NAME

dhcpcd.conf — dhcpcd configuration file

DESCRIPTION

Although **dhcpcd** can do everything from the command line, there are cases where it's just easier to do it once in a configuration file. Most of the options found in *dhcpcd*(8) can be used here. The first word on the line is the option and the rest of the line is the value. Leading and trailing whitespace for the option and value are trimmed. You can escape characters in the value using the \ character. Comments can be prefixed with the # character. String values should be quoted with the " character.

Here's a list of available options:

allowinterfaces *pattern*

When discovering interfaces, the interface name must match *pattern* which is a space or comma separated list of patterns passed to *fnmatch*(3). If the same interface is matched in **denyinterfaces** then it is still denied.

denyinterfaces *pattern*

When discovering interfaces, the interface name must not match *pattern* which is a space or comma separated list of patterns passed to *fnmatch*(3).

anonymous

Enables Anonymity Profiles for DHCP, RFC 7844. Any DUID is ignored and ClientID is set to LL only. All non essential options are then masked at this point, but they could be unmasked by explicitly requesting the option **after** the **anonymous** option is processed. As such, the **anonymous** option **should** be the last option in the configuration unless you really want to send something which could identify you. **dhcpcd** will not try and reboot an old lease, it will go straight into DISCOVER/SOLICIT.

randomise_hwaddr

Forces a hardware address randomisation when the interface is brought up or when the carrier is lost. This is generally used in tandem with the anonymous option.

arping *address* [address]

dhcpcd will arping each address in order before attempting DHCP. If an address is found, we will select the replying hardware address as the profile, otherwise the IP address. Example:

```
interface bge0
arping 192.168.0.1

# My specific 192.168.0.1 network
profile dd:ee:aa:dd:bb:ee
static ip_address=192.168.0.10/24

# A generic 192.168.0.1 network
profile 192.168.0.1
static ip_address=192.168.0.98/24
```

authprotocol *protocol* [*algorithm* [*rdm*]]

Authenticate DHCP messages. See the Supported Authentication Protocols section. If *protocol* is *token* then *algorithm is* snd_secretid/rcv_secretid so you can send and receive different tokens.

authtoken *secretid realm expire key*

Define a shared key for use in authentication. *realm* can be "" to for use with the *delayed* protocol. *expire* is the date the token expires and should be formatted "yyy-mm-dd HH:MM". You can use the keyword *forever* or *0* which means the token never expires. For the token protocol, *secretid* needs to be 0 and *realm* needs to be "". If **dhcpcd** has the error

```
dhcp_auth_encode: Invalid argument
```

then it means that **dhcpcd** could not find the correct authentication token in your configuration.

background
Fork to the background immediately. This is useful for startup scripts which don't disable link messages for carrier status.

blacklist *address*[/cidr]
Ignores all packets from *address*[/cidr].

whitelist *address*[/cidr]
Only accept packets from *address*[/cidr]. **blacklist** is ignored if **whitelist** is set.

bootp Be a BOOTP client. Basically, this just doesn't send a DHCP Message Type option and will only interact with a BOOTP server. All other DHCP options still work.

broadcast
Instructs the DHCP server to broadcast replies back to the client. Normally this is only set for non-Ethernet interfaces, such as FireWire and InfiniBand. In most cases, **dhcpcd** will set this automatically.

controlgroup *group*
Sets the group ownership of */run/dhcpcd/sock* so that users other than root can connect to **dhcpcd**.

debug Echo debug messages to the stderr and syslog.

dev *value*
Load the *value* */dev* management module. **dhcpcd** will load the first one found to work, if any.

env *value*
Push *value* to the environment for use in *dhcpcd-run-hooks*(8). For example, you can force the hostname hook to always set the hostname with **env** *force_hostname=YES*. Or set which driver *wpa_supplicant*(8) should use with **env** *wpa_supplicant_driver=nl80211*

