

Chapter 1

Configuring [MadWifi](#) using Wireless Extensions

This section describes the configuration of the Atheros wireless driver using the Wireless Extension Tools.

1.1 Using iwconfig

The generic iwconfig tool is used to set parameters which common across most drivers. For a detailed description of iwconfig, please use `man iwconfig`. In this Section, we will describe the use of iwconfig in the Madwifi driver. The formats of the iwconfig command is:

```
iwconfig      -help
iwconfig      -version
iwconfig      [interface]
iwconfig      interface [essid X] [freq F] [channel C] [sens S] [ap A] [rate R]
               [rts RT] [frag FT] [txpower T] [enc E] [key K] [retry R]
```

The first form of the iwconfig command gives a brief help message. The second form of the iwconfig command returns the current version of iwconfig along with the version of the wireless extensions with which it was built. In the third form of the iwconfig command, the current wireless status of the interface is returned. If no interface is specified, the current wireless status of every network interface is returned. Non-wireless devices will not return any wireless status.

The last form of the iwconfig command allows the user to change any of the optional parameters. Only the parameters which you wish to change need to be specified. Unspecified parameters will not be modified. Each parameter is described below.

essid - ESSID or Network Name

Set the ESSID (also known as network name) to the given value. In station mode, the driver will attempt to join the network with the same ESSID. In AP mode, the driver will use the parameter as the ESSID.

Example: The following command sets the ssid to "Atheros Wireless Network" on ath0:

```
myprompt# iwconfig ath0 essid "Atheros Wireless Network"
```

freq/channel - RF Frequency or Channel

Set the frequency or channel of the device to the given value. Values below 1000 are interpreted as channel numbers. Values above 1000 are interpreted as frequency measured in Hz. For frequency values, the suffix k, M, or G can be appended to the value to specify kilohertz, Megahertz, and Gigahertz, so that 2.412G, 2412M, and 2412000000 refer to the same frequency. Setting the channel to a specific value will override the private mode control described in Section .

Example: The following command sets the operating frequency to 5.2GHz:

```
myprompt# iwconfig ath0 freq 5.2G
```

Either of the following commands set device to operate on channel 11:

```
myprompt# iwconfig ath0 freq 11  
myprompt# iwconfig ath0 channel 11
```

sens - Sensitivity Threshold

Set the sensitivity threshold to the given value. This is the lowest signal strength at which the packets are received. Currently, this threshold is not implemented and any returned value is meaningless.

ap - Use a Specific AP

Specify which AP the device should associate with. The supplied value should be the MAC address of the desired AP or any of the keywords any, auto, or off.

Example: The following command instructs the ath0 device to associate with the AP that has MAC address 00:03:7f:03:a0:0d.

```
myprompt# iwconfig ath0 ap 00:03:7f:03:a0:0d
```

rate - Set the Data Transmit Rate

Set the bit rate for transmitted packets. The value is specified in bits per second and the values can be suffixed by k, M, or G for kilobits, megabits, and gigabits respectively. The value auto is also valid and causes the device to use the bit rate selected by the rate control module. It's also possible to supply the bit rate followed by auto, in which case the driver will automatically select from the bit rates not exceeding that rate.

Example: The following commands sets the maximum bit rate to 36Mbs. Thus, the driver will automatically select the best rate less than or equal to 36Mbs.

```
myprompt# iwconfig ath0 rate 36M auto
```

rts - Set the RTS/CTS Threshold

Set the minimum packet size for which the device sends an RTS using the RTS/CTS handshake. The parameter is the threshold in bytes or the off keyword. If it's set to off or the maximum packet size, RTS/CTS will be disabled.

Example: The following command sets the minimum packet size to use the RTS/CTS handshake to 40.

```
myprompt# iwconfig ath0 rts 40
```

frag - Set the Fragmentation Threshold

Set the maximum fragment size. The parameter is either the threshold in bytes, or the off keyword, which disables fragmentation.

Example: The following command sets ath0 to fragment all packets to at most 512 bytes.

```
myprompt# iwconfig ath0 frag 512
```

key/enc - Manipulate WEP Encryption Keys and Mode

This parameter is used to manipulate the WEP key and authentication mode. It can be used to set the key, change the key, select the active key, enable and disable WEP, and set the authentication mode. The driver can store up to 4 keys. Each instance of the key command manipulates only one key. Thus, to change all 4 keys, 4 separate commands must be used. The key value can be specified as is in the hexadecimal form. If an ASCII string is used for the key value, prepend "s:" to the key. To change a key other than the current key, prepend "[index]" to the key value, where index is the number of the key you wish to change. To select which key is active, use "[index]" without supplying any keys, where index is the desired key number. Including the keywords open or restricted changes the authentication mode between open authentication and restricted authentication. Use the off keyword to disable WEP.

Example: The following command sets the default key to be key 3:

```
myprompt# iwconfig ath0 key [3]
```

The following command sets the default key to be the hex key 0xDEAD-BEEF-AA:

```
myprompt# iwconfig ath0 key DEAD-BEEF-AA
```

The following command sets key 2 to the ASCII phrase "password" and sets the authentication type to open:

```
myprompt# iwconfig ath0 key [2] s:password open
```

The following command disables WEP:

```
myprompt# iwconfig ath0 key off
```

txpower - Set Transmit Power

Set the transmit power for data packets. Bare numbers are interpreted as values in dBm. If the number is followed by mW, the value is interpreted in milliwatts. The value can also be auto for automatic power control and off for disabling the radio transmission.

Example: The following command sets all data packets to transmit at either 30 dBm or the maximum allowed in the current regulatory domain:

```
myprompt# iwconfig ath0 txpower 30
```

retry - Set Retry Limit

This parameter sets the maximum number of retries used in the software retry algorithm. Currently, the driver does not implement software retry, thus this parameter is meaningless.

1.2 Using wlanconfig

The current [MadWifi](#) driver supports multiple APs and concurrent AP/Station mode operation on the same device. The devices are restricted to using the same underlying hardware, thus are limited to coexisting on the same channel and using the same physical layer features. Each instance of an AP or station is called a Virtual AP (or VAP). Each VAP can be in either AP mode, station mode, "special" station mode, and monitor mode. Every VAP has an associated underlying base device, which is created when the driver is loaded. Creating and destroying VAPs are done through the wlanconfig tool found in the [MadWifi](#) tools directory. Running the wlanconfig utility with no arguments returns a brief help line. The format of the wlanconfig command takes two forms:

```
wlanconfig VAP create wlandev Base Device wlanmode mode [uniquebssid]  
wlanconfig VAP destroy
```

Every Linux network device consists of a prefix followed by a number indicating the device number of the network device. For instance, the ethernet devices are named eth0, eth1, eth2, etc. Each VAP which is created is also registered as a Linux network device. The value VAP can be either a prefix name of the Linux network device, or it can be the entire device name. For instance, specifying VAP as ath lets the Linux kernel add the network device as the next device with the prefix ath. Thus, the Linux kernel appends the proper number to the end to form the full device name, e.g., ath1 if ath0 already exists. However, the full device name can also be specified. For instance, VAP

can also be ath2. In this case, the network device ath2 is registered, regardless of whether ath1 exists.

The Base Device is the underlying wireless network device name created when the driver is loaded. The [MadWifi](#) driver creates wifi0, wifi1, etc. as the underlying devices. By specifying the Base Device, the VAP is created with the Base Device as the parent device. The mode is the operating mode of the VAP. The operating mode of the VAP cannot be changed once it is created. In special cases, the operating mode of the VAP can be different from the operating mode of the underlying parent device. The first VAP which is created sets the operating mode of the underlying device. If the first VAP is deleted and a new VAP is created with a different operating mode than the original VAP, then the operating mode of the underlying device is changed to the new operating mode. The valid operating modes and their descriptions are given in Table 1.

Mode	Description
Auto	Auto select operating mode
Managed	Station mode for infrastructure networks
Master	AP mode
Monitor	Passive monitor (promiscuous) mode

Table 1: wlanconfig Operating Modes

Only one station VAP can exist on a device. If the station VAP is the first VAP created, then no other VAPs are allowed to be created. If the first VAP created is in AP (Master) mode, then one station VAP is allowed to be created. In this case, other AP VAPs can also be created after the station VAP. Prior to [r3476](#), when AP and station VAPs coexist, a "nosbeacon" parameter was required on the wlanconfig command line when creating the station. This flag disables the use of hardware beacon timers for station mode operation. This is now automatically handled. Concurrent AP and station operation implies the station should not modify the TSF clock for the APs. Creating multiple VAPs typically implies that the MAC address of each VAP is different. In point of fact, you cannot have multiple VAPs up and running concurrently. In most VAP types, unique MAC addresses are assigned to each VAP - starting with the MAC address of the underlying device. By using the uniquebssid parameter (formerly "bssid") you will skip over the underlying radio's MAC address and prevent it from being used by the first VAP created. The unique MAC addresses are assigned to the VAPs being created from a small pool of 64 addresses allocated by setting the 'locally administered' bit in the MAC and adjusting the five highest bits in the MAC address. In this way, the MAC address is mostly left unchanged and diagnostics are a bit easier. To destroy a VAP, the wlanconfig command is used with the destroy parameter. In this case, the full device name must be used, i.e. you must specify the entire name, such as "ath12" not just the prefix.

