NAME

systemd.network − Network configuration

SYNOPSIS

network.network

DESCRIPTION

A plain ini−style text file that encodes network configuration for matching network interfaces, used by **systemd-networkd**(8). See **systemd.syntax**(7) for a general description of the syntax.

The main network file must have the extension .network; other extensions are ignored. Networks are applied to links whenever the links appear.

The .network files are read from the files located in the system network directories /lib/systemd/network and /usr/local/lib/systemd/network, the volatile runtime network directory /run/systemd/network and the local administration network directory /etc/systemd/network. All configuration files are collectively sorted and processed in lexical order, regardless of the directories in which they live. However, files with identical filenames replace each other. Files in /etc/ have the highest priority, files in /run/ take precedence over files with the same name under /usr/. This can be used to override a system−supplied configuration file with a local file if needed. As a special case, an empty file (file size 0) or symlink with the same name pointing to /dev/null disables the configuration file entirely (it is "masked").

Along with the network file foo.network, a "drop−in" directory foo.network.d/ may exist. All files with the suffix ".conf" from this directory will be merged in the alphanumeric order and parsed after the main file itself has been parsed. This is useful to alter or add configuration settings, without having to modify the main configuration file. Each drop−in file must have appropriate section headers.

In addition to /etc/systemd/network, drop−in ".d" directories can be placed in /lib/systemd/network or /run/systemd/network directories. Drop−in files in /etc/ take precedence over those in /run/ which in turn take precedence over those in /lib/. Drop−in files under any of these directories take precedence over the main network file wherever located.

[MATCH] SECTION OPTIONS

The network file contains a [Match] section, which determines if a given network file may be applied to a given device; and a [Network] section specifying how the device should be configured. The first (in lexical order) of the network files that matches a given device is applied, all later files are ignored, even if they match as well.

A network file is said to match a network interface if all matches specified by the [Match] section are satisfied. When a network file does not contain valid settings in [Match] section, then the file will match all interfaces and **systemd−networkd** warns about that. Hint: to avoid the warning and to make it clear that all interfaces shall be matched, add the following:

```
Name=*
```

The following keys are accepted:

MACAddress=

A whitespace−separated list of hardware addresses. Use full colon−, hyphen− or dot−delimited hexadecimal. See the example below. This option may appear more than once, in which case the lists are merged. If the empty string is assigned to this option, the list of hardware addresses defined prior to this is reset.

Example:

```
MACAddress=01:23:45:67:89:ab 00−11−22−33−44−55 AABB.CCDD.EEFF
```

PermanentMACAddress=

A whitespace−separated list of hardware's permanent addresses. While *MACAddress=* matches the device's current MAC address, this matches the device's permanent MAC address, which may be

different from the current one. Use full colon−, hyphen− or dot−delimited hexadecimal. This option may appear more than once, in which case the lists are merged. If the empty string is assigned to this option, the list of hardware addresses defined prior to this is reset.

Path=
A whitespace−separated list of shell−style globs matching the persistent path, as exposed by the udev property *ID_PATH*.

Driver=
A whitespace−separated list of shell−style globs matching the driver currently bound to the device, as exposed by the udev property *ID_NET_DRIVER* of its parent device, or if that is not set, the driver as exposed by **ethtool −i** of the device itself. If the list is prefixed with a "!", the test is inverted.

Type=
A whitespace−separated list of shell−style globs matching the device type, as exposed by **networkctl list**. If the list is prefixed with a "!", the test is inverted. Some valid values are "ether", "loopback", "wlan", "wwan". Valid types are named either from the udev "DEVTYPE" attribute, or "ARPHRD_" macros in linux/if_arp.h, so this is not comprehensive.

Property=
A whitespace−separated list of udev property names with their values after equals sign ("="). If multiple properties are specified, the test results are ANDed. If the list is prefixed with a "!", the test is inverted. If a value contains white spaces, then please quote whole key and value pair. If a value contains quotation, then please escape the quotation with "\".

Example: if a .link file has the following:

Property=ID_MODEL_ID=9999 "ID_VENDOR_FROM_DATABASE=vendor name" "KEY=with \"quotation\""

then, the .link file matches only when an interface has all the above three properties.

Name=
A whitespace−separated list of shell−style globs matching the device name, as exposed by the udev property "INTERFACE", or device's alternative names. If the list is prefixed with a "!", the test is inverted.