If the hostname is set, it will be will set to the FQDN if possible as per RFC 4702, section 3.1. If the FQDN option is missing, **dhcpcd** will still try and set a FQDN from the hostname and domain options for consistency. To override this, set **env** *hostname_fqdn=[YES|NO|SERVER]*. A value of *SERVER* means just what the server says, don't manipulate it. This could lead to an inconsistent hostname on a DHCPv4 and DHCPv6 network where the DHCPv4 hostname is short and the DHCPv6 has an FQDN. DHCPv6 has no hostname option.

clientid *string*
Send the *clientid*. If the string is of the format 01:02:03 then it is encoded as hex. For interfaces whose hardware address is longer than 8 bytes, or if the *clientid* is an empty string then **dhcpcd** sends a default *clientid* of the hardware family and the hardware address.

duid [ll | lt | uuid | value]
Use a DHCP Unique Identifier. If persistent storage is available then a DUID-LLT (link local address + time) is generated, otherwise DUID-LL is generated (link local address). The DUID type can be hinted as an optional parameter if the file */var/lib/dhcpcd/duid* does not exist. If not*ll*, *lt* or *uuid* then *value* will be converted from 00:11:22:33 format. This, plus the IAID will be used as the **clientid**. The DUID generated will be held in*/var/lib/dhcpcd/duid* and should not be copied to other hosts. This file also takes precedence over the above rules except for setting a value.

iaid *iaid*
Set the Interface Association Identifier to *iaid*. This option must be used in an**interface** block. This defaults to the VLANID (prefixed with 0xff) for the interface if set, otherwise the last 4 bytes of the hardware address assigned to the interface. Each instance of this should be unique within the scope of the client and **dhcpcd** warns if a conflict is detected. If there is a conflict, it is only a problem if the conflicted IAIDs are used on the same network.

dhcp Enable DHCP on the interface, on by default.

dhcp6 Enable DHCPv6 on the interface, on by default.

ipv4 Enable IPv4 on the interface, on by default.

ipv6 Enable IPv6 on the interface, on by default.

request [*address*]
Request the *address* in the DHCP DISCOVER message. There is no guarantee this is the address the DHCP server will actually give. If no*address* is gi ven then the first address currently assigned to the *interface* is used.

inform [*address*[*/cidr*[*/broadcast_address*]]]
Behaves like **request** as above, but sends a DHCP INFORM instead of DISCOVER/REQUEST. This does not get a lease as such, just notifies the DHCP server of the *address* in use. You should also include the optional *cidr* network number in case the address is not already configured on the interface. **dhcpcd** remains running and pretends it has an infinite lease. **dhcpcd** will not de-configure the interface when it exits. If**dhcpcd** f ails to contact a DHCP server then it returns a failure instead of falling back on IPv4LL.

inform6
Performs a DHCPv6 Information Request. No address is requested or specified, but all other DHCPv6 options are allowed. This is normally performed automatically when an IPv6 Router Advertisement indicates that the client should perform this operation. This option is only needed when **dhcpcd** is not processing IPv6 RA messages and the need for a DHCPv6 Information Request exists.

persistent
dhcpcd normally de-configures the interface and configuration when it exits. Sometimes, this isn't desirable if, for example, you have root mounted over NFS or SSH clients connect to this host and they need to be notified of the host shutting down. You can use this option to stop this from happening.

fallback *profile*
Fall back to using this profile if DHCP fails. This allows you to configure a static profile instead of using ZeroConf.

fallback_time *seconds*
Start fallback after *seconds*. The default is 5 seconds.

hostname *name*
Sends the hostname *name* to the DHCP server so it can be registered in DNS. If *name* is an empty string then the current system hostname is sent. If *name* is a FQDN (i.e., contains a .) then it will be encoded as such.

hostname_short
Sends the short hostname to the DHCP server instead of the FQDN. This is useful because DHCP servers will not register the FQDN in their DNS if the domain part does not match theirs.

