple fields separated by ‘,’ without any space between fields. Detail parameter may be referred the following table.
   (2) If the value is the type of byte array, the format of value will be a string of ASCII of 0~f, which using 2 ASCII standing for one byte. For example, when set Tx power of CCK for path A, it will be
   “iwpriv wlan0 set_mib pwrlevelCCK_A=08080909090a0a0a0a0b0b0b0c0c”
2. Get_mib description:
   Usage: “iwpriv <iface> get_mib name”
   Iface: “wlan0”

| Name | Meaning | Value | Default | Comment |
|---|---|---|---|---|
| channel | Operation frequency used | 0 for auto channel, 1-14 for 11b/11g, 36-165 for 11a | | |
| ch_low | The lowest channel to scan and use | 1-14 for 11b/11g, 36-165 for 11a | | |
| ch_hi | The highest channel to scan and use | 1-14 for 11b/11g, 36-165 for 11a | | |
| pwrlevelCCK_A | CCK Tx power level for 14 channels (28 hex digits) for path A | RF module dependent | | Type of byte array |
| pwrlevelCCK_B | CCK Tx power level for 14 channels (28 hex digits) for path B | RF module dependent | | Type of byte array |
| pwrlevelHT40_1S_A | 40MHz mode HT OFDM 1 spatial stream Tx power level for 14 channels (28 hex digits) for path A | RF module dependent | | Type of byte array |
| pwrlevelHT40_1S_B | 40MHz mode HT OFDM 1 spatial stream Tx power level for 14 channels (28 hex digits) for path B | RF module dependent | | Type of byte array |
| pwrdiffHT40_2S | 40MHz mode HT OFDM 2 spatial stream Tx power difference between HT40_1S for 14 channels (28 hex digits). Bit[3:0] for path A. Bit[7:4] for path B. | RF module dependent | | Type of byte array |
| pwrdiffHT20 | 20MHz mode HT OFDM Tx power difference between HT40_1S for 14 channels (28 hex digits). Bit[3:0] for path A. Bit[7:4] for path B. | RF module dependent | | Type of byte array |
| pwrdiffOFDM | Legacy OFDM Tx power difference between HT40_1S for 14 channels (28 hex digits). Bit[3:0] for path A. | RF module dependent | | Type of byte array |

| | | | | |
|---|---|---|---|---|
| | Bit[7:4] for path B. | | | |
| Pwrlevel5GHT40_1S_A | 40MHz mode HT OFDM 1 spatial stream Tx power level for 5G 196 channels (392 hex digits) for path A | RF module dependent | | Type of byte array |
| Pwrlevel5GHT40_1S_B | 40MHz mode HT OFDM 1 spatial stream Tx power level for 5G 196 channels (392 hex digits) for path B | RF module dependent | | Type of byte array |
| Pwrdiff5GHT40_2S | 40MHz mode HT OFDM 2 spatial stream Tx power difference between HT40_1S for 5G 196 channels (392 hex digits). Bit[3:0] for path A. Bit[7:4] for path B. | RF module dependent | | Type of byte array |
| Pwrdiff5GHT20 | 20MHz mode HT OFDM Tx power difference between HT40_1S for 5G 196 channels (392 hex digits). Bit[3:0] for path A. Bit[7:4] for path B. | RF module dependent | | Type of byte array |
| Pwrdiff5GOFDM | Legacy OFDM Tx power difference between HT40_1S for 5G 196 channels (392 hex digits). Bit[3:0] for path A. Bit[7:4] for path B. | RF module dependent | | Type of byte array |
| pwrdiff_20BW1S_OFDM1T_A | Pwower Index Difference between BW20-1S and BW40-1S.<br>Bit[7:4]: Path A 2G Offset, Range –8~7.<br>Pwower Index Difference between OFDM-1Tx and BW40-1S.<br>Bit[3:0]: Path A 2G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| pwrdiff_40BW2S_20BW2S_A | Pwower Index Difference between BW40-2S and BW40-1S.<br>Bit[7:4]: Path A 2G Offset, Range –8~7.<br>Pwower Index Difference between BW20-2S and BW20-1S.<br>Bit[3:0]: Path A 2G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| pwrdiff_5G_20BW1S_OFDM1T_A | Pwower Index Difference between BW20-1S and BW40-1S.<br>Bit[7:4]: Path A 5G Offset, Range –8~7.<br>Pwower Index Difference between OFDM-1Tx and BW40-1S.<br>Bit[3:0]: Path A 5G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| pwrdiff_5G_40BW2S_20BW2S_A | Pwower Index Difference between BW40-2S and BW40-1S.<br>Bit[7:4]: Path A 5G Offset, Range –8~7.<br>Pwower Index Difference between | RF module dependent (for 8812) | | Type of byte array |

| | | | | |
|---|---|---|---|---|
| | BW20-2S and BW20-1S.<br>Bit[3:0]: Path A 5G Offset, Range –8~7. | | | |
| pwrdiff_5G_80BW1S_160BW1S_A | Pwower Index Difference between BW80-1S and BW40-1S (UpSide Ch + LowSide Ch)/2.<br>Bit[7:4]: Path A 5G Offset, Range –8~7.<br>Pwower Index Difference between BW160-1S and BW80-1S (UpSide Ch + LowSide Ch)/2.<br>Bit[3:0]: Path A 5G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| pwrdiff_5G_80BW2S_160BW2S_A | Pwower Index Difference between BW80-2S and BW80-1S.<br>Bit[7:4]: Path A 5G Offset, Range –8~7.<br>Pwower Index Difference between BW160-2S and BW160-1S.<br>Bit[3:0]: Path A 5G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| pwrdiff_20BW1S_OFDM1T_B | Pwower Index Difference between BW20-1S and BW40-1S.<br>Bit[7:4]: Path B 2G Offset, Range –8~7.<br>Pwower Index Difference between OFDM-1Tx and BW40-1S.<br>Bit[3:0]: Path B 2G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| pwrdiff_40BW2S_20BW2S_B | Pwower Index Difference between BW40-2S and BW40-1S.<br>Bit[7:4]: Path B 2G Offset, Range –8~7.<br>Pwower Index Difference between BW20-2S and BW20-1S.<br>Bit[3:0]: Path B 2G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| pwrdiff_5G_20BW1S_OFDM1T_B | Pwower Index Difference between BW20-1S and BW40-1S.<br>Bit[7:4]: Path B 5G Offset, Range –8~7.<br>Pwower Index Difference between OFDM-1Tx and BW40-1S.<br>Bit[3:0]: Path B 5G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| pwrdiff_5G_40BW2S_20BW2S_B | Pwower Index Difference between BW40-2S and BW40-1S.<br>Bit[7:4]: Path B 5G Offset, Range –8~7.<br>Pwower Index Difference between | RF module dependent (for 8812) | | Type of byte array |

| | | | | |
|---|---|---|---|---|
| | BW20-2S and BW20-1S.<br>Bit[3:0]: Path B 5G Offset, Range –8~7. | | | |
| pwrdiff_5G_80BW1S_160BW1S_B | Pwower Index Difference between BW80-1S and BW40-1S (UpSide Ch + LowSide Ch)/2.<br>Bit[7:4]: Path B 5G Offset, Range –8~7.<br>Pwower Index Difference between BW160-1S and BW80-1S (UpSide Ch + LowSide Ch)/2.<br>Bit[3:0]: Path B 5G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| pwrdiff_5G_80BW2S_160BW2S_B | Pwower Index Difference between BW80-2S and BW80-1S.<br>Bit[7:4]: Path B 5G Offset, Range –8~7.<br>Pwower Index Difference between BW160-2S and BW160-1S.<br>Bit[3:0]: Path B 5G Offset, Range –8~7. | RF module dependent (for 8812) | | Type of byte array |
| preamble | CCK preamble type | 0 – long preamble, 1 – short preamble | | |
| trswitch | Enable T/R switch | 0 – disable, 1 – enable | | |
| disable_ch14_ofdm | Disable OFDM sending and receiving in channel 14. It will also prevent auto channel to choose ch14. | 0 – enable, 1 – disable | | |
| disable_ch1213 | Prevent auto channel to choose ch12 and ch13 | 0 – enable, 1 – disable | | |
| pa_type | Support 8812 different pa type | 0 – skyworth-5022,<br>1 – RFMD-4501 / skyworth-85703<br>2 – SKYWORKS_5023<br>3 – RTC5634<br>16 – Internal PA | 0 | |
| xcap | Crystal Capacitor value | 0~62(0x3E) | | 0 stands the value is not calibrated yet. |
| tssi1 | Tx signal strength value of path A | 0 – 255 | | 0 stands the value is not calibrated yet. |
| tssi2 | Tx signal strength value of path B | 0 – 255 | | 0 stands the value is not calibrated yet. |
| ther | Thermal value | For 8188C/8192C/8192D: 7 < ther <= 0x1d<br>Others: 7 < ther <= 0x32 | | 0 stands the value is not calibrated yet. |
| MIMO_TR_mode | MIMO mode assignment | 1 – 1T2R, 3 – 2T2R, 4 – 1T1R | 3 | |
| tx2path | Enable tx using 2 path to send 1T rate | 0 – disable, 1 – enable | 1 | |
| bcn2path | Enable 2 path to send beacon | 0 – disable, 1 – enable | | |

| | | | | |
|---|---|---|---|---|
| add_cck1M_pwr | Add power to CCK 1M rate | 0 – disable,<br>Other – the power added to CCK 1M in unit of power level | | |
| ssid | SSID | "string_value", SSID with 32 characters in max | | |
| defssid | If don't give SSID in Ad-hoc client mode and no IBSS available, it will start an IBSS with SSID given here. | "string_value", SSID with 32 chars in max | "defaultSSID" | |
| bssid2join | Besides SSID, designate target BSSID to join | xxxxxxxxxxxx (12 digits mac address) | | Type of byte array |
| bcnint | Beacon interval in ms | 20-1024 | 100 | |
| dtimperiod | DTIM period | 1-255 | 1 | Suggest to set 1 because patent issue |
| swcrypto | S/w encryption enabled/disabled | 0 – disable, 1 – enable | | |
| aclmode | Access control mode | 0 – disable, 1 – accept, 2 – deny | | |
| aclnum | Set number of ACL | Suggest set '0' whenever driver is re-initialized | | |
| acladdr | Set access control address | xxxxxxxxxxxx (12 digits mac address) | | When acl is added, the aclnum will be increased automatically. |
| oprates | Operational rates | Bit0-bit11 for 1,2,5.5,11,6,9,12,18,24,,36,48,54 M | 0xfff | |
| basicrates | Basic rates | Bit0-bit11 for 1,2,5.5,11,6,9,12,18,24,,36,48,54 M | 0xf | |
| regdomain | Regulation domain | 1-15 (FCC, IC, ETSI, SPAIN, FRANCE, MKK, ISREAL, MKK1, MKK2, MKK3, NCC,<br>RUSSIAN, CN, GLOBAL, WORLD-WIDE) | 1 | Please refer the 2.4G channel plan table in detail. |
| txpwr_lmt_index | Set specific region domain for tx power limit | 0 – use mib regdomain for tx power limit,<br>Other – the specific region domain index for tx power limit | | |
| autorate | Auto rate adaptive | 0 – disable, 1 – enable | 1 | |
| fixrate | Fixed Tx rate | Bit0-bit11 for 1,2,5.5,11,6,9,12,18,24,,36,48,54 M<br>Bit12-Bit27 for MCS0,MCS1,…,MCS15<br>(Bit31 + 0) for NSS1-MCS0<br>(Bit31 + 1) for NSS1-MCS1<br>(Bit31 + 2) for NSS1-MCS2<br>…<br>(Bit31 + 10) for NSS2-MCS0<br>(Bit31 + 11) for NSS2-MCS1<br>(Bit31 + 12) for NSS2-MCS2<br>…etc | | Refer when auto rate is disabled |

| | | | | |
|---|---|---|---|---|
| disable_protection | Forcedly disable protection mode | 0 – auto, 1 – disable protection | | Normally when 11g is used, driver will auto detect if legacy (11b) device is existed. When 11n is used, driver will auto detect if legacy (11b/g) device is existed. If yes, it will enable protection mode automatically. |
| disable_olbc | Forcedly OLBC detection | 0 – auto, 1 – disable protection | | Normally 11g AP should detect OLBC. If disabled, AP will enter protection mode only when legacy device associated. |
| deny_legacy | Deny the association from legacy STA for corresponding band | 1 – 11b, 2 – 11g, 4 – 11a, 8 – 11n | | Set the corresponding legacy band of STA to deny |
| prob_info_enable | Enable wlan driver to collect probe request information | 0 – disable, 1 – enable | | Cat /proc/wlan0/probe_info to see the collect results |
| fast_roaming | Client mode fast roaming | 0 – disable, 1 – enable | | |
| lowestMlcstRate | Use lowest basic rate to send multicast and broadcast | 0 – disable<br>Bit0-bit11 for 1,2,5.5,11,6,9,12,18,24,,36,48,54 M<br>Bit12-Bit27 for MCS0,MCS1,...,MCS15 | | |
| stanum | Limit max associated sta number | 0-32. 0 – disable (not limit). | | |
| authtype | 802.11 Authentication type | 0 – open system, 1 – shared key, 2 – auto | 2 | |
| encmode | Encryption mode | 0 – disabled, 1 – WEP64, 2 – TKIP, 4 – AES(CCMP), 5 – WEP128 | | Set to 2 always under WPA/WPA2 mode |
| wepdkeyid | WEP default Tx key | 0-3 | | |
| psk_enable | PSK mode | 0 – disable, 1 – WPA, 2 – WPA2, 3 – WPA/WPA2 mixed | | |
| wpa_cipher | WPA PSK cipher suite | 2 –TKIP, 8 – AES(CCMP), 10 – TKIP/AES mixed | | |
| wpa2_cipher | WPA2 PSK cipher suite | 2 –TKIP, 8 – AES(CCMP), 10 – TKIP/AES mixed | | |
| passphrase | PSK key | 32 characters or 64 hex digits | | |
| gk_rekey | Group key update time | 0 – disable, >1 – enable | | Time unit is second |
| 802_1x | Flag of using 802.1x | 0 – disable, 1 – enable | | When 802.1x is enabled, the Auth daemon must be invoked |
| default_port | Default state of 802.1x control port | 0 – data packet is not allowed to | | Refer when 802_1x is |

