NAME

ifconfig - configure network interface parameters

SYNOPSIS

ifconfig [**-j** *jail*] [**-kLmn**] [**-f** *type***:***format*] *interface* [**create**] [*address_family* [*address* [*dest_address*]]] [*parameters*]
ifconfig [**-j** *jail*] *interface* **destroy**
ifconfig [**-j** *jail*] **-a** [**-dkLmuv**] [**-f** *type***:***format*] [**-G** *groupname*] [**-g** *groupname*] [*address_family*]
ifconfig -C
ifconfig [**-j** *jail*] **-g** *groupname*
ifconfig [**-j** *jail*] **-l** [**-du**] [**-g** *groupname*] [*address_family*]
ifconfig [**-j** *jail*] [**-dkLmuv**] [**-f** *type***:***format*]

DESCRIPTION

The **ifconfig** utility is used to assign an address to a network interface and/or configure network interface parameters. The **ifconfig** utility must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters.

The following options are available:

-a Display information about all interfaces in the system.

The **-a** flag may be used instead of the *interface* argument.

-C List all the interface cloners available on the system, with no additional information. Use of this flag is mutually exclusive with all other flags and commands.

-d Display only the interfaces that are down.

-f *type***:***format*[**,***type***:***format ...*]

Control the output format of **ifconfig**. The format is specified as a comma-separated list of *type***:***format* pairs (see the *EXAMPLES* section for more information).

The output format can also be specified via the IFCONFIG_FORMAT environment variable. The **-f** flag can be supplied multiple times.

The *type*s and their associated *format* strings are:

addr Adjust the display of inet and inet6 addresses:

default Default format, **numeric**
fqdn Fully qualified domain names (FQDN)
host Unqualified hostnames
numeric
Numeric format

ether Adjust the display of link-level ethernet (MAC) addresses:

colon Separate address segments with a colon
dash Separate address segments with a dash
dotted Dotted notation, for example: ‘5254.0015.4a3b’
default Default format, **colon**

inet Adjust the display of inet address subnet masks:

cidr CIDR notation, for example: ‘203.0.113.224/26’
default Default format, **hex**
dotted Dotted quad notation, for example: ‘255.255.255.192’
hex Hexadecimal format, for example: ‘0xffffffc0’

inet6
Adjust the display of inet6 address prefixes (subnet masks):

cidr CIDR notation, for example: ‘::1/128’ or ‘fe80::1%lo0/64’
default Default format, **numeric**
numeric
Integer format, for example: ‘prefixlen 64’

-G *groupname*
Exclude members of the specified *groupname* from the output. *groupname*.

Only one option **-G** should be specified as later override previous ones *groupname* may contain shell patterns in which case it should be quoted.

Setting *groupname* to **all** selects all interfaces.

-g *groupname*
Limit the output to the members of the specified *groupname*.

If **-g** is specified before other significant flags like, e.g., **-a**, **-l**, or **-C**, then **ifconfig** lists names of interfaces belonging to *groupname*. Any other flags and arguments are ignored in this case.

Only one option **-g** should be specified as later override previous ones *groupname* may contain shell patterns in which case it should be quoted.

Setting *groupname* to **all** selects all interfaces.

-j *jail* Perform the actions inside the *jail*.

The **ifconfig** will first attach to the *jail* (by jail id or jail name) before performing the effects.

This allow network interfaces of *jail* to be configured even if the **ifconfig** binary is not available in *jail*.

-k Print keying information for the *interface*, if available.

For example, the values of 802.11 WEP keys and carp(4) passphrases will be printed, if accessible to the current user.

This information is not printed by default, as it may be considered sensitive.

-L Display address lifetime for IPv6 addresses as time offset string.

-l List all available interfaces on the system, with no other additional information.

If an *address_family* is specified, only interfaces of that type will be listed.

If the *address_family* is set to **ether**, then **-l** will exclude loopback interfaces from the list of Ethernet interfaces. This is a special case, because all the other synonyms of the **link** address family will include loopback interfaces in the list.

Use of this flag is mutually exclusive with all other flags and commands, except for **-d**, **-g**, and **-u**.

-m Display the capability list and all of the supported media for the specified interface.

-n Disable automatic loading of network interface drivers.

By default if the network interface driver is not present in the kernel then **ifconfig** will attempt to load it.

-u Display only the interfaces that are up.

-v Get more verbose status for an interface.

address

For the inet family, the address is either a host name present in the host name data base, hosts(5), or an IPv4 address expressed in the Internet standard "dot notation".

It is also possible to use the CIDR notation (also known as the slash notation) to include the netmask. That is, one can specify an address like 192.168.0.1/16.

For the **inet6** family, it is also possible to specify the prefix length using the slash notation, like ::1/128. See the **prefixlen** parameter below for more information.

The link-level (**link**) address is specified as a series of colon-separated hex digits. This can be used to, for example, set a new MAC address on an Ethernet interface, though the mechanism used is not Ethernet specific.

Use the **random** keyword to set a randomly generated MAC address. A randomly-generated MAC address might be the same as one already in use in the network. Such duplications are extremely unlikely.

If the interface is already up when the link-level address is modified, it will be briefly brought down and then brought back up again in order to ensure that the receive filter in the underlying Ethernet hardware is properly reprogrammed.

address_family

Specify the address family which affects interpretation of the remaining parameters. Since an interface can receive transmissions in differing protocols with different naming schemes, specifying the address family is recommended. The address or protocol families currently supported are:

ether

Synonymous with **link** (with some exceptions, see **-l**).

inet Default, if available.

inet6

link

Default, if **inet** is not available.

lladdr
Synonymous with **link**.

dest_address
Specify the address of the correspondent on the other end of a point to point link.

interface
This parameter is a string of the form "name unit", for example, "em0".

The **ifconfig** utility displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, **ifconfig** will report only the details specific to that protocol family.

When no arguments are given, **-a** is implied.

Only the super-user may modify the configuration of a network interface.

PARAMETERS

The following *parameter*s may be set with **ifconfig**:

add Another name for the **alias** parameter. Introduced for compatibility with BSD/OS.

alias Establish an additional network address for this interface. This is sometimes useful when changing network numbers, and one wishes to accept packets addressed to the old interface. If the address is on the same subnet as the first network address for this interface, a non-conflicting netmask must be given. Usually 0xffffffff is most appropriate.

-alias Remove the network address specified. This would be used if you incorrectly specified an alias, or it was no longer needed. If you have incorrectly set an NS address having the side effect of specifying the host portion, removing all NS addresses will allow you to respecify the host portion.

anycast
(Inet6 only.) Specify that the address configured is an anycast address. Based on the current specification, only routers may configure anycast addresses. Anycast address will not be used as source address of any of outgoing IPv6 packets.

arp Enable the use of the Address Resolution Protocol (arp(4)) in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between Internet Protocol addresses and IEEE 802 48-bit MAC addresses (Ethernet addresses).

-arp Disable the use of the Address Resolution Protocol (arp(4)).

staticarp
If the Address Resolution Protocol is enabled, the host will only reply to requests for its addresses, and will never send any requests.

-staticarp
If the Address Resolution Protocol is enabled, the host will perform normally, sending out requests and listening for replies.

stickyarp
Enable the so-called sticky ARP mode for the interface. If this option is enabled on the given interface, any resolved address is marked as a static one and never expires. This may be used to increase security of the network by preventing ARP spoofing or to reduce latency for high-performance Ethernet networks where the time needed for ARP resolution is too high. Please note that a similar feature is also provided for bridges. See the sticky option in the *Bridge Interface Parameters* section. Enabling this option may impact techniques which rely on ARP expiration/overwriting feature such as load-balancers or high-availabity solutions such as carp(4).

-stickyarp
Disable the so-called sticky ARP mode for the interface (default). Resolved addresses will expire normally respecting the kernel ARP configuration.

broadcast
(Inet only.) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

debug Enable driver dependent debugging code; usually, this turns on extra console error logging.

-debug
Disable driver dependent debugging code.

promisc
Put interface into permanently promiscuous mode.

-promisc

Disable permanently promiscuous mode.

delete Another name for the **-alias** parameter.

description *value*, **descr** *value*

Specify a description of the interface. This can be used to label interfaces in situations where they may otherwise be difficult to distinguish.

-description, **-descr**

Clear the interface description.

down Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

group *groupname*

Assign the interface to a "group". The *groupname* may not be longer than 15 characters and must not end in a digit. Any interface can be in multiple groups.

Cloned interfaces are members of their interface family group by default. For example, a VLAN interface such as *vlan10* is a member of the VLAN interface family group, *vlan*.

-group *groupname*

Remove the interface from the given "group".

eui64 (Inet6 only.) Fill interface index (lowermost 64bit of an IPv6 address) automatically.

fib *fib_number*

Specify interface FIB. A FIB *fib_number* is assigned to all frames or packets received on that interface. The FIB is not inherited, e.g., vlans or other sub-interfaces will use the default FIB (0) irrespective of the parent interface’s FIB. The kernel needs to be tuned to support more than the default FIB using the *ROUTETABLES* kernel configuration option, or the *net.fibs* tunable.

tunnelfib *fib_number*

Specify tunnel FIB. A FIB *fib_number* is assigned to all packets encapsulated by tunnel interface, e.g., gif(4), gre(4) and vxlan(4).

maclabel *label*

If Mandatory Access Control support is enabled in the kernel, set the MAC label to *label*.

media *type*

If the driver supports the media selection system, set the media type of the interface to *type*. Some interfaces support the mutually exclusive use of one of several different physical media connectors. For example, a 10Mbit/s Ethernet interface might support the use of either AUI or twisted pair connectors. Setting the media type to **10base5/AUI** would change the currently active connector to the AUI port. Setting it to **10baseT/UTP** would activate twisted pair. Refer to the interfaces’ driver specific documentation or man page for a complete list of the available types.

mediaopt *opts*

If the driver supports the media selection system, set the specified media options on the interface. The *opts* argument is a comma delimited list of options to apply to the interface. Refer to the interfaces’ driver specific man page for a complete list of available options.

