

RALINK TECHNOLOGY, CORP.

RALINK RT3883RT3662 LINUX STATION RELEASE NOTES & USER'S GUIDE

Copyright © 2011 Ralink Technology, Corp.

All Rights Reserved.

This document is property of Ralink Technology Corporation. Transmittal, receipt, or possession of this document does not express, license, or imply any rights to use, sell, design, or manufacture from this information or the software documented herein. No reproduction, publication, or disclosure of this information, in whole or in part, shall be allowed, unless the prior written consent of Ralink Technology Corporation is obtained.

NOTE: THIS DOCUMENT CONTAINS SENSITIVE INFORMATION AND HAS RESTRICTED DISTRIBUTION.

**for stevej@cradlepoint.com
And Company Use Only**



Proprietary Notice and Liability Disclaimer

The confidential Information, technology or any Intellectual Property embodied therein, including without limitation, specifications, product features, data, source code, object code, computer programs, drawings, schematics, know-how, notes, models, reports, contracts, schedules and samples, constitute the Proprietary Information of Ralink (hereinafter "Proprietary Information")

All the Proprietary Information is provided "AS IS". No Warranty of any kind, whether express or implied, is given hereunder with regards to any Proprietary Information or the use, performance or function thereof. Ralink hereby disclaims any warranties, including but not limited warranties of non-infringement, merchantability, completeness, accuracy, fitness for any particular purpose, functionality and any warranty related to course of performance or dealing of Proprietary Information. In no event shall Ralink be liable for any special, indirect or consequential damages associated with or arising from use of the Proprietary Information in any way, including any loss of use, data or profits.

Ralink retains all right, title or interest in any Proprietary Information or any Intellectual Property embodied therein. The Proprietary Information shall not in whole or in part be reversed, decompiled or disassembled, nor reproduced or sublicensed or disclosed to any third party without Ralink's prior written consent.

Ralink reserves the right, at its own discretion, to update or revise the Proprietary Information from time to time, of which Ralink is not obligated to inform or send notice. Please check back if you have any question. Information or items marked as "not yet supported" shall not be relied on, nor taken as any warranty or permission of use.

Ralink Technology Corporation (Taiwan)

5F, No.5, Tai-Yuen 1st St,
Jhubei City
HsinChu County 30265, Taiwan, ROC
Tel +886-3-560-0868
Fax +886-3-560-0818

Sales Taiwan: Sales@ralinktech.com.tw

Technical Support Taiwan: F AE@ralinktech.com.tw

<http://www.ralinktech.com/>

Contents

1	VERSION HISTORY	7		
1.1.1	Version 2.4.2.1	7	2.2.36	HT_AutoBA
1.1.2	Version 2.4.2.2	7	2.2.37	HT_AMSDU
1.1.3	Version 2.4.3.0	7	2.2.38	HT_BAWinSize
1.1.4	Version 2.4.3.3	7	2.2.39	HT_Gl
1.1.5	Version 2.4.3.4	7	2.2.40	HT_MCS
1.1.6	Version 2.4.3.6	7	2.2.41	HT_MIMOPSMODE
1.1.7	Version 2.6.0.0	7	2.2.42	HT_BADecline=Value
			2.2.43	HT_DisallowTKIP
2	CONFIGURATION.....	8	2.2.44	BeaconLostTime
2.1	CONFIGURATION FILE: RT2860STA.DAT	8	2.2.45	AutoRoaming
2.2	USAGE.....	10	2.2.46	MacAddress.....
2.2.1	CountryRegion	10	2.2.47	AutoConnect.....
2.2.2	CountryRegionForABand.....	10	2.2.48	ETxBfEnCond=Value
2.2.3	CountryCode	11	2.2.49	StreamMode=Value
2.2.4	SSID	11	2.2.50	P2P_GOIntent.....
2.2.5	WirelessMode	11	2.2.51	P2P_DevName.....
2.2.6	Channel.....	11	2.2.52	P2P_ListChannel.....
2.2.7	BGProtection.....	12	2.2.53	P2P_OpChannel.....
2.2.8	TxPreamble	12		
2.2.9	RTSThreshold	12	2.3	MORE INFORMATION.....
2.2.10	FragThreshold	12	3	WIRELESS TOOLS.....
2.2.11	TxBurst	12	3.1	IWPRIV USAGE.....
2.2.12	PktAggregate	12	3.1.1	DriverVersion.....
2.2.13	NetworkType.....	12	3.1.2	CountryRegion.....
2.2.14	AuthMode	13	3.1.3	CountryRegionABand
2.2.15	EncryptType.....	13	3.1.4	SSID
2.2.16	DefaultKeyID	13	3.1.5	WirelessMode.....
2.2.17	WEP KeyType	13	3.1.6	TxBurst:
2.2.18	WEP Hex Key	13	3.1.7	PktAggregate:
2.2.19	WEP Key String.....	14	3.1.8	TxPreamble:
2.2.20	WPAPSK	14	3.1.9	TxPower:.....
2.2.21	WmmCapable	14	3.1.10	Channel
2.2.22	IEEE80211H.....	14	3.1.11	BGProtection:
2.2.23	PSMode	14	3.1.12	RTSThreshold:.....
2.2.24	AutoRoaming	14	3.1.13	FragThreshold:
2.2.25	RoamThreshold	14	3.1.14	NetworkType:
2.2.26	TGnWiFiTest.....	15	3.1.15	AuthMode:
2.2.27	WirelessEvent.....	15	3.1.16	EncryptType:
2.2.28	CarrierDetect.....	15	3.1.17	DefaultKeyID:
2.2.29	AutoReconnect.....	15	3.1.18	Key1.....
2.2.30	AdhocN.....	15	3.1.19	Key2.....
2.2.31	HT_RDG.....	15	3.1.20	Key3.....
2.2.32	HT_EXTCHA	15	3.1.21	Key4.....
2.2.33	HT_OpMode.....	16	3.1.22	WPAPSK.....
2.2.34	HT_MpduDensity	16	3.1.23	WmmCapable.....
2.2.35	HT_BW	16	3.1.24	IEEE80211H.....
			3.1.25	PSMode
			3.1.26	ResetCounter.....

3.1.27	Debug.....	26	3.2.24	WmmCapable.....	34
3.1.28	CarrierDetect.....	26	3.2.25	IEEE80211H.....	34
3.1.29	HtRdg.....	26	3.2.26	NetworkType.....	34
3.1.30	HtExtcha.....	27	3.2.27	AuthMode.....	34
3.1.31	HtOpMode.....	27	3.2.28	EncrypType.....	34
3.1.32	HtMpduDensity.....	27	3.2.29	DefaultKeyID.....	34
3.1.33	HtBw.....	27	3.2.30	Key1.....	34
3.1.34	HtAutoBa.....	27	3.2.31	Key2.....	34
3.1.35	HtAmsdu.....	28	3.2.32	Key3.....	34
3.1.36	HtBaWinSize.....	28	3.2.33	Key4.....	34
3.1.37	HtGi.....	28	3.2.34	WPAPSK.....	34
3.1.38	HtMcs.....	28	3.2.35	AutoReconnect.....	34
3.1.39	HtProtect.....	28	3.2.36	WPAPSK.....	35
3.1.40	HtMimoPs.....	28	3.2.37	PMK.....	35
3.1.41	FixedTxMode.....	29			
3.1.42	LongRetry.....	29	3.3	IWPRIV EXAMPLES.....	35
3.1.43	ShortRetry.....	29	3.3.1	Infrastructure.....	35
3.1.44	HtBaDecline=Value.....	29	3.3.2	Ad-Hoc.....	36
3.1.45	HtDisallowTKIP=Value.....	30	3.3.3	Get site survey.....	36
3.1.46	wsc_ap_band=Value.....	30	3.3.4	Get Statistics.....	36
3.1.47	BeaconLostTime=Value.....	30	3.3.5	ANY SSID.....	37
3.1.48	AutoRoaming=Value.....	30			
3.1.49	SiteSurvey=Value.....	30	3.4	IWLST.....	37
3.1.50	TdlsCapable =Value.....	31			
3.1.51	TdlsSetup=Value.....	31	3.5	IWCONFIG.....	37
3.1.52	AutoReconnect=value.....	31			
3.1.53	AdhocN=value.....	31	4 WPS – WI-FI PROTECTED SETUP.....38		
3.1.54	ETxBfEnCond=Value.....	31	4.1	IWPRIV USAGE.....	38
3.1.55	StreamMode=Value.....	31	4.1.1	wsc_conf_mode.....	38
3.2	IWPRIV SHOW USAGE.....	32	4.1.2	wsc_mode.....	39
3.2.1	SSID.....	32	4.1.3	wsc_pin.....	39
3.2.2	WirelessMode.....	32	4.1.4	wsc_ssid.....	39
3.2.3	TxBurst.....	32	4.1.5	wsc_start.....	39
3.2.4	TxPreamble.....	32	4.1.6	wsc_stop.....	39
3.2.5	TxPower.....	32	4.1.7	wsc_gen_pincode.....	40
3.2.6	Channel.....	32	4.1.8	wsc_cred_count.....	40
3.2.7	BGProtection.....	32	4.1.9	wsc_cred_ssid.....	40
3.2.8	RTSThreshold.....	32	4.1.10	wsc_cred_auth.....	40
3.2.9	FragThreshold.....	32	4.1.11	wsc_cred_encr.....	41
3.2.10	HtBw.....	32	4.1.12	wsc_cred_keyidx.....	41
3.2.11	HtMcs.....	32	4.1.13	wsc_cred_key.....	41
3.2.12	HtGi.....	33	4.1.14	wsc_cred_mac.....	42
3.2.13	HtOpMode.....	33	4.1.15	wsc_conn_by_idx.....	42
3.2.14	HtExtcha.....	33	4.1.16	wsc_auto_conn.....	42
3.2.15	HtMpduDensity.....	33	4.1.17	wsc_ap_band.....	42
3.2.16	HtBaWinSize.....	33	4.1.18	Wsc4digitPinCode.....	43
3.2.17	HtRdg.....	33	4.1.19	WscV2Support (WPS2.0).....	43
3.2.18	HtAmsdu.....	33			
3.2.19	HtAutoBa.....	33	4.2	WPS STA AS AN ENROLLEE OR REGISTRAR	43
3.2.20	CountryRegion.....	33	4.2.1	Enrollee Mode.....	43
3.2.21	CountryRegionABand.....	33	4.2.2	Registrar Mode.....	44
3.2.22	CountryCode.....	33			
3.2.23	PktAggregate.....	33	4.3	WPS IOCTL USAGE.....	45

4.3.1	<i>iwpriv commands without argument</i>		6.1.17	<i>ATERXFER</i>	61
	45		6.1.18	<i>ATESHOW</i>	61
4.3.2	<i>iwpriv commands with one INT</i>		6.1.19	<i>ATEHELP</i>	61
	<i>argument</i>	45	6.1.20	<i>ResetCounter</i>	61
4.3.3	<i>iwpriv commands with string</i>		6.1.21	<i>ATERRF</i>	61
	<i>argument</i>	46	6.1.22	<i>ATEWRF1</i>	61
4.4	WPS IOCTL SAMPLE PROGRAM	46	6.1.23	<i>ATEWRF2</i>	61
5	WIFI DIRECT - P2P COMMAND	49	6.1.24	<i>ATEWRF3</i>	62
5.1	IWPRIV USE	50	6.1.25	<i>ATEWRF4</i>	62
5.1.1	<i>P2pOpMode</i>	50	6.1.26	<i>ATEAUTOALC</i>	62
5.1.2	<i>p2pLisCh</i>	50	6.1.27	<i>ATEIPG</i>	62
5.1.3	<i>p2pOpCh</i>	50	6.1.28	<i>ATEPAYLOAD</i>	62
5.1.4	<i>p2pGoInt</i>	50	6.2	Tx MODE, MCS, BW AND GI SELECTION TABLE	62
5.1.5	<i>p2pDevName</i>	51	6.3	EXAMPLES	63
5.1.6	<i>p2pWscMode</i>	51	6.3.1	<i>Check EVM & Power</i>	64
5.1.7	<i>p2pWscConf</i>	51	6.3.2	<i>Check Carrier</i>	64
5.1.8	<i>p2pScan</i>	51	6.3.3	<i>Check spectrum mask</i>	64
5.1.9	<i>p2pInv</i>	52	6.3.4	<i>Frequency offset tuning</i>	64
5.1.10	<i>p2pDevDisc</i>	52	6.3.5	<i>Rx</i>	65
5.1.11	<i>p2pLink</i>	52	6.3.6	<i>Show all ate parameters</i>	65
5.1.12	<i>p2pCfg</i>	52	6.3.7	<i>Online help</i>	65
5.1.13	<i>p2pTab</i>	52	6.3.8	<i>Display Rx Packet Count and RSSI</i>	66
5.1.14	<i>p2pProv</i>	52	6.4	IWPRIV RAO BBP [PARAMETERS]=[VALUE]	66
5.1.15	<i>p2pStat</i>	52	6.4.1	<i>BBPID</i>	66
5.1.16	<i>p2pReset</i>	53	6.4.2	<i>BBPID=Value</i>	67
5.1.17	<i>p2pPerTab</i>	53	6.5	IWPRIV RAO MAC [PARAMETERS]=[VAL]	67
5.1.18	<i>p2pDefConfMthd</i>	53	6.5.1	<i>MAC_OFFSET</i>	67
5.1.19	<i>p2pLinkDown</i>	53	6.5.2	<i>MAC_OFFSET=Value</i>	67
5.1.20	<i>p2pSigmaEnable</i>	53	6.6	IWPRIV RAO E2P [PARAMETERS]=[VAL]	67
5.1.21	<i>P2P example:</i>	53	6.6.1	<i>EEP_ADDR</i>	67
6	ATE TEST COMMAND FORMAT	56	6.6.2	<i>EEP_ADDR=Value</i>	67
6.1	IWPRIV RAO SET [PARAMETERS]=[VAL]	58	6.7	EXAMPLE	67
6.1.1	<i>ATE</i>	58	6.7.1	<i>Hardware access</i>	68
6.1.2	<i>ATEDA</i>	58	6.7.2	<i>Statistic counter operation</i>	68
6.1.3	<i>ATESA</i>	58	6.7.3	<i>Suggestion:</i>	68
6.1.4	<i>ATEBSSID</i>	58	6.8	<i>ATED</i>	68
6.1.5	<i>ATECHANNEL</i>	59	6.8.1	<i>Introduction</i>	68
6.1.6	<i>ATETXPOW0</i>	59	6.8.2	<i>Environment setup</i>	68
6.1.7	<i>ATETXPOW1</i>	59	6.8.3	<i>How to use ated for ATE purpose</i>	69
6.1.8	<i>ATETXFREQOFFSET</i>	59	7	IOCTL	70
6.1.9	<i>ATETXLEN</i>	59	7.1	PARAMETERS FOR IWCONFIG	70
6.1.10	<i>ATETXCNT</i>	59	7.2	PARAMETERS FOR IWPRIV	73
6.1.11	<i>ATETXMODE (Refer to TxMode)</i>	59			
6.1.12	<i>ATETXBW (Refer to TxMode)</i>	60			
6.1.13	<i>ATETXGI (Refer to TxMode)</i>	60			
6.1.14	<i>ATETXMCS (Refer to TxMode)</i>	60			
6.1.15	<i>ATETXANT</i>	60			
6.1.16	<i>ATERXANT</i>	60			

7.2.1	Set Data, Parameters the same as iwpriv	73	8.1.14	GET station's wireless mode:.....	87
7.2.2	Get Data, Parameters is the same as iwpriv	74	8.1.15	GET Bss type:.....	88
7.2.3	Set Raw Data with Flags	74	8.1.16	GET Authentication Mode:.....	88
7.2.4	Get Raw Data with Flags	78	8.1.17	GET Encryption Type:	89
7.2.5	Set Raw Data with Flags	83	8.1.18	GET RSSI 0 (unit: db).....	90
8	IOCTL HOW TO	84	8.1.19	GET RSSI 1 (unit: db).....	90
8.1	GET DATA	84	8.1.20	GET RSSI 2 (unit: db).....	90
8.1.1	GET IPv4 and MAC mapping table:	84	8.1.21	GET Driver wireless extension version	91
8.1.2	GET IPv6 and MAC mapping table:	84	8.1.22	GET station WPS query status	91
8.1.3	GET value of clone MAC in Dongle mode:	84	8.2	HOW TO DISPLAY RATE, BW:.....	91
8.1.4	GET station connection status:	84	8.3	SET DATA FOR N MODE.....	93
8.1.5	GET station statistics information:	85	8.3.1	SET HT mode:	93
8.1.6	GET AP list table:.....	85	9	EFUSE BUFFER MODE HOWTO	95
8.1.7	GET scan table:	85	9.1	IN RT2860STA.DAT	95
8.1.8	GET station's MAC:	86	9.1.1	EfuseBufferMode=Value	95
8.1.9	GET station connection status:	86	9.2	BY IWPRIV COMMAND.....	95
8.1.10	GET AP's BSSID.....	86	9.2.1	efuseFreeNumber=Value	95
8.1.11	GET SSID.....	86	9.2.2	efuseDump=Value	95
8.1.12	GET station's last TX related information:.....	87	9.2.3	efuseBufferModeWriteBack=Value	96
8.1.13	GET station's last RX related information:.....	87	9.2.4	efuseLoadFromBin.....	96

1 VERSION HISTORY

1.1.1 Version 2.4.2.1

1. Add 20/40 rescan trigger function by manual.
2. Fix ApCli pkt aggregation can not work when ApCli connect to Ralink legacy AP
3. Add BF capable information to Beacon.
4. Update BF code.

1.1.2 Version 2.4.2.2

1. Fix eTxBF can not work when peer are 1x1 client.
2. Patch StreamMode code.
3. Fix when igmpsnooping enable, multicast Qos queue incorrect issue.