Example: If we wish to use the system as a station only, we would create a single station VAP once the driver is loaded. The following command creates a single station VAP named ath0 on device wifi0:

```
myprompt# wlanconfig ath create wlandev wifi0 wlanmode sta
```

Note that no other VAPs can be created since we are assuming this is the first VAP created on wifi0. Since this is the first VAP created, we only need to specify ath, not ath0. However, the following command would also be correct:

```
myprompt# wlanconfig ath0 create wlandev wifi0 wlanmode sta
```

The MAC address of the station VAP is the same as the underlying device's MAC address since it is the first VAP created.

Example: Now, we wish to create two AP VAPs on device wifi0. The first device will have a cloned MAC address taken from the underlying device. The second VAP will have a "virtual" MAC address formed from the underlying device's MAC address. The first VAP will be ath0 and the second device will be ath2.

```
myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
myprompt# wlanconfig ath2 create wlandev wifi0 wlanmode ap
```

Example: Now, we wish to create two AP VAPs on device wifi0. The first device will have the MAC address cloned from the underlying device, and the second will have the next available unique MAC. The first VAP will be ath0 and the second VAP will be ath1.

```
myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
```

NOTE: In revisions prior to [r3476](#) it was often recommended [but not necessary] to specify "bssid" parameter when creating the second AP to indicate a unique MAC should be used. Such assignments are now correctly done automatically. NOTE: To prevent the use of the underlying device MAC, you can specify the "uniquebssid" parameter and all VAPs will be allocated MAC addresses from the pool of ("locally administered") unique MAC addresses derived from the MAC assigned by the hardware manufacturer.

Example: Now, we wish to create two AP VAPs and one station VAP. The AP VAPs will be ath0 and ath2 and the station VAP will be ath1.

```
myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
myprompt# wlanconfig ath create wlandev wifi0 wlanmode sta
myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
```

NOTE: In revisions prior to [r3476](#) it was necessary to specify "nosbeacon" parameter in the station create command, but this is obvious in context and is now handled automatically by the driver.

Example: Now, we wish to destroy a VAP (regardless of its operating mode). We assume that there is a VAP named ath0, and it's the one we wish to destroy.

```
myprompt# wlanconfig ath0 destroy
```

1.3 Private (Driver Specific) Driver Commands

The following is a list of the private commands which are accessible using [iwpriv](#). The general syntax of [iwpriv](#) is

```
iwpriv device [command] [parameters].
```

The entire list of [iwpriv](#) commands can be found by running [iwpriv](#) to a device without any command. The resulting list of commands has several columns. The number of parameters allowed for each command is listed. Parameters are classified as either "set" or "get" parameters. "Set" parameters are parameters which the user supplies to the driver. "Get" parameters are parameters which the driver returns to the user.

setoptie - Set Optional Information Element

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: ??
Resets State Machine After Command: ??

This command takes a 256 byte input parameter which specifies?

getoptie - Get Optional Information Element

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: ??

This command gets the optional information element. The information element is returned as 256 bytes.

mode - Set Wireless Mode

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: auto
Resets State Machine After Command: Yes

This command sets the wireless mode, i.e. the frequency band and the protocol used. The mode can be specified either by name or by number. The allowed mode names and corresponding numbers are given in Table 2.

Mode	Number	Description
auto	0	Auto select operating mode
11a	1	802.11a (5GHz) mode (54Mbps)
11b	2	802.11b (2.4GHz) mode (11Mbps)

11g	3	802.11g (2.4GHz) mode with 802.11b compatibility (54Mbps)
fh	4	802.11 frequency hopping mode
11adt/111at	5	802.11a (5GHz) dynamic turbo mode
11gdt/11gt	6	802.11g (2GHz) dynamic turbo mode (108Mbps)
11ast	7	802.11a (5GHz) static turbo mode

Table 2: 802.11 Operating Modes

Example: Either of the following two commands will set the wireless operating mode on a device named ath0 to use 802.11a dynamic turbo:

```
myprompt# iwpriv ath0 mode 11a
```

or

```
myprompt# iwpriv ath0 mode 1
```

get_mode - Get Wireless Mode

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns the wireless mode of VAP. The returned values correspond to the modes given in Table 2.

Example: The following command retrieves the wireless mode of a device named ath0 which we will assume is operating in the 802.11g mode:

```
myprompt# iwpriv ath0 get_mode
ath0      get_mode:11g
```

hide_ssid - Enable/Disable Hiding of the 802.11 SSID

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Disabled
Resets State Machine After Command: Yes

This command enables and disables the ability to hide the 802.11 SSID in the beacon if the VAP is in AP mode. To enable hiding of the SSID, a value of 1 is passed into the driver. To disable hiding of the SSID, a value of 0 is passed into the driver.

Example: The following command enables hiding the 802.11 SSID on ath0:

```
myprompt# iwpriv ath0 hide_ssid 1
```

get_hide_ssid - Get Status of 802.11 SSID Hiding Support

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns whether the driver is currently hiding the 802.11 SSID in beacons. A value of 1 indicates that the VAP is hiding the 802.11 SSID. A value of 0 indicates the VAP is not hiding the 802.11 SSID.

Example: The following command retrieves whether ath0 is hiding the 802.11 SSID in its beacon:

```
myprompt# iwpriv ath0 get_hide_ssid  
ath0      get_hide_ssid:0
```

protmode - Enable/Disable 802.11g Protection Mode

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value:
Enabled Resets State Machine After Command: Yes

This command enables and disables the 802.11g protection mode. To enable 802.11g protection, a value of 1 is passed into the driver. To disable 802.11g protection, a value of 0 is passed into the driver.

Example: The following command disables 802.11g protection on ath0:

```
myprompt# iwpriv ath0 protmode 0
```

get_protmode - Get Status of 802.11g Protection Mode

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns whether the driver is currently using 802.11g protection mode. A value of 1 indicates that the VAP is using 802.11g protection. A value of 0 indicates the VAP is not using 802.11g protection.

Example: The following command retrieves whether ath0 is using 802.11g protection mode:

```
myprompt# iwpriv ath0 get_protmode  
ath0      get_protmode:1
```

ap_bridge - Set Internal Client Bridging

This command is used to moderate the internal bridging on the access point. Setting ap_bridge to 0 will prevent connected clients from being able to access each other. Setting the ap_bridge to 1 will allow connected clients to access each other.

```
ap_bridge (0014) : set 1 int & get 0
```

Referenced in:

```
net80211/ieee80211_wireless.c:2
```

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

```
myprompt# iwpriv ath0 ap_bridge 0
```

get_ap_bridge - Get Status of Internal Client Bridging

This command will show the current state of the ap_bridge. 0 is down, 1 is up

```
get_ap_bridge (0014) : set 0 & get 1 int
```

Referenced in:

```
net80211/ieee80211_wireless.c:1
```

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

```
myprompt# iwpriv ath0 get_ap_bridge
```

inact_init - Set Inactivity Period for INIT State

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: 30 secs Resets State Machine After Command: No

This command sets the inactivity period for when the net80211 state machine is in the INIT (initialization) state. The argument passed into the driver is the desired inactivity period in seconds.

Example: The following command sets the inactivity period for the INIT state on ath0 to 90 seconds:

```
myprompt# iwpriv ath0 inact_init 90
```

get_inact_init - Get Inactivity Period for INIT State

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A Resets State Machine After Command: No

This command gets the inactivity period for when the net80211 state machine is in the INIT (initialization) state.

Example: The following command gets the inactivity period for the INIT state on ath0:

```
myprompt# iwpriv ath0 get_inact_init  
ath0      get_inact_init:30
```

inact_auth - Set Inactivity Period for AUTH State

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: 180 secs Resets State Machine After Command: No

This command sets the inactivity period for when the net80211 state machine is in the AUTH (authorization) state. The argument passed into the driver is the desired inactivity period in seconds.

Example: The following command sets the inactivity period for the AUTH state on ath0 to 90 seconds:

```
myprompt# iwpriv ath0 inact_auth 90
```

get_inact_auth - Get Inactivity Period for AUTH State

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A Resets State Machine After Command: No

This command gets the inactivity period for when the net80211 state machine is in the AUTH (authorization) state.