<table>
<tr><td></td><td></td><td>pass through until 802.1x authentication is ok<br>1 – data packet is allowed pass through even 802.1x authentication is not ok</td><td></td><td>set to 1</td></tr>
<tr><td>wepkey1</td><td>WEP key1</td><td>10 hex digits for WEP64, 26 hex digits for WEP128</td><td></td><td>Type of byte array</td></tr>
<tr><td>wepkey2</td><td>WEP key2</td><td>10 hex digits for WEP64, 26 hex digits for WEP128</td><td></td><td>Type of byte array</td></tr>
<tr><td>wepkey3</td><td>WEP key3</td><td>10 hex digits for WEP64, 26 hex digits for WEP128</td><td></td><td>Type of byte array</td></tr>
<tr><td>wepkey4</td><td>WEP key4</td><td>10 hex digits for WEP64, 26 hex digits for WEP128</td><td></td><td>Type of byte array</td></tr>
<tr><td>opmode</td><td>Operation mode (AP or client)</td><td>16 – AP, 8 – Infrastructure client, 32 – Ad-hoc client</td><td>16</td><td></td></tr>
<tr><td>hiddenAP</td><td>Hidden AP enable/disable</td><td>0 – disabled, 1 – enabled</td><td></td><td></td></tr>
<tr><td>rtsthres</td><td>RTS threshold</td><td>0-2347</td><td>2347</td><td></td></tr>
<tr><td>fragthres</td><td>Fragment threshold</td><td>256-2346</td><td>2346</td><td></td></tr>
<tr><td>shortretry</td><td>Short retry limit</td><td>1-255</td><td>3</td><td></td></tr>
<tr><td>longretry</td><td>Long retry limit</td><td>1-255</td><td>3</td><td></td></tr>
<tr><td>expired_time</td><td>Client inactivity time in 10ms</td><td>>100</td><td>30000</td><td>Time unit is 10 ms.</td></tr>
<tr><td>led_type</td><td>WLAN LED type</td><td>For 8188RE, 8192CE, 8192DR
<table>
<tr><th></th><th>LED0</th><th>LED1</th></tr>
<tr><td>0</td><td>tx</td><td>rx</td></tr>
<tr><td>1</td><td>enable/tx/rx</td><td>n/a</td></tr>
<tr><td>2</td><td>link</td><td>tx/rx (d,m)</td></tr>
<tr><td>3</td><td>link/tx/rx (d,m)</td><td>n/a</td></tr>
<tr><td>4</td><td>link</td><td>tx/rx (d)</td></tr>
<tr><td>5</td><td>link/tx/rx (d)</td><td>n/a</td></tr>
<tr><td>6</td><td>enable</td><td>tx/rx (d)</td></tr>
<tr><td>7</td><td>enable/tx/rx (d)</td><td>n/a</td></tr>
<tr><td>8</td><td>11a tx/rx (d)</td><td>11g tx/rx (d)</td></tr>
</table>
0-1 – hw control<br>2-8 – sw control<br>d – count data frames<br>m – count management frames<br><br>For 8188RE, 8192CE
<table>
<tr><th></th><th>LED2 (GPIO8)</th></tr>
<tr><td>11</td><td>link/tx/rx (d,m)</td></tr>
<tr><td>12</td><td>enable/tx/rx (d)</td></tr>
<tr><td>15</td><td>link/tx/rx (d)</td></tr>
<tr><td>16</td><td>assoc/tx/rx (d)</td></tr>
<tr><th></th><th>LED2 (GPIO10)</th></tr>
<tr><td>13</td><td>link/tx/rx (d,m)</td></tr>
<tr><td>17</td><td>enable/tx/rx (d)</td></tr>
</table>
11-17 – sw control<br>d – count data frames</td><td></td><td></td></tr>
</table>

<table>
<tr><td></td><td></td><td>m – count management frames<br><br>For 8192DR<table><tr><td></td><td>LED2 (GPIO10)</td></tr><tr><td>50</td><td>enable/tx/rx (d)</td></tr><tr><td>52</td><td>link/tx/rx (d,m)</td></tr><tr><td></td><td>LED1 (GPIO9)</td></tr><tr><td>51</td><td>link/tx/rx (d,m)</td></tr></table>50-52 – sw control<br>d – count data frames<br>m – count management frames<br><br>For 8188ER/8192ER<table><tr><td></td><td>LED0 (GPIO5)</td></tr><tr><td>3</td><td>link/tx/rx (d,m)</td></tr><tr><td>5</td><td>link/tx/rx (d)</td></tr><tr><td>7</td><td>enable/tx/rx (d)</td></tr></table>3-7 – sw control<br>d – count data frames<br>m – count management frames<br><br>For 8812E/8812AR-VN<table><tr><td></td><td>LED1</td></tr><tr><td>3</td><td>link/tx/rx (d,m)</td></tr><tr><td>5</td><td>link/tx/rx (d)</td></tr><tr><td>7</td><td>enable/tx/rx (d)</td></tr></table>3-7 – sw control<br>d – count data frames<br>m – count management frames<br><br>For 8881A<table><tr><td></td><td>LED0</td></tr><tr><td>3</td><td>link/tx/rx (d,m)</td></tr><tr><td>5</td><td>link/tx/rx (d)</td></tr><tr><td>7</td><td>enable/tx/rx (d)</td></tr></table>3-7 – sw control<br>d – count data frames<br>m – count management frames</td><td></td><td></td></tr>
<tr><td>iapp_enable</td><td>IAPP enable/disable</td><td>0 – disable, 1 – enable</td><td></td><td></td></tr>
<tr><td>block_relay</td><td>Block packet relaying between associated clients</td><td>0 – relay, 1 – block relay and drop, 2 – block relay and indicate to bridge</td><td></td><td></td></tr>
<tr><td>deny_any</td><td>Deny the association SSID of “any” including upper and lower cases</td><td>0 – disable, 1 – enable</td><td></td><td></td></tr>
<tr><td>crc_log</td><td>Calculate CRC error packets</td><td>0 – disable, 1 – enable</td><td></td><td></td></tr>
<tr><td>wifi_specific</td><td>Do WiFi logo test specific check</td><td>0 – disable, 1 – enable, 2 – auto</td><td>2</td><td>0 for performance</td></tr>
</table>

| | | | | |
|---|---|---|---|---|
| | | | | mode; 1 for WiFi mode, 2 for auto mode. PS. For 8192DR 1x1 concurrent mode and 8188E, please set 1 to pass WiFi logo test. |
| disable_txsc | Tx shortcut enable/disable | 0 – enable, 1 – enable | | |
| disable_rxsc | Rx shortcut enable/disable | 0 – enable, 1 – enable | | |
| disable_brsc | Bridge shortcut enable/disable | 0 – enable, 1 – enable | | |
| keep_rsnie | Don't clean RSN IE while reinitialize the interface | 0 – erase, 1 – keep | | |
| guest_access | Restrict client to internet access only | 0 – disable, 1 – enable | | |
| band | Band selection | 1 – 11b, 2 – 11g, 4 – 11a, 8 – 11n 64 – 11ac | 3 | |
| cts2self | Use cts2Self for protection mode | 0 – no, 1 – yes | 1 | |
| wds_enable | WDS enable/disable | 0 – disable, 1 – enable | | |
| wds_pure | Flag to enable pure WDS mode that don't broadcast beacon and don't accept any station | 0 – disable, 1 – enable | | |
| wds_priority | Give WDS packets higher priority | 0 – disable, 1 – enable | | |
| wds_num | Set number of WDS | Suggest set '0' whenever driver is re-initialized | | |
| wds_add | Set mac address of peer WDS AP and the rate sent to the peer WDS AP | xxxxxxxxxxxx (12 digits mac address). The max entry could be added is 8 in default configuration. After mac address, there is a 32-bit variable to give the rate. Bit0-bit11 for 1,2,5.5,11,6,9,12,18,24,,36,48,54 M Bit12-Bit27 for MCS0,MCS1,…,MCS15 | | When mac address is added, the wds_num will be increased automatically. |
| wds_encrypt | WDS encryption mode | 0 – disabled, 1 – WEP64, 2 – TKIP, 4 – AES (CCMP), 5 – WEP128 | | |
| wds_wepkey | WDS WEP default key | 10 hex digits for WEP64, 26 hex digits for WEP128 | | Type of byte array |
| wds_passphrase | WDS PSK key | 32 characters or 64 hex digits | | |
| nat25_disable | Disable NAT2.5 transformation in client mode | 0 – enable, 1 – disable | | |
| macclone_enable | Enable MAC clone from the first incoming packet | 0 – disable, 1 – enable | | |
| dhcp_bcst_disable | Flag of adding broadcast flag into DHCP request | 0 – enable, 1 – disable | | |
| add_pppoe_tag | Add extra tag in PPPoE packets by NAT2.5 | 0 – disable, 1 – enable | 1 | When set to 0, NAT2.5 can only support one session buildup at the same time. |
| clone_mac_addr | Assign the target MAC to clone | xxxxxxxxxxxx (12 digits mac | | Type of byte array |

| | | | | |
|---|---|---|---|---|
| | | address) | | |
| nat25sc_disable | NAT2.5 shortcut enable/disable | 0 – enable, 1 – disable | | |
| show_hidden_bss | Show hidden BSS in site survey | 0 – disable, 1 – enable | | |
| ack_timeout | Set ACK timeout value | 0-255 | | 0 means using standard value. In unit of us. |
| private_ie | Send and get private IE | At most 64 hex digits byte array | | |
| groupID | Group ID of virtual AP (multiple SSID) | 0-65535 | | When AP (including root and virtual) set the same group ID, the wlan traffics could be relayed.<br>Root interface: wlan0<br>Virtual interface: wlan0-va0~wlan0-va3. |
| vap_enable | Tell driver if multiple AP function is enabled or disabled | 0 – disable, 1 – enable | | If multiple AP is enabled, this mib must be set to 1. |
| func_off | Temporary disable wlan function | 0 – normal, 1 – wlan off | | |
| qos_enable | Support WMM and QoS | 0 – disable, 1 – enable | | |
| apsd_enable | Support WMM APSD function | 0 – disable, 1 – enable | | |
| apsd_sta_be | Enable client mode BE queue | 0 – disable, 1 – enable | | This mid is only valid when apsd_enable is 1(enable) |
| apsd_sta_bk | Enable client mode BK queue | 0 – disable, 1 – enable | | This mid is only valid when apsd_enable is 1(enable) |
| apsd_sta_vi | Enable client mode VI queue | 0 – disable, 1 – enable | | This mid is only valid when apsd_enable is 1(enable) |
| apsd_sta_vo | Enable client mode VO queue | 0 – disable, 1 – enable | | This mid is only valid when apsd_enable is 1(enable) |
| wsc_enable | Support WiFi Protection Setup | Bit0 for client mode, Bit1 for AP mode | | |
| pin | PIN setting for WPS | “string_value” with 8 characters in max | | |
| supportedmcs | Supported MCS rates | Bit 0-15 for MCS0, …, MCS15 | 0xffff | |
| basicmcs | Basic MCS rates | Bit 0-15 for MCS0, …, MCS15 | | |
| use40M | Support 40M bandwidth in 11n mode | 0 – 20M<br>1 – 40M<br>2 – 80M | | |
| 2ndchoffset | Control sideband offset | 1 – secondary channel is below the primary channel, 2 – secondary channel is above the primary channel | 1 | |
| shortGI20M | Support short GI in 20M bandwidth | 0 – disable, 1 – enable | | |
| shortGI40M | Support short GI in 40M bandwidth | 0 – disable, 1 – enable | | |
| stbc | Support Space Time Block Coding | 0 – disable, 1 – enable | | |

| | | | | |
|---|---|---|---|---|
| ldpc | Enable ldpc | 0 – disable, 1 – enable | 1 | |
| ampdu | Support packet aggratation | 0 – disable, 1 – enable | | |
| lgyEncRstrct | Restrict legacy encryption in N mode | Bit 0: WEP, Bit 1: TKIP | | |
| coexist | Support 20M/40M coexistant mode | 0 – disable, 1 – enable | | |
| txnoack | Enable Tx without receiving ACK | 0 – disable, 1 – enable | | |
| debug_err | Flag of DEBUG_ERR() macro | Bit value defined in 8185ag_debug.h (in hex) | ffffffff | |
| debug_info | Flag of DEBUG_INFO() macro | Bit value defined in 8185ag_debug.h (in hex) | 0 | |
| debug_warn | Flag of DEBUG_WARN() macro | Bit value defined in 8185ag_debug.h (in hex) | 0 | |
| debug_trace | Flag of DEBUG_TRACE() macro | Bit value defined in 8185ag_debug.h (in hex) | 0 | |
| ledBlinkingFreq | Multiple of wlan LED blinking frequency. | 1~100 | 1 | This value will be referred only when mib value of 'led_type' is greater than 1. |
| wapiType | WAPI mode | 0 - Disable<br>1 - Certificate<br>2 – PSK | 0 | |
| wapiPsk | WAPI PSK | Up to 32 characters | | |
| wapiPsklen | WAPI PSK length | 0~32 | | |
| wapiUCastKeyType | Unicast key update mode | 1 – Disable<br>2 – Time based<br>3 – Packet based<br>4 – Mix mode(Rekey when time or packet number exceeds threshold) | | This object selects a mechanism for rekeying the unicast key. |
| wapiUCastKeyTimeout | Timeout threshold of time-based unicast key update mechanism | Unit: sec. | | |
| wapiUCastKeyPktNum | Packet number threshold of packet based unicast key update mechanism | | | |
| wapiMCastKeyType | Multicast key update mode | 1 – Disable<br>2 – Time based<br>3 – Packet based<br>4 – Mix mode(Rekey when time or packet number exceeds threshold) | | This object selects a mechanism for rekeying the multicast key. |
| wapiMCastKeyTimeout | Timeout threshold of time-based multicast key update mechanism | Unit: sec. | | |
| wapiMCastKeyPktNum | Packet number threshold of packet based multicast key update mechanism | | | |
| manual_edca | Enable / disable EDCA use manual values | 0: disable, 1: enable | 0 | |
| sta_bkq_acm | Enable / disable AP broadcasting BK queue under ACM | 0: disable, 1: enable | 0 | It is useless in general case |
| sta_bkq_aifsn | Set AIFS slot number for BK queue broadcasted by AP | 1~7 | 7 | Its value in flash is sum of SIFS and total |

