

RALINK TECHNOLOGY, CORP.

RT5572/RT3572 LINUX STATION RELEASE NOTES AND USER'S GUIDE

USB WIRELESS CARD

Copyright © 2010 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.

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 Street,
ChuPei City
HsinChu Hsien 302, Taiwan, ROC
Tel +886-3-560-0868
Fax +886-3-560-0818

Sales Taiwan: Sales@ralinktech.com.tw

Technical Support Taiwan: FAE@ralinktech.com.tw

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

Contents

Contents	3	3.2.35	HT_AMSDU	20
1 Version history	7	3.2.36	HT_BAWinSize	20
2 Readme	11	3.2.37	HT_GI	20
2.1 Build Instructions	12	3.2.38	HT_MCS	20
3 Configuration	13	3.2.39	HT_MIMOPSEnable	20
3.1 Configuration File		3.2.40	HT_MIMOPSMODE	20
RT2870STA.dat	13	3.2.41	HT_DisallowTKIP	20
3.2 Configuration file use	14	3.2.42	HT_RxStream	21
3.2.1 CountryRegion	14	3.2.43	HT_TxStream	21
3.2.2 CountryRegionForABand	15	3.2.44	HT_LinkAdapt	21
3.2.3 SSID	15	3.2.45	HT_HTC	21
3.2.4 WirelessMode	15	3.2.46	HT_DisableReordering	21
3.2.5 Channel	16	3.2.47	BeaconLostTime	21
3.2.6 HwAntDiv	16	3.2.48	AutoRoaming	21
3.2.7 BGProtection	16	3.2.49	MacAddress	22
3.2.8 TxPreamble	16	3.2.50	TDLSCapable	22
3.2.9 RTSThreshold	16	3.2.51	AutoConnect	22
3.2.10 FragThreshold	16	3.2.52	HT_40MHZ_INTOLERANT	22
3.2.11 TxBurst	16	3.2.53	AntGain	22
3.2.12 PktAggregate	16	3.2.54	BandedgeDelta	22
3.2.13 NetworkType	17	3.2.55	P2P_GOIntent	22
3.2.14 AuthMode	17	3.2.56	P2P_DevName	23
3.2.15 EncryptType	17	3.2.57	P2P_ListChannel	23
3.2.16 DefaultKeyID	17	3.2.58	P2P_OpChannel	23
3.2.17 WEP KeyType	17			
3.2.18 WEP Hex Key	17	3.3	More information	23
3.2.19 WEP Key String	18	4	Wireless Tools	24
3.2.20 WPAPSK	18	4.1	lwpri ra0 set use	24
3.2.21 WmmCapable	18	4.1.1	DriverVersion	24
3.2.22 IEEE80211H	18	4.1.2	CountryRegion	24
3.2.23 PSMODE	18	4.1.3	CountryRegionABand	24
3.2.24 FastRoaming	18	4.1.4	SSID	25
3.2.25 RoamThreshold	18	4.1.5	WirelessMode	25
3.2.26 TGNWifiTest	19	4.1.6	TxBurst:	25
3.2.27 WirelessEvent	19	4.1.7	PktAggregate:	25
3.2.28 CarrierDetect	19	4.1.8	TxPreamble:	26
3.2.29 HT_RDG	19	4.1.9	TxPower:	26
3.2.30 HT_EXTCHA	19	4.1.10	Channel	26
3.2.31 HT_OpMode	19	4.1.11	HwAntDiv	26
3.2.32 HT_MpduDensity	19	4.1.12	BGProtection:	26
3.2.33 HT_BW	19	4.1.13	RTSThreshold:	26
3.2.34 HT_AutoBA	20	4.1.14	FragThreshold:	27
		4.1.15	NetworkType:	27

4.1.16	AuthMode:	27	4.3.1	radio_off	34
4.1.17	EncrypType:	27	4.3.2	radio_on.....	34
4.1.18	DefaultKeyID:.....	27	4.4	Iwpriv Examples.....	34
4.1.19	Key1	27	4.4.1	Infrastructure.....	34
4.1.20	Key2	27	4.4.2	Ad-Hoc	35
4.1.21	Key3	28	4.4.3	Get site survey	36
4.1.22	Key4	28	4.4.4	Get Statistics	36
4.1.23	WPAPSK	28	4.4.5	ANY SSID	36
4.1.24	WmmCapable	28	4.5	iwlist	36
4.1.25	IEEE80211H.....	28	4.6	iwconfig	36
4.1.26	PSMode	28	5	WPS – Wi-Fi Protected Setup	38
4.1.27	ResetCounter.....	28	5.1	Iwpriv use	38
4.1.28	Debug	29	5.1.1	wsc_conf_mode.....	38
4.1.29	CarrierDetect	29	5.1.2	wsc_mode.....	38
4.1.30	HtRdg.....	29	5.1.3	wsc_pin	39
4.1.31	HtExtcha	29	5.1.4	wsc_ssid	39
4.1.32	HtOpMode.....	29	5.1.5	wsc_bssid	39
4.1.33	HtMpduDensity	29	5.1.6	wsc_start.....	39
4.1.34	HtBw	30	5.1.7	wsc_stop	39
4.1.35	HtAutoBa	30	5.1.8	wsc_gen_pincode	39
4.1.36	HtAmsdu	30	5.1.9	wsc_cred_count.....	39
4.1.37	HtBaWinSize	30	5.1.10	wsc_cred_ssid	39
4.1.38	HtGi.....	30	5.1.11	wsc_cred_auth.....	40
4.1.39	HtMcs	30	5.1.12	wsc_cred_encr.....	40
4.1.40	HtProtect	30	5.1.13	wsc_cred_keyidx.....	40
4.1.41	HtMimoPs.....	31	5.1.14	wsc_cred_key	40
4.1.42	FixedTxMode	31	5.1.15	wsc_cred_mac	41
4.1.43	LongRetry.....	31	5.1.16	wsc_conn_by_idx.....	41
4.1.44	ShortRetry.....	31	5.1.17	wsc_auto_conn	41
4.1.45	HtTxStream=value	31	5.1.18	wsc_ap_band	41
4.1.46	HtRxStream=value	31	5.1.19	Wsc4digitPinCode	41
4.1.47	HtDisallowTKIP=value....	32	5.2	WPS STA as an Enrollee or	
4.1.48	HtBaDecline	32	Registrar	41	
4.1.49	BeaconLostTime=value...	32	5.2.1	Enrollee Mode	42
4.1.50	AutoRoaming=value	32	5.2.2	Registrar Mode	42
4.1.51	SiteSurvey=value	32	5.3	WPS IOCTL use.....	43
4.1.52	TdlsCapable=value	32	5.3.1	iwpriv commands without	
4.1.53	TdlsSetup=value	33	argument	43	
4.1.54	AutoReconnect=value ...	33	5.3.2	iwpriv commands with one	
4.1.55	AdhocN=value	33	INT argument	44	
4.1.56	AntGain.....	33	5.3.3	iwpriv commands with	
4.1.57	ModuleTxpower	33	string argument.....	44	
4.2	Iwpriv ra0 show use	33	5.4	WPS IOCTL Sample Program....	45
4.2.1	connStatus.....	33			
4.2.2	driverVer.....	33			
4.2.3	bainfo.....	34			
4.2.4	rxbulk.....	34			
4.2.5	txbulk	34			
4.2.6	AutoReconnect	34			
4.2.7	WPAPSK	34			
4.2.8	PMK	34			
4.3	Iwpriv ra0 use	34			