Example: The following command gets the inactivity period for the AUTH state on ath0:

```
myprompt# iwpriv ath0 get_inact_auth  
ath0      get_inact_auth:180
```

inact - Set Inactivity Period for RUN State

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: 300 secs Resets State Machine After Command: No

This command sets the inactivity period for when the net80211 state machine is in the RUN (running) state. The argument passed into the driver is the desired inactivity period in seconds.

Example: The following command sets the inactivity period for the RUN state on ath0 to 90 seconds:

```
myprompt# iwpriv ath0 inact 90
```

get_inact - Get Inactivity Period for RUN State

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command gets the inactivity period for when the net80211 state machine is in the RUN (running) state.

Example: The following command gets the inactivity period for the RUN state on ath0:

```
myprompt# iwpriv ath0 get_inact  
ath0      get_inact:300
```

dtim_period - Set DTIM Period

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: Yes

This command sets the beacon DTIM period. The argument passed to the driver is the desired DTIM period in ms.

Example: The following command sets the DTIM period to 2 ms on ath0:

```
myprompt# iwpriv ath0 dtim_period 2
```

get_dtim_period - Get Beacon DTIM Period

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command gets the current beacon DTIM in ms.

Example: The following command gets the DTIM period on ath0:

```
myprompt# iwpriv ath0 get_dtim_period  
ath0      get_dtim_period:1
```

bintval - Set Beacon Interval Value

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: Yes

This command sets the beacon interval. The argument passed to the driver is the desired beacon interval in ms.

Example: The following command sets the beacon interval to 25 ms on ath0:

```
myprompt# iwpriv ath0 bintval 25
```

get_bintval - Get Beacon Interval Value

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command gets the current beacon interval in ms.

Example: The following command gets the beacon interval on ath0:

```
myprompt# iwpriv ath0 get_bintval  
ath0      get_bintval:100
```

doth - 802.11h Support Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value:
Disabled Resets State Machine After Command: Yes

This command enables and disables the 802.11h support. To enable the support, a value of 1 is passed into the driver. To disable 802.11h support, a value of 0 is passed into the driver.

Example: The following command enables 802.11h on ath0:

```
myprompt# iwpriv ath0 doth 1
```

get_doth - Get 802.11h Support Status

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns whether 802.11h support is enabled or disabled in the driver.

Example: The following command retrieves the 802.11h status on ath0:

```
myprompt# iwpriv ath0 get_doth  
ath0      get_doth:0
```

doth_reassoc - Generate a Reassociation Request

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command instructs the driver to generate a Reassociation request. A single input parameter is needed but ignored.

Example: Either of the following commands generates a reassociation request on ath0.

```
myprompt# iwpriv ath0 doth_reassoc 1
```

or

```
myprompt# iwpriv ath0 doth_reassoc 0
```

doth_pwr tgt - Set Maximum Desired Power for Transmission

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command sets the desired maximum power on the current channel. The minimum of this desired value and the regulatory maximum is used as the true transmission power. The single argument passed into the driver is the desired power level in 0.5 dBm steps.

Example: To set the desired power level on the current channel to be 13 dBm, the following command is used:

```
myprompt# iwpriv ath0 doth_pwr tgt 26
```

wpa - Enable/Disable WPA/WPA2 Support

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Disabled
Resets State Machine After Command: Yes

This command enables or disables WPA or WPA2 support. A single argument is passed to the driver indicating which encryption protocols is to be supported. Table 3 lists the arguments and the encryption protocols supported.

Argument Supported	Protocol
0	No WPA
1	WPA Supported
2	WPA2 Supported
3	Both WPA and WPA2 supported

Table 3: WPA/WPA2 Support Arguments

Example: To enable both WPA and WPA2, the following command is used:

myprompt# iwpriv ath0 wpa 3

mcastcipher - Set Group Key Length

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command sets the group key (multicast) key length. This command is used mainly by hostapd. See the driver_madwifi.c file in hostapd for details on the use of this command.

get_mcastcipher - Get Group Key Length

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns the current group key length. This command is used mainly by hostapd.

mcastcipher - Set Group Key Cipher

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command sets the group key (multicast) cipher. This command is used mainly by hostapd. See the driver_madwifi.c file in hostapd for details on the use of this command.

get_mcastcipher - Get Group Key Cipher

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns the current group key cipher. This command is used mainly by hostapd.

ucastciphers - Set Pairwise Unicast Key Ciphers

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command sets the pairwise unicast key cipher. Each bit position indicates a supported WPA pairwise cipher. The bitmask and definitions are defined in hostapd. This command is used mainly by hostapd. See the driver_madwifi.c file in hostapd for details on the use of this command.

get_ucastciphers - Get Pairwise Unicast Key Ciphers

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns the current pairwise unicast key ciphers. This command is used mainly by hostapd.

ucastcipher - Set Unicast Cipher

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command sets the unicast key cipher. Currently not used.

get_ucastcipher - Get Current Unicast cipher

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns the current unicast key cipher. Currently not used.

ucastkeylen - Set Unicast Key Length

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: 13
Resets State Machine After Command: No

This command sets the length of the unicast key. A single parameter is supplied which is the desired length of the unicast key. The desired length must be a positive number less than 16. Currently not used.

Example: To set the unicast key length on ath0 to 10, the following command is used:

```
myprompt# iwpriv ath0 ucastkeylen 10
```

get_ucastkeylen - Get Current Unicast Key Length

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns the current unicast key length. Currently not used.

Example: The following command returns the current unicast key length being used on ath0.

```
myprompt# iwpriv ath0 get_ucastkeylen  
ath0      get_ucastkeylen:13
```

keymgtaigs - Select Key Management Algorithm

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: 3
Resets State Machine After Command: Yes if WPA/WPA2 is enabled

This command selects the key management algorithm used. A single parameter is passed into the driver indicating which algorithm to use. Table 4 lists the parameter value and the corresponding algorithm. This command is used by hostapd and wpa_supplicant.

Parameter	Algorithm
0	No WPA Algorithm
1	WEP Algorithm
2	WPA TKIP Algorithm
3	WPA CCMP Algorithm

Table 4: Key Management Algorithms

Example: To set the key management algorithm to ??? on ath0, the following command is used:

```
myprompt# iwpriv ath0 keymgmtalgs 2
```

get_keymgmtalgs - Get Current Key Management Algorithm

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns the current key management algorithm. The value returned corresponds to the key management algorithm as dictated by Table 4.

Example: The following command returns the current key management algorithm being used on ath0.

```
myprompt# iwpriv ath0 get_keymgmtalgs  
ath0 get_keymgmtalgs:3
```

rsncaps - Set ???

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: ??
Resets State Machine After Command: Yes if WPA/WPA2 is enabled

This commands sets ???.

Example: The following command sets the ??? of ath0 to XX:

```
myprompt# iwpriv ath0 rsncaps XX
```

get_rsn caps - Get Current ???

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns the current value of ???.

Example: The following command returns the current value of ??? on ath0.

```
myprompt# iwpriv ath0 get_rsn caps
ath0      get_rsn caps:0
```

hostroaming - Set Roaming Mode

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Auto
Resets State Machine After Command: No

This command sets the roaming mode which is effectively who controls the operation (state transitions) of the 802.11 state machine when running as a station. Stations are either controlled by the driver (typically when management frames are processed by the hardware), the host (auto/normal operation of the 802.11 layer), or explicitly through ioctl requests when applications such as wpa_supplicant want control. A single argument is passed to the driver indicating the desired roaming mode. Table 5 lists the arguments and corresponding roaming modes.

Argument	Roaming Mode	Description
0	Device	Driver/hardware control
1	Auto	802.11 layer control
2	Manual	ioctl/application control

Table 5: Roaming Mode Arguments

Example: The following command sets the roaming mode to Auto on ath0.

```
myprompt# iwpriv ath0 hostroaming 1
```

get_hostroaming - Get Roaming Mode

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns the roaming mode of the device. The returned value corresponds to the modes given in Table 5.

Example: The following command returns the roaming mode of ath0:

```
myprompt# iwpriv ath0 get_hostroaming
ath0      get_hostroaming:1
```

privacy - Enable/Disable Privacy

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Disabled Resets State Machine After Command: No

This command enables or disables privacy on the device. Passing a value of 1 enables privacy. Passing a value of 0 disables privacy.

Example: The following command enables privacy on ath0:

```
myprompt# iwpriv ath0 privacy 1
```

get_privacy - Get Privacy Status

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A Resets State Machine After Command: No

This command returns the privacy status on the device. A value of 1 indicates privacy is enabled. A value of 0 indicates privacy is disabled.