WLANInterfaceType=
A whitespace−separated list of wireless network type. Supported values are "ad−hoc", "station", "ap", "ap−vlan", "wds", "monitor", "mesh−point", "p2p−client", "p2p−go", "p2p−device", "ocb", and "nan". If the list is prefixed with a "!", the test is inverted.

SSID=
A whitespace−separated list of shell−style globs matching the SSID of the currently connected wireless LAN. If the list is prefixed with a "!", the test is inverted.

BSSID=
A whitespace−separated list of hardware address of the currently connected wireless LAN. Use full colon−, hyphen− or dot−delimited hexadecimal. See the example in *MACAddress=*. This option may appear more than once, in which case the lists are merged. If the empty string is assigned to this option, the list is reset.

Host=
Matches against the hostname or machine ID of the host. See *ConditionHost=* in **systemd.unit**(5) for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

Virtualization=
Checks whether the system is executed in a virtualized environment and optionally test whether it is a specific implementation. See *ConditionVirtualization=* in **systemd.unit**(5) for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously

assigned value is cleared.

KernelCommandLine=
Checks whether a specific kernel command line option is set. See *ConditionKernelCommandLine=* in **systemd.unit**(5) for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

KernelVersion=
Checks whether the kernel version (as reported by **uname −r**) matches a certain expression. See *ConditionKernelVersion=* in **systemd.unit**(5) for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

Architecture=
Checks whether the system is running on a specific architecture. See *ConditionArchitecture=* in **systemd.unit**(5) for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

Firmware=
Checks whether the system is running on a machine with the specified firmware. See *ConditionFirmware=* in **systemd.unit**(5) for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, then previously assigned value is cleared.

[LINK] SECTION OPTIONS

The [Link] section accepts the following keys:

MACAddress=
The hardware address to set for the device.

MTUBytes=
The maximum transmission unit in bytes to set for the device. The usual suffixes K, M, G, are supported and are understood to the base of 1024.

Note that if IPv6 is enabled on the interface, and the MTU is chosen below 1280 (the minimum MTU for IPv6) it will automatically be increased to this value.

ARP=
Takes a boolean. If set to true, the ARP (low−level Address Resolution Protocol) for this interface is enabled. When unset, the kernel's default will be used.

For example, disabling ARP is useful when creating multiple MACVLAN or VLAN virtual interfaces atop a single lower−level physical interface, which will then only serve as a link/"bridge" device aggregating traffic to the same physical link and not participate in the network otherwise. Defaults to unset.

Multicast=
Takes a boolean. If set to true, the multicast flag on the device is enabled. Defaults to unset.

AllMulticast=
Takes a boolean. If set to true, the driver retrieves all multicast packets from the network. This happens when multicast routing is enabled. Defaults to unset.

Promiscuous=
Takes a boolean. If set to true, promiscuous mode of the interface is enabled. Defaults to unset.

Unmanaged=
Takes a boolean. When "yes", no attempts are made to bring up or configure matching links, equivalent to when there are no matching network files. Defaults to "no".

This is useful for preventing later matching network files from interfering with certain interfaces that are fully controlled by other applications.

Group=

Link groups are similar to port ranges found in managed switches. When network interfaces are added to a numbered group, operations on all the interfaces from that group can be performed at once. Takes an unsigned integer in the range 0...4294967295. Defaults to unset.

RequiredForOnline=
Takes a boolean or a minimum operational state and an optional maximum operational state. Please see **networkctl**(1) for possible operational states. When "yes", the network is deemed required when determining whether the system is online (including when running **systemd−networkd−wait−online**). When "no", the network is ignored when determining the online state. When a minimum operational state and an optional maximum operational state are set, "yes" is implied, and this controls the minimum and maximum operational state required for the network interface to be considered online.

Defaults to "yes" when *ActivationPolicy=* is not set, or set to "up", "always−up", or "bound". Defaults to "no" when *ActivationPolicy=* is set to "manual" or "down". This is forced to "no" when *ActivationPolicy=* is set to "always−down".

The network will be brought up normally (as configured by *ActivationPolicy=*), but in the event that there is no address being assigned by DHCP or the cable is not plugged in, the link will simply remain offline and be skipped automatically by **systemd−networkd−wait−online** if "RequiredForOnline=no".