-mediaopt *opts*

If the driver supports the media selection system, disable the specified media options on the interface.

mode *mode*

If the driver supports the media selection system, set the specified operating mode on the interface to *mode*. For IEEE 802.11 wireless interfaces that support multiple operating modes this directive is used to select between 802.11a (**11a**), 802.11b (**11b**), and 802.11g (**11g**) operating modes.

txrtlmt

Set if the driver supports TX rate limiting.

inst *minst*, **instance** *minst*

Set the media instance to *minst*. This is useful for devices which have multiple physical layer interfaces (PHYs).

name *name*

Set the interface name to *name*.

rxcsum, **txcsum**, **rxcsum6**, **txcsum6**

If the driver supports user-configurable checksum offloading, enable receive (or transmit) checksum offloading on the interface. The feature can be turned on selectively per protocol family. Use **rxcsum6**, **txcsum6** for ip6(4) or **rxcsum**, **txcsum** otherwise. Some drivers may not be able to enable these flags independently of each other, so setting one may also set the other. The driver will offload as much checksum work as it can reliably support, the exact level of offloading varies between drivers.

-rxcsum, **-txcsum**, **-rxcsum6**, **-txcsum6**
If the driver supports user-configurable checksum offloading, disable receive (or transmit) checksum offloading on the interface. The feature can be turned off selectively per protocol family. Use **-rxcsum6**, **-txcsum6** for ip6(4) or **-rxcsum**, **-txcsum** otherwise. These settings may not always be independent of each other.

tso If the driver supports tcp(4) segmentation offloading, enable TSO on the interface. Some drivers may not be able to support TSO for ip(4) and ip6(4) packets, so they may enable only one of them.

-tso If the driver supports tcp(4) segmentation offloading, disable TSO on the interface. It will always disable TSO for ip(4) and ip6(4).

tso6, **tso4**
If the driver supports tcp(4) segmentation offloading for ip6(4) or ip(4) use one of these to selectively enabled it only for one protocol family.

-tso6, **-tso4**
If the driver supports tcp(4) segmentation offloading for ip6(4) or ip(4) use one of these to selectively disable it only for one protocol family.

lro If the driver supports tcp(4) large receive offloading, enable LRO on the interface.

-lro If the driver supports tcp(4) large receive offloading, disable LRO on the interface.

txtls Transmit TLS offload encrypts Transport Layer Security (TLS) records and segments the encrypted record into one or more tcp(4) segments over either ip(4) or ip6(4). If the driver supports transmit TLS offload, enable transmit TLS offload on the interface. Some drivers may not be able to support transmit TLS offload for ip(4) and ip6(4) packets, so they may enable only one of them.

-txtls If the driver supports transmit TLS offload, disable transmit TLS offload on the interface. It will always disable TLS for ip(4) and ip6(4).

txtlsrtlmt
Enable use of rate limiting (packet pacing) for TLS offload.

-txtlsrtlmt

Disable use of rate limiting for TLS offload.

mextpg
If the driver supports extended multi-page mbuf(9) buffers, enable them on the interface.

-mextpg
If the driver supports extended multi-page mbuf(9) biffers, disable them on the interface.

wol, **wol_ucast**, **wol_mcast**, **wol_magic**
Enable Wake On Lan (WOL) support, if available. WOL is a facility whereby a machine in a low power state may be woken in response to a received packet. There are three types of packets that may wake a system: ucast (directed solely to the machine’s mac address), mcast (directed to a broadcast or multicast address), or magic (unicast or multicast frames with a "magic contents"). Not all devices support WOL, those that do indicate the mechanisms they support in their capabilities. **wol** is a synonym for enabling all available WOL mechanisms. To disable WOL use **-wol**.

vlanmtu, **vlanhwtag**, **vlanhwfilter**, **vlanhwcsum**, **vlanhwtso**
If the driver offers user-configurable VLAN support, enable reception of extended frames, tag processing in hardware, frame filtering in hardware, checksum offloading, or TSO on VLAN, respectively. Note that this must be configured on a physical interface associated with vlan(4), not on a vlan(4) interface itself.

-vlanmtu, **-vlanhwtag**, **-vlanhwfilter**, **-vlanhwcsum**, **-vlanhwtso**
If the driver offers user-configurable VLAN support, disable reception of extended frames, tag processing in hardware, frame filtering in hardware, checksum offloading, or TSO on VLAN, respectively.

vxlanhwcsum, **vxlanhwtso**
If the driver offers user-configurable VXLAN support, enable inner checksum offloading (receive and transmit) or TSO on VXLAN, respectively. Note that this must be configured on a physical interface associated with vxlan(4), not on a vxlan(4) interface itself. The physical interface is either the interface specified as the vxlandev or the interface hosting the vxlanlocal address. The driver will offload as much checksum work and TSO as it can reliably support, the exact level of offloading may vary between drivers.

-vxlanhwcsum, **-vxlanhwtso**
If the driver offers user-configurable VXLAN support, disable checksum offloading (receive and transmit) or TSO on VXLAN, respectively.

vnet *jail*
Move the interface to the jail(8), specified by name or JID. If the jail has a virtual network stack, the interface will disappear from the current environment and become visible to the jail.

-vnet *jail*
Reclaim the interface from the jail(8), specified by name or JID. If the jail has a virtual network stack, the interface will disappear from the jail, and become visible to the current network environment.

polling
Turn on polling(4) feature and disable interrupts on the interface, if driver supports this mode.

-polling
Turn off polling(4) feature and enable interrupt mode on the interface.

create Create the specified network pseudo-device. If the interface is given without a unit number, try to create a new device with an arbitrary unit number. If creation of an arbitrary device is successful, the new device name is printed to standard output unless the interface is renamed or destroyed in the same **ifconfig** invocation.

destroy
Destroy the specified network pseudo-device.

plumb
Another name for the **create** parameter. Included for Solaris compatibility.

unplumb
Another name for the **destroy** parameter. Included for Solaris compatibility.

metric *n*
Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (routed(8)). Higher metrics have the effect of making a route less favorable; metrics are counted as additional hops to the destination network or host.

mtu *n* Set the maximum transmission unit of the interface to *n*, default is interface specific. The MTU is used to limit the size of packets that are transmitted on an interface. Not all interfaces support setting the MTU, and some interfaces have range restrictions.

netmask *mask*
(Inet only.) Specify how much of the address to reserve for subdividing networks into sub-

networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading '0x', with a dot-notation Internet address, or with a pseudo-network name listed in the network table networks(5). The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

The netmask can also be specified in CIDR notation after the address. See the *address* option above for more information.

prefixlen *len*

(Inet6 only.) Specify that *len* bits are reserved for subdividing networks into sub-networks. The *len* must be integer, and for syntactical reason it must be between 0 to 128. It is almost always 64 under the current IPv6 assignment rule. If the parameter is omitted, 64 is used.

The prefix can also be specified using the slash notation after the address. See the *address* option above for more information.

remove

Another name for the **-alias** parameter. Introduced for compatibility with BSD/OS.

link[**0-2**]

Enable special processing of the link level of the interface. These three options are interface specific in actual effect, however, they are in general used to select special modes of operation. An example of this is to enable SLIP compression, or to select the connector type for some Ethernet cards. Refer to the man page for the specific driver for more information.

-link[**0-2**]

Disable special processing at the link level with the specified interface.

monitor

Put the interface in monitor mode. No packets are transmitted, and received packets are discarded after bpf(4) processing.

-monitor

Take the interface out of monitor mode.

pcp *priority_code_point*

Priority code point (PCP) is an 3-bit field which refers to the IEEE 802.1p class of service and maps to the frame priority level.

-pcp Stop tagging packets on the interface w/ the priority code point.

up Mark an interface "up". This may be used to enable an interface after an "**ifconfig down**". It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

ICMPv6 Neighbor Discovery Protocol Parameters

The following parameters are for ICMPv6 Neighbor Discovery Protocol. Note that the address family keyword "inet6" is needed for them:

accept_rtadv
Set a flag to enable accepting ICMPv6 Router Advertisement messages. The sysctl(8) variable *net.inet6.ip6.accept_rtadv* controls whether this flag is set by default or not.

-accept_rtadv
Clear a flag **accept_rtadv**.

no_radr
Set a flag to control whether routers from which the system accepts Router Advertisement messages will be added to the Default Router List or not. When the **accept_rtadv** flag is disabled, this flag has no effect. The sysctl(8) variable *net.inet6.ip6.no_radr* controls whether this flag is set by default or not.

-no_radr
Clear a flag **no_radr**.

auto_linklocal
Set a flag to perform automatic link-local address configuration when the interface becomes available. The sysctl(8) variable *net.inet6.ip6.auto_linklocal* controls whether this flag is set by default or not.

-auto_linklocal
Clear a flag **auto_linklocal**.

defaultif
Set the specified interface as the default route when there is no default router.

-defaultif

Clear a flag **defaultif**.

ifdisabled
Set a flag to disable all of IPv6 network communications on the specified interface. Note that if there are already configured IPv6 addresses on that interface, all of them are marked as "tentative" and DAD will be performed when this flag is cleared.

-ifdisabled
Clear a flag **ifdisabled**. When this flag is cleared and **auto_linklocal** flag is enabled, automatic configuration of a link-local address is performed.

nud Set a flag to enable Neighbor Unreachability Detection.

-nud Clear a flag **nud**.

no_prefer_iface
Set a flag to not honor rule 5 of source address selection in RFC 3484. In practice this means the address on the outgoing interface will not be preferred, effectively yielding the decision to the address selection policy table, configurable with ip6addrctl(8).

-no_prefer_iface
Clear a flag **no_prefer_iface**.

no_dad
Set a flag to disable Duplicate Address Detection.

-no_dad
Clear a flag **no_dad**.

IPv6 Parameters

The following parameters are specific for IPv6 addresses. Note that the address family keyword "inet6" is needed for them:

autoconf
Set the IPv6 autoconfigured address bit.