Example: The following command returns the privacy status on ath0:

```
myprompt# iwpriv ath0 get_privacy
ath0      get_privacy:0
```

dropunencrypted - Enable/Disable Dropping of Unencrypted non-PAE frames

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Disabled Resets State Machine After Command: No

This command enables or disables dropping of unencrypted non-PAE frames received. Passing a value of 1 enables dropping of unencrypted non-PAE frames. Passing a value of 0 disables dropping of unencrypted non-PAE frames.

Example: The following command enables dropping of unencrypted non-PAE frames on ath0:

```
myprompt# iwpriv ath0 dropunencrypted 1
```

get_dropunencry - Get Status of Dropping of Unencrypted non-PAE frames

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A Resets State Machine After Command: No

This command returns whether the device is dropping unencrypted non-PAE frames. A value of 1 indicates that unencrypted non-PAE frames are being dropped. A value of 0 indicates that unencrypted non-PAE frames are not being dropped.

Example: The following command returns whether ath0 is dropping unencrypted non-PAE frames:

```
myprompt# iwpriv ath0 get_dropunencyr  
ath0      get_dropunencyr:0
```

get_wpa - Get WPA/WPA2 Support

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command gets the current status of WPA/WPA2 support in the driver.

Example: The following command retrieves the status of WPA/WPA2 support in the driver:

```
myprompt# iwpriv ath0 get_wpa  
ath0      get_wpa:0
```

countermeasures - Enable/Disable WPA/WPA2 Countermeasures

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value:
Disabled Resets State Machine After Command: No

This command enables or disables WPA/WPA2 countermeasures. Passing a value of 1 enables countermeasures if WPA or WPA2 are enabled. Passing a value of 0 disables countermeasures.

Example: The following command enables WPA/WPA2 countermeasures in the driver:

```
myprompt# iwpriv ath0 countermeasures 1
```

get_countermeas - Get Status of WPA/WPA2 Countermeasures

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value:
Disabled Resets State Machine After Command: No

This command returns the status of WPA/WPA2 countermeasure support. A value of 1 indicates WPA/WPA2 countermeasures are enabled. A value of 0 indicates WPA/WPA2 countermeasures are disabled.

Example: The following command retrieves the status of WPA/WPA2 countermeasures in the driver:

```
myprompt# iwpriv ath0 get_countermeas
ath0      get_countermeas:0
```

get_driver_caps - Get Driver Capabilities

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command gets the current driver capabilities. The bitmask of capabilities can be found in the file net80211/ieee80211_var.h.

Example: The following command retrieves the capabilities of the driver

```
myprompt# iwpriv ath0 get_driver_caps
ath0      get_driver_caps:126018575
```

addmac - Add MAC address to ACL list

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command takes a single argument which is the MAC address to be added to the ACL list.

Example: The following command adds the MAC address 00:03:7F:03:A0:0C to the ACL list.

```
myprompt# iwpriv ath0 add_mac 00:03:7f:03:a0:0c
```

delmac - Delete MAC address to ACL list

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command takes a single argument which is the MAC address to be deleted from the ACL list.

Example: The following command deletes the MAC address 00:03:7F:03:A0:0C from the ACL list.

```
myprompt# iwpriv ath0 del_mac 00:03:7F:03:A0:0C
```

maccmd - Set or Modify the MAC/ACL Handling

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command takes a single argument which describes the action one wishes to take on the MAC/ACL list. MAC addresses can be added/deleted from the ACL list using the `addmac` and `delmac` commands. Table 6 gives the commands and their associated actions.

Argument Action

0	No ACL checking is performed
1	Only allow ACLs in the ACL list
2	Only deny ACLs in the ACL list
3	Flush the ACL database
4	Remove the ACL policy

Table 6: ACL Commands

Example: The following command denies traffic to all MAC addresses in the ACL list on `ath0`.

```
myprompt# iwpriv ath0 maccmd 2
```

kickmac - Disassociate an associated station

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command takes a single argument which is the MAC address of a currently associated client. The client is immediately sent a disassociate frame (with an unspecified reason). There is nothing to prevent the client from immediately reassociating. If you are wishing to permanently remove a client from the access point you will need to also make use of the `maccmd`, `addmac` and `delmac` commands to configure the appropriate ACL entries.

Example: The following command immediately disassociates the client with MAC address `00:03:7f:03:42:3f`.

```
myprompt# iwpriv ath0 kickmac 00:03:7f:03:42:3f
```

wmm - WMM Support Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value:
Enabled Resets State Machine After Command: Yes

This command enables or disables WMM support. Passing a value of 1 to the driver enables WMM. Passing a value of 0 to the driver disables WMM. By default, WMM is enabled.

Example: The following command disables WMM support on ath0.

```
myprompt# iwpriv ath0 wmm 0
```

get_wmm - Get WMM Support

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns whether WMM support is enabled or disabled in the driver.

Example: The following command retrieves the status of WMM support in the driver:

```
myprompt# iwpriv ath0 get_wmm  
ath0      get_wmm:1
```

cwmin - WMM CWmin Parameter

Number of Input Arguments: 3 Number of Returned Arguments: 0 Default Value: Varies
Resets State Machine After Command: No

This command sets the CWmin WMM parameter for either the AP or station parameter set. A description of the AP and station parameter set and their default values can be found in the WMM standard. The cwmin command must be followed by 3 values. The first value is the access class (AC) number as defined in Table 7 taken from the WMM standard. The second value indicates whether the CWmin value is intended for the AP or station parameter set. A value of 0 indicates the CWmin is for the AP parameter set. A value of 1 indicates the CWmin is for the station parameter set. The third value is the actual value of the CWmin in units as described in the WMM standard.

AC Number Access Class Description

0	BE - Best Effort
1	BK - Background
2	VI - Video
3	VO - Voice

Table 7: Access class (AC) Values

Example: The following command sets the CWmin in the station parameter set for the VO AC to 2.

```
myprompt# iwpriv ath0 cwmin 3 1 2
```

get_cwmin - Get WMM CWmin Parameter

Number of Input Arguments: 2 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command retrieves the CWmin WMM parameter for either the AP or station parameter set. The `get_cwmin` command must be followed by 2 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates whether to retrieve the value from the AP or station parameter set. A value of 0 indicates the CWmin is from the AP parameter set. A value of 1 indicates the CWmin is from the station parameter set.

Example: The following command gets the CWmin in the AP parameter set for the VI AC.

```
myprompt# iwpriv ath0 get_cwmin 2 0
ath0      get_cwmax:4
```

cwmax - WMM CWmax Parameter

Number of Input Arguments: 3 Number of Returned Arguments: 0 Default Value: Varies
Resets State Machine After Command: No

This command sets the CWmax WMM parameter for either the AP or station parameter set. The `cwmax` command must be followed by 3 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates whether the CWmax value is intended for the AP or station parameter set. A value of 0 indicates the CWmax is for the AP parameter set. A value of 1 indicates the CWmax is for the station parameter set. The third value is the actual value of the CWmax in units as described in the WMM standard.

Example: The following command sets the CWmax in the AP parameter set for the BK AC to 5.

```
myprompt# iwpriv ath0 cwmax 1 0 5
```

get_cwmax - Get WMM CWmax Parameter

Number of Input Arguments: 2 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command retrieves the CWmax WMM parameter for either the AP or station parameter set. The `get_cwmax` command must be followed by 2 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates whether to retrieve the value from the AP or Station parameter set. A value of 0 indicates the CWmax is from the AP parameter set. A value of 1 indicates the CWmax is from the station parameter set.

Example: The following command gets the CWmax in the station parameter set for the BE AC.

```
myprompt# iwpriv ath0 get_cwmax 0 1
ath0      get_cwmax:10
```

txoplimit - WMM [TxOp?](#) Limit Parameter

Number of Input Arguments: 3 Number of Returned Arguments: 0 Default Value: Varies
Resets State Machine After Command: No

This command sets the [TxOp?](#) limit WMM parameter for either the AP or station parameter set. The txoplimit command must be followed by 3 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates whether the [TxOp?](#) limit is intended for the AP or station parameter set. A value of 0 indicates the [TxOp?](#) limit is for the AP parameter set. A value of 1 indicates the [TxOp?](#) limit is for the station parameter set. The third value is the actual value of the [TxOp?](#) limit in units as described in the WMM standard.

Example: The following command sets the [TxOp?](#) limit in the AP parameter set for the BE AC to 1024.

```
myprompt# iwpriv ath0 txoplimit 0 0 1024
```

get_txoplimit - Get WMM [TxOp?](#) Limit Parameter

Number of Input Arguments: 2 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command retrieves the [TxOp?](#) Limit WMM parameter for either the AP or station parameter set. The get_txoplimit command must be followed by 2 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates whether to retrieve the value from the AP or station parameter set. A value of 0 indicates the [TxOp?](#) limit is from the AP parameter set. A value of 1 indicates the [TxOp?](#) limit is from the station parameter set.