Also, see the **env** option above to control how the hostname is set on the host.

ia_na [*iaid* [/ address]]
Request a DHCPv6 Normal Address for *iaid*. *iaid* defaults to the **iaid** option as described above. You can request more than one ia_na by specifying a unique *iaid* for each one.

ia_ta [*iaid*]
Request a DHCPv6 Temporary Address for *iaid*. You can request more than one ia_ta by specifying a unique *iaid* for each one.

ia_pd [*iaid* [/ *prefix* / *prefix_len*] [*interface* [/ *sla_id* [/ *prefix_len* [/ *suffix*]]]]]
Request a DHCPv6 Delegated Prefix for *iaid*. This option must be used in an **interface** block. Unless a *sla_id* of 0 is assigned with the same resultant prefix length as the dele gation, a reject route is installed for the Delegated Prefix to stop unallocated addresses being resolved upstream. If no *interface* is gi ven then we will assign a prefix to every other interface with a *sla_id* equivalent to the interface index assigned by the OS. Otherwise addresses are only assigned for each *interface* and *sla_id*. To avoid delegating to any interface, use - as the invalid interface name. Each assigned address will have a *suffix*, defaulting to 1. If the *suffix* is 0 then a SLAAC address is assigned. You cannot assign a prefix to the requesting interface unless the DHCPv6 server supports the RFC 6603 Prefix Exclude Option. **dhcpcd** has to be running for all the interfaces it is delegating to. A default *prefix_len* of 64 is assumed, unless the maximum *sla_id* does not fit. In this case *prefix_len* is increased to the highest multiple of 8 that can accommodate the *sla_id*. *sla_id* is an integer which must be unique inside the *iaid* and is added to the prefix which must fit inside *prefix_len* less the length of the delegated prefix. You can specify multiple *interface* / *sla_id* / *prefix_len* per **ia_pd**, space separated. IPv6RS should be disabled globally when requesting a Prefix Delegation.

In the following example eth0 is the externally facing interface to be configured for both IPv4 and IPv6. The DHCPv4 server will provide us with an IPv4 address and a default route. The DHCPv6 server is going to provide us with an IPv6 address, a default route and a /64 subnet to be delegated to the internal interface. The eth1 interface will be automatically configured for IPv6 using the first address (::1) from the delegated prefix. A second prefix is requested and assigned to two other interfaces. *rtadvd*(8) can be used with an empty configuration file on eth1, eth2 and eth3, to provide automatic IPv6 address configuration for the internal network.

```
noipv6rs                  # disable routing solicitation
denyinterfaces eth2       # Don't touch eth2 at all
interface eth0
  ipv6rs                  # enable routing solicitation for eth0
  ia_na 1                 # request an IPv6 address
  ia_pd 2 eth1/0          # request a PD and assign it to eth1
  ia_pd 3 eth2/1 eth3/2   # req a PD and assign it to eth2 and eth3
  ia_pd 4 -               # request a PD but don't assign it
```

ipv4only
Only configure IPv4.

ipv6only
Only configure IPv6.

fqdn [disable | none | ptr | both]
none will not ask the DHCP server to update DNS. *ptr* just asks the DHCP serv er to update the PTR record of the host in DNS, whereas *both* also updates the A record. *disable* will disable the FQDN option. The default is *both*. **dhcpcd** itself never does any DNS updates. **dhcpcd** encodes the FQDN hostname as specified in RFC 1035.

interface *interface*
Subsequent options are only parsed for this *interface*. Pattern matching is allowed by *fnmatch*(3).

ipv4ll_time *seconds*
Wait for *seconds* before starting IPv4LL. The default is 5 seconds.

ipv6ra_autoconf
Generate SLAAC addresses for each Prefix advertised by an IPv6 Router Advertisement message with the Auto flag set. On by default.

ipv6ra_noautoconf
Disables the above option.

ipv6ra_fork
By default, when **dhcpcd** receives an IPv6 Router Advertisement, **dhcpcd** will only fork to the background if the RA contains at least one unexpired RDNSS option and a valid prefix or no DHCPv6 instruction. Set this option so to make **dhcpcd** always fork on a RA.