6	Wifi direct - P2P command	47	7.1.20	ResetCounter	59
6.1	Iwpriv use	47	7.1.21	ATERRF	59
6.1.1	P2pOpMode	47	7.1.22	ATEWRF1.....	59
6.1.2	p2pLisCh	47	7.1.23	ATEWRF2.....	59
6.1.3	p2pOpCh	47	7.1.24	ATEWRF3.....	59
6.1.4	p2pGoInt	48	7.1.25	ATEWRF4.....	60
6.1.5	p2pDevName	48	7.1.26	efuseBufferModeWriteBac	
6.1.6	p2pWscMode	48	k	60	
6.1.7	p2pWscConf	48	7.1.27	efuseLoadFromBin	60
6.1.8	p2pScan	49	7.1.28	ATEAUTOALC.....	60
6.1.9	p2pInv	49	7.1.29	ATEPAYLOAD.....	60
6.1.10	p2pDevDisc	49	7.1.30	ATETSSICBA.....	60
6.1.11	p2pLink	49	7.2	Tx Mode, MCS, BW and GI	
6.1.12	p2pCfg	49		Selection Table	60
6.1.13	p2pTab	49	7.2.1	MODE = 0, Legacy CCK ...	60
6.1.14	p2pProv	50	7.2.2	MODE = 1, Legacy OFDM61	
6.1.15	p2pStat	50	7.2.3	MODE = 2, HT Mixed Mode	61
6.1.16	p2pReset	50	7.2.4	MODE = 3, HT Greenfield	61
6.1.17	p2pPerTab	50	7.3	Examples	62
6.1.18	p2pDefConfMthd.....	50	7.3.1	Check EVM & Power	62
6.1.19	p2pLinkDown	50	7.3.2	Check Carrier.....	62
6.1.20	p2pSigmaEnable	51	7.3.3	Check specturm mask	62
6.1.21	p2pAutoReconn	51	7.3.4	Frequency offset tuning.	63
6.1.22	p2pDevDiscPeriod	51	7.3.5	Rx	63
6.1.23	P2P example:	51	7.3.6	Show all ate parameters	63
7	ATE Test Command Format	55	7.3.7	Online help.....	64
7.1	iwpriv ra0 set [parameters]=[val]	56	7.3.8	Display Rx Packet Count	
7.1.1	ATE	56	and RSSI	64	
7.1.2	ATEDA	57	7.3.9	Internal ALC calibration	
7.1.3	ATESA	57	(For RT33xx, RT5370 serial chipset)	64	
7.1.4	ATEBSSID	57	7.3.10	Internal ALC function	
7.1.5	ATECHANNEL	57	testing in ATE mode (For RT33xx, RT5370		
7.1.6	ATETXPOW0	57	serial chipset)	64	
7.1.7	ATETXPOW1	57	7.4	iwpriv ra0 bbp	
7.1.8	ATETXFREQOFFSET	57		[parameters]=[Value]	65
7.1.9	ATETXLLEN	57	7.4.1	BBPID	65
7.1.10	ATETXCNT	58	7.4.2	BBPID=Value	65
7.1.11	ATETXMODE (Refer to		7.5	iwpriv ra0 mac [parameters]=[val]	
TxMode)	58		66		
7.1.12	ATETXBW (Refer to		7.5.1	MAC_OFFSET	66
TxMode)	58		7.5.2	MAC_OFFSET=Value	66
7.1.13	ATETXGI (Refer to TxMode)	58	7.6	iwpriv ra0 e2p [parameters]=[val]	
7.1.14	ATETXMCS (Refer to		66		
TxMode)	58				
7.1.15	ATETXANT	58			
7.1.16	ATERXANT	58			
7.1.17	ATERXFER	59			
7.1.18	ATESHOW	59			
7.1.19	ATEHELP	59			

7.6.1	EED_ADDR.....	66	9.1.1	GET station connection status:	96
7.6.2	EED_ADDR=Value.....	66	9.1.2	GET station statistics information:	96
7.7	Example.....	66	9.1.3	GET AP list table:.....	96
7.7.1	Hardware access.....	66	9.1.4	GET scan table:.....	96
7.7.2	Statistic counter operation	67	9.1.5	GET station's MAC:	96
7.7.3	Suggestion:	67	9.1.6	GET station connection status:	97
7.8	ated	67	9.1.7	GET AP's BSSID	97
7.8.1	Introduction.....	67	9.1.8	GET SSID	97
7.8.2	Environment setup	67	9.1.9	GET station's last TX related information:.....	97
7.8.3	How to use ated for ATE purpose	67	9.1.10	GET station's last RX related information:.....	97
8	IOCTL.....	69	9.1.11	GET station's wireless mode:	97
8.1	Parameters for iwconfig.....	69	9.1.12	GET Bss type:.....	98
8.2	Parameters for iwpriv	75	9.1.13	GET Authentication Mode:	98
8.2.1	Set Data, Parameters is Same as iwpriv	75	9.1.14	GET Encryption Type:.....	99
8.2.2	Get Data, Parameters is Same as iwpriv	76	9.1.15	GET RSSI 0 (unit: db)	99
8.2.3	Set Raw Data with Flags	77	9.1.16	GET RSSI 1 (unit: db)	99
8.2.4	Get Raw Data with Flags	85	9.1.17	GET RSSI 2 (unit: db)	99
8.2.5	Set Raw Data with Flags	94	9.1.18	GET Driver wireless extension version	99
9	IOCTL instructions	96	9.2	How to display rate, BW:.....	100
9.1	Get Data	96			

1 VERSION HISTORY

Version 2.6.1.1

1. Update cfg80211 support & wpa_supplicant 1.0 support
2. WFD code update
3. Fix WIDI_SUPPORT only compile error
4. Sync OID with WFD & Intel WIDI
5. Update DPA_S fix & Enhancement

Version 2.6.1.0

1. Add VID/PID for RT5370 & RT5572.
2. Fix the abnormal G3 Duplicated WPS & P2P IE will lead DUT crash.
3. Fix enrollee(p2p client) need to reply WSC ACK instead of WSC_NACK, when DUT receive the M2D msg.
4. Fix WiDi invitation bug.
5. Fix toggle scan sequence wrong problem.

Version 2.6.0.1

6. improve auto channel select alg is ApCnt and Rule 1 method.
7. Merge v2.6.0.0 patches.

Version 2.6.0.0

1. Add modifications of Samsung TV Project
2. P2P Related issues Fix :
3. WPS Related Fix :
4. 1). Registrars (both internal and external) shall add the AuthorizedMACs subelement,
5. this applies to all Configuration Methods, including PIN, PBC and NFC
6. 2). Add modifications for preventing complained by Coverity
7. 3). Fix WSC test plan 5.1.1 failed at step 10.
8. P2P Related Enhancement :
 - 1). Add Set_P2P_SecDevTypeList_Proc() to set 2nd device list
 - 2). Change time period of sending probe request before send Provision Discovery Request or P2P GO Nego Request from 1s to 100ms in P2P GO mode.
 - 3). Modify WscBuildAssocReqIE for P2P CLI to send ASSOC Req with WPS IE.
9. Enhance Data Rate checking mechanism
10. Fixed STA scanning issue (When scan table is full)
11. Add dynamic L2SD cmds
12. When the WLAN NIC is plug on/off, may lead DUT crash.
(Fix crash issue when set OSABL = YES. The data structure is unmatched between rt_linux.h and rt_drv.h.)
13. Fix The Infra STA's WPS process may not succeed, when the scan table is full.

Version 2.5.0.3

1. Support P2P Sigma test
2. Support P2P CLI AutoReconnect
3. Add RSSI information to P2P Table
4. Fixed P2P issue: P2P peer is not removed from P2P Table if P2P peer disappears
5. Fixed P2P issue: Wrong SSID in P2P Invite Request frame when P2P CLI sent Invite Request to P2P GO. (modify InsertP2PSubelmtTlv)

6. Fixed P2P issue: Driver may retrieve wrong data from P2P sub element data. (modify P2pParseSubElmt)
7. Fixed P2P issue: System may crash while doing P2P GO Formation. (modify P2pReceGoNegoConfirmAction)
8. Fixed P2P issue: System may crash while doing P2P Invite. (modify P2pReceInviteRspAction)
9. Fixed P2P issue: Put wrong value of Listen Channel in Go Nego Req frame if Operation Channel and Listen Channel is different
10. Fixed P2P issue: P2P persistent profile update failed if re-form P2P session with same P2P peer again. (modify P2pPerstTabInsert)
11. Fixed P2P issue: Driver may retrieve wrong persistent profile after receives P2P Invite Response from peer.
12. Fixed P2P issue: GO Nego Confirm may be sent out in operation channel. (modify P2PSendGoNegoConfirm)
13. Fixed P2P issue: Driver cannot retrieve P2P information for Probe Request of P2P peer. (modify PeerP2pProbeReqSanity)
14. Fixed P2P issue: P2P CLI cannot send WSC_NACK to P2P peer.
15. Fixed STA issue: Memory leakage when WIDI and P2P are enabled.
16. Fixed STA issue: Memory leakage while scanning if WSC is enabled. Support PPAD feature.

Version 2.5.0.0

1. Support WIFI Direct(WIFI P2P).
2. Support Linux Kernel 2.6.35.
3. Fixed Ad-hoc issue: STA didn't use legacy rate when STA is Ad-hoc OPEN-WEP or WPAPSK-TKIP.
4. Fixed WPS issue: Beacon lost when STA tries to do WPS with WPS AP or STA connects to hidden AP.
5. Fixed WPS issue: WPS procedure is taking much longer than before.
6. Corrected WPS behavior: According to WFA: In PBC, STA shall stop if more than one AP available.
7. Fixed issue: WPS failed with Ralink WpaSupplicant when STA's privacy is different from AP's privacy. (for example: AP is WPA2PSK but STA is OPEN)
8. Fixed Ralink WpaSupplicant issue - Scan table didn't include RSSI.
9. Fixed issue: STA had problem to show 32-bytes SSID when user uses "iwpriv ra0 show SSID" or "iwpriv ra0 stat" to check SSID of WSC profile.
10. Fixed connection issue: If there are two APs have same SSID, AuthMode and EncrypType
11. Refine the stack size of all functions to be smaller than 1024B.
12. Fixed WPS LED issue: When overlapping, WPS LED should turn off for LED mode 9 after timeout.
13. Fixed issue: Use wrong rate index in send_monitor_packets().
14. Improve scan mechanism.
15. Fixed issue: STA supports the WPA authentication which group cipher is WEP
16. Support to generate random WSC 4-digits PIN.
17. Fixed issue: Single SKU Bandedge Delta value decrease all channels transmit power on FCC, it should work on Channel 1 / 11.
18. Fix P2P CLIENT use PBC to connect P2P GO as negotiate by KEYPAD / DISPLAY
19. Fix P2P CLIENT difficult connect on P2P GO need scan times with wrong BSSID
20. Fix P2P CLIENT not use N-rate transmit data to GO
21. Fix P2P concurrent mode as INFRA + P2P_CLIENT with different HT_BW
22. Fix P2P Link Down not turn back to Listen Channel cause can't be found

Version 2.4.0.0

1. Rename crda.c to cfg80211.c, CRDA_SUPPORT to RT_CFG80211_SUPPORT.
2. Fix issue in 64bit CPU:
3. Fix issue: error link list usage in 11r ric module in station mode.
4. Add an ATE command for AUTOALC.
5. Add an ATE command for interpacket GAP.