RequiredFamilyForOnline=
Takes an address family. When specified, an IP address in the given family is deemed required when determining whether the link is online (including when running **systemd−networkd−wait−online**). Takes one of "ipv4", "ipv6", "both", or "any". Defaults to "any". Note that this option has no effect if "RequiredForOnline=no", or if "RequiredForOnline=" specifies a minimum operational state below "degraded".

ActivationPolicy=
Specifies the policy for **systemd−networkd** managing the link administrative state. Specifically, this controls how **systemd−networkd** changes the network device's "IFF_UP" flag, which is sometimes controlled by system administrators by running e.g., **ip link set dev eth0 up** or **ip link set dev eth0 down**, and can also be changed with **networkctl up eth0** or **networkctl down eth0**.

Takes one of "up", "always−up", "manual", "always−down", "down", or "bound". When "manual", **systemd−networkd** will not change the link's admin state automatically; the system administrator must bring the interface up or down manually, as desired. When "up" (the default) or "always−up", or "down" or "always−down", **systemd−networkd** will set the link up or down, respectively, when the interface is (re)configured. When "always−up" or "always−down", **systemd−networkd** will set the link up or down, respectively, any time **systemd−networkd** detects a change in the administrative state. When *BindCarrier=* is also set, this is automatically set to "bound" and any other value is ignored.

When the policy is set to "down" or "manual", the default value of *RequiredForOnline=* is "no". When the policy is set to "always−down", the value of *RequiredForOnline=* forced to "no".

The administrative state is not the same as the carrier state, so using "always−up" does not mean the link will never lose carrier. The link carrier depends on both the administrative state as well as the network device's physical connection. However, to avoid reconfiguration failures, when using "always−up", *IgnoreCarrierLoss=* is forced to true.

[SR−IOV] SECTION OPTIONS

The [SR−IOV] section accepts the following keys. Specify several [SR−IOV] sections to configure several SR−IOVs. SR−IOV provides the ability to partition a single physical PCI resource into virtual PCI functions which can then be injected into a VM. In the case of network VFs, SR−IOV improves north−south network performance (that is, traffic with endpoints outside the host machine) by allowing traffic to bypass the host machine’s network stack.

VirtualFunction=
Specifies a Virtual Function (VF), lightweight PCIe function designed solely to move data in and out. Takes an unsigned integer in the range 0...2147483646. This option is compulsory.

VLANId=
Specifies VLAN ID of the virtual function. Takes an unsigned integer in the range 1...4095.

QualityOfService=
Specifies quality of service of the virtual function. Takes an unsigned integer in the range 1...4294967294.

VLANProtocol=
Specifies VLAN protocol of the virtual function. Takes "802.1Q" or "802.1ad".

MACSpoofCheck=
Takes a boolean. Controls the MAC spoof checking. When unset, the kernel's default will be used.

QueryReceiveSideScaling=
Takes a boolean. Toggle the ability of querying the receive side scaling (RSS) configuration of the virtual function (VF). The VF RSS information like RSS hash key may be considered sensitive on some devices where this information is shared between VF and the physical function (PF). When unset, the kernel's default will be used.

Trust=
Takes a boolean. Allows to set trust mode of the virtual function (VF). When set, VF users can set a specific feature which may impact security and/or performance. When unset, the kernel's default will be used.

LinkState=
Allows to set the link state of the virtual function (VF). Takes a boolean or a special value "auto". Setting to "auto" means a reflection of the physical function (PF) link state, "yes" lets the VF to communicate with other VFs on this host even if the PF link state is down, "no" causes the hardware to drop any packets sent by the VF. When unset, the kernel's default will be used.

MACAddress=
Specifies the MAC address for the virtual function.

[NETWORK] SECTION OPTIONS

The [Network] section accepts the following keys:

Description=
A description of the device. This is only used for presentation purposes.

DHCP=
Enables DHCPv4 and/or DHCPv6 client support. Accepts "yes", "no", "ipv4", or "ipv6". Defaults to "no".

Note that DHCPv6 will by default be triggered by Router Advertisement, if that is enabled, regardless of this parameter. By enabling DHCPv6 support explicitly, the DHCPv6 client will be started regardless of the presence of routers on the link, or what flags the routers pass. See "IPv6AcceptRA=".

Furthermore, note that by default the domain name specified through DHCP, on Ubuntu, are used for name resolution. See option **UseDomains=** below.