ipv6rs Enables IPv6 Router Advertisement solicitation. This is on by default, but is documented here in the case where it is disabled globally but needs to be enabled for one interface.

lastlease
If **dhcpcd** cannot obtain a lease, then try to use the last lease acquired for the interface.

lastleaseextend
Same as the above, but the lease will be retained even if it expires. **dhcpcd** will give it up if any other host tries to claim it for their own via ARP. This violates RFC 2131, section 3.7, which states the lease should be dropped once it has expired.

leasetime *seconds*
Request DHCP a lease time of *seconds*. *-1* represents an infinite lease time. By default **dhcpcd** does not request any lease time and leaves it in the hands of the DHCP server. It is not possible to request a DHCPv6 lease time as this is not RFC compliant. See RFC 8415 21.4, 21.6, 21.21 and 21.22.

link_rcvbuf *size*
Override the size of the link receive buffer from the kernel default. While**dhcpcd** will reco ver from link buffer overflows, this may not be desirable on heavily loaded systems.

logfile *logfile*
Writes to the specified *logfile*. **dhcpcd** still writes to *syslog*(3). The *logfile* is reopened when **dhcpcd** receives the SIGUSR2 signal.

metric *metric*
Metrics are used to prefer an interface over another one, lowest wins. **dhcpcd** will supply a default metric of 1000 + *if_nametoindex*(3). This will be offset by 2000 for wireless interfaces, with additional offsets of 1000000 for IPv4LL and 2000000 for roaming interfaces.

mudurl *url*
Specifies the URL for a Manufacturer Usage Description (MUD). The description is used by upstream network devices to instantiate any desired access lists. See draft-ietf-opsawg-mud for more information.

noalias
Any pre-existing IPv4 addresses will be removed from the interface when adding a new IPv4 address.

noarp Don’t send any ARP requests. This also disables IPv4LL.

arp_persistdefence
Keep the IP address even if defence fails upon IP Address conflict.

noauthrequired
Don’t require authentication even though we requested it. Also allows FORCERENEW and RECONFIGURE messages without authentication.

nodelay
Don’t delay for an initial randomised time when starting protocols.

nodev Don’t load */dev* management modules.

nodhcp Don't start DHCP or listen to DHCP messages. This is only useful when allowing IPv4LL.

nodhcp6
Don't start DHCPv6 or listen to DHCPv6 messages. Normally DHCPv6 is started by an IPv6 Router Advertisement instruction or configuration.

nogateway
Don't install any default routes.

gateway
Install a default route if available (default).

nohook *script*
Don't run this hook script. Matches full name, or prefixed with 2 numbers optionally ending with *.sh*.

So to stop **dhcpcd** from touching your DNS settings or starting wpa_supplicant you would do:-
nohook resolv.conf, wpa_supplicant

noipv4 Don't attempt to configure an IPv4 address.

noipv4ll
Don't attempt to obtain an IPv4LL address if we failed to get one via DHCP. See*RFC 3927*.

noipv6 Don't solicit or accept IPv6 Router Advertisements and DHCPv6.

noipv6rs
Don't solicit or accept IPv6 Router Advertisements.

nolink Don't receive link messages about carrier status. You should only set this for buggy interface drivers.

nosyslog
Disable writing to syslog(3).

noup Don't bring the interface up when in manager mode.

option *option*
Requests the *option* from the server. It can be a variable to be used in *dhcpcd-run-hooks*(8) or the numerical value. You can specify more *option*s separated by commas, spaces or more **option** lines. Prepend dhcp6_ to *option* to request a DHCPv6 option. If no DHCPv6 options are configured, then DHCPv4 options are mapped to equivalent DHCPv6 options.

Prepend nd_ to *option* to handle ND options, but this only works for the **nooption**, **reject** and **require** options.