-autoconf
Clear the IPv6 autoconfigured address bit.

deprecated

Set the IPv6 deprecated address bit.

-deprecated
Clear the IPv6 deprecated address bit.

pltime *n*
Set preferred lifetime for the address.

prefer_source
Set a flag to prefer address as a candidate of the source address for outgoing packets.

-prefer_source
Clear a flag **prefer_source**.

vltime *n*
Set valid lifetime for the address.

IEEE 802.11 Wireless Interfaces Cloning Parameters

The following parameters are specific to cloning IEEE 802.11 wireless interfaces with the **create** request:

wlandev *device*
Use *device* as the parent for the cloned device.

wlanmode *mode*
Specify the operating mode for this cloned device. *mode* is one of **sta**, **ahdemo** (or **adhoc-demo**), **ibss** (or **adhoc**), **ap** (or **hostap**), **wds**, **tdma**, **mesh**, and **monitor**. The operating mode of a cloned interface cannot be changed. The **tdma** mode is actually implemented as an **adhoc-demo** interface with special properties.

wlanbssid *bssid*
The 802.11 mac address to use for the bssid. This must be specified at create time for a legacy **wds** device.

wlanaddr *address*
The local mac address. If this is not specified then a mac address will automatically be assigned to the cloned device. Typically this address is the same as the address of the parent device but if the **bssid** parameter is specified then the driver will craft a unique address for the device (if supported).

wdslegacy
Mark a **wds** device as operating in "legacy mode". Legacy **wds** devices have a fixed peer relationship and do not, for example, roam if their peer stops communicating. For completeness a Dynamic WDS (DWDS) interface may marked as **-wdslegacy**.

bssid Request a unique local mac address for the cloned device. This is only possible if the device supports multiple mac addresses. To force use of the parent’s mac address use **-bssid**.

beacons
Mark the cloned interface as depending on hardware support to track received beacons. To have beacons tracked in software use **-beacons**. For **hostap** mode **-beacons** can also be used to indicate no beacons should be transmitted; this can be useful when creating a WDS configuration but **wds** interfaces can only be created as companions to an access point.

Cloned IEEE 802.11 Wireless Interface Parameters

The following parameters are specific to IEEE 802.11 wireless interfaces cloned with a **create** operation:

ampdu
Enable sending and receiving AMPDU frames when using 802.11n (default). The 802.11n specification states a compliant station must be capable of receiving AMPDU frames but transmission is optional. Use **-ampdu** to disable all use of AMPDU with 802.11n. For testing and/or to work around interoperability problems one can use **ampdutx** and **ampdurx** to control use of AMPDU in one direction.

ampdudensity *density*
Set the AMPDU density parameter used when operating with 802.11n. This parameter controls the inter-packet gap for AMPDU frames. The sending device normally controls this setting but a receiving station may request wider gaps. Legal values for *density* are 0, .25, .5, 1, 2, 4, 8, and 16 (microseconds). A value of - is treated the same as 0.

ampdulimit *limit*
Set the limit on packet size for receiving AMPDU frames when operating with 802.11n. Legal values for *limit* are 8192, 16384, 32768, and 65536 but one can also specify just the unique prefix: 8, 16, 32, 64. Note the sender may limit the size of AMPDU frames to be less than the maximum specified by the receiving station.

amsdu
Enable sending and receiving AMSDU frames when using 802.11n. By default AMSDU is received but not transmitted. Use **-amsdu** to disable all use of AMSDU with 802.11n. For testing and/or to work around interoperability problems one can use **amsdutx** and **amsdurx** to

control use of AMSDU in one direction.

amsdulimit *limit*

Set the limit on packet size for sending and receiving AMSDU frames when operating with 802.11n. Legal values for *limit* are 7935 and 3839 (bytes). Note the sender may limit the size of AMSDU frames to be less than the maximum specified by the receiving station. Note also that devices are not required to support the 7935 limit, only 3839 is required by the specification and the larger value may require more memory to be dedicated to support functionality that is rarely used.

apbridge

When operating as an access point, pass packets between wireless clients directly (default). To instead let them pass up through the system and be forwarded using some other mechanism, use **-apbridge**. Disabling the internal bridging is useful when traffic is to be processed with packet filtering.

authmode *mode*

Set the desired authentication mode in infrastructure mode. Not all adapters support all modes. The set of valid modes is **none**, **open**, **shared** (shared key), **8021x** (IEEE 802.1x), and **wpa** (IEEE WPA/WPA2/802.11i). The **8021x** and **wpa** modes are only useful when using an authentication service (a supplicant for client operation or an authenticator when operating as an access point). Modes are case insensitive.

bgscan

Enable background scanning when operating as a station. Background scanning is a technique whereby a station associated to an access point will temporarily leave the channel to scan for neighboring stations. This allows a station to maintain a cache of nearby access points so that roaming between access points can be done without a lengthy scan operation. Background scanning is done only when a station is not busy and any outbound traffic will cancel a scan operation. Background scanning should never cause packets to be lost though there may be some small latency if outbound traffic interrupts a scan operation. By default background scanning is enabled if the device is capable. To disable background scanning, use **-bgscan**. Background scanning is controlled by the **bgscanidle** and **bgscanintvl** parameters. Background scanning must be enabled for roaming; this is an artifact of the current implementation and may not be required in the future.

bgscanidle *idletime*

Set the minimum time a station must be idle (not transmitting or receiving frames) before a background scan is initiated. The *idletime* parameter is specified in milliseconds. By default a station must be idle at least 250 milliseconds before a background scan is initiated. The idle time may not be set to less than 100 milliseconds.

bgscanintvl *interval*
Set the interval at which background scanning is attempted. The *interval* parameter is specified in seconds. By default a background scan is considered every 300 seconds (5 minutes). The *interval* may not be set to less than 15 seconds.

bintval *interval*
Set the interval at which beacon frames are sent when operating in ad-hoc or ap mode. The *interval* parameter is specified in TU's (1024 usecs). By default beacon frames are transmitted every 100 TU's.

bmissthreshold *count*
Set the number of consecutive missed beacons at which the station will attempt to roam (i.e., search for a new access point). The *count* parameter must be in the range 1 to 255; though the upper bound may be reduced according to device capabilities. The default threshold is 7 consecutive missed beacons; but this may be overridden by the device driver. Another name for the **bmissthreshold** parameter is **bmiss**.

bssid *address*
Specify the MAC address of the access point to use when operating as a station in a BSS network. This overrides any automatic selection done by the system. To disable a previously selected access point, supply **any**, **none**, or - for the address. This option is useful when more than one access point uses the same SSID. Another name for the **bssid** parameter is **ap**.

burst Enable packet bursting. Packet bursting is a transmission technique whereby the wireless medium is acquired once to send multiple frames and the interframe spacing is reduced. This technique can significantly increase throughput by reducing transmission overhead. Packet bursting is supported by the 802.11e QoS specification and some devices that do not support QoS may still be capable. By default packet bursting is enabled if a device is capable of doing it. To disable packet bursting, use **-burst**.

chanlist *channels*
Set the desired channels to use when scanning for access points, neighbors in an IBSS network, or looking for unoccupied channels when operating as an access point. The set of channels is specified as a comma-separated list with each element in the list representing either a single channel number or a range of the form "a-b". Channel numbers must be in the range 1 to 255 and be permissible according to the operating characteristics of the device.

channel *number*

Set a single desired channel. Channels range from 1 to 255, but the exact selection available depends on the region your adaptor was manufactured for. Setting the channel to **any**, or "**-**" will clear any desired channel and, if the device is marked up, force a scan for a channel to operate on. Alternatively the frequency, in megahertz, may be specified instead of the channel number.

When there are several ways to use a channel the channel number/frequency may be appended with attributes to clarify. For example, if a device is capable of operating on channel 6 with 802.11n and 802.11g then one can specify that g-only use should be used by specifying **6:g**. Similarly the channel width can be specified by appending it with "**/**"; e.g., **6/40** specifies a 40MHz wide channel. These attributes can be combined as in: **6:ht/40**.

The full set of flags specified following a "**:**" are:

a 802.11a
b 802.11b
d Atheros Dynamic Turbo mode
g 802.11g
h Same as **n**
n 802.11n aka HT
s Atheros Static Turbo mode
t Atheros Dynamic Turbo mode, or appended to **st** and **dt**

The full set of channel widths following a **/** are:

5 5MHz aka quarter-rate channel
10 10MHz aka half-rate channel
20 20MHz mostly for use in specifying **ht20**
40 40MHz mostly for use in specifying **ht40**

In addition, a 40MHz HT channel specification may include the location of the extension channel by appending "**+**" or "**-**" for above and below, respectively; e.g., **2437:ht/40+** specifies 40MHz wide HT operation with the center channel at frequency 2437 and the extension channel above.

country *name*
Set the country code to use in calculating the regulatory constraints for operation. In particular the set of available channels, how the wireless device will operation on the channels, and the maximum transmit power that can be used on a channel are defined by this setting. Country/Region codes are specified as a 2-character abbreviation defined by ISO 3166 or using a longer, but possibly ambiguous, spelling; e.g., "ES" and "Spain". The set of country codes are taken from */etc/regdomain.xml* and can also be viewed with the **list countries** request. Note that

not all devices support changing the country code from a default setting; typically stored in EEPROM. See also **regdomain**, **indoor**, **outdoor**, and **anywhere**.

dfs Enable Dynamic Frequency Selection (DFS) as specified in 802.11h. DFS embodies several facilities including detection of overlapping radar signals, dynamic transmit power control, and channel selection according to a least-congested criteria. DFS support is mandatory for some 5GHz frequencies in certain locales (e.g., ETSI). By default DFS is enabled according to the regulatory definitions specified in */etc/regdomain.xml* and the current country code, regdomain, and channel. Note the underlying device (and driver) must support radar detection for full DFS support to work. To be fully compliant with the local regulatory agency frequencies that require DFS should not be used unless it is fully supported. Use **-dfs** to disable this functionality for testing.

