Madwifi/Atheros Wireless Linux Driver Users Guide

Protocols Group

June 2, 2006

Contents

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 1.3.

***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, perpend "`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* [`bssid`||`-bssid`] [`nosbeacon`]
`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

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

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.

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. When AP and station VAPs coexist, the `nosbeacon` flag must be used when creating the station. This flag disables the use of hardware beacon timers for station mode operation. This is necessary because 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. However, if the `-bssid` flag is used, then the MAC address of the underlying wireless device is cloned for the VAP being created.

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, not just the device 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 the 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`. Both devices will have a the same MAC address cloned from the underlying device. The first VAP will be `ath0` and the second VAP will be `ath1`.

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

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

***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 commands 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
```

`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 commands 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 commands 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 commands 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 commands 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 commands 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 commands 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_pwrtgt` - 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_pwrtgt 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.

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

```
myprompt# iwpriv ath0 wpa 3
```

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

Table 3: WPA/WPA2 Support Arguments

`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
```

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

Table 4: Key Management Algorithms

`keymgtalgs` - 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 used by `hostapd` and `wpa_supplicant`.

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

```
myprompt# iwpriv ath0 keymgtalgs 2
```

`get_keymgtalgs` - 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_keymgtalgs
ath0      get_keymgtalgs: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_rsncaps` - 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_rsncaps
ath0      get_rsncaps: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_dropunencry
ath0      get_dropunencry: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 CW_{min} Parameter

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

This command sets the CW_{min} 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 CW_{min} value is intended for the AP or station parameter set. A value of 0 indicates the CW_{min} is for the AP parameter set. A value of 1 indicates the CW_{min} is for the station parameter set. The third value is the actual value of the CW_{min} 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 CW_{min} in the station parameter set for the VO AC to 2.

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

`get_cwmin` - Get WMM CW_{min} 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 CW_{min} 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 CW_{min} is from the AP parameter set. A value of 1 indicates the CW_{min} is from the station parameter set.

***Example*:** The following command gets the CW_{min} in the AP parameter set for the VI AC.

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

`cwmax` - WMM CW_{max} Parameter

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

This command sets the CW_{max} 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 CW_{max} value is intended for the AP or station parameter set. A value of 0 indicates the CW_{max} is for the AP parameter set. A value of 1 indicates the CW_{max} is for the station parameter set. The third value is the actual value of the CW_{max} in units as described in the WMM standard.

***Example*:** The following command sets the CW_{max} in the AP parameter set for the BK AC to 5.

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

`get_cwmax` - Get WMM CW_{max} 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 CW_{max} 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 CW_{max} is from the AP parameter set. A value of 1 indicates the CW_{max} is from the station parameter set.

***Example*:** The following command gets the CW_{max} 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.

```
myprompt# 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 commands 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 commands 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
```

2 Configuring AP using CLI

This section describes the configuration of Atheros Linux AP using CLI command sets.

2.1 CLI commands

This section lists all the supported CLI commands. Throughout this document, CLI commands will be denoted using bold, computer font as in **`command`**. Parameters to the command will be denoted by non-bold computer font as in `parameter`. Any values which are needed by the parameters will be denoted using italic Times font as in *value*.

Switching the WLAN and BSS

`config` `wlan` *wlan_index* `bss` *bss_index*
`config` `wlan` *wlan_index*
`config` `bss` *bss_index*

Use **`config`** command to switch current WLAN (radio) or virtual BSS. Except for the command **`commit`**, all other CLI commands are only effective on the current WLAN or BSS.

The shell prompt reflects the current WLAN and BSS in the following form.

```
wlan[wlan_index,bss_index] ->
```

or

```
wlan[wlan_index] ->
```

If only one index is in the prompt, it could be either that there is no BSS created for this WLAN, or there is no BSS currently selected. To select a BSS within the same WLAN, use `config bss bss_index`.

Reading the AP Configuration

get [parameter]

Use **get** command to display the current AP configurations. For description of each available parameter, please refer to section 2.2.

Modifying the AP Configuration

set parameter [*value*]

Use **set** command to modify the current AP configuration. For description of each available parameter, please refer to section 2.2.

Adding a BSS

add <bss|sta> [*bss_index*]

where *bss_index* is a value from 0-3.

Use the **add** command to add a new virtual BSS (bss) or a wireless client (sta) to the current WLAN. If the *bss_index* is not specified, the new BSS will created using the first available index.

The maximum number of BSS's is 4.