To see a list of options you can use, call **dhcpcd** with the `-V`, `--variables` argument.

nooption *option*
Remove the option from the message before it's processed.

require *option*
Requires the *option* to be present in all messages, otherwise the message is ignored. To enforce that **dhcpcd** only responds to DHCP servers and not BOOTP servers, you can **require** *dhcp_message_type*. This isn't an exact science though because a BOOTP server can send DHCP-like options.

reject *option*
Reject a message that contains the *option*. This is useful when you cannot use**require** to select / de-select BOOTP messages.

destination *option*
If **dhcpcd.conf** detects an address added to a point to point interface (PPP, TUN, etc) then it will set the listed DHCP options to the destination address of the interface.

profile *name*
Subsequent options are only parsed for this profile *name*.

quiet Suppress any dhcpcd output to the console, except for errors.

reboot *seconds*
Allow *reboot* seconds before moving to the DISCOVER phase if we have an old lease to use. Allow *reboot* seconds before starting fallback states from the DISCOVER phase. IPv4LL is started when the first *reboot* timeout is reached. The default is 5 seconds. A setting of 0 seconds causes **dhcpcd.conf** to skip the reboot phase and go straight into DISCOVER. This is desirable for mobile users because if you change from network A to network B and they use the same subnet and the address from network A isn't in use on network B, then the DHCP server will remain silent even if authoritative which means **dhcpcd** will timeout before moving back to the DISCOVER phase. This has no effect on DHCPv6 other than skipping the reboot phase.

release
dhcpcd will release the lease prior to stopping the interface.

script *script*
Use *script* instead of the default */lib/dhcpcd/dhcpcd-run-hooks*.

request_time *seconds*
Request the lease for *seconds* before going back to DISCOVER. The default is 180 seconds.

ssid *ssid*
Subsequent options are only parsed for this wireless *ssid*.

slaac hwaddr | **private** | **token** *token* [**temp** | **temporary**]
Selects the interface identifier used for SLAAC generated IPv6 addresses. If **private** is used, a RFC 7217 address is generated. If **token** *token* is used then the token is combined with the prefix to make the final address. The **temporary** directive will create a temporary address for the prefix as well.

static *value*
Configures a static *value*. If you set**ip_address** then **dhcpcd** will not attempt to obtain a lease and will just use the value for the address with an infinite lease time. If you set an empty value this removes all prior static allocations to the same value. This is useful when using profiles and in the case of **ip_address** it will remove the static allocation. Note that setting 0.0.0.0 keeps the static allocation but waits for a 3rdparty to configure the address. If you set **ip6_address**, **dhcpcd** will continue auto-configuration as normal.

Here is an example which configures two static address, overriding the default IPv4 broadcast address, an IPv4 router, DNS and disables IPv6 auto-configuration. You could also use the **inform6** command here if you wished to obtain more information via DHCPv6. For IPv4, you should use the **inform** *ipaddress* option instead of setting a static address.

```
interface eth0
noipv6rs
static ip_address=192.168.0.10/24
static broadcast_address=192.168.0.63
static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 fd51:42f8:caae:d92e::1
```

Here is an example for PPP which gives the destination a default route. It uses the special *destination* keyword to insert the destination address into the value.

```
interface ppp0
static ip_address=0.0.0.0
destination routers
```

timeout *seconds*
Time out after *seconds*, instead of the default 30. A setting of 0 *seconds* causes **dhcpcd** to wait forever to get a lease. If **dhcpcd** is working on a single interface then **dhcpcd** will exit when a timeout occurs, otherwise **dhcpcd** will fork into the background. If using IPv4LL then **dhcpcd** start the IPv4LL process after the timeout and then wait a little longer before really timing out.

userclass *string*
Tag the DHCPv4 message with the userclass. You can specify more than one.

msuserclass *string*
Tag the DHCPv4 mesasge with the Microsoft userclass. Unlike the **userclass** option, this one can only be added once. It should only be used for Microsoft DHCP servers and the **vendorclassid** should be set to "MSFT 98" or "MSFT 5.0". This option is not RFC compliant.

vendor *code*,*value*
Add an encapsulated vendor-specific information option (DHCP Option 43). *code* should be between 1 and 254 inclusive. To add a raw vendor string, omit *code* but keep the comma. Examples.