dotd Enable support for the 802.11d specification (default). When this support is enabled in station mode, beacon frames that advertise a country code different than the currently configured country code will cause an event to be dispatched to user applications. This event can be used by the station to adopt that country code and operate according to the associated regulatory constraints. When operating as an access point with 802.11d enabled the beacon and probe response frames transmitted will advertise the current regulatory domain settings. To disable 802.11d use **-dotd**.

doth Enable 802.11h support including spectrum management. When 802.11h is enabled beacon and probe response frames will have the SpectrumMgt bit set in the capabilities field and country and power constraint information elements will be present. 802.11h support also includes handling Channel Switch Announcements (CSA) which are a mechanism to coordinate channel changes by an access point. By default 802.11h is enabled if the device is capable. To disable 802.11h use **-doth**.

deftxkey *index*
Set the default key to use for transmission. Typically this is only set when using WEP encryption. Note that you must set a default transmit key for the system to know which key to use in encrypting outbound traffic. The **weptxkey** is an alias for this request; it is provided for backwards compatibility.

dtimperiod *period*
Set the DTIM period for transmitting buffered multicast data frames when operating in ap mode. The *period* specifies the number of beacon intervals between DTIM and must be in the range 1 to 15. By default DTIM is 1 (i.e., DTIM occurs at each beacon).

quiet Enable the use of quiet IE. Hostap will use this to silence other stations to reduce interference for radar detection when operating on 5GHz frequency and doth support is enabled. Use **-quiet** to disable this functionality.

quiet_period *period*
Set the QUIET *period* to the number of beacon intervals between the start of regularly scheduled quiet intervals defined by Quiet element.

quiet_count *count*
Set the QUIET *count* to the number of TBTTs until the beacon interval during which the next quiet interval shall start. A value of 1 indicates the quiet interval will start during the beacon interval starting at the next TBTT. A value 0 is reserved.

quiet_offset *offset*
Set the QUIET *offset* to the offset of the start of the quiet interval from the TBTT specified by the Quiet count, expressed in TUs. The value of the *offset* shall be less than one beacon interval.

quiet_duration *dur*
Set the QUIET *dur* to the duration of the Quiet interval, expressed in TUs. The value should be less than beacon interval.

dturbo
Enable the use of Atheros Dynamic Turbo mode when communicating with another Dynamic Turbo-capable station. Dynamic Turbo mode is an Atheros-specific mechanism by which stations switch between normal 802.11 operation and a "boosted" mode in which a 40MHz wide channel is used for communication. Stations using Dynamic Turbo mode operate boosted only when the channel is free of non-dturbo stations; when a non-dturbo station is identified on the channel all stations will automatically drop back to normal operation. By default, Dynamic Turbo mode is not enabled, even if the device is capable. Note that turbo mode (dynamic or static) is only allowed on some channels depending on the regulatory constraints; use the **list chan** command to identify the channels where turbo mode may be used. To disable Dynamic Turbo mode use **-dturbo**.

dwds Enable Dynamic WDS (DWDS) support. DWDS is a facility by which 4-address traffic can be carried between stations operating in infrastructure mode. A station first associates to an access point and authenticates using normal procedures (e.g., WPA). Then 4-address frames are passed to carry traffic for stations operating on either side of the wireless link. DWDS extends the normal WDS mechanism by leveraging existing security protocols and eliminating static binding.

When DWDS is enabled on an access point 4-address frames received from an authorized station will generate a "DWDS discovery" event to user applications. This event should be used to create a WDS interface that is bound to the remote station (and usually plumbed into a bridge). Once the WDS interface is up and running 4-address traffic then logically flows through that interface.

When DWDS is enabled on a station, traffic with a destination address different from the peer station are encapsulated in a 4-address frame and transmitted to the peer. All 4-address traffic uses the security information of the stations (e.g., cryptographic keys). A station is associated using 802.11n facilities may transport 4-address traffic using these same mechanisms; this depends on available resources and capabilities of the device. The DWDS implementation guards against layer 2 routing loops of multicast traffic.

ff Enable the use of Atheros Fast Frames when communicating with another Fast Frames-capable station. Fast Frames are an encapsulation technique by which two 802.3 frames are transmitted in a single 802.11 frame. This can noticeably improve throughput but requires that the receiving station understand how to decapsulate the frame. Fast frame use is negotiated using the Atheros 802.11 vendor-specific protocol extension so enabling use is safe when communicating with non-Atheros devices. By default, use of fast frames is enabled if the device is capable. To explicitly disable fast frames, use **-ff**.

fragthreshold *length*

Set the threshold for which transmitted frames are broken into fragments. The *length* argument is the frame size in bytes and must be in the range 256 to 2346. Setting *length* to 2346, **any**, or **-** disables transmit fragmentation. Not all adapters honor the fragmentation threshold.

hidessid

When operating as an access point, do not broadcast the SSID in beacon frames or respond to probe request frames unless they are directed to the ap (i.e., they include the ap's SSID). By default, the SSID is included in beacon frames and undirected probe request frames are answered. To re-enable the broadcast of the SSID etc., use **-hidessid**.

ht Enable use of High Throughput (HT) when using 802.11n (default). The 802.11n specification includes mechanisms for operation on 20MHz and 40MHz wide channels using different signalling mechanisms than specified in 802.11b, 802.11g, and 802.11a. Stations negotiate use of these facilities, termed HT20 and HT40, when they associate. To disable all use of 802.11n use **-ht**. To disable use of HT20 (e.g., to force only HT40 use) use **-ht20**. To disable use of HT40 use **-ht40**.

HT configuration is used to "auto promote" operation when several choices are available. For example, if a station associates to an 11n-capable access point it controls whether the station uses legacy operation, HT20, or HT40. When an 11n-capable device is setup as an access point and Auto Channel Selection is used to locate a channel to operate on, HT configuration controls whether legacy, HT20, or HT40 operation is setup on the selected channel. If a fixed channel is specified for a station then HT configuration can be given as part of the channel specification; e.g., 6:ht/20 to setup HT20 operation on channel 6.

htcompat
Enable use of compatibility support for pre-802.11n devices (default). The 802.11n protocol specification went through several incompatible iterations. Some vendors implemented 11n support to older specifications that will not interoperate with a purely 11n-compliant station. In particular the information elements included in management frames for old devices are different. When compatibility support is enabled both standard and compatible data will be provided. Stations that associate using the compatibility mechanisms are flagged in **list sta**. To disable compatibility support use **-htcompat**.

htprotmode *technique*
For interfaces operating in 802.11n, use the specified *technique* for protecting HT frames in a mixed legacy/HT network. The set of valid techniques is **off**, and **rts** (RTS/CTS, default). Technique names are case insensitive.

inact Enable inactivity processing for stations associated to an access point (default). When operating as an access point the 802.11 layer monitors the activity of each associated station. When a station is inactive for 5 minutes it will send several "probe frames" to see if the station is still present. If no response is received then the station is deauthenticated. Applications that prefer to handle this work can disable this facility by using **-inact**.

indoor
Set the location to use in calculating regulatory constraints. The location is also advertised in beacon and probe response frames when 802.11d is enabled with **dotd**. See also **outdoor**, **anywhere**, **country**, and **regdomain**.

list active
Display the list of channels available for use taking into account any restrictions set with the **chanlist** directive. See the description of **list chan** for more information.

list caps
Display the adaptor’s capabilities, including the operating modes supported.

list chan
Display the list of channels available for use. Channels are shown with their IEEE channel

number, equivalent frequency, and usage modes. Channels identified as ‘11g’ are also usable in ‘11b’ mode. Channels identified as ‘11a Turbo’ may be used only for Atheros’ Static Turbo mode (specified with **mediaopt turbo**). Channels marked with a ‘*’ have a regulatory constraint that they be passively scanned. This means a station is not permitted to transmit on the channel until it identifies the channel is being used for 802.11 communication; typically by hearing a beacon frame from an access point operating on the channel. **list freq** is another way of requesting this information. By default a compacted list of channels is displayed; if the **-v** option is specified then all channels are shown.

list countries
Display the set of country codes and regulatory domains that can be used in regulatory configuration.

list mac
Display the current MAC Access Control List state. Each address is prefixed with a character that indicates the current policy applied to it: ‘+’ indicates the address is allowed access, ‘-’ indicates the address is denied access, ‘*’ indicates the address is present but the current policy open (so the ACL is not consulted).

list mesh
Displays the mesh routing table, used for forwarding packets on a mesh network.

list regdomain
Display the current regulatory settings including the available channels and transmit power caps.

list roam
Display the parameters that govern roaming operation.

list txparam
Display the parameters that govern transmit operation.

list txpower
Display the transmit power caps for each channel.

list scan
Display the access points and/or ad-hoc neighbors located in the vicinity. This information may be updated automatically by the adapter with a **scan** request or through background scanning. Depending on the capabilities of the stations the following flags (capability codes) can be included in the output:

A Channel agility.

B PBCC modulation.

C Poll request capability.

D DSSS/OFDM capability.

E Extended Service Set (ESS). Indicates that the station is part of an infrastructure network rather than an IBSS/ad-hoc network.

I Independent Basic Service Set (IBSS). Indicates that the station is part of an ad-hoc network rather than an ESS network.

P Privacy capability. The station requires authentication and encryption for all data frames exchanged within the BSS using cryptographic means such as WEP, TKIP, or AES-CCMP.

R Robust Secure Network (RSN).

S Short Preamble. Indicates that the network is using short preambles, defined in 802.11b High Rate/DSSS PHY, and utilizes a 56 bit sync field rather than the 128 bit field used in long preamble mode. Short preambles are used to optionally improve throughput performance with 802.11g and 802.11b.

c Pollable capability.

s Short slot time capability. Indicates that the 802.11g network is using a short slot time because there are no legacy (802.11b) stations present.