See the [DHCPv4] or [DHCPv6] sections below for further configuration options for the DHCP client support.

DHCPServer=
Takes a boolean. If set to "yes", DHCPv4 server will be started. Defaults to "no". Further settings for the DHCP server may be set in the [DHCPServer] section described below.

LinkLocalAddressing=

Enables link−local address autoconfiguration. Accepts **yes**, **no**, **ipv4**, and **ipv6**. An IPv6 link−local address is configured when **yes** or **ipv6**. An IPv4 link−local address is configured when **yes** or **ipv4** and when DHCPv4 autoconfiguration has been unsuccessful for some time. (IPv4 link−local address autoconfiguration will usually happen in parallel with repeated attempts to acquire a DHCPv4 lease).

Defaults to **no** when *Bridge=yes* is set, and **ipv6** otherwise.

IPv6LinkLocalAddressGenerationMode=
Specifies how IPv6 link local address is generated. Takes one of "eui64", "none", "stable−privacy" and "random". When unset, "stable−privacy" is used if *IPv6StableSecretAddress=* is specified, and if not, "eui64" is used. Note that if *LinkLocalAddressing=* is "no" or "ipv4", then *IPv6LinkLocalAddressGenerationMode=* will be ignored. Also, even if *LinkLocalAddressing=* is "yes" or "ipv6", setting *IPv6LinkLocalAddressGenerationMode=none* disables to configure an IPv6 link−local address.

IPv6StableSecretAddress=
Takes an IPv6 address. The specified address will be used as a stable secret for generating IPv6 link−local address. If this setting is specified, and *IPv6LinkLocalAddressGenerationMode=* is unset, then *IPv6LinkLocalAddressGenerationMode=stable−privacy* is implied. If this setting is not specified, and "stable−privacy" is set to *IPv6LinkLocalAddressGenerationMode=*, then a stable secret address will be generated from the local machine ID and the interface name.

IPv4LLRoute=
Takes a boolean. If set to true, sets up the route needed for non−IPv4LL hosts to communicate with IPv4LL−only hosts. Defaults to false.

DefaultRouteOnDevice=
Takes a boolean. If set to true, sets up the default route bound to the interface. Defaults to false. This is useful when creating routes on point−to−point interfaces. This is equivalent to e.g. the following,

ip route add default dev veth99

or,

[Route]
Gateway=0.0.0.0

Currently, there are no way to specify e.g., the table for the route configured by this setting. To configure the default route with such an additional property, please use the following instead:

[Route]
Gateway=0.0.0.0
Table=1234

IPv6Token=
Specifies an optional address generation mode for the Stateless Address Autoconfiguration (SLAAC). Supported modes are "prefixstable" and "static".

When the mode is set to "static", an IPv6 address must be specified after a colon (":"), and the lower bits of the supplied address are combined with the upper bits of a prefix received in a Router Advertisement (RA) message to form a complete address. Note that if multiple prefixes are received in an RA message, or in multiple RA messages, addresses will be formed from each of them using the supplied address. This mode implements SLAAC but uses a static interface identifier instead of an identifier generated by using the EUI−64 algorithm. Because the interface identifier is static, if Duplicate Address Detection detects that the computed address is a duplicate (in use by another node on the link), then this mode will fail to provide an address for that prefix. If an IPv6 address without mode is specified, then "static" mode is assumed.

When the mode is set to "prefixstable" the **RFC 7217**[1] algorithm for generating interface identifiers will be used. This mode can optionally take an IPv6 address separated with a colon (":"). If an IPv6 address is specified, then an interface identifier is generated only when a prefix received in an RA message matches the supplied address.

If no address generation mode is specified (which is the default), or a received prefix does not match any of the addresses provided in "prefixstable" mode, then the EUI−64 algorithm will be used to form an interface identifier for that prefix. This mode is also SLAAC, but with a potentially stable interface identifier which does not directly map to the interface's hardware address.

Note that the "prefixstable" algorithm uses both the interface name and MAC address as input to the hash to compute the interface identifier, so if either of those are changed the resulting interface identifier (and address) will change, even if the prefix received in the RA message has not changed.

This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.

Examples:

IPv6Token=::1a:2b:3c:4d
IPv6Token=static:::1a:2b:3c:4d
IPv6Token=prefixstable
IPv6Token=prefixstable:2002:da8:1::

LLMNR=
Takes a boolean or "resolve". When true, enables **Link−Local Multicast Name Resolution**[2] on the link. When set to "resolve", only resolution is enabled, but not host registration and announcement. Defaults to true. This setting is read by **systemd-resolved.service**(8).