Set the vendor option 01 with an IP address.
vendor 01,192.168.0.2
Set the vendor option 02 with a hex code.
vendor 02,01:02:03:04:05
Set the vendor option 03 with an IP address as a string.
vendor 03,\"192.168.0.2\"
Set un-encapsulated vendor option to hello world.
vendor ,"hello world"

vsio *en code,value*
Add an encapsulated vendor-specific information option (DHCP Option 125) with IANA assigned Enterprise Number *en* proceeding with the *code* which should be between 1 and 255 inclusive, and the *value* after the comma. Examples:

Set the vsio for enterprise number 155 option 01 with an IPv4 address.
vsio 155 01,192.168.1.1
Set the vsio for enterprise number 155 option 02 with a string.
vsio 155 02,"hello world"
Set the vsio for enterprise number 255 option 01 with a hex code.
vsio 255 01,01:02:03:04:05

vsio6 *en code,value*
Add an encapsulated vendor-specific information option (DHCPv6 Option 17) with IANA assigned Enterprise Number *en* proceeding with the *code* which should be between 1 and 65535 inclusive, and the *value* after the comma. Examples:

Set the vsio for enterprise number 155 option 01 with an IPv6 address.
vsio6 155 01,2001:0db8:85a3:0000:0000:8a2e:0370:7334
Set the vsio for enterprise number 155 option 02 with a string.
vsio6 155 02,"hello world"
Set the vsio for enterprise number 255 option 01 with a hex code.
vsio6 255 01,01:02:03:04:05

vendorclassid *string*
Set the DHCP Vendor Class. DHCPv6 has its own option as shown below. The default is dhcpcd-<version>:<os>:<machine>:<platform>. For example
dhcpcd-5.5.6:NetBSD-6.99.5:i386:i386
If not set then none is sent. Some badly configured DHCP servers reject unknown

vendorclassids. To work around it, try and impersonate Windows by using the MSFT vendorclassid.

vendclass *en data*
Add the DHCPv6 Vendor Indetifying Vendor Class with the IANA assigned Enterprise Number *en* with the *data*. This option can be set more than once to add more data, but the behaviour, as per RFC 3925 is undefined if the Enterprise Number differs.

waitip [4 | 6]
Wait for an address to be assigned before forking to the background. 4 means wait for an IPv4 address to be assigned. 6 means wait for an IPv6 address to be assigned. If no argument is given, **dhcpcd.conf** will wait for any address protocol to be assigned. It is possible to wait for more than one address protocol and **dhcpcd.conf** will only fork to the background when all waiting conditions are satisfied.

xidhwaddr
Use the last four bytes of the hardware address as the DHCP xid instead of a randomly generated number.

Defining new options

DHCP, ND and DHCPv6 allow for the use of custom options, and RFC 3925 vendor options for DHCP can also be supplied. Each option needs to be started with the **define**, **definend**, **define6** or **vendopt** directive. This can optionally be followed by both **embed** or **encap** options. Both can be specified more than once and **embed** must come before **encap**.

define *code type variable*
Defines the DHCP option *code* of *type* with a name of *variable* exported to *dhcpcd-run-hooks*(8).

definend *code type variable*
Defines the ND option *code* of *type* with a name of *variable* exported to *dhcpcd-run-hooks*(8), with a prefix of *nd_*.

define6 *code type variable*
Defines the DHCPv6 option *code* of *type* with a name of *variable* exported to *dhcpcd-run-hooks*(8), with a prefix of *dhcp6_*.

vendopt *code type variable*
Defines the Vendor-Identifying Vendor Options. The *code* is the IANA Enterprise Number which will uniquely describe the encapsulated options. *type* is normally *encap*. *variable* names the Vendor option to be exported.

embed *type variable*
Defines an embedded variable within the defined option. The length is determined by the *type*. If the *variable* is not the same as defined in the parent option, it is prefixed with the parent *variable* first with an underscore. If the *variable* has the name of *reserved* then it is not processed.

encap *code type variable*
Defines an encapsulated variable within the defined option. The length is determined by the *type*. If the*variable* is not the same as defined in the parent option, it is prefix ed with the parent *variable* first with an underscore.