By default interesting information elements captured from the neighboring stations are displayed at the end of each row. Possible elements include: **WME** (station supports WME), **WPA** (station supports WPA), **WPS** (station supports WPS), **RSN** (station supports 802.11i/RSN), **HTCAP** (station supports 802.11n/HT communication), **ATH** (station supports Atheros protocol extensions), **VEN** (station supports unknown vendor-specific extensions). If the **-v** flag is used all the information elements and their contents will be shown. Specifying the **-v** flag also enables display of long SSIDs. The **list ap** command is another way of requesting this information.

list sta

When operating as an access point display the stations that are currently associated. When operating in ad-hoc mode display stations identified as neighbors in the IBSS. When operating in mesh mode display stations identified as neighbors in the MBSS. When operating in station mode display the access point. Capabilities advertised by the stations are described under the **scan** request. The following flags can be included in the output:

A Authorized. Indicates that the station is permitted to send/receive data frames.

E Extended Rate Phy (ERP). Indicates that the station is operating in an 802.11g network using extended transmit rates.

H High Throughput (HT). Indicates that the station is using HT transmit rates. If a '+' follows immediately after then the station associated using deprecated mechanisms supported only when **htcompat** is enabled.

P Power Save. Indicates that the station is operating in power save mode.

Q Quality of Service (QoS). Indicates that the station is using QoS encapsulation for data frame. QoS encapsulation is enabled only when WME mode is enabled.

S Short GI in HT 40MHz mode enabled. If a '+' follows immediately after then short GI in HT 20MHz mode is enabled as well.

T Transitional Security Network (TSN). Indicates that the station associated using TSN; see also **tsn** below.

W Wi-Fi Protected Setup (WPS). Indicates that the station associated using WPS.

s Short GI in HT 20MHz mode enabled.

By default information elements received from associated stations are displayed in a short form; the **-v** flag causes this information to be displayed symbolically.

list wme

Display the current channel parameters to use when operating in WME mode. If the **-v** option is specified then both channel and BSS parameters are displayed for each AC (first channel, then BSS). When WME mode is enabled for an adaptor this information will be displayed with the regular status; this command is mostly useful for examining parameters when WME mode is disabled. See the description of the **wme** directive for information on the various parameters.

maxretry *count*

Set the maximum number of tries to use in sending unicast frames. The default setting is 6 but drivers may override this with a value they choose.

mcastrate *rate*
Set the rate for transmitting multicast/broadcast frames. Rates are specified as megabits/second in decimal; e.g., 5.5 for 5.5 Mb/s. This rate should be valid for the current operating conditions; if an invalid rate is specified drivers are free to chose an appropriate rate.

mgtrate *rate*
Set the rate for transmitting management and/or control frames. Rates are specified as megabits/second in decimal; e.g., 5.5 for 5.5 Mb/s.

outdoor
Set the location to use in calculating regulatory constraints. The location is also advertised in beacon and probe response frames when 802.11d is enabled with **dotd**. See also **anywhere**, **country**, **indoor**, and **regdomain**.

powersave
Enable powersave operation. When operating as a client, the station will conserve power by periodically turning off the radio and listening for messages from the access point telling it there are packets waiting. The station must then retrieve the packets. Not all devices support power save operation as a client. The 802.11 specification requires that all access points support power save but some drivers do not. Use **-powersave** to disable powersave operation when operating as a client.

powersavesleep *sleep*
Set the desired max powersave sleep time in TU's (1024 usecs). By default the max powersave sleep time is 100 TU's.

protmode *technique*
For interfaces operating in 802.11g, use the specified *technique* for protecting OFDM frames in a mixed 11b/11g network. The set of valid techniques is **off**, **cts** (CTS to self), and **rtscts** (RTS/CTS). Technique names are case insensitive. Not all devices support **cts** as a protection technique.

pureg When operating as an access point in 802.11g mode allow only 11g-capable stations to associate (11b-only stations are not permitted to associate). To allow both 11g and 11b-only stations to associate, use **-pureg**.

puren When operating as an access point in 802.11n mode allow only HT-capable stations to associate (legacy stations are not permitted to associate). To allow both HT and legacy stations to associate, use **-puren**.

regdomain *sku*
Set the regulatory domain to use in calculating the regulatory constraints for operation. In particular the set of available channels, how the wireless device will operation on the channels, and the maximum transmit power that can be used on a channel are defined by this setting. Regdomain codes (SKU's) are taken from */etc/regdomain.xml* and can also be viewed with the **list countries** request. Note that not all devices support changing the regdomain from a default setting; typically stored in EEPROM. See also **country**, **indoor**, **outdoor**, and **anywhere**.

rifs Enable use of Reduced InterFrame Spacing (RIFS) when operating in 802.11n on an HT channel. Note that RIFS must be supported by both the station and access point for it to be used. To disable RIFS use **-rifs**.

roam:rate *rate*
Set the threshold for controlling roaming when operating in a BSS. The *rate* parameter specifies the transmit rate in megabits at which roaming should be considered. If the current transmit rate drops below this setting and background scanning is enabled, then the system will check if a more desirable access point is available and switch over to it. The current scan cache contents are used if they are considered valid according to the **scanvalid** parameter; otherwise a background scan operation is triggered before any selection occurs. Each channel type has a separate rate threshold; the default values are: 12 Mb/s (11a), 2 Mb/s (11b), 2 Mb/s (11g), MCS 1 (11na, 11ng).

roam:rssi *rssi*
Set the threshold for controlling roaming when operating in a BSS. The *rssi* parameter specifies the receive signal strength in dBm units at which roaming should be considered. If the current rssi drops below this setting and background scanning is enabled, then the system will check if a more desirable access point is available and switch over to it. The current scan cache contents are used if they are considered valid according to the **scanvalid** parameter; otherwise a background scan operation is triggered before any selection occurs. Each channel type has a separate rssi threshold; the default values are all 7 dBm.

roaming *mode*
When operating as a station, control how the system will behave when communication with the current access point is broken. The *mode* argument may be one of **device** (leave it to the hardware device to decide), **auto** (handle either in the device or the operating system--as appropriate), **manual** (do nothing until explicitly instructed). By default, the device is left to handle this if it is capable; otherwise, the operating system will automatically attempt to reestablish communication. Manual mode is used by applications such as wpa_supplicant(8) that

want to control the selection of an access point.

rtsthreshold *length*
Set the threshold for which transmitted frames are preceded by transmission of an RTS control frame. The *length* argument is the frame size in bytes and must be in the range 1 to 2346. Setting *length* to 2346, **any**, or **-** disables transmission of RTS frames. Not all adapters support setting the RTS threshold.

scan Initiate a scan of neighboring stations, wait for it to complete, and display all stations found. Only the super-user can initiate a scan. See **list scan** for information on the display. By default a background scan is done; otherwise a foreground scan is done and the station may roam to a different access point. The **list scan** request can be used to show recent scan results without initiating a new scan.

scanvalid *threshold*
Set the maximum time the scan cache contents are considered valid; i.e., will be used without first triggering a scan operation to refresh the data. The *threshold* parameter is specified in seconds and defaults to 60 seconds. The minimum setting for *threshold* is 10 seconds. One should take care setting this threshold; if it is set too low then attempts to roam to another access point may trigger unnecessary background scan operations.

shortgi
Enable use of Short Guard Interval when operating in 802.11n on an HT channel. NB: this currently enables Short GI on both HT40 and HT20 channels. To disable Short GI use **-shortgi**.

smps Enable use of Static Spatial Multiplexing Power Save (SMPS) when operating in 802.11n. A station operating with Static SMPS maintains only a single receive chain active (this can significantly reduce power consumption). To disable SMPS use **-smps**.

smpsdyn
Enable use of Dynamic Spatial Multiplexing Power Save (SMPS) when operating in 802.11n. A station operating with Dynamic SMPS maintains only a single receive chain active but switches to multiple receive chains when it receives an RTS frame (this can significantly reduce power consumption). Note that stations cannot distinguish between RTS/CTS intended to enable multiple receive chains and those used for other purposes. To disable SMPS use **-smps**.

ssid *ssid*
Set the desired Service Set Identifier (aka network name). The SSID is a string up to 32 characters in length and may be specified as either a normal string or in hexadecimal when preceded by '0x'. Additionally, the SSID may be cleared by setting it to '-'.

tdmaslot *slot*

When operating with TDMA, use the specified *slot* configuration. The *slot* is a number between 0 and the maximum number of slots in the BSS. Note that a station configured as slot 0 is a master and will broadcast beacon frames advertising the BSS; stations configured to use other slots will always scan to locate a master before they ever transmit. By default **tdmaslot** is set to 1.

tdmaslotcnt *cnt*

When operating with TDMA, setup a BSS with *cnt* slots. The slot count may be at most 8. The current implementation is only tested with two stations (i.e., point to point applications). This setting is only meaningful when a station is configured as slot 0; other stations adopt this setting from the BSS they join. By default **tdmaslotcnt** is set to 2.

tdmaslotlen *len*

When operating with TDMA, setup a BSS such that each station has a slot *len* microseconds long. The slot length must be at least 150 microseconds (1/8 TU) and no more than 65 milliseconds. Note that setting too small a slot length may result in poor channel bandwidth utilization due to factors such as timer granularity and guard time. This setting is only meaningful when a station is configured as slot 0; other stations adopt this setting from the BSS they join. By default **tdmaslotlen** is set to 10 milliseconds.

tdmabintval *intval*

When operating with TDMA, setup a BSS such that beacons are transmitted every *intval* superframes to synchronize the TDMA slot timing. A superframe is defined as the number of slots times the slot length; e.g., a BSS with two slots of 10 milliseconds has a 20 millisecond superframe. The beacon interval may not be zero. A lower setting of **tdmabintval** causes the timers to be resynchronized more often; this can be help if significant timer drift is observed. By default **tdmabintval** is set to 5.

tsn When operating as an access point with WPA/802.11i allow legacy stations to associate using static key WEP and open authentication. To disallow legacy station use of WEP, use **-tsn**.