| | | | | |
|---|---|---|---|---|
| | | | | slot time.<br>SIFS is 10 us when 11a/b/g and 16 us when 11n. Slot time is 20 us when 11a/b and 9 us when 11g/n.<br>For example, sta_bkq_aifsn=7 under 11g/n, AIFS is 9*7+16 = 79 us. |
| sta_bkq_cwmin | Set minimal contention window period for BK queue broadcasted by AP | 1~10 | 4 | Slot time will be 2^n-1, 15, by default. |
| sta_bkq_cwmax | Set maximal contention window period for BK queue broadcasted by AP | 1~10 | 10 | Slot time will be 2^n-1, 1023, by default. |
| sta_bkq_txoplimit | Set TXOP limit for BK queue broadcasted by AP | 0~256 | 0 | |
| sta_beq_acm | Enable / disable AP broadcasting BE queue under ACM | 0: disable, 1: enable | 0 | |
| sta_beq_aifsn | Set AIFS slot number for BE queue broadcasted by AP | 1~7 | 3 | Its value in flash is sum of SIFS and total slot time.<br>SIFS is 10 us when 11a/b/g and 16 us when 11n. Slot time is 20 us when 11a/b and 9 us when 11g/n.<br>For example, sta_beq_aifsn=3 under 11g/n, AIFS is 9*3+16 = 43 us. |
| sta_beq_cwmin | Set minimal contention window period for BE queue broadcasted by AP | 1~10 | 4 | Slot time will be 2^n-1, 15, by default. |
| sta_beq_cwmax | Set maximal contention window period for BE queue broadcasted by AP | 1~10 | 10 | Slot time will be 2^n-1, 1023, by default. |
| sta_beq_txoplimit | Set TXOP limit for BE queue broadcasted by AP | 0~256 | 0 | |
| sta_viq_acm | Enable / disable AP broadcasting VI queue under ACM | 0: disable, 1: enable | 0 | |
| sta_viq_aifsn | Set AIFS slot number for VI queue broadcasted by AP | 1~7 | 2 | Its value in flash is sum of SIFS and total slot time.<br>SIFS is 10 us when 11a/b/g and 16 us when 11n. Slot time is 20 us when 11a/b and 9 us when 11g/n. |

| | | | | |
|---|---|---|---|---|
| | | | | For example, sta_viq_aifsn=2 under 11g/n, AIFS is 9*2+16 = 34 us. |
| sta_viq_cwmin | Set minimal contention window period for VI queue broadcasted by AP | 1~10 | 3 | Slot time will be 2^n-1, 15, by default. |
| sta_viq_cwmax | Set maximal contention window period for VI queue broadcasted by AP | 1~10 | 4 | Slot time will be 2^n-1, 7, by default. |
| sta_viq_txoplimit | Set TXOP limit for VI queue broadcasted by AP | 0~256 | 188 | Follow SPEC in 11b |
| sta_voq_acm | Enable / disable AP broadcasting VO queue under ACM | 0: disable, 1: enable | 0 | |
| sta_voq_aifsn | Set AIFS slot number for VO queue broadcasted by AP | 1~7 | 2 | Its value in flash is sum of SIFS and total slot time.<br>SIFS is 10 us when 11a/b/g and 16 us when 11n. Slot time is 20 us when 11a/b and 9 us when 11g/n.<br>For example, sta_voq_aifsn=2 under 11g/n, AIFS is 9*2+16 = 34 us. |
| sta_voq_cwmin | Set minimal contention window period for VO queue broadcasted by AP | 1~10 | 2 | Slot time will be 2^n-1, 7, by default. |
| sta_voq_cwmax | Set maximal contention window period for VO queue broadcasted by AP | 1~10 | 3 | Slot time will be 2^n-1, 3, by default. |
| sta_voq_txoplimit | Set TXOP limit for VO queue broadcasted by AP | 0~256 | 102 | Follow SPEC in 11b |
| ap_bkq_aifsn | Set AIFS slot number for BK queue used by AP | 1~7 | 7 | Its value in flash is sum of SIFS and total slot time.<br>SIFS is 10 us when 11a/b/g and 16 us when 11n. Slot time is 20 us when 11a/b and 9 us when 11g/n.<br>For example, ap_bkq_aifsn=7 under 11g/n, AIFS is 9*7+16 = 79 us. |
| ap_bkq_cwmin | Set minimal contention window period for BK queue used by AP | 1~10 | 4 | Slot time will be 2^n-1, 15, by default. |
| ap_bkq_cwmax | Set maximal contention window period for BK queue used by AP | 1~10 | 10 | Slot time will be 2^n-1, 1023, by |

| | | | | |
|---|---|---|---|---|
| | | | | default. |
| ap_bkq_txoplimit | Set TXOP limit for BK queue used by AP | 0~256 | 0 | |
| ap_beq_aifsn | Set AIFS slot number for BE queue used by AP | 1~7 | 3 | Its value in flash is sum of SIFS and total slot time.<br>SIFS is 10 us when 11a/b/g and 16 us when 11n. Slot time is 20 us when 11a/b and 9 us when 11g/n.<br>For example, ap_beq_aifsn=3 under 11g/n, AIFS is 9*3+16 = 43 us. |
| ap_beq_cwmin | Set minimal contention window period for BE queue used by AP | 1~10 | 4 | Slot time will be 2^n-1, 15, by default. |
| ap_beq_cwmax | Set maximal contention window period for BE queue used by AP | 1~10 | 6 | Slot time will be 2^n-1, 63, by default. |
| ap_beq_txoplimit | Set TXOP limit for BE queue used by AP | 0~256 | 0 | |
| ap_viq_aifsn | Set AIFS slot number for VI queue used by AP | 1~7 | 1 | Its value in flash is sum of SIFS and total slot time.<br>SIFS is 10 us when 11a/b/g and 16 us when 11n. Slot time is 20 us when 11a/b and 9 us when 11g/n.<br>For example, ap_viq_aifsn=1 under 11g/n, AIFS is 9*1+16 = 25 us. |
| ap_viq_cwmin | Set minimal contention window period for VI queue used by AP | 1~10 | 4 | Slot time will be 2^n-1, 15, by default. |
| ap_viq_cwmax | Set maximal contention window period for VI queue used by AP | 1~10 | 3 | Slot time will be 2^n-1, 7, by default. |
| ap_viq_txoplimit | Set TXOP limit for VI queue used by AP | 0~256 | 188 | Follow SPEC in 11b |
| ap_voq_aifsn | Set AIFS slot number for VO queue used by AP | 1~7 | 1 | Its value in flash is sum of SIFS and total slot time.<br>SIFS is 10 us when 11a/b/g and 16 us when 11n. Slot time is 20 us when 11a/b and 9 us when 11g/n.<br>For example, ap_voq_aifsn=1 under 11g/n, AIFS is 9*1+16 |

| | | | | |
|---|---|---|---|---|
| | | | | = 25 us. |
| ap_voq_cwmin | Set minimal contention window period for VO queue used by AP | 1~10 | 3 | Slot time will be 2^n-1, 7, by default. |
| ap_voq_cwmax | Set maximal contention window period for VO queue used by AP | 1~10 | 2 | Slot time will be 2^n-1, 3, by default. |
| ap_voq_txoplimit | Set TXOP limit for VO queue used by AP | 0~256 | 102 | Follow SPEC in 11b |
| band5GSelected | Restrict 5G channel usage to specific bands | BIT0: 0 – disable band 1 usage,<br>1– enable band 1 usage<br>BIT1: 0 – disable band 2 usage,<br>1– enable band 2 usage<br>BIT2: 0 – disable band 3 usage,<br>1– enable band 3 usage<br>BIT3: 0 – disable band 4 usage,<br>1– enable band 4 usage | 0x0f | |
| macPhyMode | Set dual or single MAC/PHY mode | 0 – Single MAC/PHY,<br>2 – Dual MAC/PHY | 2 | Please refer to section "Dual-band configuration" |
| txbf | Enable Tx Beamforming function | 0 – disable, 1 – enable | 1 | |
| txbfer | Enable Tx Beamformer | 0 – disable, 1 – enable | 1 | This mib is only valid when txbf is 1 |
| txbfee | Enable Tx Beamformee | 0 – disable, 1 – enable | 1 | This mib is only valid when txbf is 1 |
| supportedvht | Set Tx/Rx MCS map of VHT Capabilities element which carried in beacon, probe response, association request, response.<br><br>Definition follows Draft P802.11ac_D3.0, 8.4.2.160.3 VHT Supported MCS Set field | 0xfffa – Support 2SS MCS0~9, 1SS MCS0~9<br><br>0xfff5 - Support 2SS MCS0~8, 1SS MCS0~8<br><br>0xfffe - Support 1SS MCS0~9, 2SS not support<br><br>0xfffc - Support 1SS MCS0~7, 2SS not support | 0xfffa | 1SS Support<br>b[1:0] =<br>0: MCS0~7<br>1: MCS0~8<br>2: MCS0~9<br>3: not support<br><br>2SS Support<br>b[3:2] =<br>0: MCS0~7<br>1: MCS0~8<br>2: MCS0~9<br>3: not support |
| vht_txmap | Set VHT Tx rate map for rate adaptive algorithm | 0xfffff – Support 2SS MCS9~0, 1SS MCS9~0<br><br>0x7ffdf –Support 2SS MCS8~0, 1SS MCS8~0<br><br>0x3fcff - Support 2SS2 MCS7~0, 1SS MCS7~0<br><br>0xfcff: Support 2SS2 MCS5~0, 1SS MCS7~0 | 0xfffff | b[9:0]:<br>1SS MCS9~0<br><br>b[19:10]:<br>2SS MCS9~0, |
| adaptivity_enable | Enable adaptivity test support | 0: disable, 1: enable | | If trying to pass adaptivity test in ETSI |

| | | | | |
|---|---|---|---|---|
| | | | | domain, this mib should be set to 1. |
| edcca_thd | Energy threshold for adaptivity test | Unit is dBm if 100 minus this value. For example, default value is –55dBm. | 45 | |
| mesh_enable | Enable mesh function | 0 – disable, 1 – enable | | |
| mesh_ap_enable | Enable access point function | 0 – disable, 1 – enable | | This mib only valid when mesh_enable is 1. Mesh_ap_enable is 0 means pure mesh mode. |
| mesh_id | Mesh ID | "string_value", with 32 characters in max | | This mib only valid when mesh_enable is 1 |
| mesh_privacy | Enable mesh data encryption | 0 – disable, 4 – AES(CCMP) | | This mib only valid when mesh_enable is 1 |
| mesh_passphrase | Mesh PSK key | 32 characters or 64 hex digits | | |
| mesh_max_neighbor | The number of neighbor one mesh node can have. | 0~15 | 15 | This mib only valid when mesh_enable is 1 |
| mesh_igmp_enable | Enable mesh igmp snooping | 0 – disable, 1 – enable | 1 | This mib only valid when mesh_enable is 1 |
| meshaclmode | Access control mode between mesh nodes | 0 – disable, 1 – accept, 2 – deny | | This mib only valid when mesh_enable is 1 |
| meshaclnum | The number of ACL | 0~128 | | This mib only valid when mesh_enable is 1 and meshaclmode is 1 or 2 |
| meshacladdr | The array of mac address of mesh nodes to be deny or accept | The content of this mib depends on the value of meshaclnum. If meshaclnum is 1, this mib contains one xxxxxxxxxxx (12 digits mac address). If meshaclnum is 2, this mib contains two xxxxxxxxxxx (12 digits mac address), totally 24 digits. | | This mib only valid when mesh_enable is 1 and meshaclmode is 1 or 2 |
| disable_DFS | Disable DFS function | 0 – enable DFS, 1 – disable DFS | 0 | |
| gbwcmode | Set group bandwidth control mode | 0 – disable group bandwidth control function,<br>1 – The Tx and Rx bandwidth to or from address in gbwcaddr are restricted by gbwcthrd_tx and gbwcthrd_rx,<br>2 – The Tx and Rx bandwidth to or from address NOT in gbwcaddr are restricted by gbwcthrd_tx and | | |

| | | | | |
|---|---|---|---|---|
| | | gbwcthrd_rx,<br>3 – The total Tx bandwidth to all address is restricted by gbwcthrd_tx<br>4 – The total Rx bandwidth from all address is restricted by gbwcthrd_rx<br>5 – The total Tx and total Rx bandwidth to or from all address are restricted by gbwcthrd_tx and gbwcthrd_rx | | |
| gbwcnum | Set number of group bandwidth control address | 0~64 | | This mib is only valid when gbwcmode is other then 0 (disable) |
| gbwcaddr | Set group bandwidth control address | xxxxxxxxxxxx (12 digits mac address) | | This mib is only valid when gbwcmode is other then 0 (disable). When gbwcaddr is added, the gbwcnum will be increased automatically. |
| gbwcthrd_tx | Set Tx bandwidth threshold in unit of kbps | 0 – disable Tx bandwidth control, Other – Tx bandwidth threshold in unit of kbps | 30000 | This mib is only valid when gbwcmode is other then 0 (disable) |
| gbwcthrd_rx | Set Rx bandwidth threshold in unit of kbps | 0 – disable Rx bandwidth control, Other – Rx bandwidth threshold in unit of kbps | 30000 | This mib is only valid when gbwcmode is other then 0 (disable) |
| use_efuse | Enable loading hw setting (power index, thermal, etc.) from efuse | 0 – disable, 1 – enable | | |
| ap_profile_enable | Enable/Disable multiple AP profile support | 0 – disable, 1 - enable | 0 | |
| ap_profile_num | Set profile number | Number of profile to set | 0 | When "ap_profile_add" is called, the “ap_profile_num" will be increased by 1 automatically. So, suggest to set this number to '0' first, and then issue command "ap_profile_add" to add profile subsequently |
| ap_profile_add | Add AP profile | *Note | | |

| | | | | |
|---|---|---|---|---|
| sortbyprofile | Enable sort by profile | 0 – disable, 1 - enable | 0 | This mib is only valid when ap_profile_enable is 1(enable). When sortbyporfile is 1(enabled), wlan driver will sort the profiles by SSID, and connect to APs according to the order of profiles. If sortbyporfile is 0(disable), wlan driver will connects to the AP in the profile list with the best signal strength. |
| global_vlan | Enable vlan function globally | 0 – disable, 1 – enable | | |
| is_lan | Set WAN or LAN port | 0 – WAN port, 1 – LAN port | 1 | This mib is only valid when global_vlan is 1(enable). |
| vlan_enable | Enable vlan function for this port | 0 – disable, 1 – enable | | This mib is only valid when global_vlan is 1(enable). |
| vlan_tag | Enable tag | 0 – no tag, 1 – tag | | This mib is only valid when global_vlan is 1(enable). |
| vlan_id | Set vlan id | 1~4095 | | This mib is only valid when global_vlan is 1(enable). |
| vlan_pri | Set vlan priority | 0~7 | | This mib is only valid when global_vlan is 1(enable). |
| vlan_cfi | Set vlan CFI | 0 – disable, 1 – enable | | This mib is only valid when global_vlan is 1(enable). |
| countrycode | Enable/Disable 802.11d feature | 0 – disable, 1 – enable | 1 | Currenty 802.11d is only supported in 5GHz band. Therefore this MIB always has value 0 in 2.4GHz band. |
| countrystr | Country string | “string_value”, 2 characters in max | “US” | This MIB is valid only when countrycode is 1 OR tpc_enable is 1 |
| tpc_enable | Enable/Disable 802.11h TPC feature | 0 – disable, 1 – enable | 1 | 802.11h TPC is only supported in 5GHz band. Therefore this MIB always has value 0 in 2.4GHz band. |

| | | | | |
|---|---|---|---|---|
| tpc_tx_power | The transmit power carried in the TPC report element | Two's complement integer value in units of dBm | 12 | This MIB is valid only when tpc_enable is 1 |
| tpc_link_margin | The link margin carried in TPC report element | Two's complement integer value in units of dBm | | This MIB is valid only when tpc_enable is 1 |
| lpwc | The local power constraint carried in the power constraint element | Unsigned integer value in units of dBm | | This MIB is valid only when tpc_enable is 1 |
| min_tx_power | The minimum transmit power carried in power capability element | Two's complement integer value in units of dBm | | This MIB is valid only when tpc_enable is 1. |
| max_tx_power | The maximum transmit power carried in power capability element | Two's complement integer value in units of dBm | 20 | This MIB is valid only when tpc_enable is 1. |
| ldpc_92e | Enable/Disable 92E ldpc | 0 – disable, 1 – enable | 0 | Get detail usage in section 7.44 of "Kernel_2_6_SDK_User_Guide.doc" |
| disable_txpwrlmt | Disable tw power limit function | 0 – enable tx power limit,<br>1 – disable tx power limit | | |
| disable_txpwrlmt2path | Degrade 3dB (6 tx power index) in 1T rates when tx2path and tx power limit are both enable. | 0 – disable, 1 – enable | 0 | This mib should set to 1 when the power limit table already consider the 3dB gain of tx2path. |

Note1: The default value of MIB will be '0' if it is not specified.
Note2: The values set to EDCA manually will be applied after driver close and up

# 2.3. iwconfig/iwlist

The driver supports iwconfig and iwlist (Wireless Tools v29) for getting or setting wlan configurations. Before use this feature, please do build the kernel image as following Kernel configuration:
Select *"Network device support ---> Wireless LAN (non-hamradio) ---> Wireless Extensions v18 support"* and *"Network device support ---> Wireless LAN (non-hamradio) ---> Wireless Tools v29 support"*; then rebuild kernel image.