1.1.3 Version 2.4.3.0

1. Support iTxBF.
2. Support ATE Calibration for iTxBF.
3. Fix eTxBF throughput low issue on long rang.
4. Fix throughput low when GreenAP enable.
5. Update TxBF code.

1.1.4 Version 2.4.3.3

1. Update iTxBF code.
2. Add new ATE command for iTxBF.
3. Disable iTxBF when AP config to 1x1 AP.
4. Disable STA iTxBF function.
5. Fix issue: AP is not obeying the Buffer size of Station in BA.
6. Do VCORcalibration of regular (10 secs).
7. Send out sounding after do VCORcalibration when eTxBF enable.
8. Change DMA/MAC/PBF Tx/Rx stop operation for iTxBF DividerCalibration.

1.1.5 Version 2.4.3.4

1. **Fix CountryRegion read wrong from EEPROM.**

1.1.6 Version 2.4.3.6

1. Modify to support beamforming-awared rate adaptation algorithm when enable compile flag "NEW_RATE_ADAPT_SUPPORT".
2. Add range enhancement feature when enable compile flag "RANGE_EXT_SUPPORT".
3. Fix ATE issues.
4. Fix lgmp snooping issues.

1.1.7 Version 2.6.0.0

1. Support WPS 2.0 STA.
2. Fixed link status is not connected when security is WAPI.
3. Fix does not recognize implicit block ACK buffer size.



4. Support P2P for station mode.
5. Fixed Ad-hoc issue: STA didn't use legacy rate when STA is Ad-hoc creator and security is OPEN-WEP or WPAPSK-TKIP.
6. Fixed WPS issue: Beacon lost when STA tries to do WPS with WPS AP or STA connects to hidden AP.
7. Fixed issue: STA supports the WPA authentication which group cipher is WEP.
8. Fixed Ad-hoc issue: STA didn't use legacy rate when STA is Ad-hoc creator and security is OPEN-WEP or WPAPSK-TKIP.
9. System will crash when STA is registrar in ad-hoc mode.
10. STA WPS IE is invalid in unicast probe req.
11. STA supports the WPA authentication which group cipher is WEP.
12. WPS failed with Ralink WpaSupplicant when STA's privacy is different from AP's privacy.
13. STA Update scan table maintain mechanism.
14. HT IE wasn't included in broadcast probe request.
15. Ralink WpaSupplicant - Scan table didn't include RSSI.

2 CONFIGURATION

RT3883/3662 driver can be configured via following interfaces, i.e.

1. configuration file
2. "iwconfig" command
3. "iwpriv" command

NOTE:

- modify configuration file "RT2860STA.dat" in /etc/Wireless/RT2860STA/RT2860STA.dat.
- iwconfig/iwpriv comes with kernel.
- iwpriv usage, please refer to below sections for details.

2.1 Configuration File: RT2860STA.dat

```
# Copy this file to /etc/Wireless/RT2860STA/RT2860STA.dat
# This file is a binary file and will be read on loading rt.o module.
#
# Use "vi -b RT2860STA.dat" to modify settings according to your need.
#
# 1.) set NetworkType to
#     "Adhoc" for using Adhoc-mode,
#     otherwise using Infrastructure
# 2.) set Channel to
#     "0" for auto-select on Infrastructure mode
# 3.) set SSID for connecting to your Access-point.
# 4.) AuthMode can be
#     "WEPAUTO",
#     "OPEN",
#     "SHARED",
#     "WPAPSK",
#     "WPA2PSK",
#     "WPANONE"
# 5.) EncrypType can be
#     "NONE",
#     "WEP",
#     "TKIP",
```




```
# "AES"
# for more information refer to the Readme file.
#
#The word of "Default" must not be removed
Default
CountryRegion=5
CountryRegionABand=7
CountryCode=
SSID=Dennis2860AP
NetworkType=Infra
WirelessMode=9
Channel=0
BasicRate=15
BeaconPeriod=100
TxPower=100
BGProtection=0
TxPreamble=0
RTSThreshold=2347
FragThreshold=2346
TxBurst=1
PktAggregate=0
WmmCapable=0
AckPolicy=0;0;0;0
AuthMode=OPEN
EncrypType=NONE
WPAPSK=
DefaultKeyID=1
Key1Type=0
Key1Str=
Key2Type=0
Key2Str=
Key3Type=0
Key3Str=
Key4Type=0
Key4Str=
PSMode=CAM
AutoRoaming=0
RoamThreshold=70
HT_RDG=1
HT_EXTCHA=0
HT_OpMode=1
HT_MpduDensity=4
HT_BW=1
HT_AutoBA=1
HT_AMSDU=0
HT_BAWinSize=64
HT_GI=1
HT_MCS=33
HT_MIMOPSMODE=3
IEEE80211H=0
TGnWifiTest=0
WirelessEvent=0
CarrierDetect=0
```

Note:

Ralink Website document

**for stevej@cradlepoint.com
And Company Use Only**



WMM parameters

WmmCapable ; Set it as 1 to turn on WMM Qos support
AckPolicy1~4 ; Ack policy which support normal Ack or no Ack
; (AC_BK, AC_BE, AC_VI, AC_VO)

All WMM parameters do not support iwpriv command but 'WmmCapable', please store all parameter to RT2860STA.dat, and restart driver.

2.2 Usage

Syntax is 'Param'='Value' and describes below.

SectionNumber Parameter

Value:

...
...
...

2.2.1 CountryRegion

Value:

Region	Channels
0	1-11
1	1-13
2	10-11
3	10-13
4	14
5	1-14
6	3-9
7	5-13

2.2.2 CountryRegionForABand

Value:

Region	Channels
0	36, 40, 44, 48, 52, 56, 60, 64, 149, 153, 157, 161, 165
1	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140
2	36, 40, 44, 48, 52, 56, 60, 64
3	52, 56, 60, 64, 149, 153, 157, 161
4	149, 153, 157, 161, 165
5	149, 153, 157, 161

6	36, 40, 44, 48
7	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 149, 153, 157, 161, 165
8	52, 56, 60, 64
9	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 132, 136, 140, 149, 153, 157, 161, 165
10	36, 40, 44, 48, 149, 153, 157, 161, 165
11	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 149, 153, 157, 161

2.2.3 CountryCode

Value:

AG, AR, AW, AU, AT, BS, BB, BM, BR, BE, BG, CA, KY, CL, CN, CO, CR, CY, CZ, DK, DO, EC, SV, FI, FR, DE, GR, GU, GT, HT, HN, HK, HU, IS, IN, ID, IE, IL, IT, JP, JO, LV, LI, LT, LU, MY, MT, MA, MX, NL, NZ, NO, PE, PT, PL, RO, RU, SA, CS, SG, SK, SI, ZA, KR, ES, SE, CH, TW, TR, GB, UA, AE, US, VE

NOTE:

"" : Using default setting.

2.4 G - channels 1 ~ 11

5G - channels 52 ~ 64, 100 ~ 140, 149 ~ 165

2.2.4 SSID

Value:

0~z, 1~32 ASCII characters.

2.2.5 WirelessMode

Value:

0: legacy 11b/g mixed

1: legacy 11B only

2: legacy 11A only

// Not support in RflicType=1(id=RFIC_5225)

// and RflicType=2(id=RFIC_5325)

3: legacy 11a/b/g mixed

// Not support in RflicType=1(id=RFIC_5225)

// and RflicType=2(id=RFIC_5325)

4: legacy 11G only

5: 11ABGN mixed

6: 11N only

7: 11GN mixed

8: 11AN mixed

9: 11BGN mixed

10: 11AGN mixed

2.2.6 Channel



Value:

This value depends on the CountryRegion or CountryRegionForABand setting.

2.2.7 BGProtection

Value:

- 0: Auto
- 1: Always on
- 2: Always off

2.2.8 TxPreamble

Value:

- 0: Preamble Long
- 1: Preamble Short
- 2: Auto

2.2.9 RTSThreshold

Value:

1 ~ 2347

2.2.10 FragThreshold

Value:

256 ~ 2346

2.2.11 TxBurst

Value:

- 0: Disable
- 1: Enable

2.2.12 PktAggregate

Value:

- 0: Disable
- 1: Enable

2.2.13 NetworkType

Value:

Infra: infrastructure mode
Adhoc: adhoc mode

2.2.14 AuthMode

Value:

OPEN:	For open system
SHARED:	For shared key system
WEPAUTO:	Auto switch between OPEN and SHARED
WPAPSK:	For WPA pre-shared key (Infra)
WPA2PSK:	For WPA2 pre-shared key (Infra)
WPANONE:	For WPA pre-shared key (Adhoc)
WPA:	
WPA2:	

2.2.15 EncrypType

Value:

NONE:	For AuthMode=OPEN
WEP:	For AuthMode=OPEN or AuthMode=SHARED
TKIP:	For AuthMode=WPAPSK or WPA2PSK
AES:	For AuthMode=WPAPSK or WPA2PSK

2.2.16 DefaultKeyID

Value:

1 ~ 4

2.2.17 WEP KeyType

Key1Type=value
Key2Type=value
Key3Type=value
Key4Type=value

Value:

0: hexadecimal type
1: ASCII type
(usage: reading profile only)

2.2.18 WEP Hex Key

Key1=value
Key2=value
Key3=value
Key4=value

Value:

10 or 26 hexadecimal characters e.g. 012345678
5 or 13 ASCII characters e.g. passwd
(usage : "iwpriv" only)

2.2.19 WEP Key String

Key1Str=value

Key2Str=value

Key3Str=value

Key4Str=value

Value:

10 or 26 characters (key type=0)

5 or 13 characters (key type=1)

(usage : reading profile only)

2.2.20 WPAPSK

Value:

8 ~ 63 ASCII or

64 HEX characters

2.2.21 WmmCapable

Value:

0: Disable WMM

1: Enable WMM

2.2.22 IEEE80211H

Enable IEEE802.11h support

Value:

0: Disable

1: Enable

2.2.23 PSMode

Value:

CAM Constantly Awake Mode

Max_PSP Max Power Savings

Fast_PSP Power Save Mode

2.2.24 AutoRoaming

Value:

0: Disabled

1: Enabled

2.2.25 RoamThreshold

Value:



0 ~ 255

2.2.26 TGNWiFiTest

Value:

0: Disabled

1: Enabled

2.2.27 WirelessEvent

Value:

0: Disabled

1: Enabled

2.2.28 CarrierDetect

Value:

0: Disabled

1: Enabled

2.2.29 AutoReconnect

Value:

0: Disabled

1: Enabled

2.2.30 AdhocN

Value:

0: Disabled

1: Enabled

2.2.31 HT_RDG

Value:

0: Disabled

1: Enabled

2.2.32 HT_EXTCHA

Value:

0: Below

1: Above

2.2.33 HT_OpMode

Value:

- 0: HT mixed format
- 1: HT greenfield format

2.2.34 HT_MpduDensity

Value:

0 ~ 7

2.2.35 HT_BW

Value:

- 0: 20MHz
- 1: 40MHz

2.2.36 HT_AutoBA

Value:

- 0: Disabled
- 1: Enabled

2.2.37 HT_AMSDU

Value:

- 0: Disabled
- 1: Enabled

2.2.38 HT_BAWinSize

Value:

1 ~ 64

2.2.39 HT_GI

Value:

- 0: long GI
- 1: short GI

2.2.40 HT_MCS

Value:

0 ~ 15

33: auto

2.2.41 HT_MIMOPSMMode

Value:

0: Static SM Power Save Mode
2: Reserved
1: Dynamic SM Power Save Mode
3: SM enabled
(not yet fully supported)

2.2.42 HT_BADecline=Value

Reject BA request from AP

Value:

0: Disabled
1: Enabled

2.2.43 HT_DisallowTKIP

Enable/Disable N rate with 11N AP when cipher is WEP or TKIP.

Value:

0: FALSE
1: TRUE

2.2.44 BeaconLostTime

Change Beacon Lost Time
Value:

1 ~ 60 seconds

Default value is 4 seconds

2.2.45 AutoRoaming

Enable/disable auto roaming mechanism

Value:

0: disable
1: enable

Default setting is disabled.

2.2.46 MacAddress

MacAddress=value

Value: XX:XX:XX:XX:XX:XX

2.2.47 AutoConnect

Enable/Disable driver connect to ANY AP when SSID is null.

Value:

0: disable (default)

1: enable

2.2.48 ETxBfEnCond=Value

Value:

0: Disable Explicit TX Beamforming (ETxBF)

1: Enable Explicit TX Beamforming (ETxBF)

2.2.49 StreamMode=Value

Value:

0: Disable Stream Mode

1: Enable Stream Mode

2.2.50 P2P_GOIntent

Relative value between 0 and 15 used to indicate the desire of the P2P device to be the P2P Group Owner, with a larger value indicating a higher desire.

Value:

0~15: GO Intent.

2.2.51 P2P_DevName

Define P2P device name for display

Value:

0~Z, less than 32 characters.

2.2.52 P2P_ListChannel

Set P2P device listen channel. (Channel 1, 6, 11)

Value:

1, 6, 11

2.2.53 P2P_OpChannel

Set P2P device GO operate channel.

Value:

Based on country region.

2.3 MORE INFORMATION

If you want for rt2860 driver to auto-load at boot time:



- A) choose ra0 for first RT2860 WLAN card, ra1 for second RT2860 WLAN card, etc.
- B) create(edit) 'ifcfg-ra0' file in /etc/sysconfig/network-scripts/,
edit(or add the line) in /etc/modules.conf:
 alias ra0 rt2860sta
- C) edit(create) the file /etc/sysconfig/network-scripts/ifcfg-ra0
 DEVICE='ra0'
 ONBOOT='yes'

NOTE:

- if you use dhcp, add this line too.
 BOOTPROTO='dhcp'
- D) To ease the Default Gateway setting,
add the line
 GATEWAY=x.x.x.x
in /etc/sysconfig/network

Ralink Website document

**for stevej@cradlepoint.com
And Company Use Only**

3 WIRELESS TOOLS

3.1 Iwpriv Usage

This is detailed explanation of each parameter for iwpriv.

Before reading this document, make sure you already read README.

`iwpriv ra0 set [parameters]=[Value]`

NOTE:

Execute one iwpriv/set command simultaneously.

3.1.1 DriverVersion

Check driver version by issue iwpriv set command.

Range: Any value

Value:

0

3.1.2 CountryRegion

Set country region.

Range:

0 ~ 7

Value:

Region	Channels
0	1-11
1	1-13
2	10-11
3	10-13
4	14
5	1-14
6	3-9
7	5-13

3.1.3 CountryRegionABand

Set country region for A band.



Range:

{0~10}

Value:

Region	Channels
0	36, 40, 44, 48, 52, 56, 60, 64, 149, 153, 157, 161, 165
1	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140
2	36, 40, 44, 48, 52, 56, 60, 64
3	52, 56, 60, 64, 149, 153, 157, 161
4	149, 153, 157, 161, 165
5	149, 153, 157, 161
6	36, 40, 44, 48
7	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 149, 153, 157, 161, 165
8	52, 56, 60, 64
9	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 132, 136, 140, 149, 153, 157, 161, 165
10	36, 40, 44, 48, 149, 153, 157, 161, 165
11	36, 40, 44, 48, 52, 56, 60, 64, 100, 104, 108, 112, 116, 120, 149, 153, 157, 161

3.1.4 SSID

Set AP SSID

Range:

{0 ~ z, 1 ~ 32 ASCII characters}

Value:

3.1.5 WirelessMode

Set Wireless Mode

Range:

{0 ~ 10}

Value:

- 0: legacy 11b/g mixed
- 1: legacy 11B only
- 2: legacy 11A only
- 3: legacy 11a/b/g mixed
- 4: legacy 11G only
- 5: 11ABGN mixed
- 6: 11N only



7: 11GN mixed
8: 11AN mixed
9: 11BGN mixed
10: 11AGN mixed

3.1.6 TxBurst:

Set TxBurst Enable or Disable

Range:

{0,1}

Value:

0: Disable,
1: Enable

3.1.7 PktAggregate:

Set Tx Aggregate Enable or Disable

Range:

{0,1}

Value:

0: Disable,
1: Enable

3.1.8 TxPreamble:

Set TxPreamble

Range:

{0~2}

Value:

0: Preamble Long
1: Preamble Short
2: Auto

3.1.9 TxPower:

Set Tx power in percentage

Range:

{0 ~ 100}

Value:

3.1.10 Channel

Set Channel, depends on CountryRegion or CountryRegionABand

3.1.11 BGProtection:

Set 11B/11G Protection

Range:

{0~2}

Value:

0: Auto,
1: Always on
2: Always off

3.1.12 RTSThreshold:

Set RTS Threshold

Range:

{1 ~ 2347}

Value:

3.1.13 FragThreshold:

Set Fragment Threshold

Range:

{256 ~ 2346}

Value:

3.1.14 NetworkType:

Set Network type

Range:

{Infra, Adhoc}

Value:

3.1.15 AuthMode:

Set Authentication Mode

Range:



{OPEN, SHARED, WEPAUTO, WPAPSK, WPA2PSK, WPANONE}

Value:

3.1.16 EncryptType:

Set Encryption Type

Range:

{NONE, WEP, TKIP, AES}

Value:

3.1.17 DefaultKeyID:

Set Default Key ID

Range:

{1 ~ 4}

Value:

3.1.18 Key1

Set Key1 String

Range:

5 ASCII characters or 10 hex number, or
13 ASCII characters or 26 hex numbers

Value:

3.1.19 Key2

Set Key2 String

Range:

5 ASCII characters or 10 hex number or
13 ASCII characters or 26 hex numbers

Value:

3.1.20 Key3

Set Key3 String

Range:

5 ASCII characters or 10 hex number or



13 ASCII characters or 26 hex numbers

Value:

3.1.21 Key4

Set Key4 String

Range:

5 ASCII characters or 10 hex number or
13 ASCII characters or 26 hex numbers

Value:

3.1.22 WPAPSK

WPA Pre-Shared Key

Range:

8~63 ASCII or 64 hex characters

Value:

3.1.23 WmmCapable

Set WMM Capable

Range:

0, 1

Value:

0: Disable WMM,
1: Enable WMM

3.1.24 IEEE80211H

Enable IEEE802.11h support

Range:

0, 1

Value:

0: Disable
1: Enable

3.1.25 PSMode



Set Power Saving Mode

Range:

{CAM, MAX_PSP, FAST_PSP}

Value:

3.1.26 ResetCounter

Reset statistics counter

Range:

Any value

Value:

0

3.1.27 Debug

Set on debug level

Range:

{0 ~ 5}

Value:

0: OFF	no debug message display
1: ERROR	display error message
2: WARN	display warning message
3: TRACE	display trace message, usually used.
4: INFO	display informatic message
5: LOUD	display all message

3.1.28 CarrierDetect

Value:

0: Disabled
1: Enabled

3.1.29 HtRdg

Enable HT Reverse Direction Grant.