Deleting a BSS

del bss *bss_index*
del bss *all*

in which, *bss_index* is a value from 0-3.

Use the **del** command to delete a virtual BSS. The*bss_index* must be specified to indicate which BSS to delete. If the BSS value *all* is specified, all the BSS's on the current WLAN will be deleted.

Saving the AP Configuration

commit

Use **commit** to save the current AP configuration. The saved configuration will take effect on the next AP reboot.

Getting Help

help command

Use **help** to display the usage of each command. For the get and set commands, the supported configuration parameters are displayed.

The ap_service Script

ap_service <*start*|*stop*|*restart*>

The **ap_service** script is used to start, stop and restart the Linux AP service without rebooting the entire AP. Besides configuring WLAN using CLI commands, it's also responsible for configuring network functions such as bridging and VLAN. It's located in directory /etc/wlan.

***Example*:** In system startup script (by default /etc/rc.d/rcS), **ap_service** can be used to start the AP service.

```
/etc/wlan/ap_service start
```

***Example*:** After using **`commit`** to save the configurations, instead of rebooting the entire AP, **`ap_service`** can be used to restart just the AP service.

```
wlan[0,0]-> commit
wlan[0,0]-> .  /etc/wlan/ap_service restart
```

2.2 AP Configuration Parameters

This section describes the available AP configuration parameters. All parameters will be saved by the **`commit`** command and take effect upon next AP reboot.

ACL List

`get` `acl`
`set` `acl` *acl_mode*
`add` `acl` *mac_address*
`del` `acl` *mac_address*

The supported ACL modes and their actions are listed in Table 9.

ACL Mode	Actions
Open	No ACL checking is performed
Allow	Only allow ACLs in the ACL list
Deny	Only deny ACLs in the ACL list
Flush	Flush the ACL database
Disable	Remove the ACL policy

Table 9: Supported ACL Modes

***Example*:** Set ACL mode to allow, and add one MAC address to the ACL list.

```
wlan[0,0]-> set acl allow
wlan[0,0]-> add acl 00:03:7f:03:42:3f
```

The command `get acl` will display the current ACL mode and the MAC addresses in the list.

Authentication Type

`get` `authentication`
`set` `authentication` *authmode*

The supported authentication types are listed in Table 10.

Auto Channel Select

`get` `autochannelselect`
`set` `autochannelselect` <*Enable*|*Disable*>

If `autochannelselect` is enabled, AP will scan and select the best channel available. If it's disabled, a channel must be explicitly set by the user.

Authentication Type	Description
open	Open-System authentication type
shared	Shared-Key authentication type
auto	Allow Open-System or Shared Key for authentication type
802.1x	802.1x authentication type
WPA	WPAv1 authentication type
WPA-PSK	WPAv1-PSK authentication type
WPA2	WPAv2 authentication type
WPA2-PSK	WPAv2-PSK authentication type
WPA-AUTO	Allow WPAv1 or WPAv2 for authentication type
WPA-AUTO-PSK	Allow WPAv1-PSK or WPAv2-PSK for authentication type

Table 10: Supported Authentication Types

Radio Channel

get channel
set channel <*IEEE channel*|*Frequency*>

If autochannelselect is enabled, get channel will display Auto. Otherwise, get channel will display the current channel or frequency, depending on the value user previously set.

The command set channel can accept both channel numbers and frequency values. When channel is explicitly set, autochannelselect is automatically turned off.

Cipher Suite

get cipher
set cipher *cipher_suite*

The supported *cipher_suite*s are *Auto*, *WEP*, *TKIP* and *AES*.

Display Configuration

get config

This command will display the all the AP configurations.

Country Code

get countrycode
set countrycode *CC*

The *CC* is the two-letter abbreviation of a country.

Enable/Disable Encryption

get encryption
set encryption <*Enable*|*Disable*>

Unless encryption is enabled, all the security settings (authentication, cipher, key, and etc.) will not take any effect.

Restore Default Configuration

set factory

This command will restore AP to the default factory settings.

Group Key Update Interval

get groupkeyupdate
set groupkeyupdate *interval*

The *interval* is in seconds. This parameter is valid only when authentication is one of WPA, WPA-PSK, WPA2, WPA2-PSK, WPA-AUTO and WPA-AUTO-PSK.

IP Address and Subnet Mask

set ipaddr *ip_address*
set ipmask *ip_mask*

These two commands set the IP address and subnet mask of the AP.