MulticastDNS=
Takes a boolean or "resolve". When true, enables **Multicast DNS**[3] support on the link. When set to "resolve", only resolution is enabled, but not host or service registration and announcement. Defaults to false. This setting is read by **systemd-resolved.service**(8).

DNSOverTLS=
Takes a boolean or "opportunistic". When true, enables **DNS−over−TLS**[4] support on the link. When set to "opportunistic", compatibility with non−DNS−over−TLS servers is increased, by automatically turning off DNS−over−TLS servers in this case. This option defines a per−interface setting for **resolved.conf**(5)'s global *DNSOverTLS=* option. Defaults to false. This setting is read by **systemd-resolved.service**(8).

DNSSEC=
Takes a boolean or "allow−downgrade". When true, enables **DNSSEC**[5] DNS validation support on the link. When set to "allow−downgrade", compatibility with non−DNSSEC capable networks is increased, by automatically turning off DNSSEC in this case. This option defines a per−interface setting for **resolved.conf**(5)'s global *DNSSEC=* option. Defaults to false. This setting is read by **systemd-resolved.service**(8).

DNSSECNegativeTrustAnchors=
A space−separated list of DNSSEC negative trust anchor domains. If specified and DNSSEC is enabled, look−ups done via the interface's DNS server will be subject to the list of negative trust anchors, and not require authentication for the specified domains, or anything below it. Use this to disable DNSSEC authentication for specific private domains, that cannot be proven valid using the Internet DNS hierarchy. Defaults to the empty list. This setting is read by **systemd-resolved.service**(8).

LLDP=
Controls support for Ethernet LLDP packet reception. LLDP is a link−layer protocol commonly

implemented on professional routers and bridges which announces which physical port a system is connected to, as well as other related data. Accepts a boolean or the special value "routers−only". When true, incoming LLDP packets are accepted and a database of all LLDP neighbors maintained. If "routers−only" is set only LLDP data of various types of routers is collected and LLDP data about other types of devices ignored (such as stations, telephones and others). If false, LLDP reception is disabled. Defaults to "routers−only". Use **networkctl**(1) to query the collected neighbor data. LLDP is only available on Ethernet links. See *EmitLLDP=* below for enabling LLDP packet emission from the local system.

EmitLLDP=

Controls support for Ethernet LLDP packet emission. Accepts a boolean parameter or the special values "nearest−bridge", "non−tpmr−bridge" and "customer−bridge". Defaults to false, which turns off LLDP packet emission. If not false, a short LLDP packet with information about the local system is sent out in regular intervals on the link. The LLDP packet will contain information about the local hostname, the local machine ID (as stored in **machine-id**(5)) and the local interface name, as well as the pretty hostname of the system (as set in **machine-info**(5)). LLDP emission is only available on Ethernet links. Note that this setting passes data suitable for identification of host to the network and should thus not be enabled on untrusted networks, where such identification data should not be made available. Use this option to permit other systems to identify on which interfaces they are connected to this system. The three special values control propagation of the LLDP packets. The "nearest−bridge" setting permits propagation only to the nearest connected bridge, "non−tpmr−bridge" permits propagation across Two−Port MAC Relays, but not any other bridges, and "customer−bridge" permits propagation until a customer bridge is reached. For details about these concepts, see **IEEE 802.1AB−2016**[6]. Note that configuring this setting to true is equivalent to "nearest−bridge", the recommended and most restricted level of propagation. See *LLDP=* above for an option to enable LLDP reception.

BindCarrier=

A link name or a list of link names. When set, controls the behavior of the current link. When all links in the list are in an operational down state, the current link is brought down. When at least one link has carrier, the current interface is brought up.

This forces *ActivationPolicy=* to be set to "bound".

Address=

A static IPv4 or IPv6 address and its prefix length, separated by a "/" character. Specify this key more than once to configure several addresses. The format of the address must be as described in **inet_pton**(3). This is a short−hand for an [Address] section only containing an Address key (see below). This option may be specified more than once.