Value:

0: Disabled
1: Enabled

3.1.30 HtExtcha

To locate the 40MHz channel in combination with the control.

Value:

- 0: Below
- 1: Above

3.1.31 HtOpMode

Change HT operation mode.

Value:

- 0: HT mixed format
- 1: HT greenfield format

3.1.32 HtMpduDensity

Minimum separation of MPDUs in an A-MPDU. (based on 802.11n D2.0)

Value:

0 ~ 7

- 0: no restriction
- 1: 1/4 μ s
- 2: 1/2 μ s
- 3: 1 μ s
- 4: 2 μ s
- 5: 4 μ s
- 6: 8 μ s
- 7: 16 μ s

3.1.33 HtBw

Support channel width.

Value:

- 0: 20MHz
- 1: 40MHz

3.1.34 HtAutoBa

Enable auto block acknowledgment (Block Ack).

Value:

- 0: Disabled
- 1: Enabled

3.1.35 HtAmsdu

Enable aggregation of multiple MSDUs in one MPDU.

Value:

0: Disabled
1: Enabled

3.1.36 HtBaWinSize

Set BA WinSize.

Value:

1 ~ 64

3.1.37 HtGi

Support Short/Long GI.

Value:

0: long GI
1: short GI

3.1.38 HtMcs

MCS rate selection.

Value:

0 ~ 15
33: auto

3.1.39 HtProtect

Enable HT protection for legacy device.

Value:

0: Disable
1: Enable

3.1.40 HtMimoPs

MIMO power save mode selection. (based on 802.11n D2.0)

Value:

0: Static SM (Spatial Multiplexing) Power Save Mode
1: Dynamic SM Power Save Mode
2: Reserved



3: SM enabled
(not fully support yet)

3.1.41 FixedTxMode

Set Fixed Tx Mode for fixed rate setting

Value:

Mode= CCK

MCS = 0	=> 1Mbps
MCS= 1	=> 2Mbps
MCS= 2	=> 5.5 Mbps
MCS= 3	=> 11 Mbps

Mode = OFDM

MCS = 0	=> 6Mbps
MCS= 1	=> 9Mbps
MCS= 2	=> 12Mbps
MCS= 3	=> 18Mbps
MCS= 4	=> 24Mbps
MCS= 5	=> 36Mbps
MCS= 6	=> 48Mbps
MCS= 7	=> 54Mbps

3.1.42 LongRetry

Usage:

iwpriv ra0 set LongRetry=value

Value:

0 ~ 255

3.1.43 ShortRetry

Usage:

iwpriv ra0 set ShortRetry=value

Value:

0 ~ 255

3.1.44 HtBaDecline=Value

Reject BA request from AP

Value:



0: Disabled
1: Enabled

3.1.45 HtDisallowTKIP=Value

When cipher is WEP or TKIP, STA would connect to 11N AP with legacy rate
Enable/Disable N rate with 11N AP when cipher is WEP or TKIP
Default setting is disable

Value:

0: False
1: True

3.1.46 wsc_ap_band=Value

Setting prefer band to do WPS with dual band WPS AP
Default value is auto (2)

Value:

0 : prefer 2.4G
1 : prefer 5G
2 : auto

3.1.47 BeaconLostTime=Value

Change Beacon Lost Time
Default value is 4 seconds

Value:

1 ~ 60 seconds

3.1.48 AutoRoaming=Value

Enable/Disable auto roaming mechanism
Default setting is disable.

Value:

0: Disabled
1: Enabled

3.1.49 SiteSurvey=Value

Scan with specific SSID after Link Up

Value:

0~z, 1~32 ASCII characters

3.1.50 TdlsCapable =Value

Enable/disable TDLS capable

Value:

0 : disable

1 : enable

Example : iwpriv ra0 set TdlsCapable=0

3.1.51 TdlsSetup=Value

Manually add TDLS link

Value: MAC address

Example: iwpriv ra0 set TdlsSetup=00:11:22:33:44:55

3.1.52 AutoReconnect=value

Description: Enable/Disable driver auto reconnect functionality

Valid Range: 0-1

Default Value: 1

0: Disable, 1: Enable

3.1.53 AdhocN=value

Description: Enable/Disable Adhoc to support N or not

Valid Range: 0-1

Default Value: 1

0: Disable, 1: Enable

3.1.54 ETxBfEnCond=Value

Value:

0: Disable Explicit TX Beamforming (ETxBF)

1: Enable Explicit TX Beamforming (ETxBF)

Example:

iwpriv ra0 set ETxBfEnCond =1

3.1.55 StreamMode=Value

Value:

0: Disable Stream Mode

1: Enable Stream Mode

Example:

iwpriv ra0 set StreamMode=1



3.2 Iwpriv SHOW Usage

This is the status of each parameter for “iwpriv ra0 show”.

`iwpriv ra0 show [parameters]`

3.2.1 SSID

Show AP SSID

3.2.2 WirelessMode

Show Wireless Mode

3.2.3 TxBurst

Show TxBurst

3.2.4 TxPreamble

Show TxPreamble

3.2.5 TxPower

Show TxPower

3.2.6 Channel

Show Channel

3.2.7 BGProtection

Show BGProtection

3.2.8 RTSThreshold

Show RTSThreshold

3.2.9 FragThreshold

Show FragThreshold

3.2.10 HtBw

Show HtBw

3.2.11 HtMcs



Show HtMcs

3.2.12 HtGi

Show HtGi

3.2.13 HtOpMode

Show HtOpMode

3.2.14 HtExtcha

Show HtExtcha

3.2.15 HtMpduDensity

Show HtMpduDensity

3.2.16 HtBaWinSize

Show HtBaWinSize

3.2.17 HtRdg

Show HtRdg

3.2.18 HtAmsdu

Show HtAmsdu

3.2.19 HtAutoBa

Show HtAutoBa

3.2.20 CountryRegion

Show CountryRegion

3.2.21 CountryRegionABand

Show CountryRegionABand

3.2.22 CountryCode

Show CountryCode

3.2.23 PktAggregate

Show PktAggregate

3.2.24 WmmCapable

Show WmmCapable

3.2.25 IEEE80211H

Show IEEE80211H

3.2.26 NetworkType

Show NetworkType

3.2.27 AuthMode

Show AuthMode

3.2.28 EncryptType

Show EncryptType

3.2.29 DefaultKeyID

Show DefaultKeyID

3.2.30 Key1

Show Key1

3.2.31 Key2

Show Key2

3.2.32 Key3

Show Key3

3.2.33 Key4

Show Key4

3.2.34 WPAPSK

Show WPAPSK

3.2.35 AutoReconnect

Show bAutoReconnect flag

e.g.

3.2.36 WPAPSK

Show WPA Passphrase

e.g.

```
iwpriv ra0 show WPAPSK
```

3.2.37 PMK

Show PMK

e.g.

```
iwpriv ra0 show PMK
```

3.3 Iwpriv Examples

3.3.1 Infrastructure

1.1.1.1 OPEN/NONE

Config STA to link with AP which is OPEN/NONE (Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=OPEN
3. iwpriv ra0 set EncrypType=NONE
4. iwpriv ra0 set SSID="AP's SSID"

1.1.1.2 SHARED/WEP

Config STA to link with AP which is SHARED/WEP (Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=SHARED
3. iwpriv ra0 set EncrypType=WEP
4. iwpriv ra0 set DefaultKeyID=1
5. iwpriv ra0 set Key1="AP's wep key"
6. iwpriv ra0 set SSID="AP's SSID"

1.1.1.3 WPAPSK/TKIP

Config STA to link with AP which is WPAPSK/TKIP (Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=WPAPSK
3. iwpriv ra0 set EncrypType=TKIP
4. iwpriv ra0 set SSID="AP's SSID"
5. iwpriv ra0 set WPAPSK="AP's wpa-presared key"
6. iwpriv ra0 set SSID="AP's SSID"

1.1.1.4 WPAPSK/AES

Config STA to link with AP which is WPAPSK/AES (Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=WPAPSK
3. iwpriv ra0 set EncrypType=AES
4. iwpriv ra0 set SSID="AP's SSID"
5. iwpriv ra0 set WPAPSK="AP's wpa-preshared key"
6. iwpriv ra0 set SSID="AP's SSID"

1.1.1.5 WPA2PSK/TKIP

Config STA to link with AP which is WPA2PSK/TKIP(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Infra
2. iwpriv ra0 set AuthMode=WPA2PSK
3. iwpriv ra0 set EncrypType=TKIP
4. iwpriv ra0 set SSID="AP's SSID"
5. iwpriv ra0 set WPAPSK=12345678
6. iwpriv ra0 set SSID="AP's SSID"

3.3.2 Ad-Hoc

1.1.1.6 OPEN/NONE

Config STA to create/link as adhoc mode, which is OPEN/NONE(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Adhoc
2. iwpriv ra0 set AuthMode=OPEN
3. iwpriv ra0 set EncrypType=NONE
4. iwpriv ra0 set SSID="Adhoc's SSID"

1.1.1.7 WPANONE/TKIP

Config STA to create/link as adhoc mode, which is WPANONE/TKIP(Authentication/Encryption)

1. iwpriv ra0 set NetworkType=Adhoc
2. iwpriv ra0 set AuthMode=WPANONE
3. iwpriv ra0 set EncrypType=TKIP
4. iwpriv ra0 set SSID="AP's SSID"
5. iwpriv ra0 set WPAPSK=12345678
6. iwpriv ra0 set SSID="AP's SSID"

3.3.3 Get site survey

Usage:

iwpriv ra0 get_site_survey

3.3.4 Get Statistics



Usage:

```
iwpriv ra0 stat ;read statistic counter
iwpriv ra0 set ResetCounter=0;reset statistic counter
```

3.3.5 ANY SSID

Link with an AP which is the largest strength, set ANY SSID (ssidLen=0)

Usage:

```
iwconfig ra0 essid ""
or,
iwpriv ra0 set SSID=""
```

3.4 iwlist

This is detailed explanation of each parameter for iwlist.

```
iwlist ra0 scanning ; list the results after scanning(manual rescan)
```

3.5 iwconfig

The following are our support in standard configuration – iwconfig

1. iwconfig ra0 essid {NN|on|off} ;set essid
2. iwconfig ra0 mode {managed|ad-hoc|...} ;set wireless mode
3. iwconfig ra0 freq N.NNNN[k|M|G] ;set frequency
4. iwconfig ra0 channel N ;set channel
5. iwconfig ra0 ap {N|off|auto} ;set AP address
6. iwconfig ra0 nick N ;set nickname
7. iwconfig ra0 rate {N|auto|fixed} ;set rate
8. iwconfig ra0 rts {N|auto|fixed|off} ;set RTS threshold
9. iwconfig ra0 frag {N|auto|fixed|off} ;set Fragment threshold
10. iwconfig ra0 enc {NNNN-NNNN|off} ;set encryption type
11. iwconfig ra0 power {period N|timeout N} ;set power management modes

NOTE:

Please refer to the main page of 'iwconfig', 'iwlist' and 'iwpriv' for wireless extension usage.

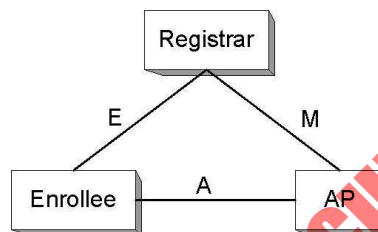
4 WPS – WI-FI PROTECTED SETUP

Simple Config Architectural Overview

This section presents a high-level description of the Simple Config architecture. Much of the material is taken directly from the Simple Config specification.

Figure 1 depicts the major components and their interfaces as defined by Wi-Fi Simple Config Spec. There are three logical components involved: the Registrar, the access point (AP), and the Enrollee.

- ◆ The **Enrollee** is a device seeking to join a WLAN domain. Once an Enrollee obtains a valid credential, it becomes a member.
- ◆ A **Registrar** is an entity with the authority to issue and revoke domain credentials. A registrar can be integrated into an AP.
- ◆ The **AP** can be either a WLAN AP or a wireless router.



Components and Interfaces

Registration initiation is ordinarily accomplished by a user action such as powering up the Enrollee and, optionally, running a setup wizard on the Registrar (PC).

4.1 Iwpriv Usage

This is detailed explanation of each iwpriv parameter.

Before reading this document, make sure you already read README.

`iwpriv ra0 [commands]=[Value]`

NOTE:

Wireless extension private handlers.

4.1.1 wsc_conf_mode

Set WPS conf mode.

Range:

{0, 1, 2}

Value:

0: WPS Disabled



- 1: Enrollee
- 2: Registrar

4.1.2 wsc_mode

Set WPS mode, PIN or PBC.

Range:

{1, 2}

Value:

- 1: PIN
- 2: PBC

4.1.3 wsc_pin

Set the enrollee's PIN Code.

Range:

{00000000 ~ 99999999}

Value:

4.1.4 wsc_ssid

Set WPS AP SSID.

Range:

{0~z, 1~32 ASCII characters}

Value:

4.1.5 wsc_start

Trigger RT2860 STA driver to do WPS process.

Range:

NULL

Value:

4.1.6 wsc_stop

Stop WPS process.

Range:

NULL



Value:

4.1.7 wsc_gen_pincode

Generate new PIN code.

Range:

NULL

Value:

4.1.8 wsc_cred_count

Set count of WPS credential.

Range:

{1 ~ 8}

Value:

4.1.9 wsc_cred_ssid

Set SSID into credential [idx].

Range:

{"idx ssid_str"}

Value:

idx: 0 ~ 7

ssid_str: 0~z, 1~32 ASCII characters

e.g.

iwpriv ra0 wsc_cred_ssid "0 wps_ap1"

4.1.10 wsc_cred_auth

Set AuthMode into credential [idx].

Range:

{"idx auth_str"}

Value:

idx: 0 ~ 7

auth_str: OPEN, WPAPSK, WPA2PSK, SHARED, WPA, WPA2

e.g.



iwpriv ra0 wsc_cred_auth "0 WPAPSK"

4.1.11 wsc_cred_encr

Set EncryptType into credential [idx].

Range:

{"idx encr_str"}

Value:

idx: 0 ~ 7

encr_str: NONE, WEP, TKIP, AES

e.g.

iwpriv ra0 wsc_cred_encr "0 TKIP"

4.1.12 wsc_cred_keyIdx

Set Key Index into credential [idx].

Range:

{"idx key_index"}

Value:

idx: 0 ~ 7

key_index: 1 ~ 4

e.g.

iwpriv ra0 wsc_cred_keyIdx "0 1"

4.1.13 wsc_cred_key

Set Key into credential [idx].

Range:

{"idx key"}

Value:

idx: 0 ~ 7

key: ASCII string (wep_key_len(=5,13), passphrase_len(=8~63))

OR

Hex string (wep_key_len(=10,26), passphrase_len(=64))

e.g.



iwpriv ra0 wsc_cred_key "0 12345678" ;; Passphrase
iwpriv ra0 wsc_cred_key "0 abcd" ;; WEP Key

4.1.14 wsc_cred_mac

Set AP's MAC into credential[idx].

Range:

{ "idx mac_str" }

Value:

idx: 0 ~ 7

mac_str: xx:xx:xx:xx:xx:xx

e.g.

iwpriv ra0 wsc_cred_mac "0 00:11:22:33:44:55"

4.1.15 wsc_conn_by_idx

Connect AP by credential index.

Range:

{ 0 ~ 7 }

Value:

idx: 0 ~ 7

4.1.16 wsc_auto_conn

Set driver to re-connecting to AP or not after registration.

Range:

{ 0, 1 }

Value:

0: Disabled, driver won't re-connect to AP with new configurations.

1: Enabled, driver will re-connect to AP with new configurations.

4.1.17 wsc_ap_band

Setting prefer band to do WPS with dual band WPS AP.

Range:

{ 0, 1, 2 }

Value:

0: prefer 2.4G

1: prefer 5G

2: auto



Default value is auto (2)

4.1.18 Wsc4digitPinCode

Generate WPS 4-digits PIN

Value:

- 0: Disable
- 1: Enable

4.1.19 WscV2Support (WPS2.0)

Enable/Disable WSC V2 support

Value:

- 0: Disable
- 1: Enable

4.2 WPS STA as an Enrollee or Registrar

Build WPS function. Please set 'HAS_WSC=y'.

4.2.1 Enrollee Mode

1.1.1.8 PIN mode:

Running Scenarios (case 'a' and 'b')

- A) Adding an Enrollee to AP+Registrar (EAP)
[AP+Registrar]<----EAP---->[Enrollee Client]
- B) Adding an Enrollee with external Registrar (UPnP/EAP)
[External Registrar]<----UPnP---->[AP_Proxy]<---EAP--->[Enrollee Client]

NOTE:

'EAP' indicates to use wireless medium and 'UPnP' indicates to use
wired or wireless medium.

- (i) [Registrar] or [AP+Registrar]
Enter the Enrollee Pin Code on the Registrar and start WPS on the Registrar.