Type prefix

These keywords come before the type itself, to describe it more fully. You can use more than one, but they must appear in the order listed below.

request Requests the option by default without having to be specified in user configuration.

norequest
This option cannot be requested, regardless of user configuration.

optional
This option is optional. Only makes sense for embedded options like the client FQDN option, where the FQDN string itself is optional.

index The option can appear more than once and will be indexed.

array The option data is split into a space separated array, each element being the same type.

truncated
The option might truncated from its normal length. The end of the normal size is zero padded on expansion. Currently this is only supported for ip6address where it's a prefix.

Types to define

The type directly affects the length of data consumed inside the option. Any remaining data is normally discarded. Lengths can be specified for string and binhex types, but this is generally with other data embedded afterwards in the same option.

ipaddress
An IPv4 address, 4 bytes.

ip6address
An IPv6 address, 16 bytes.

string [: **length**]
A NVT ASCII string of printable characters.

byte A byte.

bitflags: **flags**
A byte represented as a string of flags, most significant bit first. For example, using ABCDEFGH then A would equal 10000000, B 01000000, C 00100000, etc. If the bit is not set, the flag is not printed. A flag of 0 is not printed even if the bit position is set. This is to allow reservation of the first bits while assigning the last bits. A flag of 1 prints the bit set or unset.

int16 A signed 16bit integer, 2 bytes.

uint16 An unsigned 16bit integer, 2 bytes.

int32 A signed 32bit integer, 4 bytes.

uint32 An unsigned 32bit integer, 4 bytes.

flag A fixed value (1) to indicate that the option is present, 0 bytes.

domain An RFC 3397 encoded string.

dname An RFC 1035 validated string.

uri If an array then the first two bytes are the URI length inside the option data. Otherwise, the whole option data is the URI. As a space is not allowed in the URI encoding, the URIs are space separated.

binhex [: **length**]
Binary data expressed as hexadecimal.

embed Contains embedded options (implies encap as well).

encap Contains encapsulated options (implies embed as well).

option References an option from the global definition.

Example definition

```
# DHCP option 81, Fully Qualified Domain Name, RFC 4702
define 81 embed fqdn
embed byte flags
embed byte rcode1
embed byte rcode2
embed domain fqdn

# DHCP option 125, Vendor Specific Information Option, RFC 3925
define 125 encap vsio
embed uint32 enterprise_number
# Options defined for the enterprise number
encap 1 ipaddress ipaddress
```

Supported Authentication Protocols

token Sends a plain text token the server expects and matches a token sent by the server. The tokens do not have to be the same. If unspecified, the token with a *secretid* of 0 will be used in sending messages and validating received messages.

delayedrealm
Delayed Authentication. **dhcpcd** will send an authentication option with no key or MAC. The server will see this option, and select a key for **dhcpcd.conf**, writing the *realm* and *secretid* in it. **dhcpcd** will then look for an unexpired token with a matching *realm* and *secretid*. This token is used to authenticate all other messages.

delayed Same as above, but without a realm.

Supported Authentication Algorithms

If none specified, **hmac-md5** is the default.

hmac-md5

Supported Replay Detection Mechanisms

If none specified, **monotonic** is the default. If this is changed from what was previously used, or the means of calculating or storing it is broken, then the DHCP server will probably have to have its notion of the client's Replay Detection Value reset.

monocounter
Read the number in the file */var/lib/dhcpcd/dhcpcd-rdm.monotonic* and add one to it.

monotime
Create an NTP timestamp from the system time.

monotonic
Same as **monotime**.

SEE ALSO

fnmatch(3), *if_nametoindex*(3), *dhcpcd*(8), *dhcpcd-run-hooks*(8)

AUTHORS

Roy Marples <*roy@marples.name*>

BUGS

Please report them to https://roy.marples.name/projects/dhcpcd