If the specified address is "0.0.0.0" (for IPv4) or "::" (for IPv6), a new address range of the requested size is automatically allocated from a system−wide pool of unused ranges. Note that the prefix length must be equal or larger than 8 for IPv4, and 64 for IPv6. The allocated range is checked against all current network interfaces and all known network configuration files to avoid address range conflicts. The default system−wide pool consists of 192.168.0.0/16, 172.16.0.0/12 and 10.0.0.0/8 for IPv4, and fd00::/8 for IPv6. This functionality is useful to manage a large number of dynamically created network interfaces with the same network configuration and automatic address range assignment.

Gateway=

The gateway address, which must be in the format described in **inet_pton**(3). This is a short−hand for a [Route] section only containing a Gateway key. This option may be specified more than once.

DNS=

A DNS server address, which must be in the format described in **inet_pton**(3). This option may be specified more than once. Each address can optionally take a port number separated with ":", a network interface name or index separated with "%", and a Server Name Indication (SNI) separated with "#". When IPv6 address is specified with a port number, then the address must be in the square

brackets. That is, the acceptable full formats are "111.222.333.444:9953%ifname#example.com" for IPv4 and "[1111:2222::3333]:9953%ifname#example.com" for IPv6. If an empty string is assigned, then the all previous assignments are cleared. This setting is read by **systemd-resolved.service**(8).

Domains=
A whitespace−separated list of domains which should be resolved using the DNS servers on this link. Each item in the list should be a domain name, optionally prefixed with a tilde ("~"). The domains with the prefix are called "routing−only domains". The domains without the prefix are called "search domains" and are first used as search suffixes for extending single−label hostnames (hostnames containing no dots) to become fully qualified domain names (FQDNs). If a single−label hostname is resolved on this interface, each of the specified search domains are appended to it in turn, converting it into a fully qualified domain name, until one of them may be successfully resolved.

Both "search" and "routing−only" domains are used for routing of DNS queries: look−ups for hostnames ending in those domains (hence also single label names, if any "search domains" are listed), are routed to the DNS servers configured for this interface. The domain routing logic is particularly useful on multi−homed hosts with DNS servers serving particular private DNS zones on each interface.

The "routing−only" domain "~." (the tilde indicating definition of a routing domain, the dot referring to the DNS root domain which is the implied suffix of all valid DNS names) has special effect. It causes all DNS traffic which does not match another configured domain routing entry to be routed to DNS servers specified for this interface. This setting is useful to prefer a certain set of DNS servers if a link on which they are connected is available.

This setting is read by **systemd-resolved.service**(8). "Search domains" correspond to the *domain* and *search* entries in **resolv.conf**(5). Domain name routing has no equivalent in the traditional glibc API, which has no concept of domain name servers limited to a specific link.

DNSDefaultRoute=
Takes a boolean argument. If true, this link's configured DNS servers are used for resolving domain names that do not match any link's configured *Domains=* setting. If false, this link's configured DNS servers are never used for such domains, and are exclusively used for resolving names that match at least one of the domains configured on this link. If not specified defaults to an automatic mode: queries not matching any link's configured domains will be routed to this link if it has no routing−only domains configured.

NTP=
An NTP server address (either an IP address, or a hostname). This option may be specified more than once. This setting is read by **systemd-timesyncd.service**(8).

IPForward=
Configures IP packet forwarding for the system. If enabled, incoming packets on any network interface will be forwarded to any other interfaces according to the routing table. Takes a boolean, or the values "ipv4" or "ipv6", which only enable IP packet forwarding for the specified address family. This controls the net.ipv4.ip_forward and net.ipv6.conf.all.forwarding sysctl options of the network interface (see **ip−sysctl.txt**[7] for details about sysctl options). Defaults to "no".

Note: this setting controls a global kernel option, and does so one way only: if a network that has this setting enabled is set up the global setting is turned on. However, it is never turned off again, even after all networks with this setting enabled are shut down again.

To allow IP packet forwarding only between specific network interfaces use a firewall.

IPMasquerade=
Configures IP masquerading for the network interface. If enabled, packets forwarded from the network interface will be appear as coming from the local host. Takes one of "ipv4", "ipv6", "both", or "no".

Defaults to "no". If enabled, this automatically sets *IPForward=* to one of "ipv4", "ipv6" or "yes".

Note. Any positive boolean values such as "yes" or "true" are now deprecated. Please use one of the values in the above.