Example: The following command gets the [TxOp?](#) limit in the station parameter set for the BE AC.

```
myprompt# iwpriv ath0 get_txoplimit 0 1
ath0      get_txoplimit:2048
```

aifs - WMM AIFS Parameter

Number of Input Arguments: 3 Number of Returned Arguments: 0 Default Value: Varies
Resets State Machine After Command: No

This command sets the AIFS WMM parameter for either the AP or station parameter set. The aifs command must be followed by 3 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates whether the AIFS is intended for the AP or station parameter set. A value of 0 indicates the AIFS is for the AP parameter set. A value of 1 indicates the AIFS is for the station parameter set. The third value is the actual AIFS value in units as described in the WMM standard.

Example: The following command sets the AIFS value in the AP parameter set for the BE AC to 3.

```
myprompt# iwpriv ath0 aifs 0 0 3
```

get_aifs - Get WMM AIFS Parameter

Number of Input Arguments: 2 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command retrieves the AIFS WMM parameter for either the AP or station parameter set. The get_aifs command must be followed by 2 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates whether to retrieve the value from the AP or station parameter set. A value of 0 indicates the AIFS value is from the AP parameter set. A value of 1 indicates the AIFS value is from the station parameter set.

Example: The following command gets the AIFS value in the station parameter set for the BE AC.

```
iwpriv ath0 get_aifs 0 1  
ath0 get_aifs:2
```

acm - WMM ACM Bit Value

Number of Input Arguments: 3 Number of Returned Arguments: 0 Default Value: Varies
Resets State Machine After Command: No

This command sets the ACM bit value in the WMM parameters for the station parameter set. The acm command must be followed by 3 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates the ACM bit value is intended for the station parameter set. Thus, the second value should always be 1. The third value is the desired ACM bit value (either 0 or 1).

Example: The following command sets the ACM bit to 1 in the station parameter set for the BE AC.

```
myprompt# iwpriv ath0 acm 0 1 1
```

get_acm - Get WMM ACM Bit Value

Number of Input Arguments: 2 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command retrieves the ACM bit value in the current station WMM parameter set. The `get_acm` command must be followed by 2 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates the ACM value is from the station parameter set, thus the second value should be 1.

Example: The following command gets the ACM bit value in the station parameter set for the BE AC.

```
myprompt# iwpriv ath0 get_acm 0 1
ath0      get_acm:0
```

noackpolicy - WMM [NoAck?](#) Policy Bit Value

Number of Input Arguments: 3 Number of Returned Arguments: 0 Default Value: Varies
Resets State Machine After Command: No

This command sets the [NoAck?](#) Policy bit value in the WMM parameters for the AP parameter set. The `noackpolicy` command must be followed by 3 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates the [NoAck?](#) policy bit value is intended for the AP parameter set. Thus, the second value should always be 0. The third value is the desired [NoAck?](#) Policy bit value (either 0 or 1).

Example: The following command sets the [NoAck?](#) Policy bit to 1 in the AP parameter set for the BE AC.

```
myprompt# iwpriv ath0 noackpolicy 0 1 1
```

get_noackpolicy - Get WMM [NoAck?](#) Policy Bit Value

Number of Input Arguments: 2 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command retrieves the [NoAck?](#) Policy bit value in the current AP WMM parameter set. The `get_noackpolicy` command must be followed by 2 values. The first value is the access class (AC) number as defined in Table 7. The second value indicates the [NoAck?](#) policy value is from the AP parameter set, thus the second value should be 0.

Example: The following command gets the [NoAck?](#) Policy bit value in the AP parameter set for the BE AC.

```
myprompt# iwpriv ath0 get_noackpolicy 0 1
ath0      get_noackpolicy:0
```

ff - Atheros Fast Frame Support Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Enabled Resets State Machine After Command: No

This command enables or disables Atheros Fast Frame support. Passing a value of 1 to the driver enables fast frames. Passing a value of 0 to the driver disables fast frames. By default, fast frames is enabled if the hardware supports fast frames.

Example: The following command disables Atheros Fast Frame support on ath0.

```
myprompt# iwpriv ath0 ff 0
```

get_ff - Get Atheros Fast Frame Support

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A Resets State Machine After Command: No

This command returns whether Atheros Fast Frame support is enabled or disabled in the driver.

Example: The following command retrieves the status of Atheros Fast Frame support in the driver:

```
myprompt# iwpriv ath0 get_ff  
ath0      get_ff:1
```

xr - Atheros XR Support Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Disabled Resets State Machine After Command: No

This command enables or disables Atheros XR support in the driver. Passing a value of 1 to the driver enables XR support. Passing a value of 0 to the driver disables XR support.

Example: The following command enables Atheros XR support in the driver:

```
myprompt# iwpriv ath0 xr 1
```

get_xr - Get Atheros XR Support

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A Resets State Machine After Command: No

This command returns whether Atheros XR support is enabled or disabled in the driver.

Example: The following command retrieves the status of Atheros XR support in the driver:

```
myprompt# iwpriv ath0 get_xr
ath0      get_xr:1
```

burst - Atheros SuperA/G Bursting Support Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Enabled Resets State Machine After Command: No

This command enables or disables Atheros SuperA/G bursting support in the driver. Passing a value of 1 to the driver enables SuperG bursting. Passing a value of 0 to the driver disables SuperA/G bursting.

Example: The following command disables Atheros SuperA/G bursting in the driver:

```
myprompt# iwpriv ath0 burst 0
```

get_burst - Get Atheros SuperA/G Bursting Support

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A Resets State Machine After Command: No

This command returns whether Atheros SuperA/G bursting support is enabled or disabled in the driver.

Example: The following command retrieves the status of Atheros SuperA/G bursting support in the driver:

```
myprompt# iwpriv ath0 get_burst
ath0      get_burst:1
```

ar - Atheros SuperA/G Adaptive Radio (AR) Support Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Enabled Resets State Machine After Command: No

This command enables or disables Atheros SuperA/G Adaptive Radio (AR) support in the driver. Passing a value of 1 to the driver enables AR. Passing a value of 0 to the driver disables AR.

Example: The following command disables Atheros SuperA/G AR in the driver:

```
myprompt# iwpriv ath0 ar 0
```

get_ar - Get Atheros SuperA/G Adaptive Radio (AR) Support

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns whether Atheros SuperA/G AR support is enabled or disabled in the driver.

Example: The following command retrieves the status of Atheros SuperA/G AR support in the driver:

```
myprompt# iwpriv ath0 get_ar  
ath0      get_ar:1
```

compression - Atheros SuperA/G Compression Support Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Disabled
Resets State Machine After Command: No

This command enables or disables Atheros SuperA/G compression in the driver. Passing a value of 1 to the driver enables hardware compression. Passing a value of 0 to the driver disables hardware compression.

Example: The following command disables Atheros SuperA/G hardware compression in the driver:

```
myprompt# iwpriv ath0 compression 0
```

get_compression - Get Atheros SuperA/G Compression Support

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns whether Atheros SuperA/G hardware compression support is enabled or disabled in the driver.

Example: The following command retrieves the status of Atheros SuperA/G hardware compression support in the driver:

```
myprompt# iwpriv ath0 get_compression  
ath0      get_compression:0
```

abolt - Set ABOLT value

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Varies
Resets State Machine After Command: Yes

This command sets the abolt value used to control the Atheros proprietary features. This is a bitmask where each bit position corresponds to a feature. Setting the bit to 1 enables the feature if hardware is capable. Setting the bit to 0 disables the feature. The bitmask is described in Table 8.

Bit Position Feature

1	Static Turbo G (disabled)
2	Dynamic turbo
3	Compression
4	Fast Frames
5	Bursting
6	WMM based cwmin/cwmax/burst tuning
7	XR
8	AR

Table 8: Abolt Bit Position Definitions

pureg - Use Only 802.11g Data Rates (no legacy 802.11b support) Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Disabled Resets State Machine After Command: Yes

This command enables or disables 802.11g only operation (no legacy 802.11b rates are supported). Passing a value of 1 to the driver enables 802.11g rates only operation (rates below 6Mbps are disabled). Passing a value of 0 to the driver disables 802.11g only operation and allows legacy 802.11b rates to be supported.

Example: The following command enables 802.11g rates only operation. Thus, no rates below 6Mbps will be supported.

```
myprompt# iwpriv ath0 pureg 0
```

get_pureg - Get Status of 802.11g Only Data Rates Support

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A Resets State Machine After Command: No

This command returns whether the driver is using only 802.11g rates (no rates below 6Mbps).