## 2.3.1. iwconfig

configure a wlan interface.
Usage: "iwconfig <wlan_*interface*>"

"iwconfig <wlan_*interface*> [essid *X*] [mode *M*] [freq *F*] [channel *C*] [ap *A*] [rate *R*] [rts *RT*] [frag *FT*] [enc *E*] [key *K*] [retry *R*]"

wlan_interface: wlan interface, e.g., wlan0

"iwconfig --help"

"iwconfig --version"

Parameters of **iwconfig**

| Name | Meaning | Value | Access | Comment |
|---|---|---|---|---|
| essid | ESSID | any string, e.g. iwconfig essid "My SSID" | GET/SET | |
| mode | operating mode of the device | *Ad-Hoc*, *Managed* (client mode), *Master* (AP mode), *Repeater, Monitor* | GET | |
| freq | operating frequency | frequency in GHz | GET/SET | |
| channel | operating channel value | channel value | GET/SET | |
| ap | MAC address | e.g. *00:e0:4c:01:23:45* | GET | |
| rate/bit[rate] | maximum available bit rate | bit rate in Mb/s | GET | |
| rts[_threshold] | RTS threshold | packet size or off | GET/SET | |
| frag[mentation_threshold] | fragmentation threshold | packet size; off: based on driver setting | GET/SET | |
| key/enc[ryption] | WEP key settings | mode: open/restricted; keys in 10 or 32 hex-digit | GET | |
| retry | retry limits | number of retrys | GET | |

*Notes 1: for more detailed information, please refer to the manual of iwconfig.*

*Notes 2:* Because *'iwconfig'* cannot fully cover all the configurations of the AP, we suggest the users using *'iwpriv'* descript in section 2.2 to setup the AP.

## 2.3.2. iwlist

Get wireless information from a wlan interface

Usage: "iwlist <wlan_*interface*> <*keyword*>"

"iwlist --help"

"iwlist --version"

wlan_interface: wlan interface, e.g., wlan0

<keywords> of **iwlist**

| Name | Meaning | Value | Comment |
|---|---|---|---|
| scanning | site survey of neighboring WLAN devices | list of Access Points and Ad-Hoc cells in range. | |
| channel/frequency | supported channel and frequency | frequencies in GHz corresponding to the channels | varied as domain region changed |
| bitrate/rate | supported rate and extended supported rate announced in beacon | supported bit-rates in Mb/s | HT rates are not listed by iwlist |
| keys/encryption | WEP encryption information | key sizes, list of available keys and current transmit key | |
| ap/accsspoints/pe | Associated peer list | list of associated peers | |

| | | | |
|---|---|---|---|
| ers | | | |
| auth | Authentication capabilities | WPA, WPA2, CIPHER-TKIP, CIPHER-CCMP | |

*Notes 1: for more detailed information, please refer to the manual of iwlist.*

*Notes 2:* Because *'iwlist'* cannot fully cover all the configurations of the AP, we suggest the users using *ioctl* descript in section 17.3 to retrieve settings of the AP.

# 3. Basic Driver Operation and Setting

## 3.1. Basic Driver Operation

### 3.1.1. Bring up Interface

After inserting driver module, we can use the following command to bring up WLAN interface.

Usage: "ifconfig <iface> up"

iface: "wlan0"

### 3.1.2. Set MAC Address

Use the following command to change WLAN interface MAC address.

Usage: "ifconfig <iface> hw ether <addr>"

iface: "wlan0"

addr: "xxxxxxxxxxxx", for example, MAC address 00:23:45:67:89:ab maps to "0023456789ab"

### 3.1.3. Down Interface

Use the following command to bring down WLAN interface.

Usage: "ifconfig <iface> down"

iface: "wlan0"

## 3.2. Basic Driver Setting

As described previously, there are three methods to configure parameters in this package. One of these is using "iwpriv" to set MIB directly to driver. The following sections show some examples of basic driver setting.

### 3.2.1. 802.11 b/g/n mode

1. Related MIB:

(1) "band": This parameter is bit mask of band selection. The value of this parameter could be set to 1 to use 802.11b band, set to 2 to use 802.11g band, set to 8 to use 802.11n band, or set to 11 to use 802.11 b/g/n bands.

(2) "use40M": This parameter means 11n channel bonding. The value of this parameter could be set to 0 to use 20MHz channel, or set to 1 to 40MHz channel.

2. 802.11b mode example with wlan0 interface:
```
ifconfig wlan0 down
iwpriv wlan0 set_mib band=1
iwpriv wlan0 set_mib deny_legacy=0
iwpriv wlan0 set_mib use_40M=0
ifconfig wlan0 up
```
3. 802.11g mode example with wlan0 interface:
```
ifconfig wlan0 down
iwpriv wlan0 set_mib band=2
iwpriv wlan0 set_mib deny_legacy=1
iwpriv wlan0 set_mib use_40M=0
ifconfig wlan0 up
```
4. 802.11n mode example with wlan0 interface:
```
ifconfig wlan0 down
iwpriv wlan0 set_mib band=8
iwpriv wlan0 set_mib deny_legacy=3
iwpriv wlan0 set_mib use_40M=1
ifconfig wlan0 up
```
5. 802.11 b/g/n mode example with wlan0 interface:
```
ifconfig wlan0 down
iwpriv wlan0 set_mib band=11
iwpriv wlan0 set_mib deny_legacy=0
iwpriv wlan0 set_mib use_40M=1
ifconfig wlan0 up
```

### 3.2.2. Channel

1. Related MIB:

(1) "channel": This parameter means WLAN channel number. The value of this parameter could be set to 0 to use auto channel, set to 1-14 for 11b/11g, or set to 36-165 for 11a. See the channel plan in Section 1.2.

2. WLAN auto channel example with wlan0 interface:
```
ifconfig wlan0 down
iwpriv wlan0 set_mib channel=0
ifconfig wlan0 up
```

### 3.2.3. SSID

1. Related MIB:
   (1) "ssid": This parameter means WLAN SSID. The value of this parameter could be set to specific string you want.
2. WLAN ssid example with wlan0 interface:
   ifconfig wlan0 down
   iwpriv wlan0 set_mib ssid=Realtek_AP_Test
   ifconfig wlan0 up

### 3.2.4. TX Rate

1. Related MIB:
   (1) "autorate": This parameter means TX rate adaptive enable/disable. The value of this parameter could be set to 0 to disable rate adaptive, or set to 1 to enable rate adaptive.
   (2) "fixrate": This parameter means the fixed TX rate when "autorate" is disable. The value of this parameter is a bit map value that bit 0-11 for rate 1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54M and bit 12-27 for rate MCS0, MCS1, …, MCS15.
2. WLAN auto rate example with wlan0 interface:
   ifconfig wlan0 down
   iwpriv wlan0 set_mib autorate=1
   iwpriv wlan0 set_mib fixrate=0
   ifconfig wlan0 up
3. WLAN fixed rate example with wlan0 interface:
   Use MCS7 rate as the fixed rate and it maps to bit 19. The value of this bitmap is 0x80000 that it converts to 524288 on decimal.
   ifconfig wlan0 down
   iwpriv wlan0 set_mib autorate=0
   iwpriv wlan0 set_mib fixrate=524288
   ifconfig wlan0 up

### 3.2.5. Hidden AP

1. Related MIB:
   (1) "hiddenAP": This parameter means hidden AP enable or disable. The value of this parameter could be set to 0 to disable hidden AP, set to 1 to enable hidden AP.
2. WLAN hidden AP example with wlan0 interface:

```
ifconfig wlan0 down
iwpriv wlan0 set_mib hiddenAP=0
ifconfig wlan0 up
```

### 3.2.6. No Encryption

1. Related MIB:
   (1) “authtype”: This parameter means WLAN authentication type. The value of this parameter could be set to 0 to use open system, set to 1 to use shared key, or set to 2 to use both automatically.
   (2) “encmode”: This parameter means WLAN security mode. The value of this parameter is set to 0 to disable encryption.
   (3) “802_1x”: This parameter means 802.1X enable/disable. The value of this parameter is set to 0 to disable 802.1X.
2. WLAN No Encryption example with wlan0 interface:

```
ifconfig wlan0 down
iwpriv wlan0 set_mib authtype=0
iwpriv wlan0 set_mib encmode=0
iwpriv wlan0 set_mib 802_1x=0
ifconfig wlan0 up
```

# 4. Dual-band Configuration

Dual-band functions are only supported by RTL8192D series. To enable Dual MAC/PHY mode, please turn on the option, "RTL8192D dual-MAC-dual-PHY mode", in Linux kernel configuration. For Dual MAC/PHY mode, wlan0 is for 5G only, wlan1 is for 2G only.

## 4.1. Related MIB

1. "macPhyMode": setting the wlan interface to be started as Dual MAC/PHY (1T1R Concurrent Mode) or Single MAC/PHY (2T2R Selective Mode)
2. "band": setting the band for wlan interfaces. For example: 5G: 12 (A+N), 2G: 11 (B+G+N)
3. "channel": setting a correct channel according to the band setting.

More dual-band related MIB are listed in bellow table.

| | | | | |
|---|---|---|---|---|
| macPhyMode | Set dual or single MAC/PHY mode | 0 – Single MAC/PHY,<br>2 – Dual MAC/PHY | 2 | Please refer to section "Dual-band configuration" |
| channel | Operation frequency used | 0 for auto channel, 1-14 for 11b/11g, 36-165 for 11a | | |
| band | Band selection | 1 – 11b, 2 – 11g, 4 – 11a, 8 – 11n | wlan0:12<br>wlan1:11 | |
| pwrlevel5GHT40_1S_A | 40MHz mode HT OFDM 1 spatial stream Tx power level for 196 (channel 1~196) channels (392 hex digits) for path A | RF module dependent | | Type of byte array. E.g. Channel 36 should use the 36'th byte. |
| pwrlevel5GHT40_1S_B | 40MHz mode HT OFDM 1 spatial stream Tx power level for 196 (channel 1~196) channels (392 hex digits) for path B | RF module dependent | | Type of byte array. E.g. Channel 36 should use the 36'th byte. |
| pwrdiff5GHT40_2S | 40MHz mode HT OFDM 2 spatial<br>stream Tx power difference between HT40_1S for 196 | RF module dependent | | Type of byte array. E.g. Channel 36 should use the 36'th byte. |

| | | | | |
|---|---|---|---|---|
| | (channel 1~196) channels (392 hex digits). Bit[3:0] for path A. Bit[7:4] for path B. | | | |
| pwrdiff5GHT20 | 20MHz mode HT OFDM Tx power difference between HT40_1S for 196 (channel 1~196) channels (392 hex digits). Bit[3:0] for path A. Bit[7:4] for path B. | RF module dependent | | Type of byte array. E.g. Channel 36 should use the 36'th byte. |
| pwrdiff5GOFDM | Legacy OFDM Tx power difference between HT40_1S for 196 (channel 1~196) channels (392 hex digits). Bit[3:0] for path A. Bit[7:4] for path B. | RF module dependent | | Type of byte array. E.g. Channel 36 should use the 36'th byte. |

*Note 1:* if the value is the type of byte array, the format of value will be a string of ASCII of 0~f, which using 2 ASCII standing for one byte. For example, when set Tx power of pwrlevel5GHT40_1S_A, it will be

"iwpriv wlan0 set_mib pwrlevel5GHT40_1S_A=00000000000000000000000000000000000000000000000000000000000000000000000000000000282828282828282828282828282828282828282828282828282828282828280000000000000000000000000000000000000000000000000000000000000000000000000000000028282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828280000000000000000000000000000000000000000000000000000000000000000000000000000000028282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828282828280000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000"

# 4.2. Examples

## 4.2.1. Setting as 5G Single MAC/PHY selective mode

1. disable all wlan interfaces
   *> ifconfig wlan0 down*
2. setting related mibs
   *a. setting single MAC/PHY*
   *> iwpriv wlan0 set_mib macPhyMode=0*
   *b. setting band as A+N mode*
   *> iwpriv wlan0 set_mib band=12*
   *c. setting channel, e.g channel 44*

*> iwpriv wlan0 set_mib channel=44*
*d. setting other mib if necessary, such as 40M bandwidth, encryption, etc.*

3. enable wlan interface
*> ifconfig wlan0 up*

## 4.2.2. Setting as 2G Single MAC/PHY selective mode

1. disable all wlan interfaces
*> ifconfig wlan0 down*
2. setting related mibs
*a. setting single MAC/PHY*
*> iwpriv wlan0 set_mib macPhyMode=0*
*b. setting band as B+G+N mode*
*> iwpriv wlan0 set_mib band=11*
*c. setting channel, e.g channel 6*
*> iwpriv wlan0 set_mib channel=6*
*d. setting other mib if necessary, such as 40M bandwidth, encryption, etc.*
3. enable wlan interface
*> ifconfig wlan0 up*

## 4.2.3. Setting as the Dual MAC/PHY concurrent mode

1. disable all wlan interfaces
*> ifconfig wlan0 down*
*> ifconfig wlan1 down*
2. setting related mibs
*a. setting dual MAC/PHY*
*> iwpriv wlan0 set_mib macPhyMode=2*
*> iwpriv wlan1 set_mib macPhyMode=2*
*b. setting wlan0 band as A+N mode, setting wlan1 band as B+G+N mode*
*> iwpriv wlan0 set_mib band=12*
*> iwpriv wlan1 set_mib band=11*
*c. setting channel, e.g 5G channel 44, 2G channel 6*
*> iwpriv wlan0 set_mib channel=44*
*> iwpriv wlan1 set_mib channel=6*
*d. setting other mib if necessary, such as 40M bandwidth, encryption, etc.*

3. enable wlan interface

*> ifconfig wlan0 up*
*>ifconfig wlan1 up*

# 5. Security Policy

This section describes the security policy of WLAN by introducing security related MIB.