6. Add an ATE command for ATE payload pattern.
7. Fix WPS issue: some PIN cannot set successfully.(ex. 89134498)
8. Fix issue: STA send null frame with QoS field when STA connected to non-WMM AP.
9. Support RFKILL: DO NOT set HAS_RFKILL_HW_SUPPORT=y in config.mk
10. Reduce data packet lost while doing scan if STA already connects to AP.
11. Fix issue: If there are two APs with same SSID and security mode but have different WPAPSK mode.
12. Support TxRx ring PRE-ALLOC in probe stage: set RESOURCE_PRE_ALLOC=y in config.mk.
13. Shorten the connection time: set FastConnect=1 in DAT file.
14. Fix Adhoc issue.
15. Fix issue: Correct RSSI, SNR, Noise calculation
16. Fix issue: STA cannot connect to OPEN-WEP AP if WpaSupplicant set

Version 2.3.0.0

1. Support PSP XLINK in ad-hoc mode.
2. Add mac80211 iw utility other commands support.
3. Fix issue: The start address of HeaderBuf must be aligned by 4 when VENDOR_FEATURE1_SUPPORT is enabled.
4. WMM ACM: see history of acm_comm.c.
5. Support WpaSupplicant(v0.6.9) WPS
6. Fix WPS issue: Check SelectRegistrar is TRUE or FALSE
7. Fix WPS issue: Some AP (ex. Buffalo WZR-AG300NH) would change SSID to another SSID after push WPS PBC button.
After WPS process finish, change SSID to original SSID.
Driver needs to wait AP to re-generate Beacon; otherwise, driver will update this PBC SSID to MlmeAux.
8. Shorten the waiting time when unplug the device.
9. Fix mgmt ring full issue: It happened only in big-endian platform and quickly switch the network type between Infra and Adhoc mode.
10. Fix statistics issue: It happened in big-endian platform, tx count would be zero.
11. Support WiFi Draft_3.0 11n test items.
12. Support Linux Kernel 2.6.31
13. Support Ad-hoc WPS2PSK/AES (DPO doesn't support this item)
14. Support Samsung Auto Provision (only for Samsung Project)
15. For GPL issue: Separate three kernel modules - util module, driver module, netif module. util module and netif module could license GPL.
Note: Driver module couldn't license GPL.

Version 2.2.0.0

1. Fixed issue of showing wrong statistics value.
2. Fixed problem of showing wrong SNR in wireless extension scan list.
3. Fixed WPA2PSK failed with some AP if key length is not zero in pairwise_msg_2 and pairwise_msg_4.
4. Add HAS_STATS_COUNT compile option.
5. Support TDLS.
6. Support Client WDS.
7. Support WMM ACM (Pass WiFi Test Plan v0.30).
8. Support the MAC assignment from configuration file(.dat).
9. Support Ethernet Convert in Ad-hoc mode.

Version 2.1.2.0

1. Short GI sampling improvement.
2. Support Linux Kernel 2.6.29
3. Fixed zero IWEVASSOCREQIE to Native WpaSupplicant.
4. Fixed non-zero BSSID to Native WpaSupplicant when STA disconnecting to AP.
5. Fixed Enqueue command to command thread would fail when HAS_KTHREAD_SUPPORT=n in config.mk.

Version 2.1.0.0

1. New generation schema for multiple OS porting.
2. Fixed Ad-hoc ping failed in noisy environment. (Probe Response has too many retry packet then cause "not enough space in MgmtRing").
3. Fixed WPS failed with D-Link DIR-628 in 5GHz.
4. Change FastRoaming in DAT file to AutoRoaming.
5. Support kthread.
6. Add New A band channel list region 15 contains the whole channels in the A band region 4 and the new CE channel 167,169,171,173.
7. New IEEE802.11r functionality.

2 README

Model name: RT537x/RT3x7x Wireless LAN Linux Driver

RT537x serial driver name:

Kernel 2.4.x: rtutil5370sta.o; rt5370sta.o; rtnet5370sta.o

Kernel 2.6.x: rtutil5370sta.ko; rt5370sta.ko; rtnet5370sta.ko

Driver name:

RT3x7x serial driver name:

Kernel 2.4.x: rtutil3x7xsta.o; rt3x7xsta.o; rtnet3x7xsta.o

Kernel 2.6.x: rtutil3x7xsta.ko; rt3x7xsta.ko; rtnet3x7xsta.ko

Supporting Kernel:

Linux kernel 2.4 and 2.6 series

Tested in Redhat 7.3 or later

Description:

This is a Linux device driver for the Ralink RT5370 BGN WLAN Card.

Contents:

Makefile: Makefile

*.c: c files

*.h: header files

This driver implements basic IEEE802.11 functions

Features:

Infrastructure and Ad-Hoc mode with open or shared or WPA-PSK or WPA2-PSK authentication method.

NONE, WEP, TKIP and AES encryption.

2.1 Build Instructions

- 1> \$tar -xvzf yyyy_mmdd_RT5370_Linux_STA_x.x.x.x.tgz
go to "./yyyy_mmdd_RT5370_Linux_STA_x.x.x.x" directory.
- 2> In Makefile
 - define the linux kernel source include file path LINUX_SRC modify to meet your need.
- 3> In os/linux/config.mk
 - define the GCC and LD of the target machine.
 - define the compiler flags CFLAGS.
 - modify to meet your need.
 - ** Build for being controlled by NetworkManager**
Please set 'HAS_WPA_SUPPLICANT=y' and 'HAS_NATIVE_WPA_SUPPLICANT_SUPPORT=y'.
 - ** Build for being controlled by WpaSupplicant with Ralink Driver**
Please set 'HAS_WPA_SUPPLICANT=y' and 'HAS_NATIVE_WPA_SUPPLICANT_SUPPORT=n'.
- 4> compile driver source code
\$make
- 5> \$cp RT2870STA.dat /etc/Wireless/RT2870STA/RT2870STA.dat
- 6> load driver, execute script "load.6"
./load.6
Or
#insmod UTIL/os/linux/rtutil5370sta.ko
#insmod MODULE/os/linux/rt5370sta.ko
#insmod NETIF/os/linux/rtnet5370sta.ko
- 7> unload driver, execute script "unload"
./unload
Or
#rmmod rtnet5370sta
#rmmod rt5370sta
#rmmod rtutil5370sta

3 CONFIGURATION

RT5370 driver can be configured via following interfaces, i.e.

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

Note:

- 1) modify configuration file "RT2870STA.dat" in /etc/Wireless/RT2870STA/RT2870STA.dat.
- 2) iwconfig/iwpriv comes with kernel.
- 3) iwpriv use, please refer to below sections for details.

3.1 Configuration File RT2870STA.dat

```
# Copy this file to /etc/Wireless/RT2870STA/RT2870STA.dat
# This file will be read on loading driver module.
#
# Use "vi RT2870STA.dat" to modify settings according to your need.
#
#
#The word of "Default" must not be removed
Default
CountryRegion=5
CountryRegionABand=7
CountryCode=
ChannelGeography=1
SSID=11n-AP
NetworkType=Infra
WirelessMode=5
Channel=0
BeaconPeriod=100
TxPower=100
BGProtection=0
TxPreamble=0
RTSThreshold=2347
FragThreshold=2346
TxBurst=1
PktAggregate=0
WmmCapable=1
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
APSDCapable=0
APSDAC=0;0;0;0
HT_RDG=1
HT_EXTCHA=0
HT_OpMode=1
HT_MpduDensity=4
HT_BW=1
HT_BADecline=0
HT_AutoBA=1
HT_BADecline=0
HT_AMSDU=0
HT_BAWinSize=64
HT_GI=1
HT_MCS=33
HT_MIMOPSMODE=3
HT_DisallowTKIP=1
IEEE80211H=0
TGnWifiTest=0
WirelessEvent=0
CarrierDetect=0
AntDiversity=0
BeaconLostTime=4
FtSupport=1

```

NOTE:

WMM parameters

WmmCapable

AckPolicy1~4

Set it as 1 to turn on WMM Qos support

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 RT2870STA.dat, and restart driver.

3.2 Configuration file use

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

SectionNumber

Param

Value

...

...

...

3.2.1 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
31	1-14
32	1-11 active scan, 12 and 13 passive scan
33	1-14 all active scan, 14 b mode only

3.2.2 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

3.2.3 SSID

Value:

0~z, 1~32 ascii characters.