Example: The following command returns whether the driver supports 802.11g rates only.

```
myprompt# iwpriv ath0 get_pureg  
ath0 get_pureg:0
```

wds - Enable/Disable 4 Address (WDS) Parsing

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: N/A
Resets State Machine After Command: No

This command enables or disables 4 address parsing on the device. For Stations, enabling 4 address parsing results in the station passing up any packet received in a 4 address from to the network layer. Also, any unicast packet not destined for the AP which is passed to the station by the network layer will be sent in 4 address mode. For APs, enabling 4 address parsing will result in the AP forwarding packets to any MAC address from which it has received a 4 address packet. Passing a value of 1 will enable 4 address parsing. Passing a value of 0 will disable 4 address parsing.

Example: The following command enables 4 address parsing on ath0:

```
myprompt# iwpriv ath0 wds 1
```

get_wds - Get Status of 4 Address (WDS) Parsing

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns whether 4 address parsing is enabled or disabled in the driver. A value of 1 indicates that 4 address parsing is enabled. A value of 0 indicates that 4 address parsing is disabled.

Example: The following command returns the stats of address parsing on ath0:

```
myprompt# iwpriv ath0 get_wds  
ath0      get_wds:0
```

countryie - Enable Country IE in Beacon Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value:
Disabled Resets State Machine After Command: Yes

This command enables and disables generation of country IE in beacon and probe response. To enable the support, a value of 1 is passed into the driver. To disable, a value of 0 is passed into the driver.

Example: The following command enables country ie in beacon on ath0:

```
myprompt# iwpriv ath0 countryie 1
```

get_countryie - Get Country IE Status

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns whether country IE is enabled or disabled in the driver.

Example: The following command retrieves the country IE status on ath0:

```
myprompt# iwpriv ath0 get_countryie  
ath0      get_countryie:0
```

coverageclass - Set Coverage Class for AP

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: 0
Resets State Machine After Command: Yes

This command sets the coverage class for AP. Coverage class determines the air propagation time used in BSS operation. The coverageclass value can be between 0 to 31. The coverageclass value is sent by AP via country IE element in beacon.

Example: The following command sets the coverage class to 12. on ath0:

```
myprompt# iwpriv ath0 coverageclass 12
```

get_coveragecls - Get Coverage Class Value

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command gets current coverage class value.

Example: The following command gets coverageclass value on ath0:

```
myprompt# iwpriv ath0 get_coveragecls  
ath0      get_coveragecls:12
```

regclass - Enable Regulatory class ids to be used in country IE in Beacon. Enable/Disable

Number of Input Arguments: 1 Number of Returned Arguments: 0 Default Value: Disabled
Resets State Machine After Command: Yes

This command enables advertising regclass ids in country IE in beacon instead of regular channel triplet (chan no/no.of channels/max transmit power). If set country does not have any regclass ids defined, it reverts back to regular triplet. The option needs to be enabled when using coverageclass. To enable the support, a value of 1 is passed into the driver. To disable, a value of 0 is passed into the driver.

Example: The following command enables country ie in beacon on ath0:

```
myprompt# iwpriv ath0 regclass 1
```

get_regclass - Get Regulatory Class ID Status

Number of Input Arguments: 0 Number of Returned Arguments: 1 Default Value: N/A
Resets State Machine After Command: No

This command returns whether regulatory class ids are getting advertised in country IE in beacon.

Example: The following command retrieves the country IE status on ath0:

```
myprompt# iwpriv ath0 get_regclass  
ath0      get_regclass:0
```

1.4 IOCTLs That Need Better Documentation

Not all private controls have been well documented. The following information is a work in progress. Please repair and add to it freely.

From `/net80211/ieee80211_ioctl.h` - NB: Even-numbered ioctl numbers have set semantics and are privileged!

ackrate - Set ACK Bitrate

Number of Input Arguments: 1
Number of Returned Arguments:
Default Value: 0
Resets State Machine After Command:

This command adds the option to set ack bitrate (hi vs default), it only works on 5212, 5213 devices.

Example:

```
sysctl -w dev.wifi0.ackrate=1 # send acks at high bit-rate  
sysctl -w dev.wifi0.ackrate=0 # send it at a basic-rate (default)
```

setkey

```
setkey      (8BF2) : set 60 byte & get 0  
ieee80211_ioctl_setkey(struct net_device *dev, struct iw_request_info *info, void *w, char *extra)  
Referenced in:  
ath/if_ath.c:2  
net80211/ieee80211_crypto_ccmp.c:4  
net80211/ieee80211_crypto.c:7  
net80211/ieee80211_node.c:2  
net80211/ieee80211_crypto_none.c:3  
net80211/ieee80211_wireless.c:8
```

net80211/ieee80211_crypto_wep.c:3
net80211/ieee80211_crypto_tkip.c:3
tools/athkey.c:9
tools/80211stats.c:6

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

delkey

delkey (8BF4) : set 7 byte & get 0
ieee80211_ioctl_delkey(struct net_device *dev, struct iw_request_info *info, void *w, char *extra)

Referenced in:

ath/if_ath.c:2
net80211/ieee80211_crypto.c:7
net80211/ieee80211_node.c:1
net80211/ieee80211_wireless.c:6
tools/athkey.c:4
tools/80211stats.c:1

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

setmlme

setmlme (8BF0) : set 42 byte & get 0

Referenced in:

net80211/ieee80211_wireless.c:5

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

kickmac

kickmac (8BFE) : set 1 addr & get 0

Referenced in:

net80211/ieee80211_wireless.c:3

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

wds_add

wds_add (8BFA) : set 1 addr & get 0

Referenced in:

net80211/ieee80211_node.c:12
net80211/ieee80211_wireless.c:1
net80211/ieee80211_input.c:3
net80211/ieee80211_output.c:1
net80211/ieee80211_proto.c:1

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

wds_del

wds_del (8BFC) : set 1 addr & get 0

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

setchanlist

setchanlist (8BE6) : set 32 byte & get 0

Referenced in:

net80211/ieee80211_wireless.c:3

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

getchanlist

getchanlist (8BE7) : set 0 & get 32 byte

Referenced in:

net80211/ieee80211_wireless.c:3

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

getchaninfo

getchaninfo (8BED) : set 0 & get 2044 byte

Referenced in:

net80211/ieee80211_wireless.c:3

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

setwmmparams

setwmmparams (8BE4) : set 4 int & get 0

Referenced in:

net80211/ieee80211_wireless.c:3

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

getwmmparams

getwmmparams (8BE5) : set 3 int & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:3

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

setparam

setparam (8BE0) : set 2 int & get 0

Referenced in:

net80211/ieee80211_wireless.c:13

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

getparam

getparam (8BE1) : set 1 int & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:12

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

authmode - authentication mode

authmode (0003) : set 1 int & get 0

Referenced in:

ath/if_ath.c:1

net80211/ieee80211_node.c:7

net80211/ieee80211_wireless.c:5

net80211/ieee80211_input.c:6

net80211/ieee80211_output.c:2

net80211/ieee80211_proto.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_authmode

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

mcastkeylen

mcastkeylen (0006) : set 1 int & get 0

Referenced in:

net80211/ieee80211_node.c:1
net80211/ieee80211_wireless.c:4
net80211/ieee80211_input.c:3
net80211/ieee80211_output.c:2

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

get_mcastkeylen

get_mcastkeylen (0006) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1
Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

get_uciphers

get_uciphers (0007) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1
Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

hostroaming - roaming mode ??

hostroaming (000C) : set 1 int & get 0

Referenced in:

net80211/ieee80211_wireless.c:2

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_hostroaming

get_hostroaming (000C) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

driver_caps - driver capabilities

driver_caps (0010) : set 1 int & get 0

Referenced in:

net80211/ieee80211_wireless.c:2

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

abolt

abolt (001A) : set 1 int & get 0

Referenced in:

net80211/ieee80211_wireless.c:3

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_abolt - Atheros Adv. Capabilities

get_abolt (001A) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

turbo- turbo mode

turbo (0001) : set 1 int & get 0

Referenced in:

ath/if_ath.c:20

ath_rate/sample/sample.c:1

net80211/ieee80211.c:2

net80211/ieee80211_node.c:4

net80211/ieee80211_beacon.c:1

net80211/ieee80211_scan_sta.c:7

net80211/ieee80211_wireless.c:8

net80211/ieee80211_input.c:4

net80211/ieee80211_scan_ap.c:6

net80211/ieee80211_proto.c:3

tools/80211debug.c:1

tools/athdebug.c:2

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_turbo

get_turbo (0001) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

doth_chanswitch

doth_chanswitch (8BE8) : set 2 int & get 0

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

bgscan - bg scanning (on, off)

bgscan (0028) : set 1 int & get 0

bgscan - Enable or disable background scanning. By default it is on so it can be turned off by setting it to 0.