## 5.1. WEP

### 5.1.1. Related MIB

1. “authtype”: This parameter means WLAN authentication type. The value of this parameter could be set to 0 to use open system, set to 1 to use shared key, or set to 2 to use both automatically.
2. “encmode”: This parameter means WLAN security mode. The value of this parameter is set to 1 to use WEP 64 bit encryption. Set to 5 to use WEP 128 bit encryption.
3. “wepdkeyid”: This parameter means WEP default key id. The value of this parameter is set to 0~3 to decide which key id to use.
4. “wepkey1” ~ “wepkey4”: These parameters indicate value of the related WEP key id. The value is set to 10 hex digits for WEP64, 26 hex digits for WEP128

### 5.1.2. WEP 64 example

This example shows example for using WEP 64 bits encryption.

1. “authtype”: Set to 2 to use open system or shared key automatically.
2. “encmode”: Set to 1 to use WEP 64 encryption.
3. “wepdkeyid”: Set to 0 to use “wepkey1” as default key.
4. “wepkey1”: These parameters store the value of the related WEP key id. For example, the value of “wepkey1” is “0987654321” 10 characters in HEX.

### 5.1.3. WEP 128 example

This example shows example for using WEP 128 bits encryption.

1. “authtype”: Set to 2 to use open system or shared key automatically.
2. “encmode”: Set to 5 to use WEP 128 encryption.
3. “wepdkeyid”: Set to 0 to use “wepkey1” as default key.
4. “wepkey1”: These parameters store the value of the related WEP key id. For example, the value of “wepkey1” is “12345678901234567890123456” 26 characters in HEX.

# 5.2. WPA/WPA2 PSK

## 5.2.1. Related MIB

1. “encmode”: This parameter means WLAN security mode. The value of this parameter is set to 2 to use WPA/WPA2 mode.
2. “802_1x”: This parameter means 802.1X enable/disable. The value of this parameter is set to 0 to disable 802.1X.
3. “psk_enable”: This parameter is WPA PSK mode. The value of this parameter is set to 1 to use WPA encryption, or set to 2 to use WPA2 encryption, or set to 3 to use WPA/WPA2 mixed mode encryption.
4. “wpa_cipher”: This parameter means WPA cipher type. The value of this parameter is set to 2 to use TKIP, or set to 8 to use AES, or set to 10 to use AES/TKIP mixed mode.
5. “wpa2_cipher”: This parameter means WPA2 cipher type. The value of this parameter is set to 2 to use TKIP, or set to 8 to use AES, or set to 10 to use AES/TKIP mixed mode.
6. “passphrase”: This parameter means WPA PSK value. The value of this parameter is 32 characters or 64 hex digits.

## 5.2.2. WPA AES example

This example shows how to use following MIB setup WPA AES encryption.

1. “encmode”: Set to 2 to use WPA/WPA2 mode.
2. “802_1x”: Set to 0 to disable 802.1X.
3. “psk_enable”: Set to 1 to use WPA encryption.
4. “wpa_cipher”: Set to 8 to use AES.
5. “passphrase”: This parameter stores the value of the related WPA key. For example, the value of “passphrase” is “87654321” 8 characters.

## 5.2.3. WPA TKIP example

This example shows how to use following MIB to setup WPA TKIP encryption.

1. “encmode”: Set to 2 to use WPA/WPA2 mode.
2. “802_1x”: Set to 0 to disable 802.1X.
3. “psk_enable”: Set to 1 to use WPA encryption.
4. “wpa_cipher”: Set to 2 to use TKIP.
5. “passphrase”: This parameter stores the value of the related WPA key. For example, the value of “passphrase” is “87654321” 8 characters.

## 5.2.4. WPA2 AES example

This example shows how to use following MIB to setup WPA2 AES encryption.

1. “encmode”: Set to 2 to use WPA/WPA2 mode.
2. “802_1x”: Set to 0 to disable 802.1X.
3. “psk_enable”: Set to 2 to use WPA2 encryption.
4. “wpa2_cipher”: Set to 8 to use AES.
5. “passphrase”: This parameter stores the value of the related WPA2 key. For example, the value of “passphrase” is “87654321” 8 characters.

## 5.2.5. WPA2 TKIP example

This example shows how to use following MIB to setup WPA2 TKIP encryption.

1. “encmode”: Set to 2 to use WPA/WPA2 mode.
2. “802_1x”: Set to 0 to disable 802.1X.
3. “psk_enable”: Set to 2 to use WPA2 encryption.
4. “wpa2_cipher”: Set to 2 to use TKIP.
5. “passphrase”: This parameter stores the value of the related WPA2 key. For example, the value of “passphrase” is “87654321” 8 characters.

## 5.2.6. WPA/WPA2 AES mixed mode example

This example shows how to use following MIB to setup WPA/WPA2 AES mixed mode encryption.

1. “encmode”: Set to 2 to use WPA/WPA2 mode.
2. “802_1x”: Set to 0 to disable 802.1X.
3. “psk_enable”: Set to 3 to use WPA/WPA2 mixed encryption.
4. “wpa_cipher”: Set to 8 to use AES.
5. “wpa2_cipher”: Set to 8 to use AES.
6. “passphrase”: This parameter stores the value of the related WPA/WPA2 mixed mode key. For example, the value of “passphrase” is “87654321” 8 characters.

## 5.2.7. WPA/WPA2 TKIP mixed mode example

This example shows how to use following MIB to setup WPA/WPA2 TKIP encryption.

1. “encmdoe”: Set to 2 to use WPA/WPA2 mode.
2. “802_1x”: Set to 0 to disable 802.1X.
3. “psk_enable”: Set to 3 to use WPA/WPA2 mixed encryption.
4. “wpa_cipher”: Set to 2 to use TKIP.
5. “wpa2_cipher”: Set to 2 to use TKIP.

6. “passphrase”: This parameter stores the value of the related WPA/WPA2 mixed mode key. For example, the value of “passphrase” is “87654321” 8 characters.

### 5.2.8. WPA/WPA2 auto mixed mode example

This example shows how to use following MIB to setup WPA AES encryption.

1. “encmode”: Set to 2 to use WPA/WPA2 mode.
2. “802_1x”: Set to 0 to disable 802.1X.
3. “psk_enable”: Set to 3 to use WPA/WPA2 mixed encryption.
4. “wpa_cipher”: Set to 10 to use AES/TKIP.
5. “wpa2_cipher”: Set to 10 to use AES/TKIP.
6. “passphrase”: This parameter stores the value of the related WPA/WPA2 mixed mode key. For example, the value of “passphrase” is “87654321” 8 characters.

## 5.3. Mix Security Setting for Client Mode

In wireless client mode, our SDK support mix security setting, which is convenient to user, that only the passphrase and wep key need to be set for connecting to the remote AP. The encryption method used by the client is automatically chosen according to the encryption method and cipher suites of the remote AP which the user want to connect to. Users don’t bother to specify a particular encryption method and therefore alleviate the cumbersome of setting process.
The client would choose the highest security setting supported by remote AP. The rule is: when AP is using WPA or WPA2, the SDK will use MIB “passphrase“, and choose the encryption method and a cipher suite according to the order AES > WPA2 > OTHER that the AP have supported; when AP is using WEP, the SDK will use MIBs “wepdkeyid “ and “wepkey1”~“wepkey4” to connect to it; when AP is an open system, the client just connect to it directly. For example:
If AP is using WPA2/WPA - AES/TKIP, the client will use WPA2-AES to connect.
If AP is using WPA - AES/TKIP, the client will use WPA-AES to connect.
If AP is using WPA2/WPA-TKIP, the client will use WPA2-TKIP to connect.
Though, the AP setting of the last of two examples above is not allowed nowadays.

To enable this feature, please go to “make menuconfig” and enable the option “Client mix security Support” as follows:

```
[*] Config kernel (NEW)      --->
    Device Drivers    --->
      Network device support    --->
        Wireless LAN    --->
          [*]     Client Mode support
            [*]           Client mix security Support
```

### 5.3.1. Related MIB

Only following MIBs matter when mix security support is enabled.

1. “passphrase”: This parameter means WPA PSK value. The value of this parameter is 32 characters or 64 hex digits.
2. “wepdkeyid”: This parameter means WEP default key id. The value of this parameter is set to 0~3 to decide which key id to use.
3. “wepkey1” ~ “wepkey4”: These parameters indicate value of the related WEP key id. The value is set to 10 hex digits for WEP64, 26 hex digits for WEP128

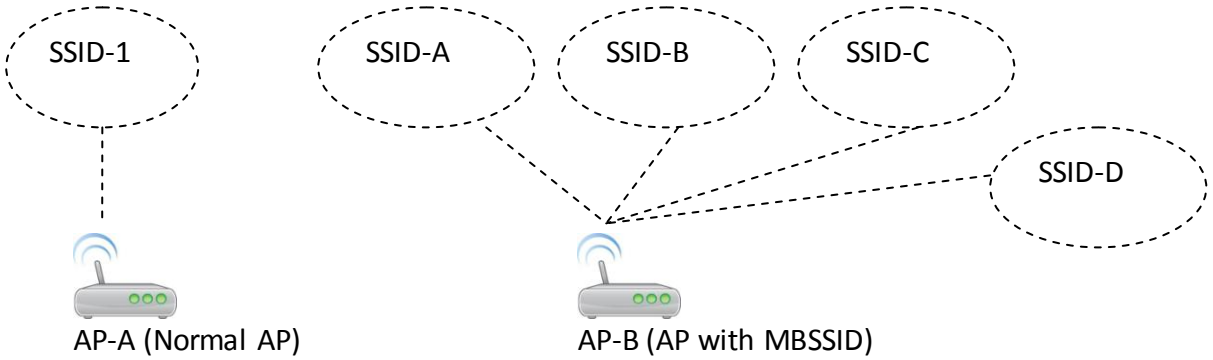
# 6. Dynamic Frequency Selection (DFS)

DFS is enabled if Linux kernel configuration "DFS support" is enabled. To obey regulation, DFS channels can ONLY be selected by auto-channel selection. The user can see "Auto (DFS)" on the channel column on web UI. If the user want to force the DUT set in a DFS channel for evaluation purpose, one should set console command with "**flash set WLAN0_CHANNEL *<channel #>***", and then reboot. See 5G channel plan in section 1.2.2.

Note: Alternatively, the user can use http://192.168.1.254/syscmd.asp to input the command.

# 7. Multiple BSSID

In the usual, the Access-Point (AP) has one SSID, we can regard it as a wireless LAN (WLAN). If an AP has feature of multiple-BSSID (MBSSID) and enable the feature with 4 SSIDs, the AP has four WLANs. Use below figure as an example: When Wi-Fi client scanned the APs, there are five WLANs scanned by the Wi-Fi client, but there are only two real APs in the environment.




The feature of MBSSID would let one AP to look like more than one, so the feature sometimes also is called virtual access point (VAP).

## 7.1. Related MIB

1. "vap_enable": This parameter is used to turn MBSSID on/off. The value of this parameter is set to 0 to turn off MBSSID feature, and is set to 1 to turn on MBSSID feature

MBSSID feature would create several interfaces which depend by user setup in the system. The first interface is called root-AP interface (wlan0), and the other interfaces are called VAP interfaces (wlan0-va0, wlan0-va1…and etc).

Some settings of VAP interfaces use the same settings with root-AP such like radio parameters, and some settings of VAP interfaces should be disable such like WDS parameters. The other settings of VAP interfaces usually do not need to modify, but some parameters should be set its value with different interface (Please see below table)

| Name | Meaning | Value | Default | Comment |
|---|---|---|---|---|
| ssid | SSID | "string_value", SSID with 32 characters in max | | |
| opmode | Operation mode (AP) | 16 – AP | 16 | This should be always AP. |
| block_relay | Block packet relaying between associated clients | 0 – relay, 1 – block relay and drop, 2 – block relay and indicate to bridge | | |
| authtype | 802.11 Authentication type | 0 – open system, 1 – shared key, 2 – auto | 2 | |
| encmode | Encryption mode | 0 – disabled, 1 – WEP64, 2 – TKIP, 4 – AES(CCMP), 5 – WEP128 | | Set to 2 always under WPA/WPA2 mode |
| wepdkeyid | WEP default Tx key | 0-3 | | |
| psk_enable | PSK mode | 0 – disable, 1 – WPA, 2 – WPA2, 3 – WPA/WPA2 mixed | | |
| wpa_cipher | WPA PSK cipher suite | 2 –TKIP, 8 – AES(CCMP), 10 – TKIP/AES mixed | | |
| wpa2_cipher | WPA2 PSK cipher suite | 2 –TKIP, 8 – AES(CCMP), 10 – TKIP/AES mixed | | |
| 802_1x | Flag of using 802.1x | 0 – disable, 1 – enable | | When 802.1x is enabled, the Auth daemon must be invoked |
| wepkey1 | WEP key1 | 10 hex digits for WEP64, 26 hex digits for WEP128 | | Type of byte array |
| wepkey2 | WEP key2 | 10 hex digits for WEP64, 26 hex digits for WEP128 | | Type of byte array |
| wepkey3 | WEP key3 | 10 hex digits for WEP64, 26 hex digits for WEP128 | | Type of byte array |
| wepkey4 | WEP key4 | 10 hex digits for WEP64, 26 hex digits for WEP128 | | Type of byte array |

# 7.2. Example of Setup

## 7.2.1. Enable MBSSID feature and Open two VAP interfaces

ifconfig wlan0-va0 down
ifconfig wlan0-va1 down

```
ifconfig wlan0 down
iwpriv wlan0 set_mib vap_enable=1
ifconfig wlan0 up
ifconfig wlan0-va0 up
ifconfig wlan0-va1 up
brctl addif br0 wlan0-va0
brctl addif br0 wlan0-va1
```

### 7.2.2. Stop one VAP interface

```
ifconfig wlan0-va1 down
brctl delif br0 wlan0-va1
```

### 7.2.3. Disable MBSSID feature and stop two VAP interfaces

```
ifconfig wlan0-va0 down
ifconfig wlan0-va1 down
ifconfig wlan0 down
brctl delif br0 wlan0-va0
brctl delif br0 wlan0-va1
iwpriv wlan0 set_mib vap_enable=0
ifconfig wlan0 up
```

### 7.2.4. Set VAP SSID

The other setup processes are alike as Chapter 5 and Chapter 6, but setup commands need to change interface name. Here, use MIB-“ssid” in VAP0 to be an example:

```
ifconfig wlan0-va0 down
iwpriv wlan0-va0 set_mib ssid=Realtek_AP_VAP0_Test
ifconfig wlan0-va0 up
```

## 7.3. Important Notes

- Wi-Fi Interface limitation:
  - If user wants to use MBSSID feature, the root-AP (wlan0) interface must not be stopped.
  - User should use “func_off” to enable/disable Wi-Fi functions on root-AP.
- Hardware limitation: Max size of MBSSID is 8 including root-AP.
- Software limitation:
  - User could modify the definition —“RTL8192CD_NUM_VWLAN” to change the number of supported VAP. The max size of MBSSID is “RTL8192CD_NUM_VWLAN” + 1.
  - The value of “RTL8192CD_NUM_VWLAN” should be smaller than 8(hardware limitation)

and bigger than -1.