NOTE:

How to get the Enrollee Pin Code? Use 'iwpriv ra0 stat' on the Enrollee.

- (ii) [RT2860 Linux WPS STA]
iwpriv ra0 wsc_conf_mode 1 ;; Enrollee
iwpriv ra0 wsc_mode 1 ;; PIN
iwpriv ra0 wsc_ssid "AP's SSID"
iwpriv ra0 wsc_start
- (iii) If the registration is successful, the Enrollee will be re-configured with the new parameters,
and will connect to the AP with these new parameters.

1.1.1.9 PBC mode:

Running Scenarios (case 'a' only)



- a. Adding an Enrollee to AP+Registrar (EAP)
[AP+Registrar]<----EAP---->[Client]
- (i) [AP+Registrar]
Start PBC on the Registrar.
- (ii) [RT2860 Linux WPS STA]
iwpriv ra0 wsc_conf_mode 1 ;; Enrollee
iwpriv ra0 wsc_mode 2 ;; PBC
iwpriv ra0 wsc_start
- (iii) If the registration is successful, the Enrollee will be re-configured with the new parameters, and will connect to the AP with these new parameters.

4.2.2 Registrar Mode

1.1.1.10 PIN mode:

Running Scenarios (case 'a' and 'b')

- a. Configure the un-configured AP
[Unconfigured AP]<----EAP---->[Registrar]
- b. Configure the configured AP
Configured AP]<----EAP---->[Registrar]
- (i) [AP]
Start PIN on the Enrollee WPS AP.
- (ii) [RT2860 Linux WPS STA]
iwpriv ra0 wsc_conf_mode 2 ;; Registrar
iwpriv ra0 wsc_mode 1 ;; PIN
iwpriv ra0 wsc_pin xxxxxxxx ;; AP's PIN Code
iwpriv ra0 wsc_ssid "AP's SSID"
iwpriv ra0 wsc_start
- (iii) If the registration is successful;

in case 'a':

The Registrar will be re-configured with the new parameters, and will connect to the AP with these new parameters;

in case 'b':

The Registrar will be re-configured with AP's configurations, and will connect to the AP with these new parameters.

1.1.1.11 PBC mode:

Running Scenarios (case 'a' and 'b')

- a. Configure the un-configured AP
[Unconfigured AP]<----EAP---->[Registrar]
- b. Configure the configured AP
Configured AP]<----EAP---->[Registrar]
- (i) [AP]
Start PBC on the Enrollee WPS AP.
- (ii) [RT2860 Linux WPS STA]
iwpriv ra0 wsc_conf_mode 2 ;; Registrar
iwpriv ra0 wsc_mode 2 ;; PBC
iwpriv ra0 wsc_start



- (iii) If the registration is successful;
in case 'a':

The Registrar will be re-configured with the new parameters, and will connect to the AP with these new parameters;

in case 'b':

The Registrar will be re-configured with AP's configurations, and will connect to the AP with these new parameters.

4.3 WPS IOCTL Usage

Detail parameters and arguments; please refer to above section for detail.

4.3.1 iwpriv commands without argument

1. iwpriv ra0 wsc_start
2. iwpriv ra0 wsc_stop
3. iwpriv ra0 wsc_gen_pincode

e.g.

```
memset(&lwreq, 0, sizeof(lwreq));
sprintf(lwreq.ifr_name, "ra0", 3);
lwreq.u.mode = WSC_STOP;

/* Perform the private ioctl */
if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM, &lwreq) < 0)
{
    fprintf(stderr, "Interface doesn't accept private ioctl...\n");
    return -1;
}
```

4.3.2 iwpriv commands with one INT argument

1. iwpriv ra0 wsc_cred_count 1
2. iwpriv ra0 wsc_conn_by_idx 1
3. iwpriv ra0 wsc_auto_conn 1
4. iwpriv ra0 wsc_conf_mode 1
5. iwpriv ra0 wsc_mode 1
6. iwpriv ra0 wsc_pin 12345678

e.g.

```
memset(&lwreq, 0, sizeof(lwreq));
lwreq.u.data.length = 1;
cred_count = 1;
((int *) buffer)[i] = (int) cred_count;
offset = sizeof(int);

sprintf(lwreq.ifr_name, "ra0", 3);
lwreq.u.mode = WSC_CREDENTIAL_COUNT;
memcpy(lwreq.u.name + offset, buffer, IFNAMSIZ - offset);

/* Perform the private ioctl */
```



```
if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM, &lwreq) < 0)
{
    fprintf(stderr, "Interface doesn't accept private ioctl...\n");
    return -1;
}
```

4.3.3 iwpriv commands with string argument

1. iwpriv ra0 wsc_ssid "0 xxxxx"
2. iwpriv ra0 wsc_cred_ssid "0 xxxxx"
3. iwpriv ra0 wsc_cred_auth "0 WPA2PSK"
4. iwpriv ra0 wsc_cred_encr "0 TKIP"
5. iwpriv ra0 wsc_cred_keyidx "0 1"
6. iwpriv ra0 wsc_cred_key "0 12345"
7. iwpriv ra0 wsc_cred_mac "0 00:11:22:33:44:55"

e.g.

```
memset(&lwreq, 0, sizeof(lwreq));
memset(buffer, 0, 2048);
sprintf(lwreq.ifr_name, "ra0", 3);
sprintf(buffer, "0 wps_ssid_1");
lwreq.u.data.length = strlen(buffer) + 1;
lwreq.u.data.pointer = (caddr_t) buffer;
lwreq.u.data.flags = WSC_CREDENTIAL_SSID;
```

```
/* Perform the private ioctl */
if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_STRING_ITEM, &lwreq) < 0)
{
    fprintf(stderr, "Interface doesn't accept private ioctl...\n");
    return -1;
}
```

4.4 WPS IOCTL Sample Program

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <assert.h>

#include <netinet/in.h> /* for sockaddr_in */

#include <fcntl.h>

#include <time.h>

#include <sys/times.h>

#include <unistd.h>

#include <sys/socket.h> /* for connect and socket */

#include <sys/stat.h>

#include <err.h>

#include <errno.h>
```



```
#include <asm/types.h>

#include </usr/include/linux/wireless.h>

#include <sys/ioctl.h>

#define IFNAMSIZ 16

#define RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM    (SIOCWFIRSTPRIV + 0x14)

#define RTPRIV_IOCTL_SET_WSC_PROFILE_STRING_ITEM (SIOCWFIRSTPRIV + 0x16)

enum {

    WSC_CREDENTIAL_COUNT = 1,

    WSC_CREDENTIAL_SSID = 2,

    WSC_CREDENTIAL_AUTH_MODE = 3,

    WSC_CREDENTIAL_ENCR_TYPE = 4,

    WSC_CREDENTIAL_KEY_INDEX = 5,

    WSC_CREDENTIAL_KEY = 6,

    WSC_CREDENTIAL_MAC = 7,

    WSC_SET_DRIVER_CONNECT_BY_CREDENTIAL_IDX = 8,

    WSC_SET_DRIVER_AUTO_CONNECT = 9,

    WSC_SET_CONF_MODE = 10, // Enrollee or Registrar

    WSC_SET_MODE = 11,    // PIN or PBC

    WSC_SET_PIN = 12,

    WSC_SET_SSID = 13,

    WSC_START = 14,

    WSC_STOP = 15,

    WSC_GEN_PIN_CODE = 16,

};

int main()

{

    struct iwreq lwreq;

    char    buffer[2048] = {0};

    int     cred_count;

    int     offset = 0;    /* Space for sub-ioctl index */
```



```
int      skfd, i = 0;          /* generic raw socket desc. */

skfd = socket(AF_INET, SOCK_DGRAM, 0);

if (skfd < 0)

    return -1;

////////// WSC_STOP //////////

memset(&lwreq, 0, sizeof(lwreq));

sprintf(lwreq.ifr_name, "ra0", 3);

lwreq.u.mode = WSC_STOP;

/* Perform the private ioctl */

if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM, &lwreq) < 0)

{

    fprintf(stderr, "Interface doesn't accept private ioctl...\n");

    return -1;

}

////////////////////////////////////

////////// WSC_CREDENTIAL_COUNT //////////

memset(&lwreq, 0, sizeof(lwreq));

lwreq.u.data.length = 1;

cred_count = 1;

((int *) buffer)[i] = (int) cred_count;

offset = sizeof(int);

sprintf(lwreq.ifr_name, "ra0", 3);

lwreq.u.mode = WSC_CREDENTIAL_COUNT;

memcpy(lwreq.u.name + offset, buffer, IFNAMSIZ - offset);

/* Perform the private ioctl */

if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_U32_ITEM, &lwreq) < 0)

{

    fprintf(stderr, "Interface doesn't accept private ioctl...\n");
```




```
        return -1;
    }

    //////////////////////////////////////

    ////////// WSC_CREDENTIAL_SSID //////////

    memset(&lwreq, 0, sizeof(lwreq));

    memset(buffer, 0, 2048);

    sprintf(lwreq.ifr_name, "ra0", 3);

    sprintf(buffer, "0 wps_ssid_1");

    lwreq.u.data.length = strlen(buffer) + 1;

    lwreq.u.data.pointer = (caddr_t) buffer;

    lwreq.u.data.flags = WSC_CREDENTIAL_SSID;

    /* Perform the private ioctl */
    if(ioctl(skfd, RTPRIV_IOCTL_SET_WSC_PROFILE_STRING_ITEM, &lwreq) < 0)
    {
        fprintf(stderr, "Interface doesn't accept private ioctl...\n");

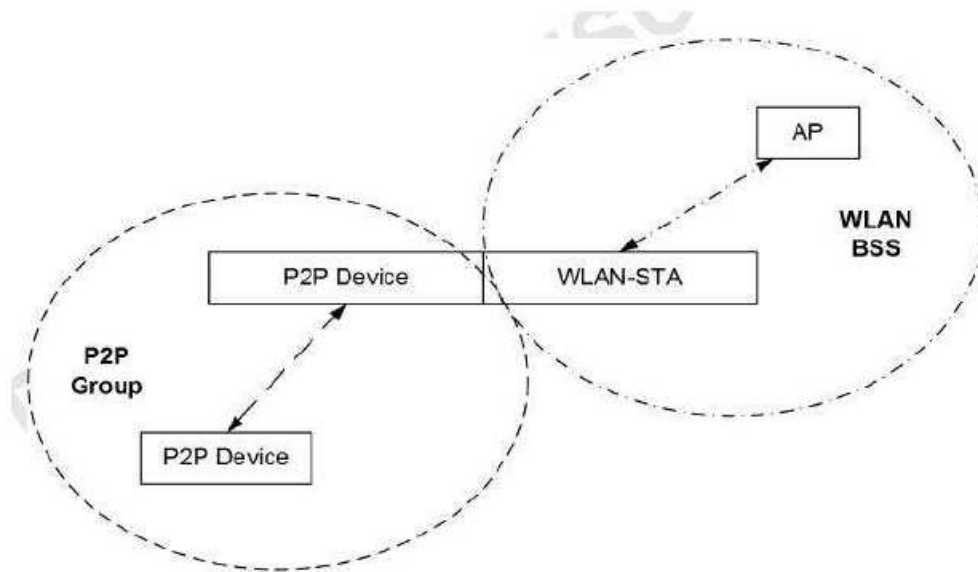
        return -1;
    }

    //////////////////////////////////////

    close(skfd);

    return 0;
}
```

5 WIFI DIRECT - P2P COMMAND



- A Wi-fi Direct Device may operate concurrently with a WLAN (infrastructure network)
- A P2P Group may operate in the same or different regulatory class and channel as a concurrently operating WLAN BSS

Wifi direct feature Makes direct connections to one another quickly and conveniently to do things like print, sync, and share content even when an access point or router is unavailable.

5.1 Iwpriv use

5.1.1 P2pOpMode

Set p2p interface operate mode to GO.

Value:

1: Auto (Force) GO mode

Example:

```
#iwpriv p2p0 set P2pOpMode=1
```

5.1.2 p2pLisCh

Set p2p device Channel in Listen stage.

Value:

1, 6, 11 (Define in P2P spec Page 26 & 36)

Example:

```
#iwpriv p2p0 set P2pLisCh=x
```

5.1.3 p2pOpCh

Set p2p Operation Channel if negotiate as GO

Value:

Based on country region

Example:

```
iwpriv p2p0 set p2pOpCh=1
```

5.1.4 p2pGoInt

Set p2p device GO Intent value

- This value is set to nego the art for become GO or Client

x1 = Group Owner Intent Value of P2P Device 1
x2 = Group Owner Intent Value of P2P Device 2

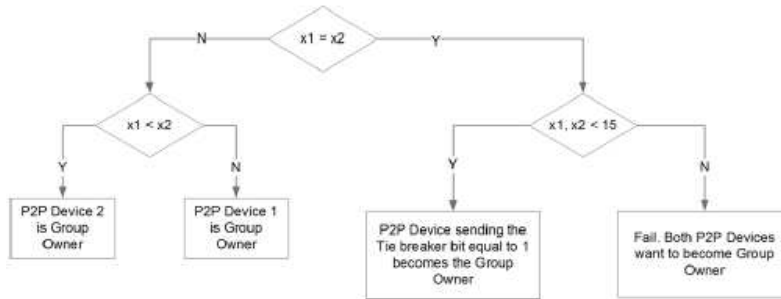


Figure 7—Group Owner determination flowchart

Value:

0~15

Example:

#iwpriv p2p0 set p2pGoInt=x

5.1.5 p2pDevName

Set p2p device display Device Name

Value:

0~Z, less than 32 characters.

Example:

#iwpriv p2p0 set p2pDevName=xxxxx

5.1.6 p2pWscMode

Set p2p device WSC Mode for P2P negotiate.

Value:

1: PIN

2: PBC

Example:

#iwpriv p2p0 set p2pWscMode=x

5.1.7 p2pWscConf

Set p2p device WSC Configure Method

Value:

1: Display

2: KeyPad

3: PBC

Example:

#iwpriv p2p0 set p2pWscConf =x

5.1.8 p2pScan

Set p2p device to start P2P Scanning.

Value:

0: Disable (Force Stop this action and cancel timer)

1: Enable (Do action)

Example:

#iwpriv p2p0 set p2pScan=x

5.1.9 p2pInv

Select p2p device ID to Invite (send p2p Invite to peer p2p device join our p2p Group)

Value:

0~29 (Software setting)

Example:

#iwpriv p2p0 set p2pInv =x

5.1.10 p2pDevDisc

Send p2p device discoverability to GO query specific CLIENT is in Group or not(debug use)

Value:

0~29 (Software setting)

Example:

#iwpriv p2p0 set p2pDevDisc=x

5.1.11 p2pLink

Select p2p device ID to do GO Negotiation

Value:

0~29 (Software setting)

Example:

#iwpriv p2p0 set p2pLink =x

5.1.12 p2pCfg

Dump/Show p2p configuration (In kernel background message)

Value:

Any

Example:

#iwpriv p2p0 set p2pCfg

5.1.13 p2pTab

Show Group Table status in kernel background message.

Value:

Any

Example:

#iwpriv p2p0 set p2pTab

5.1.14 p2pProv

Select p2p device ID to Provision

Value:

0~29 (Software setting)

Example:

#iwpriv p2p0 set p2pProv=x

5.1.15 p2pStat

Dump/Show p2p current rule, state machine status(In kernel background message)

Value:

Any

Example:



#iwpriv p2p0 set p2pStat

5.1.16 p2pReset

Reset p2p configuration the stat machine set to initial stage.

Value:

Any

Example:

#iwpriv p2p0 set p2pReset

5.1.17 p2pPerTab

Show p2p Persistent Table in kernel background message.

Value:

Any

Example:

#iwpriv p2p0 set p2pPerTab

5.1.18 p2pDefConfMthd

Set default WSC Config Method to Provision

Value:

1: Display

2: KeyPad

3: PBC

Example:

#iwpriv p2p0 set p2pDefConfMthd=x

5.1.19 p2pLinkDown

Tear down p2p session and change as p2p device mode.

Value:

Any

Example:

#iwpriv p2p0 set p2pLinkDown=x

5.1.20 p2pSigmaEnable

For p2p Sigma auto testing, we need enable some flag to pass WPS.

Value:

Any

Example:

#iwpriv p2p0 set p2pSigmaEnable=x

5.1.21 P2P example:

P2P device enable as autonomous GO :

autonomous GO:

#iwpriv p2p0 set p2pOpCh=11

#iwpriv p2p0 set P2pOpMode=1

P2P device reset to default setting:

#iwpriv p2p0 set p2pReset=1

P2P device start to scan and Listen as Channel 11:



```
#iwpriv p2p0 set p2pLisCh=11
#iwpriv p2p0 set p2pScan=1
```

P2P Device Start Device Discovery.

```
#iwpriv p2p0 set p2pScan=1
```

P2P Device Stop Device Discovery.

```
#iwpriv p2p0 set p2pScan=0
```

Connect the P2P Device on Scan Table of index 0.