Static WEP Key

set key *<1-4> <40|104|124> key_material*
set key *<1-4> default*

In which, *<1-4>* is the valid key index and *<40|104|124>* is the valid key length. Before setting the default key, user must set the key at the index first. For example, set key 1 40 1234567890 must precede set key 1 default.

The set key command is only valid when AP is using static WEP, i.e., authentication is either open or shared.

Operating Mode

get opmode
set opmode *mode*

The supported modes are listed in Table 11.

Operation Mode	Description
AP	Operating as Wireless Access Point
STA	Operating as Wireless Client

Table 11: Supported Operation Mode

Set WPA Passphrase

set passphrase
<ascii_passphrase|hex_key>

The set passphrase command is only valid when authentication is WPA-PSK, WPA2-PSK, or WPA-AUTO-PSK. The passphrase could be either an ASCII passphrase of length from 8 to 63 characters, or a 256-bit key in hexadecimal form.

Set Power

set power *value*
set power *<full|min>*

This commands set the Tx Power of the current radio. The tx power can be set to a specific *value* in dB, or to the maximum or minimum by *full* or *min*.

Port VLAN ID

get pvid
set pvid *vlan_id*

This command set the default port VLAN Id. When vlan is enable but pvid is not set for a BSS. This BSS will be added to the guest VLAN. Traffic from guest VLAN will be propagated to distribution system untagged.

RADIUS Server Configuration

get radiusname
set radiusname *radius_ip*
get radiusport
set radiusport *port*
set radiussecret *secret*

These commands are used to specify the RADIUS server tow which the AP is talking. The parameters radiusname and radiusport are used to specify the IP address and port number of the RADIUS server. The parameter radiussecret is used to specify the shared secret between AP and RADIUS server.

Data Rate

get rate
set rate *value*
set rate *best*

To display or set the data rate, in MB/sec. When *best* is specified, it's up to the AP to select the best data rate.

Repeater

get repeater
set repeater <*Enable*|*Disable*>
set repeater *ssid ssid-text*
set repeater *remote-ssid ssid-text*
set repeater *param value*

The command set repeater enable works as follows: it first deletes all the old VAPs on this WLAN, then creates a STA VAP and an AP VAP. The wds mode is enabled implicitly on the STA VAP.

To let the STA VAP associate to the remote AP, use the command set repeater remote-ssid. To set the SSID of repeater itself, use command set repeater ssid instead.

For other parameters, you can use command set repeater param value, in which param and *value* are exactly the same as in a normal CLI command. But with repeater before them, every parameter is actually set twice, once on STA, once on AP.

The command set repeater disable will just delete all VAPs.

For an example of how to set up a Linux repeater, please refer to section 4.1

The Service Set ID

get ssid
set ssid *ssid-text*

The *ssid-text* can be no longer than 32 characters.

Association Table

get sta

This command will display the association table.

Enable/Disable VLAN

get `vlan`
set `vlan` <*Enable*|*Disable*>

Enable or disable `vlan`.

WLAN State

get `wlanstate`
set `wlanstate` <*Enable*|*Disable*>

Enable or disable a BSS. Note `wlanstate` indicates whether a BSS will be up or down on the NEXT reboot, not the current status! To make `wlanstate` take effect, a reboot is still required.

Wireless Modes

get `wirelessmode`
set `wirelessmode` *mode*

The valid wireless modes are listed in Table 12. auto, 11a, 11b, 11g, turbo dynamic (11a with dynamic turbo), turbo static (11a with static turbo) and 108g dynamic (11g with dynamic turbo).

Wireless Mode	Description
auto	Auto select wireless mode
11a	802.11a wireless mode (5GHz, 54Mbps)
11b	802.11b wireless mode (2.4GHz, 11Mbps)
11g	802.11g wireless mode (2.4GHz, 54Mbps) with 802.11b compatibility
turbo dynamic	802.11a Dynamic Turbo mode (5GHz, 108Mbps)
turbo static	802.11a Static Turbo mode (5GHz, 108Mbps)
108g dynamic	802.11g Dynamic Turbo mode (2.4GHz, 108Mbps)

Table 12: Supported Wireless Modes

Enable/Disable WMM

get `wmm`
set `wmm` <*Enable*|*Disable*>

Turn on and off WMM.

Enable/Disable WDS

get `wds`
set `wds` <*Enable*|*Disable*>

Turn on and off WDS. For AP acting as a Linux repeater or remote AP, `WDS` must be set.

Coverage Class

get `coverageclass`
set `coverageclass` *value*

Set the coverage class.