- If user wants VAP to be independent with other VAPs, user should implement blocking method between VAPs. The method should be implemented by user in different system.

# 8. WMM

This section describes the WMM (Wireless Multimedia) feature and the related configuration.

## 8.1. Feature Description

WMM is a Wi-Fi Alliance interoperability certification, based on the IEEE 802.11e standard. It provides basic Quality of service (QoS) features to IEEE 802.11 networks. WMM prioritizes traffic according to four Access Categories (AC) - voice, video, best effort, and background. However, it does not provide guaranteed throughput. It is suitable for simple applications that require QoS, such as Voice over IP (VoIP) on Wi-Fi phones (VoWLAN).

The Wi-Fi Alliance has added Power Save Certification (WMM-APSD) to the WMM specification. Power Save uses mechanisms from 802.11e and legacy 802.11 to save power (for battery powered equipment) and fine-tune power consumption. The certification provides an indication that the certified product is targeted for power critical applications like hand-phones and portable power devices.

## 8.2. Related MIB

1. “qos_enable”: This parameter means WMM enable/disable. The value of this parameter could be set to 0 to disable WMM, or set to 1 to enable WMM.
2. “apsd_enable”: This parameter means WMM-APSD enable/disable. The value of this parameter could be set to 0 to disable WMM-APSD, or set to 1 to enable WMM-APSD.

# 9. WDS

## 9.1. Feature Description

The WDS (Wireless Distribution System) is wireless interconnection of 802.11 APs. It uses MAC address of peer APs to create wireless connection to expand network topology instead of wired cable that is most used in distribution system.

The AP which enables WDS is just like traditional AP that can accept wireless clients. The APs which create WDS connection must use the same channel. And the disadvantage of WDS is throughput will be halved for clients which connect wirelessly.

In Realtek WLAN system architecture, we can support 8 WDS number in current configuration.

## 9.2. Related MIB

1. “wds_enable”: This parameter means WDS enable/disable. The value of this parameter could be set to 0 to disable WDS, or set to 1 to enable WDS.
2. “wds_pure”: This parameter means WDS pure mode. The AP which enable WDS pure mode will not broadcast beacon and will not accept any station to connect. The value of this parameter is set to 0 to disable WDS pure mode, or set to 1 to enable WDS pure mode.
3. “wds_priority”: This parameter gives WDS packets higher priority. The value of this parameter could be set to 0 to disable WDS packets higher priority, or set to 1 to enable WDS packets higher priority.
4. “wds_num”: This parameter means WDS setting numbers. The value of this parameter is set to 0 if driver is re-initialized and adds by one if adding a new WDS setting.
5. “wds_add”: This parameter is used for setting MAC address of peer WDS AP and the rate sent to the peer WDS AP. The value of this parameter includes two parts. The first part is 12 digits MAC address. After MAC address, there is a 32-bit variable to give the rate that bit 0-11 for 1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54M and bit 12-27 for MCS0, MCS1, …, MCS15.
6. “wds_encrypt”: This parameter means WDS encryption mode. The value of this parameter could be set to 0 to disable encryption, set to 1 to use WEP 64 bit encryption, set to 2 to use TKIP, set to 4 to use AES, or set to 5 to use WEP 128 bit encryption.
7. “wds_wepkey”: This parameter indicates value of the WDS WEP default key. The value is set to 10 hex digits for WEP64 or 26 hex digits for WEP128.
8. “wds_passphrase”: This parameter indicates value of the WDS PSK key. The value is set to 32 characters or 64 hex digits.

# 9.3. WDS Example

These examples show how to use following MIB to setup WDS environment with different encryption types. We assume there are two AP, one is AP A and another is AP B. And we will setup WDS connection between AP A and AP B. The MAC address of AP A is 01:23:45:67:89:AA. The MAC address of AP B is 01:23:45:67:89:BB. The WLAN interface of both AP is wlan0.

## 9.3.1. WDS with No Encryption Example

The AP A needs to setup the following MIB. Assume the encryption of WDS connection is no encryption and the transmission rate is set to auto. The wds_add MIB need to set the WDS peer's MAC address, that is, use MAC address of AP B in this example.

```
ifconfig wlan0 down
iwpriv wlan0 set_mib wds_enable=1
iwpriv wlan0 set_mib wds_pure=0
iwpriv wlan0 set_mib wds_priority=1
iwpriv wlan0 set_mib wds_num=0
iwpriv wlan0 set_mib wds_encrypt=0
iwpriv wlan0 set_mib wds_add=0123456789BB,0
ifconfig wlan0 up
```

The AP B needs to setup the following MIB.

```
ifconfig wlan0 down
iwpriv wlan0 set_mib wds_enable=1
iwpriv wlan0 set_mib wds_pure=0
iwpriv wlan0 set_mib wds_priority=1
iwpriv wlan0 set_mib wds_num=0
iwpriv wlan0 set_mib wds_encrypt=0
iwpriv wlan0 set_mib wds_add=0123456789AA,0
ifconfig wlan0 up
```

## 9.3.2. WDS with WEP 64 bit Encryption Example

The AP A needs to setup the following MIB. Assume the encryption of WDS connection is WEP 64 bit encryption and the transmission rate is set to auto. The wds_add MIB need to set the WDS peer's MAC address, that is, use MAC address of AP B in this example.

```
ifconfig wlan0 down
iwpriv wlan0 set_mib wds_enable=1
iwpriv wlan0 set_mib wds_pure=0
iwpriv wlan0 set_mib wds_priority=1
```

```
iwpriv wlan0 set_mib wds_num=0
iwpriv wlan0 set_mib wds_encrypt=1
iwpriv wlan0 set_mib wds_wepkey=12345
iwpriv wlan0 set_mib wds_add=0123456789BB,0
ifconfig wlan0 up
```

The AP B needs to setup the following MIB.

```
ifconfig wlan0 down
iwpriv wlan0 set_mib wds_enable=1
iwpriv wlan0 set_mib wds_pure=0
iwpriv wlan0 set_mib wds_priority=1
iwpriv wlan0 set_mib wds_num=0
iwpriv wlan0 set_mib wds_encrypt=1
iwpriv wlan0 set_mib wds_wepkey=12345
iwpriv wlan0 set_mib wds_add=0123456789AA,0
ifconfig wlan0 up
```

### 9.3.3. WDS with AES Encryption Example

The AP A needs to setup the following MIB. Assume the encryption of WDS connection is AES and the transmission rate is set to auto. The wds_add MIB need to set the WDS peer's MAC address, that is, use MAC address of AP B in this example.

```
ifconfig wlan0 down
iwpriv wlan0 set_mib wds_enable=1
iwpriv wlan0 set_mib wds_pure=0
iwpriv wlan0 set_mib wds_priority=1
iwpriv wlan0 set_mib wds_num=0
iwpriv wlan0 set_mib wds_encrypt=4
iwpriv wlan0 set_mib wds_passphrase =12345678
iwpriv wlan0 set_mib wds_add=0123456789BB,0
ifconfig wlan0 up
```

The AP B needs to setup the following MIB.

```
ifconfig wlan0 down
iwpriv wlan0 set_mib wds_enable=1
iwpriv wlan0 set_mib wds_pure=0
iwpriv wlan0 set_mib wds_priority=1
iwpriv wlan0 set_mib wds_num=0
iwpriv wlan0 set_mib wds_encrypt=4
iwpriv wlan0 set_mib wds_passphrase =12345678
iwpriv wlan0 set_mib wds_add=0123456789AA,0
```

ifconfig wlan0 up

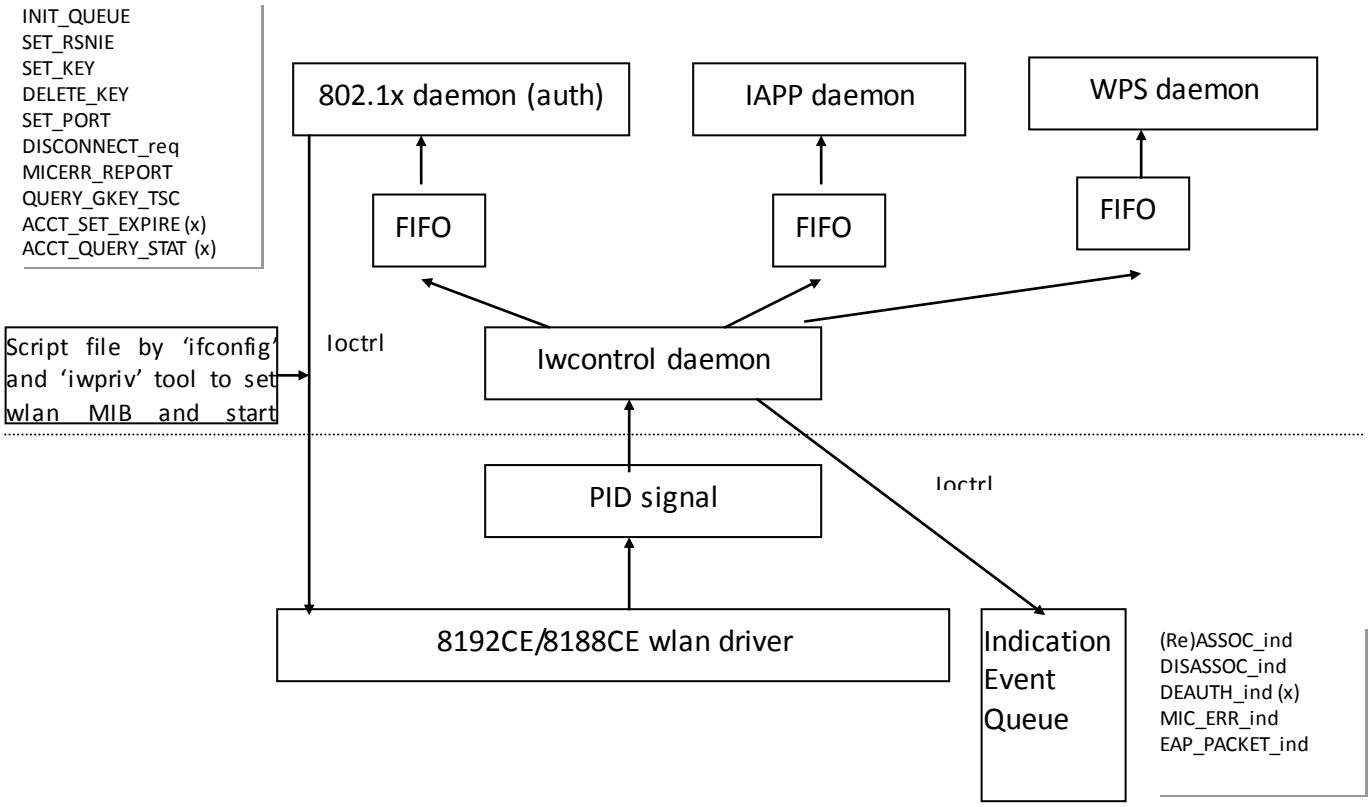
## 9.4. Important Notes

After configuring WDS successfully, system will add a new interface wlan0-wds0 which maps to wds_number=0. Currently, our system supports 8 wds number now.

# 10. 802.1X

## 10.1. System Architecture

Below block diagram shows the system architecture of the RTL819X driver and software package.




## 10.2. 802.1X Related Daemons Introduction

### 10.2.1. iwcontrol Daemon Introduction

As the block diagram of the system architecture shows, the iwcontrol daemon should be started after 802.1X, IAPP, or WPS daemon is running. We can start iwcontrol daemon by running the following command.

"iwcontrol <wlan_interface> …"

wlan_interface: wlan interface, e.g., wlan0

The iwcontrol daemon will parse the pid files in "/var/run" and create FIFO files to do IPC with WPS, IAPP, and 1x daemon. And multiple wireless interfaces can be supported in iwcontrol parameters.

### 10.2.2. 802.1X Daemon Introduction

As the block diagram of the system architecture shows, the iwcontrol daemon should be started after 802.1X, IAPP, or WPS daemon is running. We can start 802.1X daemon by running the following command.

"auth <wlan_interface> <lan_interface> auth <wpa_conf> &"
wlan_interface: wlan interface, e.g., wlan0
lan_interface: lan interface, which connects to Radius server, e.g., br0
auth: denote to act as authenticator
wpa_conf: path of wpa config file, e.g., /var/wpa-wlan0.conf

The 802.1X daemon will create PID file "/var/run/auth-wlanx.pid" for each 802.1X daemon. And multiple 802.1x daemons will be created for different wireless interfaces.

## 10.3. 802.1X Start Procedure

The start procedure is to start the 802.1X and iwcontrol daemons. Before starting the 802.1X daemon, it should configure the 802.1X related parameters and decide the role of the authentication procedure. After the parameters are configured, it will call flash utility to generate configuration file.

"flash wpa <wlan_interface> < wpa_conf > <wlan_interface>"
wlan_interface: wlan interface, e.g., wlan0
wpa_conf: path of wpa config file, e.g., /var/wpa-wlan0.conf

Then it should start the 802.1X and iwcontrol daemons by previous described commands. We can use "ps" command to check if these two daemon started successfully. The last step of the procedure is to use brctl utility to configure the correct network interface information to connect to the radius server.