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_bgscan

get_bgscan (0028) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

bgscanidle - bg scan idle threshold

bgscanidle (0029) : set 1 int & get 0

bgscanidle - How often is background scanning done. Measured in milliseconds.

By default it is set to 250. The minimum is 100.

This measures how long the medium must be idle for before it begins a background scan.

The idle time is probably based upon unicast traffic sent/received by this node. Background scan tells the AP that it's going to sleep for a specified duration while it does the scan... so no inbound traffic is lost. The issue is really one of user experience.

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_bgscanidle

get_bgscanidle (0029) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

bgscanintvl - bg scan interval

bgscanintvl (002A) : set 1 int & get 0

bgscanintvl - How often a background scan is performed. Measured in seconds. By default it is set to 300s or 5 mins. The minimum is 15 seconds.

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_bgscanintvl

get_bgscanintvl (002A) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1
Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

mcast_rate -Multicast Tx Rate

mcast_rate (002B) : set 1 int & get 0
Referenced in:
net80211/ieee80211_wireless.c:1
Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

get_mcast_rate

get_mcast_rate (002B) : set 0 & get 1 int
Referenced in:
net80211/ieee80211_wireless.c:1
Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

scanvalid = scan cache valid threshold

scanvalid (002E) : set 1 int & get 0
> scanvalid - Scan cache valid threshold. By default it is 60. Not really
> sure what this does.

scanvalid - How long (seconds) are the scan cache contents regarded valid.
If the last scan is too old the cache is refreshed with a new scan before an AP is selected, e.g. after losing AP association (beacon miss) or manually with 'iwconfig ath0 ap auto'. Preventive roaming in sta_roam_check() also uses scanvalid. Any single scan will not see all the nodes because of the brief window of passive scanning for some DFS channels and intermittent hidden node problems.

Referenced in:
net80211/ieee80211_wireless.c:1
Number of Input Arguments:

Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

get_scanvalid

get_scanvalid (002E) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

rss11a - rssi threshold in 11a

rss11a (002F) : set 1 int & get 0

Referenced in:

net80211/ieee80211_scan.c:1

net80211/ieee80211_scan_sta.c:1

net80211/ieee80211_wireless.c:3

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_rssi11a

get_rssi11a (002F) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

rss11b - rssi threshold in 11b

rss11b (0030) : set 1 int & get 0
Referenced in:
net80211/ieee80211_scan.c:1
net80211/ieee80211_scan_sta.c:1
net80211/ieee80211_wireless.c:3
Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

get_rssi11b - rssi threshold in 11g

get_rssi11b (0030) : set 0 & get 1 int
Referenced in:
net80211/ieee80211_wireless.c:1
Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

rssi11g

rssi11g (0031) : set 1 int & get 0
Referenced in:
net80211/ieee80211_wireless.c:1
Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

get_rssi11g

get_rssi11g (0031) : set 0 & get 1 int
Referenced in:
net80211/ieee80211_wireless.c:1
Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

rate11a - tx rate threshold in 11a

rate11a (0032) : set 1 int & get 0

Referenced in:

net80211/ieee80211_scan.c:1

net80211/ieee80211_scan_sta.c:1

net80211/ieee80211_wireless.c:3

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_rate11a

get_rate11a (0032) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

rate11b - tx rate threshold in 11b

rate11b (0033) : set 1 int & get 0

Referenced in:

net80211/ieee80211_scan.c:1

net80211/ieee80211_scan_sta.c:1

net80211/ieee80211_wireless.c:3

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_rate11b

get_rate11b (0033) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:
Number of Returned Arguments:
Default Value:
Resets State Machine After Command:

Example:

rate11g - tx rate threshold in 11g

rate11g (0034) : set 1 int & get 0

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_rate11g

get_rate11g (0034) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

uapsd

Unscheduled Automatic Power Save Delivery (UAPSD) is a strategy for the delivery of downlink buffered frames in the AP to the appropriate station without the need for PS Polls to indicate that the PS station is awake and ready to receive transmitted frames. Unscheduled Automatic Power Save Delivery defaults to on if the hardware is capable and we have sufficient hardware queues to do proper priority scheduling (per comments in if_ath.c). The UAPSD queing and state can be reported by ath_debug. While there is extensive logic to implement UAPSD in if_ath.c, I haven't figured out what the uapsd ioctl does explicitly.

uapsd (0035) : set 1 int & get 0 Referenced in: ath/if_ath.c:1
net80211/ieee80211_wireless.c:1 tools/athdebug.c:2

Number of Input Arguments: Number of Returned Arguments: Default Value: Resets State Machine After Command:

Example:

```
}}}
```

get_uapsd

get_uapsd (0035) : set 0 & get 1 int

Unscheduled Automatic Power Save Delivery (APSD) is a strategy for the delivery of downlink buffered frames in the AP to the appropriate station without the need for PS Polls to indicate that the PS station is awake and ready to receive transmitted frames.

This ioctl sets IEEE80211_PARAM_UAPSDINFO, which is used in ieee80211_wireless.c as follows:

```
case IEEE80211_PARAM_UAPSDINFO:
    if (vap->iv_opmode == IEEE80211_M_HOSTAP) {
        if (ic->ic_caps & IEEE80211_C_UAPSD) {
            if (value)
                IEEE80211_VAP_UAPSD_ENABLE(vap);
            else
                IEEE80211_VAP_UAPSD_DISABLE(vap);
            retv = ENETRESET;
        }
    } else if (vap->iv_opmode == IEEE80211_M_STA) {
        vap->iv_uapsdinfo = value;
        IEEE80211_VAP_UAPSD_ENABLE(vap);
        retv = ENETRESET;
    }
}
```

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

sleep - force sleep/wake

sleep (0036) : set 1 int & get 0

This ioctl sets IEEE80211_PARAM_SLEEP and forces sleep for testing. It does not actually place the HW in sleep mode yet. This only makes sense for STAs.

```
if (value) {
    /* goto sleep */
    IEEE80211_VAP_GOTOSLEEP(vap);
} else {
```

```
        /* wakeup */
        IEEE80211_VAP_WAKEUP(vap);
    }
```

Referenced in:

ath/if_ath.c:9

net80211/ieee80211_wireless.c:4

net80211/ieee80211_power.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_sleep

get_sleep (0036) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

qosnull

qosnull (0037) : set 1 int & get 0

This var sets IEEE80211_PARAM_QOSNULL and forces a QoS Null for testing.

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

pspoll - force ps-poll generation (sta only)

pspoll (0038) : set 1 int & get 0

This var sets IEEE80211_PARAM_PSPOLL and forces a PS-POLL for testing.

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

eospdrop - force uapsd EOSP drop (ap only)

eospdrop (0039) : set 1 int & get 0

Unscheduled Automatic Power Save Delivery (APSD) is a strategy for the delivery of downlink buffered frames in the AP to the appropriate station without the need for PS Polls to indicate that the PS station is awake and ready to receive transmitted frames.

The AP sets the EOSP (end of service period) bit to 1 in the last frame it transmits in order to signal to the station that it will not transmit any more frames downlink until the next service period. This signals the station that it can go back to sleep.

See " Delivery of buffered frames to power saving stations in wireless local area networks",
<http://64.233.167.104/search?q=cache:ls3gYzROElkJ:www.freepatentsonline.com/20050213534.html+EOSP+frame&hl=en&ct=clnk&cd=2&gl=us> for more discussion of power saving.

Referenced in:

net80211/ieee80211_wireless.c:1

Ticket #900: ath-mib-irq-status-debug.diff

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_eospdrop

get_eospdrop (0039) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

markdfs - mark a dfs interference channel when found

markdfs (003A) : set 1 int & get 0

Dynamic Frequency Selection (DFS) is a requirement for IEEE 802.11h for 5 GHz outdoor use in ETSI regulation domain. DFS ensures that channels containing radar are avoided by an Access Point (AP) and energy is spread across the band to reduce interference to satellites.

Referenced in:

Changeset 2214 <http://madwifi.org/changeset/2214> Strip out old DFS processing code due to HAL API changes, HAL 0.9.20.3 has no procdfs method. The dfstest timer has been removed. Still wait before transmitting on a channel to see if the channel is clear.

Ticket [#678](#) Radar Detection does not work. Apparently, the presence of a radar signal should invoke `ath_hal_radar_event`, but as of 5/13/07, it does not (mike.taylor@apprion.com indicates that he's currently working on a fix). The PHY chip detected radar perfectly when manipulated directly, but none of the radar related HAL functions seemed to do anything. To get interrupted after a radar pulse: * enable `HAL_INT_RXPHY` in the interrupt mask (via HAL), see `ar5212reg.h` from `openhal`.

net80211/ieee80211.c:1

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Example:

get_markdfs

get_markdfs (003A) : set 0 & get 1 int

Referenced in:

net80211/ieee80211_wireless.c:1

Number of Input Arguments:

Number of Returned Arguments:

Default Value:

Resets State Machine After Command:

Chapter 2

Common Configuration Examples using Wireless Extensions

In this section, we present common configurations for both AP and stations supported by the [MadWifi](#) driver. We assume the driver and all necessary modules have already been loaded. The underlying wireless device is assumed to be `wifi0` unless otherwise noted. The ethernet device is assumed to be `eth0`.