Enable/Disable XR

get xr
set xr <*Enable*|*Disable*>

Turn on and off XR.

3 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`.

3.1 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 "Atheros 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 "Atheros 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="Atheros 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
```

3.2 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 ath0
myprompt# brctl addif ath1
```

```
myprompt# brctl addif 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
```

3.3 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 ath0`
`myprompt# brctl addif ath1`
`myprompt# brctl setfd br0 1`
`myprompt# ifconfig br0 up`

3.4 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
```

3.5 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
```

4 CLI Configuration Examples

In this section, we present some special AP configurations. We assume CLI is used and underlying WLAN to be 0 unless otherwise noted. The ethernet device is assumed to be `eth0`.

4.1 Linux Repeater

In this section, we give an example how to configure the Linux AP as a wireless repeater. Shared-key authentication and static WEP are used in this example.

To setup and test repeater, we must have two Linux APs, one as the remote AP, one as the repeater.

Configuring Remote AP

The remote AP should be set up as usual, but with `wds` mode enabled.

***Example*:** Set up remote AP with `wds` enabled, and with shared-key authentication.

```
wlan[0,0]-> set ssid remote-ap-ssid
wlan[0,0]-> set wds enabled
```

```
wlan[0,0]-> set encryption enable
wlan[0,0]-> set authentication shared
wlan[0,0]-> set key 1 40 1234567890
wlan[0,0]-> set key 1 default
wlan[0,0]-> commit
wlan[0,0]-> .  /etc/wlan/ap_service restart # or just reboot
```

Configuring Repeater

Use the `set repeater` command set. For a detailed explanation of this command, please refer to section 2.2.

***Example*:**

```
wlan[0,0]-> set repeater enable
wlan[0,0]-> set repeater remote-ap-ssid
wlan[0,0]-> set repeater ssid repeater-ssid
wlan[0,0]-> set repeater encryption enable
wlan[0,0]-> set repeater authentication shared
wlan[0,0]-> set repeater key 1 40 1234567890
wlan[0,0]-> set repeater key 1 default
wlan[0,0]-> commit
wlan[0,0]-> .  /etc/wlan/ap_service restart # or just reboot
```

Some additional repeater configurations can't be set by using CLI alone.

***Example*:** If repeater and remote AP are on the same wired LAN, you probably don't want to have `eth0` in your repeater bridge. In this case, you can do:

```
wlan[0,0]-> brctl delif br0 eth0
```

***Example*:** If you want to test multicast (you don't have to anything if only testing broadcast), you need to enable `ALLMULTI` flag on STA.

```
wlan[0,0]-> config bss 0
wlan[0,0]-> ifconfig 'get interface' allmulti
```

4.2 Linux P2P/P2MP Bridge

Currently there is no specific mode called "WBR". The P2P or P2MP bridge is configured using standard STA and AP, partly leveraging Linux native bridging facilities. The basic idea is to set up on AP as root AP, and set up other APs as STA or "wireless client mode". The let every AP in client mode associate to root AP. The WLAN driver will make forwarding decisions on the wireless media, and Linux native bridging layer will make forwarding decisions on the ethernet side.

Configuring the Root AP

The remote AP should be set up as usual, but with `wds` mode enabled.

***Example*:** Set up remote AP with `wds` enabled, and with shared-key authentication.

```
wlan[0,0]-> set ssid root-ap-ssid
wlan[0,0]-> set wds enabled
wlan[0,0]-> commit
wlan[0,0]-> .  /etc/wlan/ap_service restart # or just reboot
```

Configuring Wireless Client

A wireless client is just a STA VAP with `wds` enabled.

***Example*:**

```
wlan[0,0]-> del bss all
wlan0-> add sta
wlan0-> config bss 0
wlan[0,0]-> set ssid root-ap-ssid
wlan[0,0]-> set wds enable
wlan[0,0]-> commit
wlan[0,0]-> .  /etc/wlan/ap_service restart # or just reboot
```

Some additional configurations on the wireless client can't be set by using CLI alone.

***Example*:** If you want to test multicast (you don't have to anything if only testing broadcast), you need to enable `ALLMULTI` flag on the wireless client.

```
wlan[0,0]-> config bss 0
wlan[0,0]-> ifconfig ‘get interface‘ allmulti
```

After wireless client is up, use `get sta` command to check for association. Once it's associated, the P2P bridge is up and running.

To configure P2MP, just set up another wireless client and repeat the example above.