## 10.4. WPA Config File

### 10.4.1. Config File Related Parameters

The parameter format in wpa config file is like "keyword = value". Below table shows wpa parameters in wpa config file.

| keyword | value | Comment |
|---|---|---|
| encryption | 0 – disable, 1 – WEP, 2 – WPA, 4 – WPA2 only, 6 | |

| | | |
|---|---|---|
| | – WPA2 mixed | |
| ssid | "string_value", 1-32 char | |
| enable1x | 0/1 – disable/enable 1x Radius authentication | Refer when encryption is set to 0, 1 |
| enableMacAuth | 0/1 – disable/enable MAC authentication | |
| SupportNonWpaClient | 0/1 – disable/enable none WPA client support when WPA is set | This feature is not supported now |
| wepKey | 1 – WEP64, 2 – WEP128 | Refer when encryption is set 1 (wep) |
| wepGroupKey | set "" as default | No use |
| authentication | 1 – Radius, 2 – PSK (pre-shared key) | |
| unicastCipher | 1 – TKIP, 2 – AES | |
| wpa2UnicastCipher | 1 – TKIP, 2 – AES | |
| usePassphrase | 0 – use psk value as key in raw data, 1 – use passphrase algorithm to convert psk value | |
| psk | "string_value", if usePassphrase=0 (raw data), it should be 64 hex digits. If usePassphrase=1, the string length should be >=8 and <=64. | |
| groupRekeyTime | Group key re-key time | No use |
| rsReAuthTo | The time in second which force client to do re-auth to the server. Set to 0 to disable this function. | |
| rsPort | UDP Port number of radius server | Normally 1812 is used |
| rsIP | IP address of radius server (e.g., 192.168.1.1) | |
| rsPassword | "string_value", password of radius server with 31 char in max | |
| rs2Port | UDP Port number of radius server set 2 | Normally 1812 is used |
| rs2IP | IP address of radius server (e.g., 192.168.1.1) set 2 | |
| rs2Password | "string_value", password of radius server with 31 char in max set 2 | |
| rsMaxReq | Max retry number of request packet with radius server | Set 3 as default |
| rsAWhile | Timeout time (in second) of waiting rsp packet of radius server | Set 5 as default |
| accountRsEnabled | 0/1 – disable/enable accounting radius server | |
| accountRsPort | UDP Port number of accounting radius server | |
| accountRsIP | IP address of accounting radius server | |
| accountRsPassword | "string_value", password of accounting radius server with 31 char in max | |
| accountRsUpdateEnabled | 0/1 – disable/enable the feature of statistic update with accounting server | |
| accountRsUpdateTime | Update time in seconds | |
| accountMaxReq | Max retry number of request packet with accounting radius server | |

| | | |
|---|---|---|
| accountAWhile | Timeout time (in second)of waiting rsp packet of accounting radius server | |
| MaxPmksa | Maximum cache number of PMKSA | 0 is default, then PMK cache mechanism is not enabled. If not 0, AP will maintain tuples of (station MAC address, pmksa) until reboot. When exceed limit, the oldest entry will be flushed. |

# 10.5. 802.1X Example

The following section shows WAP config file samples for the 802.1x deamon using.

We assume the IP address of the radius server is 172.20.10.2. The port number of the radius server is 1812. The password of the radius server is 12345678.

## 10.5.1. 802.1X with WPA AES Example

**encryption = 2**
ssid = "RTK_AP"
**enable1x = 1**
enableMacAuth = 0
supportNonWpaClient = 0
wepKey = 2
wepGroupKey = ""
**authentication = 1**
**unicastCipher = 2**
wpa2UnicastCipher = 3
usePassphrase = 1
psk = "12345678"
groupRekeyTime = 86400
rsReAuthTO = 0
**rsPort = 1812**
**rsIP = 172.20.10.2**
**rsPassword = "12345678"**
rsMaxReq = 3
rsAWhile = 5
accountRsEnabled = 0
accountRsPort = 0
accountRsIP = 0.0.0.0
accountRsPassword = ""
accountRsUpdateEnabled = 0
accountRsUpdateTime = 0

accountRsMaxReq = 0
accountRsAWhile = 0

## 10.5.2. 802.1X with WPA2 AES Example

**encryption = 4**
ssid = "RTK_AP"
**enable1x = 1**
enableMacAuth = 0
supportNonWpaClient = 0
wepKey = 2
wepGroupKey = ""
**authentication = 1**
unicastCipher = 1
**wpa2UnicastCipher = 2**
usePassphrase = 1
psk = "12345678"
groupRekeyTime = 86400
rsReAuthTO = 0
**rsPort = 1812**
**rsIP = 172.20.10.2**
**rsPassword = "12345678"**
rsMaxReq = 3
rsAWhile = 5
accountRsEnabled = 0
accountRsPort = 0
accountRsIP = 0.0.0.0
accountRsPassword = ""
accountRsUpdateEnabled = 0
accountRsUpdateTime = 0
accountRsMaxReq = 0
accountRsAWhile = 0ifconfig wlan0 down

## 10.5.3. 802.1X with WPA/WPA2 AES/TKIP mixed mode Example

**encryption = 6**
ssid = "RTK_AP"
**enable1x = 1**
enableMacAuth = 0
supportNonWpaClient = 0

wepKey = 2
wepGroupKey = ""
**authentication = 1**
**unicastCipher = 3**
**wpa2UnicastCipher = 3**
enablePreAuth = 0
usePassphrase = 1
psk = "12345678"
groupRekeyTime = 86400
rsReAuthTO = 0
**rsPort = 1812**
**rsIP = 172.20.10.2**
**rsPassword = "12345678"**
rsMaxReq = 3
rsAWhile = 5
accountRsEnabled = 0
accountRsPort = 0
accountRsIP = 0.0.0.0
accountRsPassword = ""
accountRsUpdateEnabled = 0
accountRsUpdateTime = 0
accountRsMaxReq = 0
accountRsAWhile = 0

# 11. IAPP

As the block diagram of the system architecture shown in section 10.1, the iwcontrol daemon should be started after 802.1X, IAPP, or WPS daemon is running. We can start IAPP daemon by running the following command.

"iapp <lan_interface> <wlan_interface> &"

lan_interface: lan interface which IAPP deamon used to send IAPP packet, e.g., br0

wlan_interface: wlan interface, e.g., wlan0

The IAPP daemon will create PID file "/var/run/iapp.pid" for each IAPP daemon. And multiple IAPP daemons will be created for different wireless interfaces.

# 12. WPS

Please refer to *"Realtek_WPS_user_guide.doc"* for detail explanation and usages.

# 13. IGMP Snooping

According to IEEE 802.11 specification, the broadcast/multicast frames sent by an AP are not acknowledged by the STA. It implied the broadcast/multicast frame may not be received by the STA. When the wireless client joins a multicast group to play video streaming, missing multicast frames will cause the video picture indistinct.

In contrast with the broadcast/multicast frame, the unicast frame in which sender requires acknowledgement. The acknowledgement and retransmission mechanism can greatly reduce the possibility of frame lost in the receiver. When an AP does multicast-to-unicast translation, it will improve the quality of the streaming video. In this case, the AP must record which wireless client had joined which multicast group, and then sent translated unicast frames to those belong to this multicast group. Otherwise, the AP sent translated unicast frames to wireless clients that have not joined this multicast group. It would resume the network bandwidth of Wi-Fi, especially for the streaming video application.

Wi-Fi IGMP snooping, as implied by name, is a feature that allows an AP to listen on the IGMP conversation between Wireless clients and multicast server. By listening to these IGMP conversations the AP maintains a map of which wireless client need which IP multicast streams.

## 13.1. The process flow of IGMP snooping

Step1. After translating 802.11 frame to 802.3 Ethernet frame, do the following check:

(1) Is destination MAC address multicast address?

(2) Is the EtherType field of the MAC header 0x0800 (IPv4)?

(3) Is the Protocol field of the IPv4 header 0x02 (IGMP)?

If all above conditions are true, then move Step2. Otherwise, exit the process.

Step2. Check the content of IGMP packet (reference the function ___igmp_type_check)

(1) When receiving the IGMP packet belongs to the following type, then add the group multicast address to the corresponding wireless client's multicast table.

- IGMP Type = 0x12 (IGMP_HOST_MEMBERSHIP_REPORT)
- IGMP Type = 0x16 (IGMPV2_HOST_MEMBERSHIP_REPORT)
- IGMP Type = 0x22 (IGMPV3_HOST_MEMBERSHIP_REPORT) and Group Record Type = 0x04 (IGMPV3_CHANGE_TO_EXCLUDE)

(2) When receiving the IGMP packet belongs to the following type, then remove the group multicast address from the corresponding wireless client's multicast table.

- IGMP Type = 0x17 (IGMP_HOST_LEAVE_MESSAGE)
- IGMP Type = 0x22 (IGMPV3_HOST_MEMBERSHIP_REPORT) and Group Record Type = 0x03 (IGMPV3_CHANGE_TO_INCLUDE)

(3) When receiving the IGMP packet not belong to the above (1)(2) type, or the IP header

contain illegal content, then exit the process.

Step3. Update the wireless client's multicast table.

(1) Convert the group multicast IP address to the multicast MAC address

(2) Call ioctl() with ioctl cmd 0x8B80(SIOCGIMCAST_ADD) / 0x8B81(SIOCGIMCAST_DEL), multicast MAC address and source MAC address to update the corresponding wireless client's multicast table.

Step4. When the AP receives the multicast frame from other interface, it will check the multicast table for each wireless client. If the specific multicast MAC address exists in the certain wireless clients' multicast table, the AP will do multicast-to-unicast translation and send the translated unicast frame to these. If the specific multicast MAC address not found in the wireless client's multicast table, the AP will not send any frame to it.

## 13.2. Related MIB

1. "mc2u_disable": This parameter is used to turn the Wi-Fi IGMP snooping on/off. The value of this parameter is set to 0 to turn on Wi-Fi IGMP snooping function. The other value is to turn off Wi-Fi IGMP snooping function.

## 13.3. IGMP snooping Example

If you did not do multicast-to-unicast translation, you must turn off the Wi-Fi IGMP snooping function. After turning off, the AP will directly send the multicast frame without translating it. The corresponding commands, as below:

```
ifconfig wlan0 down
iwpriv wlan0 set_mib mc2u_disable=1
ifconfig wlan0 up
```

If you want to turn on the Wi-Fi IGMP snooping function, you must do the following commands, as below:

```
ifconfig wlan0 down
iwpriv wlan0 set_mib mc2u_disable=0
ifconfig wlan0 up
```

# 14. Multiple AP profile

In wireless client mode, our SDK could provide the feature to set multiple AP profiles (e.g., SSID and security setting) into driver. When booting up, wlan driver will look for AP according to these profiles. If any one AP is found, it will associate to it with the configured security.
Before using this feature, please do build the kernel image as following kernel configuration:
Run kernel menuconfig in SDK. Enable Multiple AP profile support as follows:

```
Network device support    --->
  Wireless LAN (non-hamradio)    --->
    [*]    RTL8192C/D 802.11b/g/n support
    [*]    Client Mode support
    [ ]        Repeater Mode support
    [*]        Multiple AP profile Support
```

Enable "Client mode support" and then “Multiple AP profile support”. Save the kernel config and rebuild the image.

## 14.1. Related MIB

| Name | Meaning | Value | Default | Comment |
|---|---|---|---|---|
| ap_profile_enable | Enable/Disable multiple AP profile support | 0 – disable, 1 - enable | 0 | |
| ap_profile_num | Set profile number | Number of profile to set | 0 | When "ap_profile_add" is called, the “ap_profile_num" will be increased by 1 automatically. So, suggest to set this number to '0' first, and then issue command "ap_profile_add" to add profile subsequently |
| ap_profile_add | Add AP profile | *Note | | |
| sortbyprofile | Enable sort by profile | 0 – disable, 1 - enable | 0 | This mib is only valid when ap_profile_enable is 1(enable). When |

| | | | | |
|---|---|---|---|---|
| | | | | sortbyporfile is 1(enabled), wlan driver will sort the profiles by SSID, and connect to APs according to the order of profiles. If sortbyporfile is 0(disable), wlan driver will connects to the AP in the profile list with the best signal strength. |

**Note:**

If wireless security is open system, its value format is:

**ssid,sec_type,auth_type**

**ssid** - SSID of associated AP, which length is from 1~32 bytes

**sec_type** - Security type. 0 - open, 1 - wep64, 2- wep128, 3 - wpa, 4 - wpa2

**auth_type** - Authentication type. 0 - open, 1 - shared key, 2 - auto. Please note, only in wep mode, you could set auth_type in 1 or 2.

Example:

Add AP profile with SSID=”open-ssid” and open authentication:

iwpriv wlan0 set_mib ap_profile_add="open-ssid",0,0

If wireless security is wep 64-bit encryption, its value format is:

**ssid,sec_type,auth_type,default_key,key1,key2,key3,key4**

**default_key** - Default Tx key id (0~4).

**key1~key4**: WEP key in 10 bytes hex string.

Example:

Add AP profile with SSID=”wep64-ssid”, open authentication and key in “111111111” hex

iwpriv wlan0 set_mib ap_profile_add="wep64-ssid",1,0,0,"1111111111","1111111111","1111111111","1111111111"

If wireless security is wep 128-bit encryption, its value format is:

**ssid,sec_type,auth_type,default_key,key1,key2,key3,key4**

**key1~key4**: WEP key in 13 bytes hex string.

Example:

Add AP profile with SSID=”wep124-ssid”, open authentication and key in “111111111” hex

iwpriv wlan0 set_mib ap_profile_add="wep128-ssid",2,0,0,"1111111111111111111111111",
"1111111111111111111111111","1111111111111111111111111",,"11111111111111111111111"

If wireless security is wpa psk, its value format is:

**ssid,sec_type,auth_type,cipher,psk**

**cipher**: WPA cipher. 2 - TKIP, 8 - AES.

**psk** - WPA pre-shared-key in ASCII string (length 8~63). When length is set to 64, it will be thought as hex value,

and its binary value will be used directly without converting PSK in wifi driver.

Example:

Add AP profile with SSID=”wpa-ssid”, open authentication, TKIP with PSK in “1234567890”

iwpriv wlan0 set_mib ap_profile_add="wpa-ssid",3,0,2,"1234567890"

If wireless security is wpa2 psk, its value format is:

**ssid,sec_type,auth_type,cipher,psk**

Example:

Add AP profile with SSID=”wpa2-ssid”, open authentication, AES with PSK in “1234567890”

iwpriv wlan0 set_mib ap_profile_add="wpa2-ssid",4,0,8,"1234567890"

Note1: The maximum number of profile is set to 5.

Note2: In our SDK, the wlan profile mib is configured by ioctl (0x8b43) directly through Flash utility. Not by iwpriv command described above.

## 14.2. How to debug

You may issue the following command from console to dump the all profile information and show which profile is used now as:

cat /proc/wlan0/mib_ap_profile

# 15. Mesh

According to 802.11s, each mesh nodes do passive scan (listen to beacon) to find other candidate mesh nodes, to witch peer link would be established. We compare Mesh ID and other mesh capabilities to search candidate mesh nodes. If we found, the peering procedures would be started. Only when the peering is successfully established, we then call these nodes "neighbor", data can be transmitted to it. For mesh routing, 802.11s requires Hybrid Wireless Mesh Protocol, or HWMP be supported as a default routing protocol, which use MAC addresses (layer 2 routing) and uses a radio-aware routing metric for the calculation of best paths to every mesh nodes.