3.2.4 WirelessMode

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
- 11: 11N only in 5G band only

3.2.5 Channel

Value:

Depends on CountryRegion or CountryRegionForABand

3.2.6 HwAntDiv

Value:

- 0: Disable
- 1: HW RX antenna diversity
- 2: Fixed RX at AUX ANT
- 3: Fixed RX at main ANT

3.2.7 BGProtection

Value:

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

3.2.8 TxPreamble

Value:

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

3.2.9 RTSThreshold

Value:

1~2347

3.2.10 FragThreshold

Value:

256~2346

3.2.11 TxBurst

Value:

- 0: Disable
- 1: Enable

3.2.12 PktAggregate

Value:

0: Disable
1: Enable

3.2.13 NetworkType

Value:

Infra: infrastructure mode
Adhoc: adhoc mode

3.2.14 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

3.2.15 EncryptType

Value:

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

3.2.16 DefaultKeyID

Value:

1~4

3.2.17 WEP KeyType

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

Value:

0 hexadecimal type
1 assic type
(use: reading profile only)

3.2.18 WEP Hex Key

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

Value:

10 or 26 hexadecimal characters eg: 012345678
5 or 13 ascii characters eg: passwd
(use: "iwpriv" only)

3.2.19 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)
(use: reading profile only)

3.2.20 WPAPSK

Value:

8~63 ASCII or
64 HEX characters

3.2.21 WmmCapable

Value:

0: Disable WMM
1: Enable WMM

3.2.22 IEEE80211H

Enabel IEEE802.11h support

Value:

0:Disable
1:Enable

3.2.23 PSMode

Value:

CAM	Constantly Awake Mode
Max_PSP	Max Power Saving
Fast_PSP	Fast Power Saving
Legacy_PSP	Legacy Power Saving

3.2.24 FastRoaming

Value:

0: Disabled
1: Enabled

3.2.25 RoamThreshold

Value:
0 ~ 255

3.2.26 TGNWifiTest

Value:
0: Disabled
1: Enabled

3.2.27 WirelessEvent

Value:
0: Disabled
1: Enabled (send custom wireless event)

3.2.28 CarrierDetect

Value:
0: Disabled
1: Enabled

3.2.29 HT_RDG

Value:
0: Disabled
1: Enabled

3.2.30 HT_EXTCHA

Value:
0: Below
1: Above

3.2.31 HT_OpMode

Value:
0: HT mixed format
1: HT greenfield format
(Note) If you want to do TGN WIFI green field item, please set HT_OpMode=1

3.2.32 HT_MpduDensity

Value:
0 ~ 7

3.2.33 HT_BW

Value:
0: 20MHz
1: 40MHz

3.2.34 HT_AutoBA

Value:

- 0: Disabled
- 1: Enabled

3.2.35 HT_AMSDU

Value:

- 0: Disabled
- 1: Enabled

3.2.36 HT_BAWinSize

Value:

- 1 ~ 64

3.2.37 HT_GI

Value:

- 0: long GI
- 1: short GI

3.2.38 HT_MCS

Value:

- 0 ~ 15
- 33: auto

3.2.39 HT_MIMOPSEnable

Enable/Disable the 802.11n SM power save function.

Value:

- 0: Disable
- 1: Enable (Default)

3.2.40 HT_MIMOPSMode

Value:

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

3.2.41 HT_DisallowTKIP

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

Value:

- 0: FALSE
- 1: TRUE

Default setting is disable.

3.2.42 HT_RxStream

Set the number of spatial streams for reception

Value:

- 1: 1 Rx stream
- 2: 2 Rx stream

3.2.43 HT_TxStream

Set the number of spatial streams for transimtion

Value:

- 1: 1 Tx stream
- 2: 2 Tx stream

3.2.44 HT_LinkAdapt

Enable/Disable HT Link Adaptation Control

Value:

- 0:Disable (Default)
- 1:Enable

3.2.45 HT_HTC

Enable/disable HTC field of data frames send with 802.11n data rates

Value:

- 0:Disable (Default)
- 1:Enable

3.2.46 HT_DisableReordering

Disable AMPDU re-ordering handling mechanism

Value:

- 0:Disable (Default)
- 1:Enable

3.2.47 BeaconLostTime

Change Beacon Lost Time

Value:

- 1 ~ 60 seconds
- Default value is 4 seconds

3.2.48 AutoRoaming

Enable/disable auto roaming mechanism

Value:

- 0: disable
- 1: enable
- Default setting is disable.

3.2.49 MacAddress

MacAddress=value

Value:

XX:XX:XX:XX:XX:XX

3.2.50 TDLSCapable

Enable/disable TDLS Capable function

Value:

0: disable

1: enable

3.2.51 AutoConnect

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

Value:

0: disable (default)

1: enable

3.2.52 HT_40MHZ_INTOLERANT

Set to disable the 40MHz channel bandwidth operation and also indicate other 20/40MHz BSS Coex aware

Value:

0:Disable (default)

1:Enable

3.2.53 AntGain

Define peak antenna gain (dBi) for Single SKU setting.

Value:

0: Disable Single SKU TxPower Adjustment.

1~255: Enable Single SKU TxPower Adjustment.

3.2.54 BandedgeDelta

Define delta conducted power value which can pass bandedge of FCC certification at Ch1 and Ch11 (dBm) within HT_40 Bandwidth for Single SKU setting.

Value:

1~255: Delta value between HT_20 and HT_40 power value.

3.2.55 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.

3.2.56 P2P_DevName

Define P2P device name for display

Value:

0~Z, less than 32 characters.

3.2.57 P2P_ListChannel

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

Value:

1, 6, 11

3.2.58 P2P_OpChannel

Set P2P device GO operate channel.

Value:

Based on country region

3.3 More information

The subsequent instructions are for automatically loading the rt5370 driver at boot time

- A) choose ra0 for first RT5370 WLAN card, ra1 for second RT5370 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 rt5370sta
- C) edit(create) the file /etc/sysconfig/network-scripts/ifcfg-ra0
DEVICE='ra0'
ONBOOT='yes'

Note: if you use dhcp, add "BOOTPROTO='dhcp'"

- D) To ease the Default Gateway setting, add the line
GATEWAY=x.x.x.x
in /etc/sysconfig/network

4 WIRELESS TOOLS

4.1 Iwpriv ra0 set use

This section describes parameters set using iwpriv. Please refer to the Readme section for more general data.

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

Note: Execute one iwpriv/set command at a time.

4.1.1 DriverVersion

Check driver version by issue iwpriv set command.

Range:

Any value

Value:

0

Example:

`#iwpriv ra0 set DriverVersion=1`

4.1.2 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
31	1-14
32	1-11 active scan, 12 and 13 passive scan
33	1-14 all active scan, 14 b mode only

4.1.3 CountryRegionABand

Set country region for A band.

Range:

{0~9}

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

4.1.4 SSID

Set AP SSID

Range:

{0~z, 1~32 ascii characters}

Value:

4.1.5 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
- 11: 11N only in 5G band only

4.1.6 TxBurst:

Set TxBurst Enable or Disable

Range:

{0,1}

Value:

- 0:Disable,
- 1:Enable

4.1.7 PktAggregate:

Set Tx Aggregate Enable or Disable

Range:

{0,1}

Value:

0:Disable,

1:Enable

4.1.8 TxPreamble:

Set TxPreamble

Range:

{0~2}

Value:

0:Preamble Long,

1:Preamble Short,

2:Auto

4.1.9 TxPower:

Set Tx power in percentage

Range:

{0~100}

Value:

4.1.10 Channel

Set Channel, depends on CountryRegion or CountryRegionABand

4.1.11 HwAntDiv

Setting H/W Antenna Diversity Diabale or Enable

Value:

0: Disable

1: HW RX antenna diversity

2: Fixed RX at AUX ANT

3: Fixed RX at main ANT

4.1.12 BGProtection:

Set 11B/11G Protection

Range:

{0~2}

Value:

0:Auto,

1:Always on,

2:Always off

4.1.13 RTSThreshold:

Set RTS Threshold

Range:

{1~2347}



4.1.14 FragThreshold:

Range:

Value:

Set Network type

Range:

{Infra,Adhoc}

Value:

Set Authentication Mode

Range:

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

Value:

Set Encryption Type

Range:

{NONE, WEP, TKIP, AES}

Value:

Set Default Key ID

Range:

 $\{1 \sim 4\}$

Value:

Set Key1 String

Range:

{5 ascii characters or 10 hex number or
13 ascii characters or 26 hex numbers}

Value:

Set Key2 String

Range:

{5 ascii characters or 10 hex number or
13 ascii characters or 26 hex numbers}

Value:

4.1.21 Key3

Set Key3 String

Range:

{5 ascii characters or 10 hex number or
13 ascii characters or 26 hex numbers}

Value:

4.1.22 Key4

Set Key4 String

Range:

{5 ascii characters or 10 hex number or
13 ascii characters or 26 hex numbers}

Value:

4.1.23 WPAPSK

WPA Pre-Shared Key

Range:

{8~63 ascii or 64 hex characters}

Value:

4.1.24 WmmCapable

Set WMM Capable

Range:

{0,1}

Value:

0:Disable WMM,
1:Enable WMM

4.1.25 IEEE80211H

Enabel IEEE802.11h support

Range:

{0,1}

Value:

0:Disable
1:Enable

4.1.26 PSMode

Set Power Saving Mode

Range:

{CAM, MAX_PSP, FAST_PSP}

Value:

4.1.27 ResetCounter

Reset statistics counter

Range:

Any vlaue

Value:

0

4.1.28 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

4.1.29 CarrierDetect

Value

- 0: Disabled
- 1: Enabled

4.1.30 HtRdg

Enable HT Reverse Direction Grant.