txpower *power*

Set the power used to transmit frames. The *power* argument is specified in .5 dBm units. Out of range values are truncated. Typically only a few discrete power settings are available and the driver will use the setting closest to the specified value. Not all adapters support changing the transmit power.

ucastrate *rate*

Set a fixed rate for transmitting unicast frames. Rates are specified as megabits/second in decimal; e.g., 5.5 for 5.5 Mb/s. This rate should be valid for the current operating conditions; if an invalid rate is specified drivers are free to chose an appropriate rate.

wepmode *mode*

Set the desired WEP mode. Not all adapters support all modes. The set of valid modes is **off**, **on**, and **mixed**. The **mixed** mode explicitly tells the adaptor to allow association with access points which allow both encrypted and unencrypted traffic. On these adapters, **on** means that the access point must only allow encrypted connections. On other adapters, **on** is generally another name for **mixed**. Modes are case insensitive.

weptxkey *index*

Set the WEP key to be used for transmission. This is the same as setting the default transmission key with **deftxkey**.

wepkey *key*|*index*:*key*

Set the selected WEP key. If an *index* is not given, key 1 is set. A WEP key will be either 5 or 13 characters (40 or 104 bits) depending on the local network and the capabilities of the adaptor. It may be specified either as a plain string or as a string of hexadecimal digits preceded by '0x'. For maximum portability, hex keys are recommended; the mapping of text keys to WEP encryption is usually driver-specific. In particular, the Windows drivers do this mapping differently to FreeBSD. A key may be cleared by setting it to '-'. If WEP is supported then there are at least four keys. Some adapters support more than four keys. If that is the case, then the first four keys (1-4) will be the standard temporary keys and any others will be adaptor specific keys such as permanent keys stored in NVRAM.

Note that you must set a default transmit key with **deftxkey** for the system to know which key to use in encrypting outbound traffic.

wme Enable Wireless Multimedia Extensions (WME) support, if available, for the specified interface. WME is a subset of the IEEE 802.11e standard to support the efficient communication of realtime and multimedia data. To disable WME support, use **-wme**. Another name for this parameter is **wmm**.

The following parameters are meaningful only when WME support is in use. Parameters are specified per-AC (Access Category) and split into those that are used by a station when acting as an access point and those for client stations in the BSS. The latter are received from the access point and may not be changed (at the station). The following Access Categories are recognized:

AC_BE (or **BE**) best effort delivery,
AC_BK (or **BK**) background traffic,

AC_VI (or **VI**) video traffic,

AC_VO

(or **VO**) voice traffic.

AC parameters are case-insensitive. Traffic classification is done in the operating system using the vlan priority associated with data frames or the ToS (Type of Service) indication in IP-encapsulated frames. If neither information is present, traffic is assigned to the Best Effort (BE) category.

ack *ac*

Set the ACK policy for QoS transmissions by the local station; this controls whether or not data frames transmitted by a station require an ACK response from the receiving station. To disable waiting for an ACK use **-ack**. This parameter is applied only to the local station.

acm *ac*

Enable the Admission Control Mandatory (ACM) mechanism for transmissions by the local station. To disable the ACM use **-acm**. On stations in a BSS this parameter is read-only and indicates the setting received from the access point. NB: ACM is not supported right now.

aifs *ac count*

Set the Arbitration Inter Frame Spacing (AIFS) channel access parameter to use for transmissions by the local station. On stations in a BSS this parameter is read-only and indicates the setting received from the access point.

cwmin *ac count*

Set the CWmin channel access parameter to use for transmissions by the local station. On stations in a BSS this parameter is read-only and indicates the setting received from the access point.

cwmax *ac count*

Set the CWmax channel access parameter to use for transmissions by the local station. On stations in a BSS this parameter is read-only and indicates the setting received from the access point.

txoplimit *ac limit*

Set the Transmission Opportunity Limit channel access parameter to use for transmissions by the local station. This parameter defines an interval of time when a WME station has the right to initiate transmissions onto the wireless medium. On stations in a BSS this parameter is read-only and indicates the setting received from the access point.

bss:aifs *ac count*
Set the AIFS channel access parameter to send to stations in a BSS. This parameter is meaningful only when operating in ap mode.

bss:cwmin *ac count*
Set the CWmin channel access parameter to send to stations in a BSS. This parameter is meaningful only when operating in ap mode.

bss:cwmax *ac count*
Set the CWmax channel access parameter to send to stations in a BSS. This parameter is meaningful only when operating in ap mode.

bss:txoplimit *ac limit*
Set the TxOpLimit channel access parameter to send to stations in a BSS. This parameter is meaningful only when operating in ap mode.

wps Enable Wireless Privacy Subscriber support. Note that WPS support requires a WPS-capable supplicant. To disable this function use **-wps**.

MAC-Based Access Control List Parameters

The following parameters support an optional access control list feature available with some adapters when operating in ap mode; see wlan_acl(4). This facility allows an access point to accept/deny association requests based on the MAC address of the station. Note that this feature does not significantly enhance security as MAC address spoofing is easy to do.

mac:add *address*
Add the specified MAC address to the database. Depending on the policy setting association requests from the specified station will be allowed or denied.

mac:allow
Set the ACL policy to permit association only by stations registered in the database.

mac:del *address*
Delete the specified MAC address from the database.

mac:deny
Set the ACL policy to deny association only by stations registered in the database.

mac:kick *address*
Force the specified station to be deauthenticated. This typically is done to block a station after updating the address database.

mac:open
Set the ACL policy to allow all stations to associate.

mac:flush
Delete all entries in the database.

mac:radius
Set the ACL policy to permit association only by stations approved by a RADIUS server. Note that this feature requires the hostapd(8) program be configured to do the right thing as it handles the RADIUS processing (and marks stations as authorized).

Mesh Mode Wireless Interface Parameters

The following parameters are related to a wireless interface operating in mesh mode:

meshid *meshid*
Set the desired Mesh Identifier. The Mesh ID is a string up to 32 characters in length. A mesh interface must have a Mesh Identifier specified to reach an operational state.

meshttl *ttl*
Set the desired "time to live" for mesh forwarded packets; this is the number of hops a packet may be forwarded before it is discarded. The default setting for **meshttl** is 31.

meshpeering
Enable or disable peering with neighbor mesh stations. Stations must peer before any data packets can be exchanged. By default **meshpeering** is enabled.

meshforward
Enable or disable forwarding packets by a mesh interface. By default **meshforward** is enabled.

meshgate
This attribute specifies whether or not the mesh STA activates mesh gate announcements. By default **meshgate** is disabled.

meshmetric *protocol*
Set the specified *protocol* as the link metric protocol used on a mesh network. The default protocol is called *AIRTIME*. The mesh interface will restart after changing this setting.

meshpath *protocol*

Set the specified *protocol* as the path selection protocol used on a mesh network. The only available protocol at the moment is called *HWMP* (Hybrid Wireless Mesh Protocol). The mesh interface will restart after changing this setting.

hwmprootmode *mode*

Stations on a mesh network can operate as "root nodes". Root nodes try to find paths to all mesh nodes and advertise themselves regularly. When there is a root mesh node on a network, other mesh nodes can setup paths between themselves faster because they can use the root node to find the destination. This path may not be the best, but on-demand routing will eventually find the best path. The following modes are recognized:

DISABLED Disable root mode.

NORMAL Send broadcast path requests every two seconds. Nodes on the mesh without a path to this root mesh station with try to discover a path to us.

PROACTIVE Send broadcast path requests every two seconds and every node must reply with a path reply even if it already has a path to this root mesh station.

RANN Send broadcast root announcement (RANN) frames. Nodes on the mesh without a path to this root mesh station with try to discover a path to us.

By default **hwmprootmode** is set to *DISABLED*.

hwmpmaxhops *cnt*

Set the maximum number of hops allowed in an HMWP path to *cnt*. The default setting for **hwmpmaxhops** is 31.

Compatibility Parameters

The following parameters are for compatibility with other systems:

nwid *ssid*

Another name for the **ssid** parameter. Included for NetBSD compatibility.

stationname *name*

Set the name of this station. The station name is not part of the IEEE 802.11 protocol though some interfaces support it. As such it only seems to be meaningful to identical or virtually identical equipment. Setting the station name is identical in syntax to setting the SSID. One can also use **station** for BSD/OS compatibility.

wep Another way of saying **wepmode on**. Included for BSD/OS compatibility.

-wep Another way of saying **wepmode off**. Included for BSD/OS compatibility.

nwkey key
Another way of saying: "wepmode on weptxkey 1 wepkey 1:key wepkey 2:- wepkey 3:- wepkey 4:-". Included for NetBSD compatibility.

nwkey *n*:*k1*,*k2*,*k3*,*k4*
Another way of saying "wepmode on weptxkey n wepkey 1:k1 wepkey 2:k2 wepkey 3:k3 wepkey 4:k4". Included for NetBSD compatibility.

-nwkey
Another way of saying **wepmode off**. Included for NetBSD compatibility.

Bridge Interface Parameters

The following parameters are specific to bridge interfaces:

addm *interface*
Add the interface named by *interface* as a member of the bridge. The interface is put into promiscuous mode so that it can receive every packet sent on the network.

deletem *interface*
Remove the interface named by *interface* from the bridge. Promiscuous mode is disabled on the interface when it is removed from the bridge.

maxaddr *size*
Set the size of the bridge address cache to *size*. The default is 2000 entries.

timeout *seconds*
Set the timeout of address cache entries to *seconds* seconds. If *seconds* is zero, then address cache entries will not be expired. The default is 1200 seconds.

addr Display the addresses that have been learned by the bridge.

static *interface-name address*
Add a static entry into the address cache pointing to *interface-name*. Static entries are never aged out of the cache or re-placed, even if the address is seen on a different interface.

deladdr *address*
Delete *address* from the address cache.

flush Delete all dynamically-learned addresses from the address cache.

flushall
Delete all addresses, including static addresses, from the address cache.

discover *interface*
Mark an interface as a "discovering" interface. When the bridge has no address cache entry (either dynamic or static) for the destination address of a packet, the bridge will forward the packet to all member interfaces marked as "discovering". This is the default for all interfaces added to a bridge.