```
Table.Client[0]: DeviceName[R A L I N K - P C ]
                DevCapability = 32.
                  ServiceDiscovery = 0.      P2P Client Discoverability = 0
                  ConcurrentOperation = 0.    InfraManaged = 0.
                  DeviceLimit = 0.           InvitationProcedure = 1.
                GroupCapability = 8.
                  GroupOwner = 0.             PersistentP2PGroup = 0
                  GroupLimit = 0.             IntraBSS = 1.
                  CrossConnection = 0.        PersistentReconnect = 0.
                  GroupFormation = 0.
                Addr[00:0c:43:21:64:81]
                BSSID[00:0c:43:21:64:81]
                InterfaceAddr[00:0c:43:21:64:81]
                SSID[D I R E C T - 1 2 ]
                WscMode = PBC.                PIN = 00 00 00 00 00 00 00 00.
                CfgMethod = PBC PBC.          GoIntent = 6.
                PrimDevType[00 01 00 50 f2 04 00 01] SecDevType[00 00 00 00 00 00 00 00]
                ChNumber = 0.                  OpChannel = 1.      ListenChannel = 1.
                P2pClientState = P2PSTATE_GO_WPS. MyGOIndex = 255.
                P2pIP = 00000000.             P2pFlag = 2.
                Dpid = 65535.                  StateCount = 1.     Rule = I am P2P GO.
                GeneralToken = 76.             NoAToken = 0.
                RegClass = 0.                  ConfigTimeOut = 0.
                ExtListenPeriod = 0.           ExtListenInterval = 0.
```

PIN:

```
#iwpriv p2p0 set p2pWscMode=1
#iwpriv p2p0 set p2pLink=0
```

PBC:

```
#iwpriv p2p0 set p2pWscMode=2
#iwpriv p2p0 set p2pLink=0
```

P2P scan and select p2p device do link:

```
#iwpriv p2p0 set p2pScan=1
#sleep 10
#iwpriv p2p0 set p2pTab=1
#iwpriv p2p0 set p2pLink=2
```

P2P device Show P2P Scan Table:

```
#iwpriv p2p0 set p2pTab=1
```



```
Table.Client[0]: DeviceName[R A L I N K - P C ]
                DevCapability = 32.
                ServiceDiscovery = 0.      P2P Client Discoverability = 0
                ConcurrentOperation = 0.    InfraManaged = 0.
                DeviceLimit = 0.           InvitationProcedure = 1.
                GroupCapability = 8.
                GroupOwner = 0.             PersistentP2PGroup = 0
                GroupLimit = 0.             IntraBSS = 1.
                CrossConnection = 0.        PersistentReconnect = 0.
                GroupFormation = 0.
                Addr[00:0c:43:21:64:81]
                BSSID[00:0c:43:21:64:81]
                InterfaceAddr[00:0c:43:21:64:81]
                SSID[D I R E C T - 1 2 ]
                WscMode = PBC.              PIN = 00 00 00 00 00 00 00 00.
                CfgMethod = PBC PBC.         GoIntent = 6.
                PrimDevType[00 01 00 50 f2 04 00 01] SecDevType[00 00 00 00 00 00 00 00]
                ChNumber = 0.                OpChannel = 1.          ListenChannel = 1.
                P2pClientState = P2PSTATE_GO_WPS.      MyGOIndex = 255.
                P2pIP = 00000000.            P2pFlag = 2.
                Dpid = 65535.                StateCount = 1.         Rule = I am P2P GO.
                GeneralToken = 76.            NoAToken = 0.
                RegClass = 0.                ConfigTimeOut = 0.
                ExtListenPeriod = 0.          ExtListenInterval = 0.
```

P2P device Show P2P configuration:

```
#iwpriv p2p0 set p2pCfg
```

```
P2P Device Config :
=====
Device Name[9] = P2P-linux.
Device Addr = 00:0c:43:32:40:45.
OpChannel = 1.          Listen Channel = 11.
My Go Intent = 0.
WscMode = PBC.          ConfigMethod = PBC PBC.
SSID[9] = DIRECT-12.
NoA_Count = 1.          NoA_Duration = 51200.          NoA_Interval = 102400.          StartTime = 0.
ExtListenPeriod = 0.    ExtListenInterval = 0.
Intra-Bss = 0.
ConenctMAC = 00:00:00:00:00:00.
```

P2P Show current rule and state machine status

```
#iwpriv p2p0 set p2pStat
```

```
P2P Current State
=====
My Rule = I am P2P Client
CTRL Machine State = P2P_CTRL_IDLE.
DISC Machine State = P2P_DISC_IDLE.
GO_FORM Machine State = P2P_GO_FORM_IDLE.
```

P2P device GO security setting change:

```
#iwpriv p2p0 set p2pReset=1
#iwpriv p2p0 set p2pOpCh=1
#iwpriv p2p0 set P2pOpMode=1
#iwpriv p2p0 set p2pWscMode=2
#iwpriv p2p0 set p2pWscConf=3
#iwpriv p2p0 set p2pDevName=Ralink-P2P-Device
#iwpriv p2p0 set SSID=DIRECT- Ralink
#iwpriv p2p0 set AuthMode=WPA2PSK
#iwpriv p2p0 set EncrypType=AES
#iwpriv p2p0 set WPAPSK=12345678
# iwpriv p2p0 set SSID=DIRECT- Ralink
#iwpriv p2p0 set p2pScan=1
```

Ralink P2P module provides three WPS configuration methods such as PBC, PIN-Display, PIN-Keypad.

Case 1: Enable autonomous GO on Channel 11 start WPS (PBC):

```
#iwpriv p2p0 set p2pOpCh=11
```



```
#iwpriv p2p0 set P2pOpMode=1
#iwpriv p2p0 set p2pWscMode=2
#iwpriv p2p0 set p2pWscConf=3
#iwpriv p2p0 set WscConfMode=7
#iwpriv p2p0 set WscMode=2
#iwpriv p2p0 set WscGetConf=1
#iwpriv p2p0 set p2pScan=1
```

Case 2: Enable autonomous GO on Channel 11 start WPS (PIN-Display):

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set P2pOpMode=1
#iwpriv p2p0 set p2pWscMode=1
#iwpriv p2p0 set p2pWscConf=1
#iwpriv p2p0 set WscConfMode=7
#iwpriv p2p0 set WscMode=1
#iwpriv p2p0 set WscGetConf=1
#iwpriv p2p0 set p2pScan=1
```

Case 3: Enable autonomous GO on Channel 11 start WPS (PIN-Keypad):

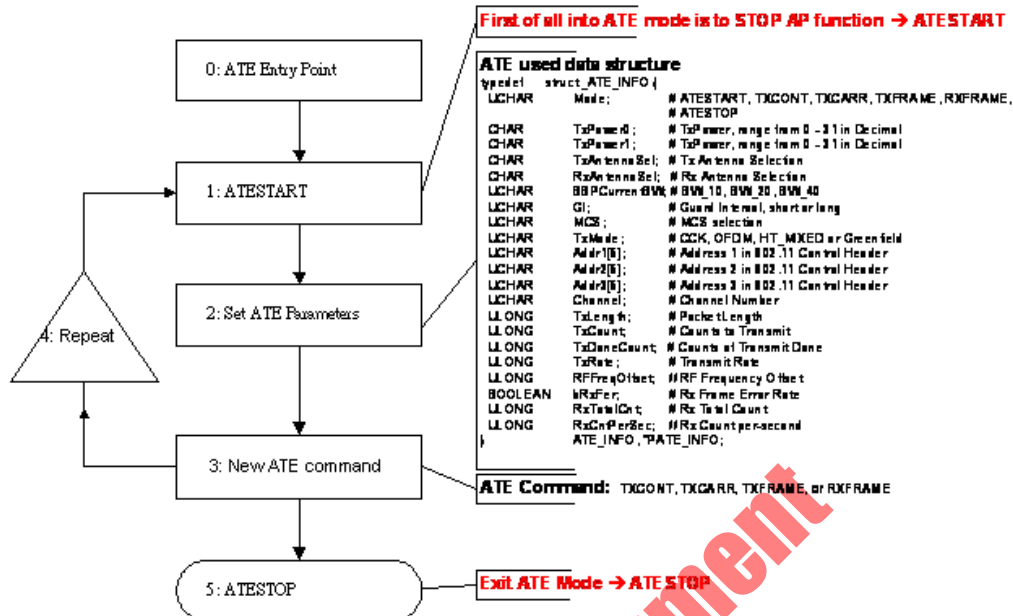
```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set p2pWscMode=1
#iwpriv p2p0 set p2pWscConf=2
#iwpriv p2p0 set WscConfMode=7
#iwpriv p2p0 set WscMode=1
#iwpriv p2p0 set p2pLink=0 (The index on P2P Scan Table)
#iwpriv p2p0 set WscPinCode=12345670 (read from enrollee's PIN Code)
#iwpriv p2p0 set WscGetConf=1
#iwpriv p2p0 set p2pScan=1
```

6 ATE TEST COMMAND FORMAT

IMPORTANT

IF YOU ARE NOT FAMILIAR WITH HARDWARE, IT IS RECOMMENDED NOT TO MODIFY HARDWARE DEFAULT VALUE.

Ralink ATE Operation Flow



Note:

1. Channel setting would take effect on next ATE command.
2. TxPower would take effect after frame transmit start.
TxPower can be changed dynamically on any ATE command operating.
3. Any ATE parameters have to be included into ATE_INFO structure.
4. Enter ATE mode by set ATE command "ATESTART".
 - a. Abort all TX rings
 - b. AsicDisableSync → Stop Beacon
 - c. Stop REKEYTimer
 - d. Stop CounterMeasureTimer
 - e. MacTableReset
5. Use TXCONT to check transmit power mask.
6. Use TXCARR to check frequency lock (under 25ppm).

6.1 iwpriv ra0 set [parameters]=[val]

Syntax:	Example
Section# parameters	11.1.5 ATECHANNEL
Explanation	Set ATE channel.
Value:	Value:
0: ...	1:
1: ...	2:
..::

6.1.1 ATE

Set ATE actions.

Value:

- ATESTART - Stop AP & ATE function.
- ATESTOP - Start AP function.
- TXCONT - Start AP continuous TX, for power mask.
- TXCARR - Start AP carrier test, for frequency calibration.
- TXFRAME - Transmit frame, for EVM.
- RXFRAME - Continuous RX, for PER/FER.

6.1.2 ATEDA

Set ATE frame header addr1.

Value:

xx:xx:xx:xx:xx:xx ; hex

6.1.3 ATESA

Set ATE frame header addr2.

Value:

xx:xx:xx:xx:xx:xx ; hex

6.1.4 ATEBSSID

Set ATE frame header addr3.

Value:



XX:XX:XX:XX:XX:XX

; hex

6.1.5 ATECHANNEL

Set ATE Channel, decimal.

Value:

802.11b/g: 1 ~ 14 depends on CountryRegion setting

6.1.6 ATETXPOW0

Set ATE Tx power for Antenna 1.

Value:

0 ~ 31 ; 5-bits only, decimal

6.1.7 ATETXPOW1

Set ATE Tx power for Antenna 2.

Value: 0 ~ 31 ; 5-bits only, decimal

6.1.8 ATETXFREQOFFSET

Set ATE RF frequency offset.

Value:

0 ~ 63 ; unit: 2KHz, decimal

6.1.9 ATETXLEN

Set ATE frame length.

Value:

24 ~ 1500 ; decimal

6.1.10 ATETXCNT

Set ATE frame Tx count.

Value:

1 ~ ; 32-bit, decimal

6.1.11 ATETXMODE (Refer to [TxMode](#))

Set ATE Tx Mode.



Value:

- | | |
|----------------|-------------|
| 0: CCK | 802.11b |
| 1: OFDM | 802.11g |
| 2: HT_MIX | 802.11b/g/n |
| 3: Green Field | 802.11n |

6.1.12 ATETXBW (Refer to [TxMode](#))

Set ATE Tx Bandwidth.

Value:

- 0: 20MHz
- 1: 40MHz

6.1.13 ATETXGI (Refer to [TxMode](#))

Set ATE Tx Guard Interval.

Value:

- 0: Long
- 1: Short

6.1.14 ATETXMCS (Refer to [TxMode](#))

Set ATE Tx MCS type.

Value:

0 ~ 15

6.1.15 ATETXANT

Set ATE TX antenna.

Value:

- 0: All
- 1: Antenna one
- 2: Antenna two

6.1.16 ATERXANT

Set ATE RX antenna.

Value:

- 0: All
- 1: Antenna one
- 2: Antenna two
- 3: Antenna three

6.1.17 ATERXFER

Set ATE to periodic show up RxCount (per-second) and RxTotalCount.

Value:

- 0: Disable counter show up
- 1: Enable counter show up

6.1.18 ATESHOW

Show all parameters of ATE.

Value:

1

6.1.19 ATEHELP

List all commands of ATE.

Value:

1

6.1.20 ResetCounter

Reset statistic counter.

Value:

0

6.1.21 ATERRF

Read all of the RF registers.

Value:

1

6.1.22 ATEWRF1

Write the RF register 1.

Value:

xxxxxxxx ;32-bit, hex

6.1.23 ATEWRF2

Write the RF register 2.



Value:

xxxxxxx ;32-bit, hex

6.1.24 ATEWRF3

Write the RF register 3.

Value:

xxxxxxx ;32-bit, hex

6.1.25 ATEWRF4

Write the RF register 4.

Value:

xxxxxxx ;32-bit, hex

6.1.26 ATEAUTOALC

Enable ATE auto Tx alc (Tx auto level control).

Value:

0/1

6.1.27 ATEIPG

Set ATE Tx frame Interpacket gap.

Value: 200 ; decimal

6.1.28 ATEPAYLOAD

Set ATE payload pattern for TxFrame.

Value: x ; only one octet acceptable

6.2 Tx Mode, MCS, BW and GI Selection Table

MODE = 0, Legacy CCK	
MCS = 0	Long Preamble CCK 1Mbps
MCS = 1	Long Preamble CCK 2Mbps
MCS = 2	Long Preamble CCK 5.5Mbps
MCS = 3	Long Preamble CCK 11Mbps
MCS = 8	Short Preamble CCK 1Mbps, * illegal rate
MCS = 9	Short Preamble CCK 2Mbps
MCS = 10	Short Preamble 5.5Mbps
MCS = 11	Short Preamble 11Mbps
Notes:	
Other MCS codes are reserved in legacy CCK mode.	
BW, SGI and STBC are reserved in legacy CCK mode.	
MODE = 1, Legacy OFDM	

MCS = 0	6Mbps
MCS = 1	9Mbps
MCS = 2	12Mbps
MCS = 3	18Mbps
MCS = 4	24Mbps
MCS = 5	36Mbps
MCS = 6	48Mbps
MCS = 7	54Mbps
Notes:	
Other MCS code in legacy CCK mode is reserved.	
When BW = 1, duplicate legacy OFDM is sent.	
SGI, STBC are reserved in legacy OFDM mode.	
MODE = 2, HT Mixed Mode	
MODE = 3, HT Greenfield	
MCS = 0 (1S)	(BW=0, SGI=0) 6.5Mbps
MCS = 1	(BW=0, SGI=0) 13Mbps
MCS = 2	(BW=0, SGI=0) 19.5Mbps
MCS = 3	(BW=0, SGI=0) 26Mbps
MCS = 4	(BW=0, SGI=0) 39Mbps
MCS = 5	(BW=0, SGI=0) 52Mbps
MCS = 6	(BW=0, SGI=0) 58.5Mbps
MCS = 7	(BW=0, SGI=0) 65Mbps
MCS = 8 (2S)	(BW=0, SGI=0) 13Mbps
MCS = 9	(BW=0, SGI=0) 26Mbps
MCS = 10	(BW=0, SGI=0) 39Mbps
MCS = 11	(BW=0, SGI=0) 52Mbps
MCS = 12	(BW=0, SGI=0) 78Mbps
MCS = 13	(BW=0, SGI=0) 104Mbps
MCS = 14	(BW=0, SGI=0) 117Mbps
MCS = 15	(BW=0, SGI=0) 130Mbps
MCS = 16 (3S)	(BW=0, SGI=0) 19.5Mbps
MCS = 17	(BW=0, SGI=0) 39Mbps
MCS = 18	(BW=0, SGI=0) 58.5Mbps
MCS = 19	(BW=0, SGI=0) 78Mbps
MCS = 20	(BW=0, SGI=0) 117Mbps
MCS = 21	(BW=0, SGI=0) 156Mbps
MCS = 22	(BW=0, SGI=0) 175Mbps
MCS = 23	(BW=0, SGI=0) 195Mbps
MCS = 32	(BW=1, SGI=0) HT duplicate 6Mbps
Notes:	
When BW=1, PHY_RATE = PHY_RATE * 2	
When SGI=1, PHY_RATE = PHY_RATE * 10/9	
The effects of BW and SGI are accumulative.	
When MCS=0~7(1S, One Tx Stream), STBC option is supported. SGI option is supported. BW option is supported.	
When MCS=8~15(2S, Two Tx Stream), STBC option is NOT supported. SGI option is supported. BW option is supported.	
When MCS=32, only SGI option is supported. BW and STBC option are not supported. (BW =1, STBC=0)	
Other MCS code in HT mode is reserved.	
When STBC is supported. Only STBC = 1 is allowed. STBC will extend the transmission range but will not increase transmission rate.	