Value:

- 0: Disabled
- 1: Enabled

4.1.31 HtExtcha

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

Value:

- 0: Below
- 1: Above

4.1.32 HtOpMode

Change HT operation mode.

Value:

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

4.1.33 HtMpduDensity

Minimum separation of MPDUs in an A-MPDU.

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

4.1.34 HtBw

Support channel width.

Value:

- 0: 20MHz
- 1: 40MHz

4.1.35 HtAutoBa

Enable auto block acknowledgment (Block Ack).

Value:

- 0: Disabled
- 1: Enabled

4.1.36 HtAmsdu

Enable aggregation of multiple MSDUs in one MPDU.

Value:

- 0: Disabled
- 1: Enabled

4.1.37 HtBaWinSize

Set BA WinSize.

Value:

- 1 ~ 64

4.1.38 HtGi

Support Short/Long GI.

Value:

- 0: long GI
- 1: short GI

4.1.39 HtMcs

MCS rate selection.

Value:

- 0 ~ 15
- 33: auto

4.1.40 HtProtect

Enable HT protection for legacy device.

Value:

0: Disable

1: Enable

4.1.41 HtMimoPs

MIMO power save.

Value:

0: Disable

1: Enable

4.1.42 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

4.1.43 LongRetry

USE:

iwpriv ra0 set LongRetry=value

Value:

0~255

4.1.44 ShortRetry

USE:

iwpriv ra0 set ShortRetry=value

Value:

0~255

4.1.45 HtTxStream=value

Value:

1: Support 1-Tx Stream for MCS0 ~ MCS7

2: Support 2-Tx Stream for MCS0 ~ MCS15

4.1.46 HtRxStream=value

Value:

- 1: Support 1-Rx Stream for MCS0 ~ MCS7
- 2: Support 2-Rx Stream for MCS0 ~ MCS15

4.1.47 HtDisallowTKIP=value

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

Value:

- 0: FALSE
 - 1: TRUE
- Default setting is disable.

4.1.48 HtBaDecline

Reject all Recipient's BA requests.

Value:

- 0: Disable (Default)
- 1: Enable

4.1.49 BeaconLostTime=value

Change Beacon Lost Time

Value:

- 1 ~ 60 seconds
- Default value is 4 seconds

4.1.50 AutoRoaming=value

Enable/disable auto roaming mechanism

Value:

- 0: disable
 - 1: enable
- Default setting is disable.

4.1.51 SiteSurvey=value

Scan with specific SSID after link up

Value:

- 0~z, 1~32 ascii characters

4.1.52 TdlsCapable=value

Enable/disable TDLS capable

Value:

- 0: disable
 - 1: enable
- Example: iwpriv ra0 set TdlsCapable=0

4.1.53 TdlsSetup=value

Manually add TDLS link

Value: MAC address

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

4.1.54 AutoReconnect=value

Description: Enable/Disable driver auto reconnect functionality

Valid Range: 0-1

Default Value: 1

0: Disable, 1: Enable

4.1.55 AdhocN=value

Description: Enable/Disable Adhoc to support N or not

Valid Range: 0-1

Default Value: 1

0: Disable, 1: Enable

4.1.56 AntGain

Define peak antenna gain (dBi) for Single SKU setting.

Value:

0: Disable Single SKU TxPower Adjustment.

1~255: Enable Single SKU TxPower Adjustment.

4.1.57 ModuleTxpower

Define max module Tx power.

Value

decimal value

ModuleTxpower will override DefineMaxTxPwr (value from EEPROM) if ModuleTxpower is not zero.

4.2 iwpriv ra0 show use

This section describes parameters set using iwpriv. Please refer to the Readme section for more general data.

A detailed explanation of each parameter for iwpriv is shown subsequently. Refer to the Readme before using this section.

`iwpriv ra0 show [parameters]`

4.2.1 connStatus

Show STA connection Status

4.2.2 driverVer

Show STA current driver version

4.2.3 bainfo

Show STA current BA information

4.2.4 rxbulk

Show STA current rxbluk information

4.2.5 txbulk

Show STA current txbluk information

4.2.6 AutoReconnect

Show bAutoReconnect flag

4.2.7 WPAPSK

Show WPA Passphrase

4.2.8 PMK

Show PMK key

4.3 Iwpriv ra0 use

This section describes parameters set using iwpriv. Please refer to the Readme section for more general data.

`iwpriv ra0 show [parameters]`

4.3.1 radio_off

Turn STA radio off

4.3.2 radio_on

Turn STA radio on

4.4 Iwpriv Examples

4.4.1 Infrastructure

4.4.1.1 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"

4.4.1.2 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"

4.4.1.3 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-preskared key"
6. iwpriv ra0 set SSID="AP's SSID"

4.4.1.4 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-preskared key"
6. iwpriv ra0 set SSID="AP's SSID"

4.4.1.5 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"

4.4.2 Ad-Hoc

4.4.2.1 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"

4.4.2.2 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"

4.4.3 Get site survey

use:

```
iwpriv ra0 get_site_survey
```

4.4.4 Get Statistics

use:

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

4.4.5 ANY SSID

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

use:

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

4.5 iwlist

This section describes parameters set using iwlist. Please refer to the Readme section for more general data.

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

4.6 iwconfig

The subsequent settings are used in the standard iwconfig configuration

- | | |
|---|------------------------------|
| 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: Refer to the 'iwconfig', 'iwlist' and 'iwpriv' sections for wireless extension instructions.

FOR 29500201@
MediaTek Confidential
CONFIDENTIAL
USE ONLY
© 2011 MediaTek Corporation. All rights reserved.

5 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.

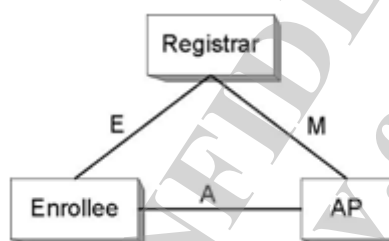


Figure 1. Components and Interfaces

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).

5.1 Iwpriv use

This section describes parameters set using iwpriv. Please refer to the Readme section for more general data.

`iwpriv ra0 [commands]=[Value]`

Note: Wireless extension private handlers.

5.1.1 wsc_conf_mode

Set WPS conf mode.

Range:

{0, 1, 2}

Value:

0: WPS Disabled

1: Enrollee

2: Registrar

5.1.2 wsc_mode

Set WPS mode, PIN or PBC.

Range:

{1, 2}

Value:

1: PIN

2: PBC

5.1.3 wsc_pin

Set Enrollee's PIN Code.

Range:

{00000000 ~ 99999999}

Value:

5.1.4 wsc_ssid

Set WPS AP SSID.

Range:

{0~z, 1~32 ascii characters}

Value:

5.1.5 wsc_bssid

BSSID of WSC AP that STA wants to do WPS with

Value:

xx:xx:xx:xx:xx:xx

5.1.6 wsc_start

Trigger RT5370 STA driver to do WPS process.

Range:

NULL

Value:

5.1.7 wsc_stop

Stop WPS process and don't wait upon two-minute timeout.

Range:

NULL

Value:

5.1.8 wsc_gen_pincode

Generate new PIN code.

Range:

NULL

Value:

5.1.9 wsc_cred_count

Set count of WPS credential, only support one credential for M8 in Registrar mode.

Range:

{1 ~ 8}

Value:

5.1.10 wsc_cred_ssid

Set SSID into credtentail[idx].

Range:

{"idx ssid_str"}

Value:

idx: 0 ~ 7

ssid_str: 0~z, 1~32 ascii characters

Example:

iwpriv ra0 wsc_cred_ssid "0 wps_ap1"

5.1.11 wsc_cred_auth

Set AuthMode into credtentail[idx].

Range:

{"idx auth_str"}

Value:

idx: 0 ~ 7

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

Example:

iwpriv ra0 wsc_cred_auth "0 WPAPSK"

5.1.12 wsc_cred_encr

Set EncryptType into credtentail[idx].

Range:

{"idx encr_str"}

Value:

idx: 0 ~ 7

encr_str: NONE, WEP, TKIP, AES

Example:

iwpriv ra0 wsc_cred_encr "0 TKIP"

5.1.13 wsc_cred_keyIdx

Set Key Index into credtentail[idx].

Range:

{"idx key_index"}

Value:

idx: 0 ~ 7

key_index: 1 ~ 4

Example:

iwpriv ra0 wsc_cred_keyIdx "0 1"

5.1.14 wsc_cred_key

Set Key into credtentail[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))

Example:

iwpriv ra0 wsc_cred_key "0 12345678" ;; Passphrase


```
iwpriv ra0 wsc_cred_key "0 abcd" ;; WEP Key
```

5.1.15 wsc_cred_mac

Set AP's MAC into credtentail[idx].

Range:

{"idx mac_str"}

Value:

idx: 0 ~ 7

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

Example:

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

5.1.16 wsc_conn_by_idx

Connect AP by credential index.

Range:

{0 ~ 7}

Value:

idx: 0 ~ 7

5.1.17 wsc_auto_conn

If the registration is successful, driver will re-connect to AP or not.