-discover *interface*
Clear the "discovering" attribute on a member interface. For packets without the "discovering" attribute, the only packets forwarded on the interface are broadcast or multicast packets and packets for which the destination address is known to be on the interface’s segment.

learn *interface*
Mark an interface as a "learning" interface. When a packet arrives on such an interface, the source address of the packet is entered into the address cache as being a destination address on the interface’s segment. This is the default for all interfaces added to a bridge.

-learn *interface*
Clear the "learning" attribute on a member interface.

sticky *interface*
Mark an interface as a "sticky" interface. Dynamically learned address entries are treated at static once entered into the cache. Sticky entries are never aged out of the cache or replaced, even if the address is seen on a different interface.

-sticky *interface*
Clear the "sticky" attribute on a member interface.

private *interface*
Mark an interface as a "private" interface. A private interface does not forward any traffic to any other port that is also a private interface.

-private *interface*
Clear the "private" attribute on a member interface.

span *interface*
Add the interface named by *interface* as a span port on the bridge. Span ports transmit a copy of every frame received by the bridge. This is most useful for snooping a bridged network

passively on another host connected to one of the span ports of the bridge.

-span *interface*
Delete the interface named by *interface* from the list of span ports of the bridge.

stp *interface*
Enable Spanning Tree protocol on *interface*. The if_bridge(4) driver has support for the IEEE 802.1D Spanning Tree protocol (STP). Spanning Tree is used to detect and remove loops in a network topology.

-stp *interface*
Disable Spanning Tree protocol on *interface*. This is the default for all interfaces added to a bridge.

edge *interface*
Set *interface* as an edge port. An edge port connects directly to end stations cannot create bridging loops in the network, this allows it to transition straight to forwarding.

-edge *interface*
Disable edge status on *interface*.

autoedge *interface*
Allow *interface* to automatically detect edge status. This is the default for all interfaces added to a bridge.

-autoedge *interface*
Disable automatic edge status on *interface*.

ptp *interface*
Set the *interface* as a point to point link. This is required for straight transitions to forwarding and should be enabled on a direct link to another RSTP capable switch.

-ptp *interface*
Disable point to point link status on *interface*. This should be disabled for a half duplex link and for an interface connected to a shared network segment, like a hub or a wireless network.

autoptp *interface*
Automatically detect the point to point status on *interface* by checking the full duplex link status. This is the default for interfaces added to the bridge.

-autoptp *interface*
Disable automatic point to point link detection on *interface*.

maxage *seconds*
Set the time that a Spanning Tree protocol configuration is valid. The default is 20 seconds. The minimum is 6 seconds and the maximum is 40 seconds.

fwddelay *seconds*
Set the time that must pass before an interface begins forwarding packets when Spanning Tree is enabled. The default is 15 seconds. The minimum is 4 seconds and the maximum is 30 seconds.

hellotime *seconds*
Set the time between broadcasting of Spanning Tree protocol configuration messages. The hello time may only be changed when operating in legacy stp mode. The default is 2 seconds. The minimum is 1 second and the maximum is 2 seconds.

priority *value*
Set the bridge priority for Spanning Tree. The default is 32768. The minimum is 0 and the maximum is 61440.

proto *value*
Set the Spanning Tree protocol. The default is rstp. The available options are stp and rstp.

holdcnt *value*
Set the transmit hold count for Spanning Tree. This is the number of packets transmitted before being rate limited. The default is 6. The minimum is 1 and the maximum is 10.

ifpriority *interface value*
Set the Spanning Tree priority of *interface* to *value*. The default is 128. The minimum is 0 and the maximum is 240.

ifpathcost *interface value*
Set the Spanning Tree path cost of *interface* to *value*. The default is calculated from the link speed. To change a previously selected path cost back to automatic, set the cost to 0. The minimum is 1 and the maximum is 200000000.

ifmaxaddr *interface size*
Set the maximum number of hosts allowed from an interface, packets with unknown source addresses are dropped until an existing host cache entry expires or is removed. Set to 0 to disable.

Link Aggregation and Link Failover Parameters

The following parameters are specific to lagg interfaces:

laggtype *type*

When creating a lagg interface the type can be specified as either **ethernet** or **infiniband**. If not specified ethernet is the default lagg type.

laggport *interface*

Add the interface named by *interface* as a port of the aggregation interface.

-laggport *interface*

Remove the interface named by *interface* from the aggregation interface.

laggproto *proto*

Set the aggregation protocol. The default is failover. The available options are failover, lacp, loadbalance, roundrobin, broadcast and none.

lagghash *option*[,*option*]

Set the packet layers to hash for aggregation protocols which load balance. The default is "l2,l3,l4". The options can be combined using commas.

l2 src/dst mac address and optional vlan number.
l3 src/dst address for IPv4 or IPv6.
l4 src/dst port for TCP/UDP/SCTP.

-use_flowid

Enable local hash computation for RSS hash on the interface. The loadbalance and lacp modes will use the RSS hash from the network card if available to avoid computing one, this may give poor traffic distribution if the hash is invalid or uses less of the protocol header information. **-use_flowid** disables use of RSS hash from the network card. The default value can be set via the *net.link.lagg.default_use_flowid* sysctl(8) variable. 0 means "disabled" and 1 means "enabled".

use_flowid

Use the RSS hash from the network card if available.

flowid_shift *number*

Set a shift parameter for RSS local hash computation. Hash is calculated by using flowid bits in a packet header mbuf which are shifted by the number of this parameter.

use_numa
Enable selection of egress ports based on the native numa(4) domain for the packets being transmitted. This is currently only implemented for lacp mode. This works only on numa(4) hardware, running a kernel compiled with the numa(4) option, and when interfaces from multiple numa(4) domains are ports of the aggregation interface.

-use_numa
Disable selection of egress ports based on the native numa(4) domain for the packets being transmitted.

lacp_fast_timeout
Enable lacp fast-timeout on the interface.

-lacp_fast_timeout
Disable lacp fast-timeout on the interface.

lacp_strict
Enable lacp strict compliance on the interface. The default value can be set via the *net.link.lagg.lacp.default_strict_mode* sysctl(8) variable. 0 means "disabled" and 1 means "enabled".

-lacp_strict
Disable lacp strict compliance on the interface.

rr_limit *number*
Configure a stride for an interface in round-robin mode. The default stride is 1.

Generic IP Tunnel Parameters

The following parameters apply to IP tunnel interfaces, gif(4):

tunnel *src_addr dest_addr*
Configure the physical source and destination address for IP tunnel interfaces. The arguments *src_addr* and *dest_addr* are interpreted as the outer source/destination for the encapsulating IPv4/IPv6 header.

-tunnel
Unconfigure the physical source and destination address for IP tunnel interfaces previously configured with **tunnel**.

deletetunnel

Another name for the **-tunnel** parameter.

accept_rev_ethip_ver
Set a flag to accept both correct EtherIP packets and ones with reversed version field. Enabled by default. This is for backward compatibility with FreeBSD 6.1, 6.2, 6.3, 7.0, and 7.1.

-accept_rev_ethip_ver
Clear a flag **accept_rev_ethip_ver**.

ignore_source
Set a flag to accept encapsulated packets destined to this host independently from source address. This may be useful for hosts, that receive encapsulated packets from the load balancers.

-ignore_source
Clear a flag **ignore_source**.

send_rev_ethip_ver
Set a flag to send EtherIP packets with reversed version field intentionally. Disabled by default. This is for backward compatibility with FreeBSD 6.1, 6.2, 6.3, 7.0, and 7.1.

-send_rev_ethip_ver
Clear a flag **send_rev_ethip_ver**.

GRE Tunnel Parameters

The following parameters apply to GRE tunnel interfaces, gre(4):

tunnel *src_addr dest_addr*
Configure the physical source and destination address for GRE tunnel interfaces. The arguments *src_addr* and *dest_addr* are interpreted as the outer source/destination for the encapsulating IPv4/IPv6 header.

-tunnel
Unconfigure the physical source and destination address for GRE tunnel interfaces previously configured with **tunnel**.

deletetunnel
Another name for the **-tunnel** parameter.

grekey *key*
Configure the GRE key to be used for outgoing packets. Note that gre(4) will always accept

GRE packets with invalid or absent keys. This command will result in a four byte MTU reduction on the interface.

Packet Filter State Table Sychronisation Parameters

The following parameters are specific to pfsync(4) interfaces:

syncdev *iface*
Use the specified interface to send and receive pfsync state synchronisation messages.

-syncdev
Stop sending pfsync state synchronisation messages over the network.

syncpeer *peer_address*
Make the pfsync link point-to-point rather than using multicast to broadcast the state synchronisation messages. The peer_address is the IP address of the other host taking part in the pfsync cluster.

-syncpeer
Broadcast the packets using multicast.

maxupd *n*
Set the maximum number of updates for a single state which can be collapsed into one. This is an 8-bit number; the default value is 128.

defer Defer transmission of the first packet in a state until a peer has acknowledged that the associated state has been inserted.

-defer Do not defer the first packet in a state. This is the default.

VLAN Parameters

The following parameters are specific to vlan(4) interfaces:

vlan *vlan_tag*
Set the VLAN tag value to *vlan_tag*. This value is a 12-bit VLAN Identifier (VID) which is used to create an 802.1Q or 802.1ad VLAN header for packets sent from the vlan(4) interface. Note that **vlan** and **vlandev** must both be set at the same time.

vlanproto *vlan_proto*
Set the VLAN encapsulation protocol to *vlan_proto*. Supported encapsulation protocols are currently:

802.1Q
Default.

802.1ad

QinQ
Same as **802.1ad**.

vlanpcp *priority_code_point*
Priority code point (PCP) is an 3-bit field which refers to the IEEE 802.1p class of service and maps to the frame priority level.