Single AP on a Preselected Channel

In this section, we give an example on how to configure a single [MadWifi](#) AP in 802.11a on channel 36 with ESSID "Åtheros Wireless Network". The desired IP address for the AP is 192.168.0.20.

Example:

```
myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap ath0
myprompt# iwconfig ath0 essid "Åtheros Wireless Network" channel 36
myprompt# brctl addbr br0
myprompt# brctl addif br0 eth0
myprompt# brctl addif ath0
myprompt# brctl setfd br0 1
myprompt# ifconfig ath0 up
myprompt# ifconfig eth0 up
myprompt# ifconfig br0 192.168.0.20 up
```

Single AP with hostapd on an Automatically Chosen Channel

In this example, we configure a single [MadWifi](#) AP in 802.11g using the auto channel select. The AP will use WPA-PSK via hostapd. The user space program hostapd requires a configuration file. The AP will have an IP address of 192.168.0.20.

Example: The configuration file (named `/etc/hostapd.conf`) is shown below.

```
interface=ath0 bridge=br0
driver=madwifi
logger_syslog=0
logger_syslog_level=0
logger_stdout=0
logger_stdout_level=0
debug=0
eapol_key_index_workaround=0
dump_file=/tmp/hostapd.dump.0.0
ssid="Åtheros Wireless Network"
wpa=1
wpa_passphrase=mypassphrase
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
```

```
wpa_group_rekey=600
```

Now, the following commands will create the AP.

```
myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
ath0
myprompt# iwconfig ath0 essid "Atheros Wireless Network"
myprompt# iwpriv ath0 mode 11g
myprompt# brctl addbr br0
myprompt# brctl addif br0 eth0
myprompt# brctl addif ath0
myprompt# brctl setfd br0 1
myprompt# ifconfig ath0 up
myprompt# ifconfig eth0 up
myprompt# ifconfig br0 192.168.0.20 up
myprompt# hostapd -dd /etc/hostapd.conf
```

WPA-PSK Station Using wpa_supplicant

In this example, we will configure the driver to be a station attempting to associate with the WPA-PSK AP in the example above. The station will have an IP address of 192.168.0.100.

Example: The user space program wpa_supplicant requires a configuration file. The file used in this example is shown below and named /tmp/my_psk.conf.

```
network={
    ssid="Atheros Wireless Network"
    scan_ssid=1
    key_mgmt=WPA-PSK
    psk="mypassphrase"
}
```

Now, the following commands will create the station which will scan for the AP with an SSID of Atheros Wireless Network.

```
myprompt# wlanconfig ath create wlandev wifi0 wlanmode sta
ath0
myprompt# iwconfig ath0 essid "Atheros Wireless Network"
myprompt# ifconfig ath0 192.168.0.100 up
myprompt# wpa_supplicant -iath0 -c /tmp/my_psk.conf -d
```

Three APs on a Preselected Channel

In this section, we give an example on how to configure a three [MadWifi](#) APs in 802.11a on channel 36 with ESSIDs "Atheros_AP1", "Atheros_AP2", and "Atheros_AP3". All three APs are bridged together. The desired IP address for the AP is 192.168.0.20.

```
Example: myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
ath0
```

```

myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
ath1
myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
ath2
myprompt# iwconfig ath0 essid "Atheros_AP1" channel 36
myprompt# iwconfig ath1 essid "Atheros_AP2"
myprompt# iwconfig ath2 essid "Atheros_AP3"
myprompt# brctl addbr br0
myprompt# brctl addif br0 eth0
myprompt# brctl addif br0 ath0
myprompt# brctl addif br0 ath1
myprompt# brctl addif br0 ath2
myprompt# brctl setfd br0 1
myprompt# ifconfig ath0 up
myprompt# ifconfig ath1 up
myprompt# ifconfig ath2 up
myprompt# ifconfig eth0 up
myprompt# ifconfig br0 192.168.0.20 up

```

Single Wireless Device AP Repeater

In this section, we give an example on how to configure [MadWifi](#) as a repeater operating with a single wireless device (e.g., wifi0). We assume there is an existing AP with an SSID of Atheros_Base_AP. We wish to "repeat" this AP using our device. To do this, we will create two VAPs. The first VAP will be an AP VAP which will serve all the clients in our range. The second VAP will be a station VAP used to association with the existing AP named Atheros_Base_AP. In order to have both a station and AP VAP coexist on one base device (e.g., wifi0), the AP VAP must be created first followed by the station VAP with the nosbeacon option selected for the station VAP. However, the station VAP must be brought up first and allowed to associate since it will choose the channel of the existing AP (Atheros_Base_AP). In this example, our AP will not be able to "receive" IP traffic because no IP address will be assigned to the bridge device. However, by assigning an IP address, the AP can also receive traffic (using 3 address frames). To support the 4 address frame format needed for repeating, the wds option must be enabled. It is assumed the existing AP understands how to handle 4 address frames.

```

Example: myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
ath0
myprompt# wlanconfig ath create wlandev wifi0 wlanmode sta nosbeacon
ath1
myprompt# iwconfig ath0 essid "Atheros_Base_AP"
myprompt# iwconfig ath1 essid "Atheros_Base_AP"
myprompt# iwpriv ath0 wds 1
myprompt# iwpriv ath1 wds 1
myprompt# ifconfig ath1 up
Now, wait for association.
myprompt# ifconfig ath0 up
myprompt# ifconfig eth0 up
myprompt# brctl addbr br0
myprompt# brctl addif br0 eth0
myprompt# brctl addif br0 ath0

```

```
myprompt# brctl addif ath1
myprompt# brctl setfd br0 1
myprompt# ifconfig br0 up
```

Dual Wireless Device AP Repeater

In this section, we give an example on how to configure [MadWifi](#) as a "repeater" operating with dual wireless devices (e.g., wifi0 and wifi1). We assume there is an existing AP with an SSID of Atheros_Base_AP in the 802.11a band. We wish to "repeat" this AP using our device but will use a different SSID (Atheros_Repeater_AP) for distribution in our coverage area. Furthermore, the local distribution will be in the 802.11g band, not the 802.11a band. Thus, we are assuming that wireless device wifi0 is capable of operating in the 802.11a band, and wifi1 is capable of operating in the 802.11b band. To do this, we will create two VAPs. The VAP associated with wifi0 will be a station VAP and the VAP associated with wifi1 will be the AP VAP. Note, the order of creation does not matter now since the VAPs are on different wireless devices. The AP VAP must serve all the clients in our range and forward their traffic to the base AP via 4 address format. In this example, our AP will also be able to "receive" IP traffic and will have the IP address of 192.168.0.20.

```
Example: myprompt# wlanconfig ath create wlandev wifi0 wlanmode sta
ath0
myprompt# wlanconfig ath create wlandev wifi1 wlanmode ap
ath1
myprompt# iwconfig ath0 essid "Atheros_Base_AP"
myprompt# iwconfig ath1 essid "Atheros_Repeater_AP"
myprompt# iwpriv ath0 wds 1
myprompt# iwpriv ath1 wds 1
myprompt# ifconfig ath1 up
myprompt# ifconfig ath0 up
myprompt# ifconfig eth0 up
myprompt# brctl addbr br0
myprompt# brctl addif br0 eth0
myprompt# brctl addif ath0
myprompt# brctl addif ath1
myprompt# brctl setfd br0 1
myprompt# ifconfig br0 192.168.0.20 up
```

Base AP Which Understands WDS (4 Address) Frames

In this section, we give an example of how to configure [MadWifi](#) to be a base AP which handles 4 address 802.11 frames. The AP will use Auto channel selection in the 802.11a band and will have an ssid of Atheros_Base_AP. The AP will have an IP address of 192.168.0.20. The underlying wireless device is assumed to be wifi0.

```
Example: myprompt# wlanconfig ath create wlandev wifi0 wlanmode ap
ath0
myprompt# iwconfig ath0 essid "Atheros_Base_AP"
myprompt# iwpriv ath0 wds 1
myprompt# ifconfig ath0 up
```

```
myprompt# ifconfig eth0 up
myprompt# brctl addbr br0
myprompt# brctl addif br0 eth0
myprompt# brctl addif ath0
myprompt# brctl setfd br0 1
myprompt# ifconfig br0 192.168.0.20 up
```