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.

5.1.18 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)

5.1.19 Wsc4digitPinCode

Generate WPS 4-digits PIN

Value:

0: Disable

1: Enable

5.2 WPS STA as an Enrollee or Registrar

Build WPS function. Please set the "HAS_WSC" parameter value to "y".

5.2.1 Enrollee Mode

5.2.1.1 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 PinCode on the Registrar and start WPS on the Registrar.
Note:
How to get the Enrollee PinCode? Use 'iwpriv ra0 stat' on the Enrollee.
- (ii) [RT5370 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.

5.2.1.2 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) [RT5370 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.

5.2.2 Registrar Mode

5.2.2.1 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) [RT5370 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.

5.2.2.2 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) [RT5370 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.

5.3 WPS IOCTL use

This section describes specific parameters and arguments. Please refer to the previous section for more general data.

5.3.1 iwpriv commands without argument

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

Example:

```
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;
}
```

5.3.2 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

Example:

```
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;
}
```

5.3.3 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 WPAESK"
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"

Example:

```
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;
}

```

5.4 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 (SIOCIFIRSTPRIV + 0x14)
#define RTPRIV_IOCTL_SET_WSC_PROFILE_STRING_ITEM (SIOCIFIRSTPRIV + 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;
}
//////////

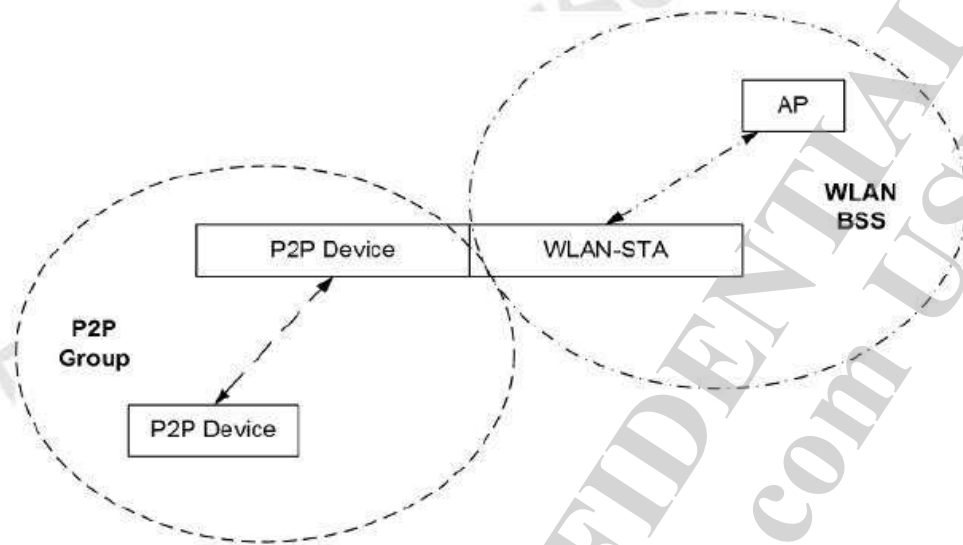
////////// WSC_CREDENTIAL_SSID //////////
memset(&lwreq, 0, sizeof(lwreq));
memset(buffer, 0, 2048);
sprintf(lwreq.ifr_name, "ra0", 3);
sprintf(buffer, "0 wps_ssid_1");
lwreq.u.data.length = strlen(buffer) + 1;
lwreq.u.data.pointer = (caddr_t) buffer;
lwreq.u.data.flags = WSC_CREDENTIAL_SSID;

/* 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;
}

```

6 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.

6.1 Iwpriv use

6.1.1 P2pOpMode

Set p2p interface operate mode to GO.

Value:

1: Auto (Force) GO mode

Example:

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

6.1.2 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
```

6.1.3 p2pOpCh

Set p2p Operation Channel if negotiate as GO

Value:

Based on country region

Example:

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

6.1.4 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

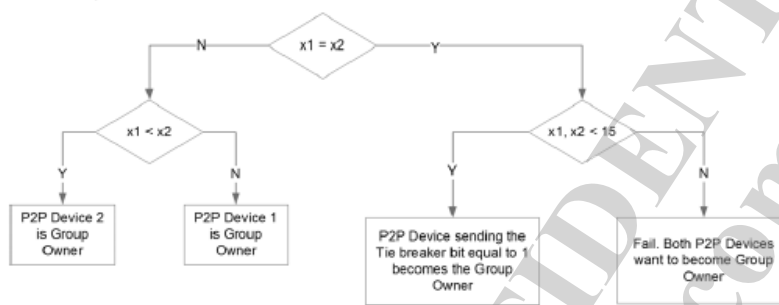


Figure 7—Group Owner determination flowchart

Value:

0~15

Example:

```
#iwpriv p2p0 set p2pGoInt=x
```

6.1.5 p2pDevName

Set p2p device display Device Name

Value:

0~Z, less than 32 characters.

Example:

```
#iwpriv p2p0 set p2pDevName=xxxxx
```

6.1.6 p2pWscMode

Set p2p device WSC Mode for P2P negotiate.

Value:

1: PIN

2: PBC

Example:

```
#iwpriv p2p0 set p2pWscMode=x
```

6.1.7 p2pWscConf

Set p2p device WSC Configure Method

Value:

1: Display

2: KeyPad

3: PBC

Example:

```
#iwpriv p2p0 set p2pWscConf =x
```

6.1.8 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
```

6.1.9 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
```

6.1.10 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
```

6.1.11 p2pLink

Select p2p device ID to do GO Negotiation

Value:

- 0~29 (Software setting)

Example:

```
#iwpriv p2p0 set p2pLink =x
```

6.1.12 p2pCfg

Dump/Show p2p configuration (In kernel background message)

Value:

- Any

Example:

```
#iwpriv p2p0 set p2pCfg
```

6.1.13 p2pTab

Show Group Table status in kernel background message.

Value:

- Any

Example:

```
#iwpriv p2p0 set p2pTab
```

6.1.14 p2pProv

Select p2p device ID to Provision

Value:

0~29 (Software setting)

Example:

#iwpriv p2p0 set p2pProv=x

6.1.15 p2pStat

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

Value:

Any

Example:

#iwpriv p2p0 set p2pStat

6.1.16 p2pReset

Reset p2p configuration the stat machine set to initial stage.

Value:

Any

Example:

#iwpriv p2p0 set p2pReset

6.1.17 p2pPerTab

Show p2p Persistent Table in kernel background message.

Value:

Any

Example:

#iwpriv p2p0 set p2pPerTab

6.1.18 p2pDefConfMthd

Set default WSC Config Method to Provision

Value:

1: Display

2: KeyPad

3: PBC

Example:

#iwpriv p2p0 set p2pDefConfMthd=x

6.1.19 p2pLinkDown

Tear down p2p session and change as p2p device mode.

Value:

Any

Example:

#iwpriv p2p0 set p2pLinkDown=x

6.1.20 p2pSigmaEnable

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

Value:

Any

Example:

```
#iwpriv p2p0 set p2pSigmaEnable=x
```

6.1.21 p2pAutoReconn

For p2p auto reconnection.

Value:

0 - Disable

1 - Enable

Default value is Disable

6.1.22 p2pDevDiscPeriod

For p2p device disconnect period

Value:

decimal

unit: 100ms

Default value is 200

6.1.23 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-Displya, 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
```

P2P device GO Negotiation as GO or CLIENT:**Case 1: To Do P2P GO Negotiation start WPS (PBC):**

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set p2pLisCh=1
#iwpriv p2p0 set p2pGoInt=0 (Default is 0)
#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: To Do P2P GO Negotiation start WPS (PIN-Display):

```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set p2pLisCh=1
#iwpriv p2p0 set p2pGoInt=0 (Default is 0)
#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: To Do P2P GO Negotiation start WPS (PIN-Keypad):

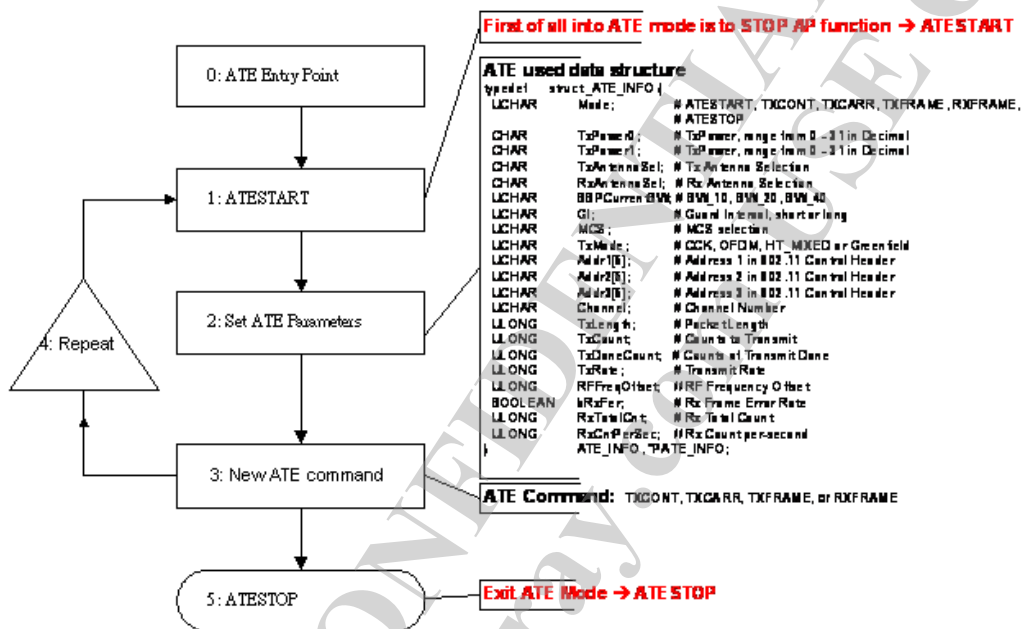
```
#iwpriv p2p0 set p2pOpCh=11
#iwpriv p2p0 set p2pLisCh=1
#iwpriv p2p0 set p2pGoInt=0 (Default is 0)
#iwpriv p2p0 set p2pWscMode=1
#iwpriv p2p0 set p2pWscConf=2
```