6.3 Examples

6.3.1 Check EVM & Power

```
iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATEDA=00:11:22:33:44:55
iwpriv ra0 set ATESA=00:aa:bb:cc:dd:ee
iwpriv ra0 set ATEBSSID=00:11:22:33:44:55
iwpriv ra0 set ATECHANNEL=1           ; set Channel
iwpriv ra0 set ATETXMODE=1           ; set TX-Mode.
iwpriv ra0 set ATETXMCS=7            ; set MCS type.
iwpriv ra0 set ATETXBW=0             ; set Bandwidth
iwpriv ra0 set ATETXGI=0             ; set Long GI.
iwpriv ra0 set ATETXLEN=1024         ; set packet length.
iwpriv ra0 set ATETXPOW0=18
iwpriv ra0 set ATETXPOW1=18
iwpriv ra0 set ATETXCNT=100000
iwpriv ra0 set ATETXFRAME
...
iwpriv ra0 set ATETXPOW0=19
...
iwpriv ra0 set ATETXPOW0=20
...
iwpriv ra0 set ATE=ATESTART
```

6.3.2 Check Carrier

```
iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATECHANNEL=1           ; set Channel
iwpriv ra0 set ATETXMODE=1           ; set TX-Mode.
iwpriv ra0 set ATETXMCS=7            ; set MCS type.
iwpriv ra0 set ATETXBW=0             ; set Bandwidth
iwpriv ra0 set ATETXCNT=200          ; Tx frame count(decimal)
iwpriv ra0 set ATE=TXFRAME           ; Start Tx Frame(inform BBP to change, modulation mode)
iwpriv ra0 set ATE=TXCARR            ; Start Tx carrier, Measure carrier with instrument
iwpriv ra0 set ATETXPOW0=05
iwpriv ra0 set ATETXPOW1=05
iwpriv ra0 set ATETXFREQOFFSET=19
iwpriv ra0 set ATE=ATESTART
```

6.3.3 Check spectrum mask

```
iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATECHANNEL=1           ; set Channel
iwpriv ra0 set ATETXMODE=1           ; set TX-Mode.
iwpriv ra0 set ATETXMCS=7            ; set MCS type.
iwpriv ra0 set ATETXBW=0             ; set Bandwidth
iwpriv ra0 set ATETXCNT=200          ; Tx frame count(decimal)
iwpriv ra0 set ATE=TXFRAME           ; Start Tx Frame(inform BBP to change, modulation mode)
iwpriv ra0 set ATE=TXCONT            ; Start continuous TX, Measure spectrum mask with instrument
iwpriv ra0 set ATETXPOW0=5
iwpriv ra0 set ATETXPOW1=5
iwpriv ra0 set ATE=ATESTART
```

6.3.4 Frequency offset tuning

```
iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATECHANNEL=1           ; set Channel
```




iwpriv ra0 set ATETXMODE=1	; set TX-Mode.
iwpriv ra0 set ATETXMCS=7	; set MCS type.
iwpriv ra0 set ATETXCNT=200	; Tx frame count(decimal)
iwpriv ra0 set ATETXFREQOFFSET=0	; Set frequency offset 0(decimal)
iwpriv ra0 set ATE=TXFRAME	; Start Tx Frame
iwpriv ra0 set ATE=TXCARR	; Start Tx carrier, Measure carrier frequency with instrument
iwpriv ra0 set ATETXFREQOFFSET=10	; Dynamic turning frequency offset, 10(decimal)
iwpriv ra0 set ATETXFREQOFFSET=20	; Dynamic turning frequency offset, 20(decimal)
iwpriv ra0 set ATE=ATESTART	; Stop, Store the tuning result to EEPROM

6.3.5 Rx

iwpriv ra0 set ATE=ATESTART	
iwpriv ra0 set ATECHANNEL=1	; set Channel
iwpriv ra0 set ResetCounter=0	; Reset statistic counter
iwpriv ra0 set ATETXMODE=1	; set TX-Mode.
iwpriv ra0 set ATETXMCS=7	; set MCS type.
iwpriv ra0 set ATETXBW=0	; set Bandwidth
iwpriv ra0 set ATE=RXFRAME	; Start Rx,
iwpriv ra0 set ATERXFER=1	; show RxCnt and RSSI/per-antenna, Transmit test packets
iwpriv ra0 set ATE=ATESTART	; Stop
iwpriv ra0 stat	; get statistics counter
iwpriv ra0 set ATERXFER=1	
iwpriv ra0 set ATERXANT=1	

iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATERXANT=0
iwpriv ra0 set ATE=RXFRAME

6.3.6 Show all ate parameters

iwpriv ra0 set ATESHOW=1

Mode=4
TxPower0=0
TxPower1=0
TxAntennaSel=0
RxAntennaSel=0
BBPCurrentBW=0
GI=0
MCS=7
TxMode=1
Addr1=00:11:22:aa:bb:cc
Addr2=00:11:22:aa:bb:cc
Addr3=00:11:22:aa:bb:cc
Channel=1
TxLength=1024
TxCount=40000
TxRate=11
RFFreqOffset=0

6.3.7 Online help

iwpriv ra0 set ATEHELP=1

ATE=ATESTART, ATESTOP, TXCONT, TXCARR, TXFRAME, RXFRAME



ATEDA
ATESA
ATEBSSID
ATECHANNEL, range:0~14
ATETXPOW0, set power level of antenna 1.
ATETXPOW1, set power level of antenna 2.
ATETXANT, set TX antenna. 0:all, 1:antenna one, 2:antenna two.
ATERXANT, set RX antenna.0:all, 1:antenna one, 2:antenna tow, 3:antenna three.
ATETFREQOFFSET, set frequency offset, range 0~63
ATETXBW, set BandWidth, 0:20MHz, 1:40MHz.
ATETXLEN, set Frame length, range 24~1500
ATETXCNT, set how many frame going to transmit.
ATETXRATE, set rate, reference to rate table.
ATETXMCS, set MCS, reference to rate table.
ATETXMODE, set Mode 0:CCK, 1:OFDM, 2:HT-Mix, 3:GreenField, reference to rate table.
ATETXGI, set GI interval, 0:Long, 1:Short
ATERXFER, 0:disable Rx Frame error rate. 1:enable Rx Frame error rate.
ATESHOW, display all parameters of ATE.
ATEHELP, online help.

6.3.8 Display Rx Packet Count and RSSI

iwpriv ra0 set ATE=RXFRAME → Start Rx
iwpriv ra0 set ATERXANT=0 → Enable All Three Rx Antennas
iwpriv ra0 set ATERXFER=1 → Enable Rx Frame Error Rate: RxCnt/RxTotal

MImePeriodicExec: Rx packet cnt = 2/4
MImePeriodicExec: Rx AvgRssi0=-88, AvgRssi1=-80, AvgRssi2=-91

MImePeriodicExec: Rx packet cnt = 2/6
MImePeriodicExec: Rx AvgRssi0=-86, AvgRssi1=-77, AvgRssi2=-89...
...

iwpriv ra0 set ATE=RXFRAME → Start Rx
iwpriv ra0 set ATERXANT=1 → Enable Three Rx Antenna-1
iwpriv ra0 set ATERXFER=1 → Enable Rx Frame Error Rate: RxCnt/RxTotal

MImePeriodicExec: Rx packet cnt = 0/7
MImePeriodicExec: Rx AvgRssi=-87

MImePeriodicExec: Rx packet cnt = 7/14
MImePeriodicExec: Rx AvgRssi=-90
...
...

6.4 iwpriv ra0 bbp [parameters]=[Value]

Read/Write BBP register by ID number.

6.4.1 BBPID

Read BBP register, BBPID only, no "=" symbol.

BBPID:

0 ~ xx ; decimal, 8-bit

6.4.2 BBPID=Value

Write BBP register.

BBPID:

0 ~ xx ; decimal, 8-bit

Value:

00 ~ FF ; hexadecimal, 8-bit

6.5 iwpriv ra0 mac [parameters]=[val]

Read/Write MAC register by offset.

6.5.1 MAC_OFFSET

Read MAC register, MAC_OFFSET only, no "=" symbol.

MAC_OFFSET:

0000 ~ FFFF ; hexadecimal, 16-bit

6.5.2 MAC_OFFSET=Value

Write MAC register.

MAC_OFFSET:

0000 ~ FFFF ; hexadecimal, 16-bit

Value:

0000 ~ FFFF ; hexadecimal, 32-bit

6.6 iwpriv ra0 e2p [parameters]=[val]

Read/Write EEPROM content by address.

6.6.1 EEP_ADDR

Read EEPROM content, EEP_ADDR only, no "=" symbol.

EEP_ADDR:

00 ~ FF ; hexadecimal, 16-bit alignment (0, 2, 4, 6, 8, A, C, ...)

6.6.2 EEP_ADDR=Value

Write EEPROM content.

EEP_ADDR:

00 ~ FF ; hexadecimal, 16-bit alignment (0, 2, 4, 6, 8, A, C, ...)

Value:

0000 ~ FFFF ; hexadecimal, 16-bit

6.7 Example

6.7.1 Hardware access

```
iwpriv ra0 bbp 0           # read BBP register 0
iwpriv ra0 bbp 0=12        # write BBP register 0 as 0x12
iwpriv ra0 mac 0           # read MAC register 0
iwpriv ra0 mac 0=1234abcd  # write MAC register 0 as 0x1234abcd
iwpriv ra0 e2p 0           # read E2PROM 0
iwpriv ra0 e2p c=12ab      # write E2PROM 0xc as 0x12ab
```

6.7.2 Statistic counter operation

```
iwpriv ra0 stat           # read statistic counter
iwpriv ra0 set ResetCounter=0 # reset statistic counter
```

6.7.3 Suggestion:

1. To turn on ATE functionality, you have to add compile flag "RALINK_ATE" to Makefile
2. Before doing ATE testing, please stop AP function
3. If you want to test another ATE action, prefer to stop AP & ATE function
4. All ATE function settings will lose efficacy after reboot.
5. Before hardware register access, please reference hardware spec.

Note.

In ATE mode, the channel must set via "ATECHANNEL"

6.8 ated

ated - user space ATE agent program for RT2860 linux driver, Ralink Tech. Corp.

RT2860 ATE daemon - ated, which comes with RT2860 linux driver.

Here will explain the relationship between the linux driver, Windows QA GUI and RT2860 ATE daemon.

In addition, this will teach you how to use this ATE daemon.

6.8.1 Introduction

The ated is an optional user space component for RT2860 linux driver.

When ated starts, AP enters ATE mode (i.e. ATESTART) immediately.

It behaves as a proxy between Windows QA GUI and RT2860 linux driver when ATE process proceeds.

And ated will be killed automatically when Windows QA GUI is closed.

You can kill it manually, too (for example, type '\$killall ated').

RT2860 linux driver will leave ATE mode either ated is killed or QA GUI is closed.

6.8.2 Environment setup

1. Connect the platform you want to test directly with a Windows host by ether network line.
2. In the Windows host, run WinPcap_4_0.exe for the QA GUI.

6.8.3 How to use ated for ATE purpose

1. First you should set both "HAS_ATE=y" and "HAS_2860_QA=y" in the file `~/Module/os/linux/config.mk` and compile the driver.
2. Modify the Makefile according to our target "PLATFORM".
3. Change the path of "CROSS_COMPILE" if needed.
4. Remove "-I\$(INCLUDE)" about in line 39 if your target "PLATFORM" is not "PC".
5. Then type 'make' command to compile the source code of the daemon.
6. After the driver interface "ra0" has started up, attach both of "ra0" and the Ethernet interface to the bridge interface "br0".
7. Manually start ated, type '\$ated -bbrX -iraX'. (For further usage of options, type \$ated -h)
8. In the Windows host, run RT2860QA_ATE.exe.
9. Select the wired network adapter.
10. Choose 2860_ATE, and then press OK.

NOTE:

The names of WLAN interface (default is "ra0") and Bridge interface (default is "br0") must be specified manually (for example : '\$ated -b br1 -ira2') if your WLAN interface or Bridge interface is not "ra0" or "br0" respectively !

Ralink Website document

for stevej@cradlepoint.com
And Company Use Only

7 IOCTL

7.1 Parameters for iwconfig

Access	Description	ID	Parameters
Get	BSSID, MAC Address	SIOCGIFHWADDR	wrq->u.name, (length = 6)
	WLAN Name	SIOCGIWNAME	wrq->u.name = "RT2860 Wireless", length = strlen(wrq->u.name)
	SSID	SIOCGIWESSID	<pre> erq = &wrq->u.essid; if(OPSTATUS_TEST_FLAG(pAd,fOP_STATUS_MEDIA_STATE_CONNECTED)) { erq->flags=1; erq->length = pAd-> CommonCfg.SsidLen; Status = copy_to_user(erq->pointer, pAd-> CommonCfg.Ssid, erq->length); } else { erq->flags=0; erq->length=0; } </pre>
	Channel / Frequency (Hz)	SIOCGIWFREQ	<pre> wrq->u.freq.m = pAd-> CommonCfg.Channel; wrq->u.freq.e = 0; wrq->u.freq.i = 0; </pre>
	Node name/nickname	SIOCGIWNICKN	<pre> erq = &wrq->u.data; erq->length = strlen(pAd->u.nickn); Status = copy_to_user(erq->pointer, pAd->u.nickn, erq->length); </pre>
	Bit Rate (bps)	SIOCGIWRATE	<pre> wrq->u.bitrate.value = RateIdTo500Kbps[pAd-> CommonCfg.TxRate] * 500000; wrq->u.bitrate.disabled = 0; </pre>
	RTS/CTS threshold	SIOCGIWRTS	<pre> wrq->u.rts.value = (INT) pAd-> CommonCfg.RtsThreshold; wrq->u.rts.disabled = (wrq->u.rts.value == MAX_RTS_THRESHOLD); wrq->u.rts.fixed = 1; </pre>
	Fragmentation threshold (bytes)	SIOCGIWFRAG	<pre> wrq->u.frag.value = (INT) pAd-> CommonCfg.FragmentThreshold; wrq->u.frag.disabled = (wrq->u.frag.value >= MAX_FRAG_THRESHOLD); wrq->u.frag.fixed = 1; </pre>
	Encoding token & mode	SIOCGIWENCODE	<pre> index = (wrq->u.encoding.flags & IW_ENCODE_INDEX) - 1; if ((index < 0) (index >= NR_WEP_KEYS)) index = pAd-> CommonCfg.DefaultKeyId; // Default key for tx (shared key) if (pAd-> CommonCfg.AuthMode == Ndis802_11AuthModeOpen) wrq->u.encoding.flags = IW_ENCODE_OPEN; else if (pAd-> CommonCfg.AuthMode == Ndis802_11AuthModeShared) wrq->u.encoding.flags = IW_ENCODE_RESTRICTED; if (pAd-> CommonCfg.WepStatus == Ndis802_11WEPDisabled) wrq->u.encoding.flags = IW_ENCODE_DISABLED; else { if(wrq->u.encoding.pointer) { wrq->u.encoding.length = pAd->SharedKey[index].KeyLen; Status = copy_to_user(wrq->u.encoding.pointer, pAd->SharedKey[index].Key, pAd->SharedKey[index].KeyLen); wrq->u.encoding.flags = (index + 1); } } </pre>
	AP's MAC address	SIOCGIWAP	<pre> wrq->u.ap_addr.sa_family = ARPHRD_ETHER; memcpy(wrq->u.ap_addr.sa_data, &pAd-> CommonCfg.Bssid, ETH_ALEN); </pre>
	Operation Mode	SIOCGIWMODE	<pre> if (ADHOC_ON(pAd)) { BssType = Ndis802_11IBSS; } </pre>

			<pre> wrq->u.mode = IW_MODE_ADHOC; } else if (INFRA_ON(pAd)) { BssType = Ndis802_11Infrastructure; wrq->u.mode = IW_MODE_INFRA; } else { BssType = Ndis802_11AutoUnknown; wrq->u.mode = IW_MODE_AUTO; } </pre>
Access	Description	ID	Parameters
Set	SSID	SIOCSIWESSID	<pre> erq = &wrq->u.essid; memset(&Ssid, 0x00, sizeof(NDIS_802_11_SSID)); if (erq->flags) { if (erq->length > IW_ESSID_MAX_SIZE) { Status = -E2BIG; break; } Status = copy_from_user(Ssid.Ssid, erq->pointer, (erq->length - 1)); Ssid.SsidLength = erq->length - 1; //minus null character. } else { Ssid.SsidLength = 0; // ANY ssid memcpy(pSsid->Ssid, "", 0); pAd->CommonCfg.BssType = BSS_INFRA; pAd->CommonCfg.AuthMode = Ndis802_11AuthModeOpen; pAd->CommonCfg.WepStatus = Ndis802_11EncryptionDisabled; } pSsid = &Ssid; if (pAd->Mlme.CntlMachine.CurrState != CNTL_IDLE) { MlmeRestartStateMachine(pAd); } pAd->MlmeAux.CurrReqIsFromNdis = FALSE; MlmeEnqueue(pAd, MLME_CNTL_STATE_MACHINE, OID_802_11_SSID, sizeof(NDIS_802_11_SSID), (VOID *)pSsid); Status = NDIS_STATUS_SUCCESS; StateMachineTouched = TRUE; </pre>
	Channel / Frequency (Hz)	SIOCSIWFREQ	<pre> frq = &wrq->u.freq; if((frq->e == 0) && (frq->m <= 1000)) chan = frq->m; // Setting by channel number else MAP_KHZ_TO_CHANNEL_ID((frq->m /100) , chan); pAd->CommonCfg.Channel = chan; </pre>
	node name/nickname	SIOCSIWNICKN	<pre> erq = &wrq->u.data; if (erq->flags) { if (erq->length <= IW_ESSID_MAX_SIZE) Status = copy_from_user(pAd->nickn, erq->pointer, erq->length); else Status = -E2BIG; } </pre>
	Bit Rate (bps)	SIOCSIWRATE	<pre> RTMPSetDesiredRates(pAd, wrq->u.bitrate.value); </pre>
	RTS/CTS threshold	SIOCSIWRTS	<pre> RtsThresh = wrq->u.rts.value; if (wrq->u.rts.disabled) RtsThresh = MAX_RTS_THRESHOLD; </pre>