In our driver, we use a special interface "wlan-msh" for mesh connectivity, which should be open after the mesh-related MIB is correctly set. These mesh-related MIB setup mesh capabilities and decide which mesh network to join. Beside these MIB, which is listed in 15.1, every mesh nodes should also be setup with same band, control sideband and channel number. If the configuration is correct, mesh network will construct automatically. Several proc files can be used to check mesh connect status (See 15.4)

We use pathsel deamon to realize mesh routing protocol. Therefore, after the wlan-msh is open, we need to start the pathsel deamon as well. Section 15.2 shows how to start up the pathsel deamon. Section 15.3 shows some examples to setup the mesh network. All these mesh connect status can be retrieved using proc files for debugging or web display (See 15.4).

## 15.1. Related MIB

1. "mesh_enable": set to 1 to enable mesh function, set to 0 to disable it. The following MIB are only valid when mesh_enable is 1.
2. "mesh_ap_enable": set to 1 to enable access point function; set to 0 to disable it. If this parameter is 0, it means this mesh station is a pure mesh node (without access point).
3. "mesh_id": a string with max 32 characters which is used to identify a mesh network according to 802.11s. We use this parameter to find candidate mesh nodes, to which we would try to establish peering.
4. "mesh_privacy": set to 4 to enable mesh data encryption using WPA2(AES); set to 0 to disable it. Currently, mesh only support WPA2(AES) encryption.
5. "mesh_passphrase": this parameter indicates value of the Mesh PSK key. The value is set to up to 32 characters or 64 hex digits. Mesh securities settings are independent to the settings used by AP. Be sure of all the mesh nodes use the exactly pre-shared key or the mesh nodes will be connected but data is unreachable.
6. "mesh_max_neightbor": the number of peer link this mesh node can establish. You can set this parameter up to 15. The default value is 15. If current peer link count reaches the

mesh_max_neightbor, additional mesh peering would not be accepted.

7. “mesh_igmp_enable”: set to 1 to enable mesh igmp function, set to 0 to disable it. The default value is 1.
8. “meshaclmode”: Access control mode between mesh nodes. Set to 0 to disable ACL, set to 1 to accept, set to 2 to deny. Default is 0. “meshaclnum” and “meshacladdr” are valid only when this parameter is 1 or 2.
9. “meshaclnum”: the number of ACL. Up to 128 ACL can be set.
10. “meshacladdr”: the array of mac address of mesh nodes to be deny or accept. The content of this mib depends on the value of “meshaclnum”. If “meshaclnum” is 1, this mib contains one xxxxxxxxxxxx (12 digits mac address). If “meshaclnum” is 2, this mib contains two xxxxxxxxxxxx (12 digits mac address), totally 24 digits. If there is one mesh node in the deny list or not in the accept list, the peer link to this mesh node would not be accept.

# 15.2. Pathsel Deamon

As mesh use pathsel deamon to create and maintain the mesh routing table. It should be start after mesh interface, ”wlan-msh”, is open.
Run following command to start pathsel daemon.

“pathsel -i wlan-msh -P -d”

The pathsel daemon will create a PID file “/var/run/pathsel-wlan-msh.pid”. Please note that only one pathsel deamon need to be start even though we have more than one wireless interfaces. The command, “ps”, can be used to check if the pathsel daemon started successfully.

# 15.3. Example

## 15.3.1. Enable Mesh

Step 1: Enable mesh function

```
iwpriv wlan0 set_mib mesh_enable=1
```

Step 2: Enable ap function (optional)

```
iwpriv wlan0 set_mib mesh_ap_enable=1
```

Step 3: Set Mesh ID

```
iwpriv wlan0 set_mib mesh_id=RTK_mesh_5g
```

Step 4: Enable mesh data encryption (optional)

```
iwpriv wlan0 set_mib mesh_privacy=4
iwpriv wlan0 set_mib mesh_passphrase=12345678
```

Step 5: Set ACL (optional)

```
iwpriv wlan0 set_mib meshaclmode=1
iwpriv wlan0 set_mib meshaclnum=2
```

iwpriv wlan0 set_mib meshacladdr=00e04c424bd300e04c424bd5

Step 6: Start up wlan0 and wlan-msh interface

ifconfig wlan0 up
ifconfig wlan-msh up
brctl addif br0 wlan0
brctl addif br0 wlan-msh

Step 7: Start up pathsel deamon

pathsel -i wlan-msh -P –d

## 15.3.2. Disable Mesh

Step 1: Stop pathsel deamon, the pid of pathsel deamon can be found using “ps” command

kill <pid of pathsel deamon>

Step 2: Stop wlan0 and wlan-msh interface

ifconfig wlan-msh down
ifconfig wlan0 down
brctl delif br0 wlan-msh

Step 3: Disable mesh function, and then bring up wlan0 interface

iwpriv wlan0 set_mib mesh_enable=0
ifconfig wlan0 up

## 15.3.3. Enable Dual-Band Mesh

Step 1: Set mesh-related MIB for both wireless interfaces

iwpriv wlan0 set_mib mesh_enable=1
iwpriv wlan0 set_mib mesh_ap_enable=1 (optional)
iwpriv wlan0 set_mib mesh_id=RTK_mesh_5g
iwpriv wlan0 set_mib mesh_privacy=4 (optional)
iwpriv wlan0 set_mib mesh_passphrase=12345678 (optional)
iwpriv wlan0 set_mib meshaclmode=1 (optional)
iwpriv wlan0 set_mib meshaclnum=2 (optional)
iwpriv wlan0 set_mib meshacladdr=00e04c424bd300e04c424bd5 (optional)

iwpriv wlan1 set_mib mesh_enable=1
iwpriv wlan1 set_mib mesh_ap_enable=1 (optional)
iwpriv wlan1 set_mib mesh_id=RTK_mesh_2g
iwpriv wlan1 set_mib mesh_privacy=4 (optional)
iwpriv wlan1 set_mib mesh_passphrase=12345678 (optional)
iwpriv wlan1 set_mib meshaclmode=1 (optional)
iwpriv wlan1 set_mib meshaclnum=2 (optional)
iwpriv wlan1 set_mib meshacladdr=00e04c424bd300e04c424bd5 (optional)

Step 2: Start up wlan0, wlan1 and wlan-msh interface

ifconfig wlan0 up
ifconfig wlan1 up
ifconfig wlan-msh up
brctl addif br0 wlan0
brctl addif br0 wlan1
brctl addif br0 wlan-msh

Step 3: Start up pathsel deamon
pathsel -i wlan-msh -P –d

### 15.3.4. Disable Dual-Band Mesh

Step 1: stop pathsel deamon, the pid of pathsel deamon can be found using “ps” command
kill <pid of pathsel deamon>

Step 2: stop wlan0, wlan1, and wlan-msh interface
ifconfig wlan-msh down
ifconfig wlan0 down
ifconfig wlan1 down
brctl delif br0 wlan-msh

Step 3: disable mesh function, and then bring up wlan0 and wlan1 interface
iwpriv wlan0 set_mib mesh_enable=0
iwpriv wlan1 set_mib mesh_enable=0
ifconfig wlan0 up
ifconfig wlan1 up

## 15.4. Debug and Web Information

CAT following files under ‘/proc/wlan0’ and ‘/proc/wlan1’(in dual-band mode) for debugging or web display:

1. “mesh_stats”: show mesh statistics and capabilities
2. “mesh_assoc_mpinfo”: show information of all mesh neighbors
3. “mesh_portal_table”: show all the mesh portal
4. “mesh_pathsel_routetable”: show mesh routing table
5. “mesh_proxy_table”: show mesh proxy table

In dual-band mode, the information in “mesh_portal_table”, “mesh_pathsel_routetable” and “mesh_proxy_table” would be same in both interfaces.

# 16. Mass Production

Please refer to *"8192C Linux Driver MP.doc"* for detail explanation and usages.

# 17. Other User Space Utilities

## 17.1. iwpriv Utility

### 17.1.1. Read WLAN register

Usage: "iwpriv <iface> read_reg <type,offset>"
iface: "wlan0"
type: b - for byte, w – for word, dw – for double word
offset: the register offset in hex

### 17.1.2. Write WLAN register

Usage: "iwpriv <iface> write_reg <type,offset,value>"
iface: "wlan0"
type: b - for byte, w – for word, dw – for double word
offset: the register offset in hex
value: value for write in hex

### 17.1.3. Read WLAN memory

Usage: "iwpriv <iface> read_mem <type,start,len>"
iface: "wlan0"
type: b - for byte, w – for word, dw – for double word
offset: the memory start address in hex
len: read length in hex

### 17.1.4. Write WLAN memory

Usage: "iwpriv <iface> write_mem <type,start,len,value>"
iface: "wlan0"
type: b - for byte, w – for word, dw – for double word

offset: the memory start address in hex
len: read length in hex
value: value for write in hex

## 17.2. Proc Files

Files under '/proc/wlan0':

6. "cam_info": dump h/w encryption cam content
7. "mib_xxx": show mib info
8. "sta_info": show all associated station info
9. "sta_keyinfo": show the encryption keys of all associated station info
10. "txdesc": show tx descriptor contents for queue 0 to queue 5
11. "rxdesc": show rx descriptor contents
12. "buf_info": show the internal buffer pointers and counts
13. "desc_info": show tx and rx descriptor pointers, indexes, and register contents
14. "stats": show Tx, Rx, and beacon statistics as follow:
    - *up_time – driver uptime*
    - *tx_packets – total tx packet numbers*
    - *tx_bytes – total tx byte counts*
    - *tx_retrys – total tx retry counts*
    - *tx_fails – total tx failed numbers*
    - *tx_drops – total tx dropped counts*
    - *rx_packets – total rx packet numbers*
    - *rx_bytes – total rx byte counts*
    - *rx_retrys – total rx retry counts*
    - *rx_crc_errors – total rx CRC error packet numbers*
    - *rx_errors – total rx error packet numbers (including CRC error, ICV error, etc.)*
    - *rx_data_drops – total rx data dropped counts other than sequence number issue*
    - *rx_decache – total rx data dropped counts due to sequence number duplicated*
    - *rx_fifoO – total rx fifo overflow counts*
    - *rx_rdu – total rx buffer under run counts*
    - *beacon_ok – total transmitted OK beacons*
    - *beacon_er – total transmitted failed beacons*
    - *freeskb_err – total error pointers of tx skb numbers*
    - *dz_queue_len – total queued packet numbers for sleeping sta*
    - *check_cnt_tx – internal tx status check counts*

- *check_cnt_rst – internal driver status check counts*
- *reused_skb – reused skb numbers*
- *skb_free_num – free skb numbers*
- *tx_average – average of tx flow*
- *rx_average – average of rx flow*
- *cur_tx_rate – current tx rate*

15. “mib_EDCA”: show the EDCA parameters will be applied when enabled
16. “*.txt“: MAC and PHY parameter files

# 17.3. IOCTL

IOCTL commands (for web display):

| id | Meaning | Input | output | comment |
|---|---|---|---|---|
| 0x8b30 | Get station info | None | 64 array of sta_info_2_web (note1) | |
| 0x8b31 | Get associated station number | None | 1 word (2 bytes) | |
| 0x8b32 | Get version information | None | 2 byte of version infomation | |
| 0x8b33 | Issue scan request | None | 1 byte of result (-1:fail, 0: success) | |
| 0x8b34 | Get scan result and scanned BSS database | 1 byte flag (get BSS database or not) | 4 bytes of number of entries and array of bss_desc (note4) with flag set to 0 | |
| 0x8b35 | Issue join request | bss_desc to join | 1 byte of result (0: success, 1: scanning, 2: fail) | |
| 0x8b36 | Get join result | None | 1 byte of result (note5) | |
| 0x8b37 | Get BSS info | None | Bss_info_2_web structure (note2) | This is used typically in client mode |
| 0x8b38 | Get WDS info | None | 8 array of wds_info (note3) | |

Note1:

```
typedef struct _sta_info_2_web {
        unsigned short      aid;
        unsigned char  addr[6];
```

```
	unsigned long  tx_packets;
	unsigned long  rx_packets;
	unsigned long  expired_time;
	unsigned short	flags; // bit2 indicate whether this entry is valid, bit3 indicates if sta is in sleeping
	unsigned char  TxOperaRate; // current used tx rate in 500 k bps (e.g., 108 for 55M)
	unsigned char  rssi; // received signal strength indication
	unsigned long  link_time; // 1 sec unit
	unsigned long  tx_fail;
	unsigned long  tx_bytes;
	unsigned long  rx_bytes;
	unsigned char  network;
	unsigned char  ht_info;
	unsigned char  RxOperaRate; // current used tx rate in 500 k bps (e.g., 108 for 55M)
	unsigned char  resv[3];
	unsigned char  acTxOperaRate; // for AC capable WLAN IC
} sta_info_2_web;

Note2:
typedef enum _wlan_mac_state {
	STATE_DISABLED=0, STATE_IDLE, STATE_SCANNING, STATE_STARTED, STATE_CONNECTED, STATE_WAITFORKEY
} wlan_mac_state;

typedef struct _bss_info_2_web {
	unsigned char state;	// defined in wlan_mac_state
	unsigned char channel;
	unsigned char txRate;
	unsigned char bssid[6];
	unsigned char rssi, sq;
	unsigned char ssid[33];
} bss_info_2_web;

Note3:
typedef struct _wds_info {
	unsigned char  state;
	unsigned char  addr[6];
	unsigned long  tx_packets;
	unsigned long  rx_packets;
	unsigned long  tx_errors;
	unsigned char  TxOperaRate;
} wds_info;
```

Note4:

```
struct ibss_priv {
        unsigned short      atim_win;    };
struct bss_desc {
        unsigned char  bssid[6];
        unsigned char  ssid[32];
        unsigned char  *ssidptr;
        unsigned short      ssidlen;
        unsigned char  meshid[32];
        unsigned char  *meshidptr;
        unsigned short      meshidlen;
        unsigned int    bsstype;
        unsigned short      beacon_prd;
        unsigned char  dtim_prd;
        unsigned long  t_stamp[2];
        struct ibss_priv      ibss_par;
        unsigned short      capability;
        unsigned char  channel;
        unsigned long  basicrate;
        unsigned long  supportrate;
        unsigned char  bdsa[6];
        unsigned char  rssi;
        unsigned char  sq;
        unsigned char  network;
};
```

Note5:
0xff: pending
2-4: success
others: fail

# 18. Hardware Limitation

## 18.1. Limitation

1. H/W encryption CAM size is 32
2. Multiple BSSID CAM size is 8
3. Tx SKB buffer must have 8 bytes space in tail for TKIP MIC
4. Support 31 wlan clients in current configuration for 8192CE/8188RE/8192DE/8812E and 63 wlan clients for 8188ER
5. Support 8 WDS number in current configuration