```
#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
```

7 ATE TEST COMMAND FORMAT

DO NOT MODIFY THE DEFAULT HARDWARE SETTINGS IF YOU ARE NOT FAMILIAR WITH THE HARDWARE.

Ralink ATE Operation Flow



Note:

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

7.1 iwpriv ra0 set [parameters]=[val]

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

7.1.1 ATE

Set ATE actions.

Value:

ATESTART

- Stop STA & ATE function.

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

7.1.2 ATEDA

Set ATE frame header addr1.

Value: `xx:xx:xx:xx:xx:xx` ; hex

7.1.3 ATESA

Set ATE frame header addr2.

Value: `xx:xx:xx:xx:xx:xx` ; hex

7.1.4 ATEBSSID

Set ATE frame header addr3.

Value: `xx:xx:xx:xx:xx:xx` ; hex

7.1.5 ATECHANNEL

Set ATE Channel, deimal.

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

7.1.6 ATETXPOW0

Set ATE Tx power for Antenna 1.

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

7.1.7 ATETXPOW1

Set ATE Tx power for Antenna 2.

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

7.1.8 ATETXFREQOFFSET

Set ATE RF frequency offset.

Value: 0 ~ 63 ; unit: 2KHz, deimal

7.1.9 ATETXLEN

Set ATE frame length.

Value:
24 ~ 1500 ; decimal

7.1.10 ATETXCNT

Set ATE frame Tx count.

Value:
1 ~ ; 32-bit, decimal

7.1.11 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

7.1.12 ATETXBW (Refer to TxMode)

Set ATE Tx Bandwidth.

Value:

0:	20MHz
1:	40MHz

7.1.13 ATETXGI (Refer to TxMode)

Set ATE Tx Guard Interval.

Value:

0:	Long
1:	Short

7.1.14 ATETXMCS (Refer to TxMode)

Set ATE Tx MCS type.

Value:
0 ~ 15

7.1.15 ATETXANT

Set ATE TX antenna.

Value:

0:	All
1:	Antenna one
2:	Antenna two

7.1.16 ATERXANT

Set ATE RX antenna.

Value:
0: All

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

7.1.17 ATERXFER

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

Value:

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

7.1.18 ATESHOW

Show all parameters of ATE.

Value:

1

7.1.19 ATEHELP

List all commands of ATE.

Value:

1

7.1.20 ResetCounter

Reset statistic counter.

Value:

0

7.1.21 ATERRF

Read all of the RF registers.

Value:

1

7.1.22 ATEWRF1

Write the RF register 1.

Value:

xxxxxxxx ;32-bit, hex

7.1.23 ATEWRF2

Write the RF register 2.

Value:

xxxxxxxx ;32-bit, hex

7.1.24 ATEWRF3

Write the RF register 3.

Value:

xxxxxxx ;32-bit, hex

7.1.25 ATEWRF4

Write the RF register 4.

Value:

xxxxxxx ;32-bit, hex

7.1.26 efuseBufferModeWriteBack

When using the E-fuse buffer mode, the data of EEPROM are all temporary and will disappear after bring down the interface. In order to save the current data of EEPROM, use this command to store all data.

Value:

0: Do nothing (Reserved)
1: Write Back

7.1.27 efuseLoadFromBin

use this command to write data into effuse

Ex: iwpriv ra0 set efuseLoadFromBin=/tmp/RT35xxEEPROM.bin

7.1.28 ATEAUTOALC

Enable ATE legacy Tx auto level control

Value:

1: Activate this command

7.1.29 ATEPAYLOAD

Set ATE payload pattern in TXFRAME mode

Value:

xx

note : x is 0~f

7.1.30 ATETSSICBA

Write temperature compensation reference value into EEPROM(0x6E)

Value:

DAC@CH1

(this value is reference to the value of channel 1 Tx0 power in EEPROM 0x52 bit7~bit0)

Ex:

Iwpriv ra0 e2p 52

[0x52]: 0x12

Iwpriv ra0 set ATETSSICBA=18

7.2 Tx Mode, MCS, BW and GI Selection Table

7.2.1 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:

1. Other MCS codes are reserved in legacy CCK mode.
2. BW, SGI and STBC are reserved in legacy CCK mode.

7.2.2 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:

1. Other MCS code in legacy CCK mode are reserved.
2. When BW = 1, duplicate legacy OFDM is sent.
3. SGI, STBC are reserved in legacy OFDM mode.

7.2.3 MODE = 2, HT Mixed Mode

7.2.4 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 = 32	(BW=1, SGI=0) HT duplicate 6Mbps

Notes:

1. When BW=1, PHY_RATE = PHY_RATE * 2
2. When SGI=1, PHY_RATE = PHY_RATE * 10/9
3. The effects of BW and SGI are accumulative.
4. When MCS=0~7(1S, One Tx Stream), STBC option is supported. SGI option is supported. BW option is supported.
5. When MCS=8~15(2S, Two Tx Stream), STBC option is NOT supported. SGI option is supported. BW option is supported.
6. When MCS=32, only SGI option is supported. BW and STBC option are not supported. (BW =1, STBC=0)
7. Other MCS code in HT mode are reserved.
8. When STBC is supported. Only STBC = 1 is allowed. STBC will extend the transmission range but will not increase transmission rate.

7.3 Examples

7.3.1 Check EVM & Power

```
iwpriv ra0 set ATE=AESTART
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 ATE=TXFRAME
...
iwpriv ra0 set ATETXPOW0=19
...
iwpriv ra0 set ATETXPOW0=20
...
iwpriv ra0 set ATE=AESTART
```

7.3.2 Check Carrier

```
iwpriv ra0 set ATE=AESTART
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=AESTART
```

7.3.3 Check spectrum mask

```
iwpriv ra0 set ATE=AESTART
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(decmlal)
iwpriv ra0 set ATE=TXFRAME ; Start Tx Frame(inform BBP to change, modulation mode)
iwpriv ra0 set ATE=TXCONT ; Start continuous TX, Measure specturm mask with instrument
iwpriv ra0 set ATETXPOW0=5
iwpriv ra0 set ATETXPOW1=5
iwpriv ra0 set ATE=AESTART

```

7.3.4 Frequency offset tuning

```

iwpriv ra0 set ATE=AESTART
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(decmlal)
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=AESTART ; Stop, Store the tuning result to EEPROM

```

7.3.5 Rx

```

iwpriv ra0 set ATE=AESTART
iwpriv ra0 set ATECHANNEL=1 ; set Channel
iwpriv ra0 set ResetCounter=0 ; Reset statistic counter
iwpriv ra0 set ATETXFREQOFFSET=value ; To use the "value"(decimal) you got in tx calibration
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=AESTART ; Stop
iwpriv ra0 stat ; get statistics counter
iwpriv ra0 set ATERXFER=1
iwpriv ra0 set ATERXANT=1

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

```

7.3.6 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

```

7.3.7 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 two, 3:antenna three.
 ATETXFREQOFFSET, 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.

7.3.8 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**

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

MlmePeriodicExec: Rx packet cnt = 2/6
 MlmePeriodicExec: 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**

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

MlmePeriodicExec: Rx packet cnt = 7/14
 MlmePeriodicExec: Rx AvgRssi=-90

7.3.9 Internal ALC calibration (For RT33xx, RT5370 serial chipset)

iwpriv ra0 set ATE=ATESTART
iwpriv ra0 set ATETSSICBA=DAC@CH1

(Note : DAC@CH1 is referred to the value of channel 1 TX0 power, stored in EEPROM 0x52 b7~b0.
 When user is finish this procedure, the EEPROM 0x6E will be stuffed the reference value for internal ALC function)

7.3.10 Internal ALC function testing in ATE mode (For RT33xx, RT5370 serial chipset)


```
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
iwpriv ra0 set ATETXMODE=1
iwpriv ra0 set ATETXMCS=7
iwpriv ra0 set ATETXBW=0
iwpriv ra0 set ATETXGI=0
iwpriv ra0 set ATETXLEN=1024
iwpriv ra0 set ATETXPOW=12
iwpriv ra0 set ATETXCNT=10000000
iwpriv ra0 set ATE TXFRAME
iwpriv ra0 set ATEAUTOALC=1 (Note:Enable temperature compensation)
```

Below is recommend testing flow :

Make sure the device is calibrated already.