		<pre> if((RtsThresh > 0) && (RtsThresh <= MAX_RTS_THRESHOLD)) pAd->CommonCfg.RtsThreshold = (USHORT)RtsThresh; else if (RtsThresh == 0) pAd->CommonCfg.RtsThreshold = MAX_RTS_THRESHOLD; </pre>
Fragmentation threshold (bytes)	SIOCSIWFRAG	<pre> FragThresh = wrq->u.frag.value; if (wrq->u.rts.disabled) FragThresh = MAX_FRAG_THRESHOLD; if ((FragThresh >= MIN_FRAG_THRESHOLD) && (FragThresh <= MAX_FRAG_THRESHOLD)) pAd->CommonCfg.FragmentThreshold = (USHORT)FragThresh; else if (FragThresh == 0) pAd->CommonCfg.FragmentThreshold = MAX_FRAG_THRESHOLD; if (pAd->CommonCfg.FragmentThreshold == MAX_FRAG_THRESHOLD) pAd->CommonCfg.bFragmentZeroDisable = TRUE; else pAd->CommonCfg.bFragmentZeroDisable = FALSE; </pre>
Encoding token & mode	SIOCSIWENCOD	<pre> index = (wrq->u.encoding.flags & IW_ENCODE_INDEX) - 1; if((index < 0) (index >= NR_WEP_KEYS)) index = pAd->CommonCfg.DefaultKeyId; // Default key for tx (shared key) if(wrq->u.encoding.pointer) { len = wrq->u.encoding.length; if(len > WEP_LARGE_KEY_LEN) len = WEP_LARGE_KEY_LEN; memset(pAd->SharedKey[index].Key, 0x00, MAX_LEN_OF_KEY); Status = copy_from_user(pAd->SharedKey[index].Key, wrq->u.encoding.pointer, len); pAd->SharedKey[index].KeyLen = len <= WEP_SMALL_KEY_LEN ? WEP_SMALL_KEY_LEN : WEP_LARGE_KEY_LEN; } pAd->CommonCfg.DefaultKeyId = (UCHAR) index; if (wrq->u.encoding.flags & IW_ENCODE_DISABLED) pAd->CommonCfg.WepStatus = Ndis802_11WEPDisabled; else pAd->CommonCfg.WepStatus = Ndis802_11WEPEnabled; if (wrq->u.encoding.flags & IW_ENCODE_RESTRICTED) pAd->CommonCfg.AuthMode = Ndis802_11AuthModeShared; else pAd->CommonCfg.AuthMode = Ndis802_11AuthModeOpen; if(pAd->CommonCfg.WepStatus == Ndis802_11WEPDisabled) pAd->CommonCfg.AuthMode = Ndis802_11AuthModeOpen; </pre>
AP's MAC address	SIOCSIWAP	<pre> Status = copy_from_user(&Bssid, &wrq->u.ap_addr.sa_data, sizeof(NDIS_802_11_MAC_ADDRESS)); if (pAd->Mlme.CntlMachine.CurrState != CNTL_IDLE) { MlmeRestartStateMachine(pAd); } pAd->MlmeAux.CurrReqIsFromNdis = FALSE; MlmeEnqueue(pAd, MLME_CNTL_STATE_MACHINE, OID_802_11_BSSID, sizeof(NDIS_802_11_MAC_ADDRESS), (VOID *)&Bssid); Status = NDIS_STATUS_SUCCESS; StateMachineTouched = TRUE; </pre>

	Operation Mode	SIOCSIWMODE	<pre> if(wrq->u.mode == IW_MODE_ADHOC) { if (pAd->CommonCfg.BssType != BSS_ADHOC) { pAd->bConfigChanged = TRUE; } pAd->CommonCfg.BssType = BSS_ADHOC; } else if (wrq->u.mode == IW_MODE_INFRA) { if (pAd->CommonCfg.BssType != BSS_INFRA) { pAd->bConfigChanged = TRUE; } pAd->CommonCfg.BssType = BSS_INFRA; } else { Status = -EINVAL; } pAd->CommonCfg.WpaState = SS_NOTUSE; </pre>
--	----------------	-------------	---

7.2 Parameters for iwpriv

Please refer section 3 to have iwpriv parameters and values.

Parameters:

```

int    socket_id;
char   name[25];           // interface name
char   data[255];         // command string
struct iwreq wrq;

```

Default setting:

```

wrq.ifr_name = name = "ra0"; // interface name
wrq.u.data.pointer = data;   // data buffer of command string
wrq.u.data.length = strlen(data); // length of command string
wrq.u.data.flags = 0;

```

Data Structure:

Please refer to `./include/oid.h` for update and detail definition.

7.2.1 Set Data, Parameters the same as iwpriv

Command and IOCTL Function		
Set Data		
Function Type	Command	IOCTL
RTPRIV_IOCTL_SET	<code>iwpriv ra0 set SSID=RT2860AP</code>	<pre> sprintf(name, "ra0"); strcpy(data, "SSID=RT2860AP"); strcpy(wrq.ifr_name, name); wrq.u.data.length = strlen(data); wrq.u.data.pointer = data; wrq.u.data.flags = 0; ioctl(socket_id, RTPRIV_IOCTL_SET, &wrq); </pre>

7.2.2 Get Data, Parameters is the same as iwpriv

Command and IOCTL Function		
Get Data		
Function Type	Command	IOCTL
RTPRIV_IOCTL_STATISTICS	Iwpriv ra0 stat	<pre>sprintf(name, "ra0"); strcpy(data, "stat"); strcpy(wrq.ifr_name, name); wrq.u.data.length = strlen(data); wrq.u.data.pointer = data; wrq.u.data.flags = 0; ioctl(socket_id, RTPRIV_IOCTL_STATISTICS, &wrq);</pre>
RTPRIV_IOCTL_GSITESURVEY	Iwpriv ra0 get_site_survey	<pre>sprintf(name, "ra0"); strcpy(data, "get_site_survey"); strcpy(wrq.ifr_name, name); wrq.u.data.length = strlen(data); wrq.u.data.pointer = data; wrq.u.data.flags = 0; ioctl(socket_id, RTPRIV_IOCTL_GSITESURVEY, &wrq);</pre>

7.2.3 Set Raw Data with Flags

IOCTL Function	
Set Raw Data by I/O Control Interface with Flags	
Function Type	IOCTL
RT_OID_802_11_COUNTRY_REGION	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(UCHAR)); wrq.u.data.length = sizeof(UCHAR); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_COUNTRY_REGION; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_BSSID_LIST_SCAN	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); wrq.u.data.length = 0; wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_BSSID_LIST_SCAN; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_SSID	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_SSID)); wrq.u.data.length = sizeof(NDIS_802_11_SSID); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_SSID; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_BSSID	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_MAC_ADDRESS)); wrq.u.data.length = sizeof(NDIS_802_11_MAC_ADDRESS); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_BSSID; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_RADIO	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(BOOLEAN)); wrq.u.data.length = sizeof(BOOLEAN); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_RADIO; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_PHY_MODE	<pre>sprintf(name, "ra0");</pre>

	<pre>strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RT_802_11_PHY_MODE)); wrq.u.data.length = sizeof(RT_802_11_PHY_MODE); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_PHY_MODE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_STA_CONFIG	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RT_802_11_STA_CONFIG)); wrq.u.data.length = sizeof(RT_802_11_STA_CONFIG); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_STA_CONFIG; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_DESIRED_RATES	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_RATES)); wrq.u.data.length = sizeof(NDIS_802_11_RATES); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_DESIRED_RATES; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_PREAMBLE	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RT_802_11_PREAMBLE)); wrq.u.data.length = sizeof(RT_802_11_PREAMBLE); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_PREAMBLE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_WEP_STATUS	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_WEP_STATUS)); wrq.u.data.length = sizeof(NDIS_802_11_WEP_STATUS); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_WEP_STATUS; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_AUTHENTICATION_MODE	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_AUTHENTICATION_MODE)); wrq.u.data.length = sizeof(NDIS_802_11_AUTHENTICATION_MODE); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_AUTHENTICATION_MODE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_INFRASTRUCTURE_MODE	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_NETWORK_INFRASTRUCTURE)); wrq.u.data.length = sizeof(NDIS_802_11_NETWORK_INFRASTRUCTURE); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_INFRASTRUCTURE_MODE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_REMOVE_WEP	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_KEY_INDEX)); wrq.u.data.length = sizeof(NDIS_802_11_KEY_INDEX); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_REMOVE_WEP; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_RESET_COUNTERS	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); wrq.u.data.length = 0; wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_RESET_COUNTERS; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_RTS_THRESHOLD	<pre>printf(name, "ra0");</pre>

	<pre>strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_RTS_THRESHOLD)); wrq.u.data.length = sizeof(NDIS_802_11_RTS_THRESHOLD); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_RTS_THRESHOLD; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_FRAGMENTATION_THRESHOLD	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_FRAGMENTATION_THRESHOLD)); wrq.u.data.length = sizeof(NDIS_802_11_FRAGMENTATION_THRESHOLD); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_FRAGMENTATION_THRESHOLD; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_POWER_MODE	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_POWER_MODE)); wrq.u.data.length = sizeof(NDIS_802_11_POWER_MODE); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_POWER_MODE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_TX_POWER_LEVEL	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_TX_POWER_LEVEL)); wrq.u.data.length = sizeof(NDIS_802_11_TX_POWER_LEVEL); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_TX_POWER_LEVEL; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_TX_POWER_LEVEL_1	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_TX_POWER_LEVEL_1; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_NETWORK_TYPE_IN_USE	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_NETWORK_TYPE)); wrq.u.data.length = / sizeof(NDIS_802_11_NETWORK_TYPE); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_NETWORK_TYPE_IN_USE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_RX_ANTENNA_SELECTED	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_ANTENNA)); wrq.u.data.length = sizeof(NDIS_802_11_ANTENNA); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_RX_ANTENNA_SELECTED; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_TX_ANTENNA_SELECTED	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_ANTENNA)); wrq.u.data.length = sizeof(NDIS_802_11_ANTENNA); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_TX_ANTENNA_SELECTED; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_ADD_WPA	<pre>printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, 32); wrq.u.data.length = 32; wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_ADD_WPA; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>

OID_802_11_REMOVE_KEY	printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_REMOVE_KEY)); wrq.u.data.length = sizeof(NDIS_802_11_REMOVE_KEY); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_REMOVE_KEY; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
OID_802_11_ADD_KEY	printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, keylength); //5,10,13,26 wrq.u.data.length = keylength L; wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_ADD_KEY; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
OID_802_11_SET_IEEE8021X	printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(BOOLEAN)); wrq.u.data.length = sizeof(BOOLEAN); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_SET_IEEE8021X; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
OID_802_11_SET_IEEE8021X_REQUIRE_KEY	printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(BOOLEAN)); wrq.u.data.length = sizeof(BOOLEAN); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_SET_IEEE8021X_REQUIRE_KEY; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
OID_802_11_ADD_WEP	printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, keylength); //5,10,13,26 wrq.u.data.length = keylength; wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_RADIO; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
OID_802_11_CONFIGURATION	printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_CONFIGURATION)); wrq.u.data.length = sizeof(NDIS_802_11_CONFIGURATION); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_CONFIGURATION; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
OID_SET_COUNTERMEASURES	printf(name, "ra0"); strcpy(wrq.ifr_name, name); wrq.u.data.length = 0; wrq.u.data.pointer = data; wrq.u.data.flags = OID_SET_COUNTERMEASURES; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
OID_802_11_DISASSOCIATE	printf(name, "ra0"); strcpy(wrq.ifr_name, name); wrq.u.data.length = 0; wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_DISASSOCIATE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
OID_802_11_PMKID	printf(name, "ra0"); strcpy(wrq.ifr_name, name); wrq.u.data.length = keylength; //follow your setting wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_PMKID; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
RT_OID_WPA_SUPPLICANT_SUPPORT	printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(BOOLEAN));

	<pre> wrq.u.data.length = sizeof(BOOLEAN); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_WPA_SUPPLICANT_SUPPORT; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
RT_OID_WPA_SUPPLICANT_SUPPORT	<pre> printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_WPA_SUPPLICANT_SUPPORT; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
RT_SET_DEL_MAC_ENTRY	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0xdd, 6); strcpy(wrq.ifr_name, name); wrq.u.data.length = 6; wrq.u.data.pointer = data; wrq.u.data.flags = RT_SET_DEL_MAC_ENTRY; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
RT_OID_802_11_SET_HT_PHYMODE OID_GET_SET_TOGGLE	<pre> typedef struct { RT_802_11_PHY_MODE PhyMode; UCHAR TransmitNo; UCHAR HtMode; //HTMODE_GF or HTMODE_MM UCHAR ExtOffset; //extension channel above or below UCHAR MCS; UCHAR BW; UCHAR STBC; UCHAR SHORTGI; UCHAR rsv; } OID_SET_HT_PHYMODE; RT_802_11_PHY_MODE tmp_ht_mode; sprintf(wrq.ifr_name, "ra0"); wrq.u.data.pointer = (caddr_t) &tmp_ht_mode; wrq.u.data.length = sizeof(RT_802_11_PHY_MODE); wrq.u.data.flags = RT_OID_802_11_SET_HT_PHYMODE OID_GET_SET_TOGGLE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>

7.2.4 Get Raw Data with Flags

IOCTL Function	
Get Raw Data by I/O Control Interface with Flags	
Function Type	IOCTL
RT_OID_DEVICE_NAME	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, 255); wrq.u.data.length = 255; wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_DEVICE_NAME; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
RT_OID_VERSION_INFO	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RT_VERSION_INFO)); wrq.u.data.length = sizeof(RT_VERSION_INFO); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_VERSION_INFO; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_BSSID_LIST	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, BssLen); </pre>

	<pre>wrq.u.data.length = BssLen; wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_BSSID_LIST; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_3_CURRENT_ADDRESS	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(CurrentAddress)); wrq.u.data.length = sizeof(CurrentAddress); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_3_CURRENT_ADDRESS; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_GEN_MEDIA_CONNECT_STATUS	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_MEDIA_STATE)); wrq.u.data.length = sizeof(NDIS_MEDIA_STATE); wrq.u.data.pointer = data; wrq.u.data.flags = OID_GEN_MEDIA_CONNECT_STATUS; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_BSSID	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_MAC_ADDRESS)); wrq.u.data.length = sizeof(NDIS_802_11_MAC_ADDRESS); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_BSSID; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_SSID	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_SSID)); wrq.u.data.length = sizeof(NDIS_802_11_SSID); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_SSID; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_QUERY_LINK_STATUS	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RT_802_11_LINK_STATUS)); wrq.u.data.length = sizeof(RT_802_11_LINK_STATUS); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_QUERY_LINK_STATUS; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_CONFIGURATION	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_CONFIGURATION)); wrq.u.data.length = sizeof(NDIS_802_11_CONFIGURATION); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_CONFIGURATION; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_RSSI_TRIGGER	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ullInfo)); wrq.u.data.length = sizeof(ullInfo); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_RSSI_TRIGGER; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_RSSI	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ullInfo)); wrq.u.data.length = sizeof(ullInfo); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_RSSI; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_RSSI_1	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ullInfo));</pre>

	<pre> wrq.u.data.length = sizeof(ullInfo); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_RSSI_1; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
RT_OID_802_11_RSSI_2	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ullInfo)); wrq.u.data.length = sizeof(ullInfo); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_RSSI_2; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_STATISTICS	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_STATISTICS)); wrq.u.data.length = sizeof(NDIS_802_11_STATISTICS); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_STATISTICS; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_GEN_RCV_OK	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ullInfo)); wrq.u.data.length = sizeof(ullInfo); wrq.u.data.pointer = data; wrq.u.data.flags = OID_GEN_RCV_OK; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_GEN_RCV_NO_BUFFER	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ullInfo)); wrq.u.data.length = sizeof(ullInfo); wrq.u.data.pointer = data; wrq.u.data.flags = OID_GEN_RCV_NO_BUFFER; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
RT_OID_802_11_PHY_MODE	<pre> typedef enum RT_802_11_PHY_MODE { PHY_11BG_MIXED = 0, PHY_11B, PHY_11A, PHY_11ABG_MIXED, PHY_11G, PHY_11ABGN_MIXED, // both band 5 PHY_11N, // 6 PHY_11GN_MIXED, // 2.4G band 7 PHY_11AN_MIXED, // 5G band 8 PHY_11BGN_MIXED, // if check 802.11b. 9 PHY_11AGN_MIXED, // if check 802.11b. 10 } RT_802_11_PHY_MODE sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ullInfo)); wrq.u.data.length = sizeof(ullInfo); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_PHY_MODE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
RT_OID_802_11_STA_CONFIG	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RT_802_11_STA_CONFIG)); wrq.u.data.length = sizeof(RT_802_11_STA_CONFIG); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_STA_CONFIG; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_RTS_THRESHOLD	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RtsThresh)); wrq.u.data.length = sizeof(RtsThresh); </pre>

	<pre> wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_RTS_THRESHOLD; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_FRAGMENTATION_THRESHOLD	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(FragThresh)); wrq.u.data.length = sizeof(FragThresh); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_FRAGMENTATION_THRESHOLD; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_POWER_MODE	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(PowerMode)); wrq.u.data.length = sizeof(PowerMode); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_POWER_MODE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
RT_OID_802_11_RADIO	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RadioState)); wrq.u.data.length = sizeof(RadioState); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_RADIO; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_INFRASTRUCTURE_MODE	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(BssType)); wrq.u.data.length = sizeof(BssType); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_INFRASTRUCTURE_MODE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
RT_OID_802_11_PREAMBLE	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(PreamType)); wrq.u.data.length = sizeof(PreamType); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_PREAMBLE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_AUTHENTICATION_MODE	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(AuthMode)); wrq.u.data.length = sizeof(AuthMode); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_AUTHENTICATION_MODE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_WEP_STATUS	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(WepStatus)); wrq.u.data.length = sizeof(WepStatus); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_WEP_STATUS; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_TX_POWER_LEVEL	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_TX_POWER_LEVEL; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_TX_POWER_LEVEL_1	<pre> sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); </pre>

	<pre>wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_TX_POWER_LEVEL_1; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_NETWORK_TYPES_SUPPORTED	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, 16); wrq.u.data.length = 16; wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_NETWORK_TYPES_SUPPORTED; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
OID_802_11_NETWORK_TYPE_IN_USE	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_NETWORK_TYPE_IN_USE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_QUERY_EEPROM_VERSION	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_QUERY_EEPROM_VERSION; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_QUERY_FIRMWARE_VERSION	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_QUERY_FIRMWARE_VERSION; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_QUERY_NOISE_LEVEL	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(UCHAR)); wrq.u.data.length = sizeof(UCHAR); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_QUERY_NOISE_LEVEL; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_EXTRA_INFO	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_EXTRA_INFO; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_QUERY_PIDVID	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(ULONG)); wrq.u.data.length = sizeof(ULONG); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_802_11_QUERY_PIDVID; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_WE_VERSION_COMPILED	<pre>sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(UINT)); wrq.u.data.length = sizeof(UINT); wrq.u.data.pointer = data; wrq.u.data.flags = RT_OID_WE_VERSION_COMPILED; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_QUERY_LAST_TX_RATE	<pre>HTTRANSMIT_SETTING tmpHT; sprintf(wrq.ifr_name, "ra0"); wrq.u.data.pointer = (caddr_t) &tmpHT;</pre>