Values in order of priority are: **1** (Background (lowest)), **0** (Best effort (default)), **2** (Excellent effort), **3** (Critical applications), **4** (Video, < 100ms latency and jitter), **5** (Voice, < 10ms latency and jitter), **6** (Internetwork control), **7** (Network control (highest)).

vlandev *iface*
Associate the physical interface *iface* with a vlan(4) interface. Packets transmitted through the vlan(4) interface will be diverted to the specified physical interface *iface* with 802.1Q VLAN encapsulation. Packets with 802.1Q encapsulation received by the parent interface with the correct VLAN Identifier will be diverted to the associated vlan(4) pseudo-interface. The vlan(4) interface is assigned a copy of the parent interface's flags and the parent's Ethernet address. The **vlandev** and **vlan** must both be set at the same time. If the vlan(4) interface already has a physical interface associated with it, this command will fail. To change the association to another physical interface, the existing association must be cleared first.

Note: if the hardware tagging capability is set on the parent interface, the vlan(4) pseudo interface's behavior changes: the vlan(4) interface recognizes that the parent interface supports insertion and extraction of VLAN tags on its own (usually in firmware) and that it should pass packets to and from the parent unaltered.

-vlandev [*iface*]
If the driver is a vlan(4) pseudo device, disassociate the parent interface from it. This breaks the link between the vlan(4) interface and its parent, clears its VLAN Identifier, flags and its link address and shuts the interface down. The *iface* argument is useless and hence deprecated.

Virtual eXtensible LAN Parameters

The following parameters are used to configure vxlan(4) interfaces.

vxlanid *identifier*

This value is a 24-bit VXLAN Network Identifier (VNI) that identifies the virtual network segment membership of the interface.

vxlanlocal *address*
The source address used in the encapsulating IPv4/IPv6 header. The address should already be assigned to an existing interface. When the interface is configured in unicast mode, the listening socket is bound to this address.

vxlanremote *address*
The interface can be configured in a unicast, or point-to-point, mode to create a tunnel between two hosts. This is the IP address of the remote end of the tunnel.

vxlangroup *address*
The interface can be configured in a multicast mode to create a virtual network of hosts. This is the IP multicast group address the interface will join.

vxlanlocalport *port*
The port number the interface will listen on. The default port number is 4789.

vxlanremoteport *port*
The destination port number used in the encapsulating IPv4/IPv6 header. The remote host should be listening on this port. The default port number is 4789. Note some other implementations, such as Linux, do not default to the IANA assigned port, but instead listen on port 8472.

vxlanportrange *low high*
The range of source ports used in the encapsulating IPv4/IPv6 header. The port selected within the range is based on a hash of the inner frame. A range is useful to provide entropy within the outer IP header for more effective load balancing. The default range is between the sysctl(8) variables *net.inet.ip.portrange.first* and *net.inet.ip.portrange.last*

vxlantimeout *timeout*
The maximum time, in seconds, before an entry in the forwarding table is pruned. The default is 1200 seconds (20 minutes).

vxlanmaxaddr *max*
The maximum number of entries in the forwarding table. The default is 2000.

vxlandev *dev*
When the interface is configured in multicast mode, the **dev** interface is used to transmit IP

multicast packets.

vxlanttl *ttl*
The TTL used in the encapsulating IPv4/IPv6 header. The default is 64.

vxlanlearn
The source IP address and inner source Ethernet MAC address of received packets are used to dynamically populate the forwarding table. When in multicast mode, an entry in the forwarding table allows the interface to send the frame directly to the remote host instead of broadcasting the frame to the multicast group. This is the default.

-vxlanlearn
The forwarding table is not populated by received packets.

vxlanflush
Delete all dynamically-learned addresses from the forwarding table.

vxlanflushall
Delete all addresses, including static addresses, from the forwarding table.

CARP Parameters

The following parameters are used to configure carp(4) protocol on an interface:

vhid *n*
Set the virtual host ID. This is a required setting to initiate carp(4). If the virtual host ID does not exist yet, it is created and attached to the interface, otherwise configuration of an existing vhid is adjusted. If the **vhid** keyword is supplied along with an "inet6" or "inet" address, then this address is configured to be run under control of the specified vhid. Whenever a last address that refers to a particular vhid is removed from an interface, the vhid is automatically removed from interface and destroyed. Any other configuration parameters for the carp(4) protocol should be supplied along with the **vhid** keyword. Acceptable values for vhid are 1 to 255.

advbase *seconds*
Specifies the base of the advertisement interval in seconds. The acceptable values are 1 to 255. The default value is 1.

advskew *interval*
Specifies the skew to add to the base advertisement interval to make one host advertise slower than another host. It is specified in 1/256 of seconds. The acceptable values are 1 to 254. The default value is 0.

pass *phrase*
Set the authentication key to *phrase*.

state *state*
Forcibly change state of a given vhid. The following states are recognized: **MASTER** and **BACKUP**.

peer *address*
Set the address to send (IPv4) carp(4) announcements to.

mcast Restore the default destination address for (IPv4) carp(4) announcements, which is 224.0.0.18.

peer6 *address*
Set the address to send (IPv6) carp(4) announcements to.

mcast6
Restore the default destination address for (IPv4) carp(4) announcements, which is ff02::12.

ENVIRONMENT

The following environment variables affect the execution of **ifconfig**:

IFCONFIG_FORMAT This variable can contain a specification of the output format. See the description of the **-f** flag for more details.

EXAMPLES

Assign the IPv4 address 192.0.2.10, with a network mask of 255.255.255.0, to the interface em0:

```
# ifconfig em0 inet 192.0.2.10 netmask 255.255.255.0
```

Add the IPv4 address 192.0.2.45, with the CIDR network prefix /28, to the interface em0:

```
# ifconfig em0 inet 192.0.2.45/28 alias
```

Remove the IPv4 address 192.0.2.45 from the interface em0:

```
# ifconfig em0 inet 192.0.2.45 -alias
```

Enable IPv6 functionality of the interface:

```
# ifconfig em0 inet6 -ifdisabled
```

Add the IPv6 address 2001:DB8:DBDB::123/48 to the interface em0:

```
# ifconfig em0 inet6 2001:db8:bdbd::123 prefixlen 48 alias
```

Note that lower case hexadecimal IPv6 addresses are acceptable.

Remove the IPv6 address added in the above example, using the / character as shorthand for the network prefix:

```
# ifconfig em0 inet6 2001:db8:bdbd::123/48 -alias
```

Configure a single CARP redundant address on igb0, and then switch it to be master:

```
# ifconfig igb0 vhid 1 10.0.0.1/24 pass foobar up
# ifconfig igb0 vhid 1 state master
```

Configure the interface xl0, to use 100baseTX, full duplex Ethernet media options:

```
# ifconfig xl0 media 100baseTX mediaopt full-duplex
```

Label the em0 interface as an uplink:

```
# ifconfig em0 description "Uplink to Gigabit Switch 2"
```

Create the software network interface gif1:

```
# ifconfig gif1 create
```

Destroy the software network interface gif1:

```
# ifconfig gif1 destroy
```

Display available wireless networks using wlan0:

```
# ifconfig wlan0 list scan
```

Display inet and inet6 address subnet masks in CIDR notation

```
# ifconfig -f inet:cidr,inet6:cidr
```

Display interfaces that are up with the exception of loopback

```
# ifconfig -a -u -G lo
```

Display a list of interface names beloning to the wlan group:

```
# ifconfig -g wlan
wlan0
wlan1
```

Display details about the interfaces belonging to the wlan group:

```
# ifconfig -a -g wlan
wlan0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 75:4c:61:6b:7a:73
        inet6 fe80::4c75:636a:616e:ffd8%wlan0 prefixlen 64 scopeid 0x3
        inet6 2001:5761:6e64:6152:6f6d:616e:fea4:ffe2 prefixlen 64 autoconf
```

```
        inet 192.168.10.5 netmask 0xffffff00 broadcast 192.168.10.255
        groups: wlan
        ssid "Hotspot" channel 11 (2462 MHz 11g) bssid 12:34:ff:ff:43:21
        regdomain ETSI country DE authmode WPA2/802.11i privacy ON
        deftxkey UNDEF AES-CCM 2:128-bit AES-CCM 3:128-bit txpower 30 bmiss 10
        scanvalid 60 protmode CTS wme roaming MANUAL
        parent interface: iwm0
        media: IEEE 802.11 Wireless Ethernet DS/2Mbps mode 11g
        status: associated
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
wlan1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        ether 00:50:69:6f:74:72
        groups: wlan
        ssid "" channel 2 (2417 MHz 11g)
        regdomain FCC country US authmode OPEN privacy OFF txpower 30 bmiss 7
        scanvalid 60 bgscan bgscanintvl 300 bgscanidle 250 roam:rssi 7
        roam:rate 5 protmode CTS wme bintval 0
        parent interface: rum0
        media: IEEE 802.11 Wireless Ethernet autoselect (autoselect)
        status: no carrier
        nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
```

Set a randomly-generated MAC address on tap0:

```
# ifconfig tap0 ether random
```

DIAGNOSTICS

Messages indicating the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface’s configuration.

SEE ALSO

netstat(1), carp(4), gif(4), netintro(4), pfsync(4), polling(4), vlan(4), vxlan(4), devd.conf(5), devd(8), jail(8), rc(8), routed(8), sysctl(8)

HISTORY

The **ifconfig** utility appeared in 4.2BSD.

BUGS

Basic IPv6 node operation requires a link-local address on each interface configured for IPv6. Normally, such an address is automatically configured by the kernel on each interface added to the system or enabled; this behavior may be disabled by setting per-interface flag **-auto_linklocal**. The default value of this flag is 1 and can be disabled by using the sysctl MIB variable *net.inet6.ip6.auto_linklocal*.

Do not configure IPv6 addresses with no link-local address by using **ifconfig**. It can result in unexpected behaviors of the kernel.