Record the channel 1 power DAC value such as #iwpriv ra0 e2p 52 which is 0x0C

Run below command for temperature compensation process:

```
#iwpriv ra0 set ATE=ATE ATESTART
```

```
# iwpriv ra0 set ATETSSICBA=12 ( Note : 12 is the decimal value of 0x0C)
```

Measure the Tx power status in room temperature. (The output power should be +/- 1dBm)

If the output power is normal, please change the temperature and check the Tx power status.

```
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
iwpriv ra0 set ATETXMODE=1
iwpriv ra0 set ATETXMCS=7
iwpriv ra0 set ATETXBW=0
iwpriv ra0 set ATETXGI=0
iwpriv ra0 set ATETXLEN=1024
iwpriv ra0 set ATETXPOW=12
iwpriv ra0 set ATETXCNT=10000000
iwpriv ra0 set ATE TXFRAME
iwpriv ra0 set ATEAUTOALC=1 (Note:Enable temperature compensation)
```

7.4 iwpriv ra0 bbp [parameters]=[Value]

Read/Write BBP register by ID number.

7.4.1 BBPID

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

BBPID:
0 ~ xx ; decimal, 8-bit

7.4.2 BBPID=Value

Write BBP register.

BBPID:
0 ~ xx ; decimal, 8-bit

Value:
00 ~ FF ; hexadecimal, 8-bit

7.5 iwpriv ra0 mac [parameters]=[val]

Read/Write MAC register by offset.

7.5.1 MAC_OFFSET

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

MAC_OFFSET:
0000 ~ FFFF ; hexadecimal, 16-bit

7.5.2 MAC_OFFSET=Value

Write MAC register.

MAC_OFFSET:
0000 ~ FFFF ; hexadecimal, 16-bit
Value:
0000 ~ FFFF ; hexadecimal, 32-bit

7.6 iwpriv ra0 e2p [parameters]=[val]

Read/Write EEPROM content by address.

7.6.1 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, ...)

7.6.2 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

7.7 Example

7.7.1 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
```

7.7.2 Statistic counter operation

```
iwpriv ra0 stat
```

```
# read statistic counter
```

```
iwpriv ra0 set ResetCounter=0
```

```
# reset statistic counter
```

7.7.3 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"

7.8 ated

- ated - user space ATE agent program for RT5370 linux driver, Ralink Tech. Corp.
- RT5370 ATE daemon - ated, comes with RT5370 Linux driver.

The section describes the use of the Linux driver, Windows QA GUI and RT5370 ATE daemon.

7.8.1 Introduction

The ated is an optional user space component for the RT5370 linux driver. AP immediately starts working in ATE mode when "ated" starts (i.e. ATESTART). It behaves as a proxy between Windows QA GUI and RT5370 linux driver when ATE process proceeds.

ATED AUTOMATICALLY STOPS WHEN THE WINDOWS QA GUI IS CLOSED.

Ated can be stopped manually (key in '\$killall ated'). The RT5370 linux driver will stop working in ATE mode when ated is stopped or the QA GUI is closed.

7.8.2 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.

7.8.3 How to use ated for ATE purpose

1. First you should set **both "HAS_ATE=y" and "HAS_28XX_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 use of options, type '\$ated -h')

8. In the Windows host, run RT2870QA_ATE.exe.
9. Select the wired network adapter.
10. Choose 2870_ATE, 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 !

8 IOCTL

8.1 Parameters for iwconfig

Access	Description	ID	Parameters
Get	BSSID, MAC Address	SIOCGIFHWADDR	wrq->u.name, (length = 6)
	WLAN Name	SIOCGIWNAME	wrq->u.name = "RT5370Wireless", 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>
	SSID Name, Nickname	SIOCGIWNICKN	<pre> erq = &wrq->u.data; erq->length = strlen(pAd->nickn); Status = copy_to_user(erq->pointer, pAd->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	SIOCGIWFRAG	wrq->u.frag.value = (INT) pAd-> CommonCfg.FragmentThreshold;

	(bytes)		<pre> 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; wrq->u.mode = IW_MODE_ADHOC; } else if (INFRA_ON(pAd)) { BssType = Ndis802_11Infrastructure; } </pre>

			<pre> 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) { </pre>

			<pre> 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; 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; </pre>

			<pre> 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.CurrReqlsFromNdis = 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) { </pre>

			<pre> pAd->bConfigChanged = TRUE; } pAd->CommonCfg.BssType = BSS_INFRA; } else { Status = -EINVAL; } pAd->CommonCfg.WpaState = SS_NOTUSE; </pre>
--	--	--	---

8.2 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.

8.2.1 Set Data, Parameters is Same as iwpriv

Command and IOCTL Function

Set Data		
Function Type	Command	IOCTL
RTPRIV_IOCTL_SET	iwpriv ra0 set SSID=RT3572AP	<pre> sprintf(name, "ra0"); strcpy(data, "SSID=RT3572AP"); 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>

8.2.2 Get Data, Parameters is Same as iwpriv

Command and IOCTL Function		
Get Data		
Function Type	Command	IOCTL
RTPRIV_IOCTL_STATISTICS	lwpriv 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	lwpriv 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>

8.2.3 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; </pre>

	ioctl(socket_id, RT_PRIV_IOCTL , &wrq);
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"); 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)); </pre>

	<pre> 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; </pre>

	ioctl(socket_id, RT_PRIV_IOCTL , &wrq);
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"); 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)); </pre>

	<pre> 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	<pre> 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); </pre>
OID_802_11_ADD_KEY	<pre> 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); </pre>
OID_802_11_SET_IEEE8021X	<pre> 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); </pre>
OID_802_11_SET_IEEE8021X_REQUIRE_KEY	<pre> printf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(BOOLEAN)); wrq.u.data.length = sizeof(BOOLEAN); </pre>

	<pre> wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_SET_IEEE8021X_REQUIRE_KEY; ioctl(socket_id, RT_PRIV_IOCTL, &wrq); </pre>
OID_802_11_ADD_WEP	<pre> 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); </pre>
OID_802_11_CONFIGURATION	<pre> 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); </pre>
OID_SET_COUNTERMEASURES	<pre> 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); </pre>
OID_802_11_DISASSOCIATE	<pre> 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); </pre>
OID_802_11_PMKID	<pre> printf(name, "ra0"); strcpy(wrq.ifr_name, name); wrq.u.data.length = keylength; //follow your setting </pre>

	<pre> wrq.u.data.pointer = data; wrq.u.data.flags = OID_802_11_PMKID; 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(BOOLEAN)); 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; </pre>

	<pre> 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>
--	--

8.2.4 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; </pre>

	ioctl(socket_id, RT_PRIV_IOCTL , &wrq);
OID_802_11_BSSID_LIST	sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, BssLen); 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);
OID_802_3_CURRENT_ADDRESS	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);
OID_GEN_MEDIA_CONNECT_STATUS	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);
OID_802_11_BSSID	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);
OID_802_11_SSID	sprintf(name, "ra0"); strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(NDIS_802_11_SSID));

	<pre> 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(ulInfo)); wrq.u.data.length = sizeof(ulInfo); 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(ulInfo)); wrq.u.data.length = sizeof(ulInfo); 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)); 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); </pre>

	<pre> 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(ulInfo)); wrq.u.data.length = sizeof(ulInfo); 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"); </pre>

	<pre>strcpy(wrq.ifr_name, name); memset(data, 0, sizeof(RtsThresh)); wrq.u.data.length = sizeof(RtsThresh); 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;</pre>

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

	<pre>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_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;</pre>

	ioctl(socket_id, RT_PRIV_IOCTL , &wrq);
RT_OID_802_11_QUERY_NOISE_LEVEL	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);
RT_OID_802_11_EXTRA_INFO	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);
RT_OID_802_11_QUERY_PIDVID	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);
RT_OID_WE_VERSION_COMPILED	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);
RT_OID_802_11_QUERY_LAST_TX_RATE	HTTRANSMIT_SETTING tmpHT; sprintf(wrq.ifr_name, "ra0");

	<pre> 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); </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_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>

8.2.5 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; </pre>

	<pre> 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>
--	---

9 IOCTL INSTRUCTIONS

9.1 Get Data

9.1.1 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);
```

9.1.2 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);
```

9.1.3 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);
```

9.1.4 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;
```

9.1.5 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);
```

9.1.6 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);
```

9.1.7 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);
```

9.1.8 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);
```

9.1.9 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);
```

9.1.10 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);
```

9.1.11 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);
```

9.1.12 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);
```

9.1.13 GET Authentication Mode:

Sample code =>

```
typedef enum _NDIS_802_11_AUTHENTICATION_MODE
{
    Ndis802_11AuthModeOpen,
    Ndis802_11AuthModeShared,
    Ndis802_11AuthModeAutoSwitch,
    Ndis802_11AuthModeWPA,
    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);
```

9.1.14 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);
```

9.1.15 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);
```

9.1.16 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);
```

9.1.17 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);
```

9.1.18 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);
```

9.2 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];
}
```

```
        *dRate = HTTxRate40_400[nMCS];  
    else  
        return 0; //false  
  
    return 1; //true  
}
```