	<pre>wrq.u.data.flags = RT_OID_802_11_QUERY_LAST_TX_RATE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
RT_OID_802_11_QUERY_LAST_RX_RATE	<pre>HTTRANSMIT_SETTING tmpHT; sprintf(wrq.ifr_name, "ra0"); wrq.u.data.pointer = (caddr_t) &tmpHT; wrq.u.data.flags = RT_OID_802_11_QUERY_LAST_RX_RATE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>
SHOW_IPV4_MAT_INFO	<pre>u_char buffer[IW_PRIV_SIZE_MASK]; sprintf(wrq.ifr_name, "ra0"); wrq.u.data.pointer = (caddr_t) buffer; wrq.u.data.flags = SHOW_IPV4_MAT_INFO; ioctl(socket_id, RTPRIV_IOCTL_SHOW, &wrq);</pre>
SHOW_IPV6_MAT_INFO	<pre>u_char buffer[IW_PRIV_SIZE_MASK]; sprintf(wrq.ifr_name, "ra0"); wrq.u.data.pointer = (caddr_t) buffer; wrq.u.data.flags = SHOW_IPV6_MAT_INFO; ioctl(socket_id, RTPRIV_IOCTL_SHOW, &wrq);</pre>
SHOW_ETH_CLONE_MAC	<pre>u_char buffer[IW_PRIV_SIZE_MASK]; sprintf(wrq.ifr_name, "ra0"); wrq.u.data.pointer = (caddr_t) buffer; wrq.u.data.flags = SHOW_ETH_CLONE_MAC; ioctl(socket_id, RTPRIV_IOCTL_SHOW, &wrq);</pre>
SHOW_CONN_STATUS	<pre>u_char buffer[IW_PRIV_SIZE_MASK]; sprintf(wrq.ifr_name, "ra0"); wrq.u.data.pointer = (caddr_t) buffer; wrq.u.data.flags = SHOW_CONN_STATUS; ioctl(socket_id, RTPRIV_IOCTL_SHOW, &wrq);</pre>

7.2.5 Set Raw Data with Flags

IOCTL Function	
Get Raw Data by I/O Control Interface with Flags	
Function Type	IOCTL
RT_OID_802_11_SET_HT_PHYMODE OID_GET_SET_TOGGLE	<pre>typedef struct { RT_802_11_PHY_MODE PhyMode; UCHAR TransmitNo; UCHAR HtMode; //HTMODE_GF or HTMODE_MM UCHAR ExtOffset; //extension channel above or below UCHAR MCS; UCHAR BW; UCHAR STBC; UCHAR SHORTGI; UCHAR rsv; } OID_SET_HT_PHYMODE ; RT_802_11_PHY_MODE tmp_ht_mode; sprintf(wrq.ifr_name, "ra0"); wrq.u.data.pointer = (caddr_t) &tmp_ht_mode; wrq.u.data.length = sizeof(RT_802_11_PHY_MODE); wrq.u.data.flags = RT_OID_802_11_SET_HT_PHYMODE OID_GET_SET_TOGGLE; ioctl(socket_id, RT_PRIV_IOCTL, &wrq);</pre>

8 IOCTL HOW TO

8.1 Get Data

8.1.1 GET IPv4 and MAC mapping table:

Linux console command: **iwpriv ra0 ipv4_matinfo**

sample code =>

```
u_char buffer[IW_PRIV_SIZE_MASK];

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) buffer;

wrq.u.data.flags = SHOW_IPV4_MAT_INFO;

ioctl(socket_id, RTPRIV_IOCTL_SHOW, &wrq);
```

8.1.2 GET IPv6 and MAC mapping table:

Linux console command: **iwpriv ra0 ipv6_matinfo**

sample code =>

```
u_char buffer[IW_PRIV_SIZE_MASK];

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) buffer;

wrq.u.data.flags = SHOW_IPV6_MAT_INFO;

ioctl(socket_id, RTPRIV_IOCTL_SHOW, &wrq);
```

8.1.3 GET value of clone MAC in Dongle mode:

Linux console command: **iwpriv ra0 cloneMAC**

sample code =>

```
u_char buffer[IW_PRIV_SIZE_MASK];

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) buffer;

wrq.u.data.flags = SHOW_ETH_CLONE_MAC;

ioctl(socket_id, RTPRIV_IOCTL_SHOW, &wrq);
```

8.1.4 GET station connection status:



Linux console command: iwpriv ra0 connStatus

sample code =>

```
u_char buffer[IW_PRIV_SIZE_MASK];

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) buffer;

wrq.u.data.flags = SHOW_CONN_STATUS;

ioctl(socket_id, RTPRIV_IOCTL_SHOW, &wrq);
```

8.1.5 GET station statistics information:

Linux console command: iwpriv ra0 stat

sample code =>

```
u_char buffer[IW_PRIV_SIZE_MASK];

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) buffer;

wrq.u.data.flags = 0;

ioctl(socket_id, RTPRIV_IOCTL_STATISTICS, &wrq);
```

8.1.6 GET AP list table:

Linux console command: iwpriv ra0 get_site_survey

sample code =>

```
u_char buffer[4096];

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) buffer;

wrq.u.data.flags = 0;

ioctl(socket_id, RTPRIV_IOCTL_GSITESURVEY, &wrq);
```

8.1.7 GET scan table:

sample code =>

```
u_char buffer[4096];

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) buffer;
```



```
wrq.u.data.length = 4096;

wrq.u.data.flags = OID_802_11_BSSID_LIST;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);

PNDIS_802_11_BSSID_LIST_EX pBssidList = (PNDIS_802_11_BSSID_LIST_EX) buffer ;
```

8.1.8 GET station's MAC:

sample code =>

```
u_char buffer[6];

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) buffer;

wrq.u.data.flags = OID_802_3_CURRENT_ADDRESS;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.9 GET station connection status:

Sample code =>

```
#define NdisMediaStateConnected      1

#define NdisMediaStateDisconnected  0

NDIS_MEDIA_STATE MediaState;

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) &MediaState;

wrq.u.data.flags = OID_GEN_MEDIA_CONNECT_STATUS;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.10 GET AP's BSSID

Sample code =>

```
char BSSID[6];

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) BSSID;

wrq.u.data.flags = OID_802_11_BSSID;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.11 GET SSID

Sample code =>



```
NDIS_802_11_SSID SSID;

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) &SSID;

wrq.u.data.flags = OID_802_11_SSID;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.12 GET station's last TX related information:

Sample code =>

```
HTTRANSMIT_SETTING tmpHT;

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) &tmpHT;

wrq.u.data.flags = RT_OID_802_11_QUERY_LAST_TX_RATE;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.13 GET station's last RX related information:

Sample code =>

```
HTTRANSMIT_SETTING tmpHT;

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) &tmpHT;

wrq.u.data.flags = RT_OID_802_11_QUERY_LAST_RX_RATE;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.14 GET station's wireless mode:

Sample code =>

```
typedef enum _RT_802_11_PHY_MODE {

    PHY_11BG_MIXED = 0,

    PHY_11B,

    PHY_11A,

    PHY_11ABG_MIXED,

    PHY_11G,

    PHY_11ABGN_MIXED,           // both band    5

    PHY_11N,                   //           6
```



```
PHY_11GN_MIXED,           // 2.4G band           7
PHY_11AN_MIXED,           // 5G band           8
PHY_11BGN_MIXED,          // if check 802.11b.  9
PHY_11AGN_MIXED,          // if check 802.11b.  10

} RT_802_11_PHY_MODE
```

```
unsigned long tmp_mode;

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) &tmp_mode;

wrq.u.data.flags = RT_OID_802_11_PHY_MODE;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.15 GET Bss type:

Sample code =>

```
typedef enum _NDIS_802_11_NETWORK_INFRASTRUCTURE
{
    Ndis802_11IBSS,
    Ndis802_11Infrastructure,
    Ndis802_11AutoUnknown,
    Ndis802_11Monitor,
    Ndis802_11InfrastructureMax // Not a real value, defined as upper bound
} NDIS_802_11_NETWORK_INFRASTRUCTURE

NDIS_802_11_NETWORK_INFRASTRUCTURE BssType;

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) &BssType;

wrq.u.data.flags = OID_802_11_INFRASTRUCTURE_MODE;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.16 GET Authentication Mode:

Sample code =>

```
typedef enum _NDIS_802_11_AUTHENTICATION_MODE
```



```
{
    Ndis802_11AuthModeOpen,

    Ndis802_11AuthModeShared,

    Ndis802_11AuthModeAutoSwitch,

    Ndis802_11AuthModeWPA,

    Ndis802_11AuthModeWPAPSK,

    Ndis802_11AuthModeWPANone,

    Ndis802_11AuthModeWPA2,

    Ndis802_11AuthModeWPA2PSK,

    Ndis802_11AuthModeWPA1WPA2,

    Ndis802_11AuthModeWPA1PSKWPA2PSK,

    Ndis802_11AuthModeMax    // Not a real mode, defined as upper bound
} NDIS_802_11_AUTHENTICATION_MODE

NDIS_802_11_AUTHENTICATION_MODE AuthMode;

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) & AuthMode;

wrq.u.data.flags = OID_802_11_AUTHENTICATION_MODE;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.17 GET Encryption Type:

Sample code =>

```
typedef enum _NDIS_802_11_WEP_STATUS
{
    Ndis802_11WEPEnabled,

    Ndis802_11Encryption1Enabled = Ndis802_11WEPEnabled,

    Ndis802_11WEPDisabled,

    Ndis802_11EncryptionDisabled = Ndis802_11WEPDisabled,

    Ndis802_11WEPKeyAbsent,

    Ndis802_11Encryption1KeyAbsent = Ndis802_11WEPKeyAbsent,

    Ndis802_11WEPNotSupported,
```



```
Ndis802_11EncryptionNotSupported = Ndis802_11WEPNotSupported,  
  
Ndis802_11Encryption2Enabled,  
  
Ndis802_11Encryption2KeyAbsent,  
  
Ndis802_11Encryption3Enabled,  
  
Ndis802_11Encryption3KeyAbsent,  
  
Ndis802_11Encryption4Enabled,    // TKIP or AES mix  
  
Ndis802_11Encryption4KeyAbsent,  
  
} NDIS_802_11_WEP_STATUS, *PNDIS_802_11_WEP_STATUS,
```

```
NDIS_802_11_WEP_STATUS  WepStatus;  
  
sprintf(wrq.ifr_name, "ra0");  
  
wrq.u.data.pointer = (caddr_t) & WepStatus;  
  
wrq.u.data.flags = OID_802_11_WEP_STATUS;  
  
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.18 GET RSSI 0 (unit: db)

Sample code =>

```
long rssi_0  
  
sprintf(wrq.ifr_name, "ra0");  
  
wrq.u.data.pointer = (caddr_t) & rssi_0;  
  
wrq.u.data.flags = RT_OID_802_11_RSSI;  
  
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.19 GET RSSI 1 (unit: db)

Sample code =>

```
long rssi_1  
  
sprintf(wrq.ifr_name, "ra0");  
  
wrq.u.data.pointer = (caddr_t) & rssi_1;  
  
wrq.u.data.flags = RT_OID_802_11_RSSI_1;  
  
ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.20 GET RSSI 2 (unit: db)



Sample code =>

```
long rssi_2

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) &rssi_2;

wrq.u.data.flags = RT_OID_802_11_RSSI_2;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.21 GET Driver wireless extension version

Sample code =>

```
Unsigned int wext_version;

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) &wext_version;

wrq.u.data.flags = RT_OID_WE_VERSION_COMPILED;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.1.22 GET station WPS query status

Sample code =>

```
int data;

strcpy(wrq.ifr_ifrn.ifrn_name, "ra0");

wrq.u.data.length = sizeof(int);

wrq.u.data.pointer = &data;

wrq.u.data.flags = RT_OID_WSC_QUERY_STATUS;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

8.2 How to display rate, BW:

```
HTTRANSMIT_SETTING HTSetting;

Double Rate;

double b_mode[] = {1, 2, 5.5, 11};

float g_Rate[] = { 6,9,12,18,24,36,48,54};

switch(HTSetting.field.MODE)

{

    case 0:
```



```
if (HTSetting.field.MCS >=0 && HTSetting.field.MCS<=3)

    Rate = b_mode[HTSetting.field.MCS];

else if (HTSetting.field.MCS >=8 && HTSetting.field.MCS<=11)

    Rate = b_mode[HTSetting.field.MCS-8];

else

    Rate = 0;

break;

case 1:

    if ((HTSetting.field.MCS >= 0) && (HTSetting.field.MCS < 8))

        Rate = g_Rate[HTSetting.field.MCS];

    else

        Rate = 0;

    break;

case 2:

case 3:

    if (0 == bGetHTTxRateByBW_GI_MCS(HTSetting.field.BW,

        HTSetting.field.ShortGI,

        HTSetting.field.MCS,

        &Rate))

        Rate = 0;

        break;

default:

    Rate = 0;

    break;

}
```

```
char bGetHTTxRateByBW_GI_MCS(int nBW, int nGI, int nMCS, double* dRate)
```

```
{
```

```
double HTTxRate20_800[16]={6.5, 13.0, 19.5, 26.0, 39.0, 52.0, 58.5, 65.0, 13.0, 26.0, 39.0, 52.0, 78.0, 104.0, 117.0, 130.0};
```

```
double HTTxRate20_400[16]={7.2, 14.4, 21.7, 28.9, 43.3, 57.8, 65.0, 72.2, 14.444, 28.889, 43.333, 57.778,
86.667, 115.556, 130.000, 144.444};
```

```
double HTTxRate40_800[18]={13.5, 27.0, 40.5, 54.0, 81.0, 108.0, 121.5, 135.0, 27.0, 54.0, 81.0, 108.0,
162.0, 216.0, 243.0, 270.0, 6.0, 39.0};
```

```
double HTTxRate40_400[18]={15.0, 30.0, 45.0, 60.0, 90.0, 120.0, 135.0, 150.0, 30.0, 60.0, 90.0, 120.0,
180.0, 240.0, 270.0, 300.0, 6.7, 43.3};
```

```
// no TxRate for (BW = 20, GI = 400, MCS = 32) & (BW = 20, GI = 400, MCS = 32)
```

```
if (((nBW == BW_20) && (nGI == GI_400) && (nMCS == 32)) ||
```

```
((nBW == BW_20) && (nGI == GI_800) && (nMCS == 32)))
```

```
return 0; //false
```

```
if( nBW == BW_20 && nGI == GI_800)
```

```
    *dRate = HTTxRate20_800[nMCS];
```

```
else if( nBW == BW_20 && nGI == GI_400)
```

```
    *dRate = HTTxRate20_400[nMCS];
```

```
else if( nBW == BW_40 && nGI == GI_800)
```

```
    *dRate = HTTxRate40_800[nMCS];
```

```
else if( nBW == BW_40 && nGI == GI_400)
```

```
    *dRate = HTTxRate40_400[nMCS];
```

```
else
```

```
    return 0; //false
```

```
return 1; //true
```

```
}
```

8.3 Set Data for N mode

8.3.1 SET HT mode:

Sample code =>

```
typedef struct {
```

```
    RT_802_11_PHY_MODE    PhyMode;
```

```
    UCHAR                 TransmitNo;
```



```
    UCHAR      HtMode;          //HTMODE_GF or HTMODE_MM

    UCHAR      ExtOffset;       //extension channel above or below

    UCHAR      MCS;

    UCHAR      BW;

    UCHAR      STBC;

    UCHAR      SHORTGI;

    UCHAR      rsv;

} OID_SET_HT_PHYMODE ;
```

```
RT_802_11_PHY_MODE tmp_ht_mode;

sprintf(wrq.ifr_name, "ra0");

wrq.u.data.pointer = (caddr_t) & tmp_ht_mode;

wrq.u.data.length = sizeof(RT_802_11_PHY_MODE);

wrq.u.data.flags = RT_OID_802_11_SET_HT_PHYMODE | OID_GET_SET_TOGGLE;

ioctl(socket_id, RT_PRIV_IOCTL, &wrq);
```

Ralink Website document

for stevej@cradlepoint.com
And Company Use Only

9 EFUSE BUFFER MODE HOWTO

9.1 In RT2860STA.dat

9.1.1 EfuseBufferMode=Value

Description: Enable Efuse Buffer Mode

Value Type: Decimal

1:Enable

0:Disable

Default Value:0

*Please enable the ATE compile flag before using it.

*The priority of efuse-buffer mode is higher than original efuse.

9.2 By iwpriv command

9.2.1 efuseFreeNumber=Value

Description: Get the Free Block number of efuse

Value Type: Decimal number

Valid Range: any Default Value

(To Do)

0: Display the Free number in Decimal number format

1: Display the Free number in Hexdecimal number format

Ex.iwpriv ra0 set efuseFreeNumber=0

9.2.2 efuseDump=Value

Description: Dump the efuse

Value Type: Hexdecimal number

Valid Range: any Default

Value(To Do)

0: Display in Decimal number format

1: Display in Hexdecimal number format



Ex.iwpriv ra0 set efuseDump=0

9.2.3 efuseBufferModeWriteBack=Value

Description: In buffer mode, Write back all data to the original .bin file

Value Type: Hexdecimal number

Valid Range: 1

1: Write

Ex.iwpriv ra0 set efuseBufferModeWriteBack=1

9.2.4 efuseLoadFromBin

Description: Load data into efuse from a specified file

Value Type: Characters Absolute path

Ex. lwpriv ra0 set efuseLoadFromBin=path/filename

Ralink Website document

**for stevej@cradlepoint.com
And Company Use Only**