IPv6PrivacyExtensions=
Configures use of stateless temporary addresses that change over time (see **RFC 4941**[8], Privacy Extensions for Stateless Address Autoconfiguration in IPv6). Takes a boolean or the special values "prefer−public" and "kernel". When true, enables the privacy extensions and prefers temporary addresses over public addresses. When "prefer−public", enables the privacy extensions, but prefers public addresses over temporary addresses. When false, the privacy extensions remain disabled. When "kernel", the kernel's default setting will be left in place. Defaults to "no".

IPv6AcceptRA=
Takes a boolean. Controls IPv6 Router Advertisement (RA) reception support for the interface. If true, RAs are accepted; if false, RAs are ignored. When RAs are accepted, they may trigger the start of the DHCPv6 client if the relevant flags are set in the RA data, or if no routers are found on the link. The default is to disable RA reception for bridge devices or when IP forwarding is enabled, and to enable it otherwise. Cannot be enabled on bond devices and when link local addressing is disabled.

Further settings for the IPv6 RA support may be configured in the [IPv6AcceptRA] section, see below.

Also see **ip−sysctl.txt**[7] in the kernel documentation regarding "accept_ra", but note that systemd's setting of **1** (i.e. true) corresponds to kernel's setting of **2**.

Note that kernel's implementation of the IPv6 RA protocol is always disabled, regardless of this setting. If this option is enabled, a userspace implementation of the IPv6 RA protocol is used, and the kernel's own implementation remains disabled, since **systemd−networkd** needs to know all details supplied in the advertisements, and these are not available from the kernel if the kernel's own implementation is used.

IPv6DuplicateAddressDetection=
Configures the amount of IPv6 Duplicate Address Detection (DAD) probes to send. When unset, the kernel's default will be used.

IPv6HopLimit=
Configures IPv6 Hop Limit. For each router that forwards the packet, the hop limit is decremented by 1. When the hop limit field reaches zero, the packet is discarded. When unset, the kernel's default will be used.

IPv4AcceptLocal=
Takes a boolean. Accept packets with local source addresses. In combination with suitable routing, this can be used to direct packets between two local interfaces over the wire and have them accepted properly. When unset, the kernel's default will be used.

IPv4RouteLocalnet=
Takes a boolean. When true, the kernel does not consider loopback addresses as martian source or destination while routing. This enables the use of 127.0.0.0/8 for local routing purposes. When unset, the kernel's default will be used.

IPv4ProxyARP=
Takes a boolean. Configures proxy ARP for IPv4. Proxy ARP is the technique in which one host, usually a router, answers ARP requests intended for another machine. By "faking" its identity, the router accepts responsibility for routing packets to the "real" destination. See **RFC 1027**[9]. When unset, the kernel's default will be used.

IPv6ProxyNDP=
Takes a boolean. Configures proxy NDP for IPv6. Proxy NDP (Neighbor Discovery Protocol) is a technique for IPv6 to allow routing of addresses to a different destination when peers expect them to

be present on a certain physical link. In this case a router answers Neighbour Advertisement messages intended for another machine by offering its own MAC address as destination. Unlike proxy ARP for IPv4, it is not enabled globally, but will only send Neighbour Advertisement messages for addresses in the IPv6 neighbor proxy table, which can also be shown by **ip −6 neighbour show proxy**. systemd−networkd will control the per−interface ‘proxy_ndp‘ switch for each configured interface depending on this option. When unset, the kernel's default will be used.

IPv6ProxyNDPAddress=
An IPv6 address, for which Neighbour Advertisement messages will be proxied. This option may be specified more than once. systemd−networkd will add the **IPv6ProxyNDPAddress=** entries to the kernel's IPv6 neighbor proxy table. This option implies **IPv6ProxyNDP=yes** but has no effect if **IPv6ProxyNDP** has been set to false. When unset, the kernel's default will be used.

IPv6SendRA=
Whether to enable or disable Router Advertisement sending on a link. Takes a boolean value. When enabled, prefixes configured in [IPv6Prefix] sections and routes configured in [IPv6RoutePrefix] sections are distributed as defined in the [IPv6SendRA] section. If *DHCPv6PrefixDelegation=* is enabled, then the delegated prefixes are also distributed. See *DHCPv6PrefixDelegation=* setting and the [IPv6SendRA], [IPv6Prefix], [IPv6RoutePrefix], and [DHCPv6PrefixDelegation] sections for more configuration options.

DHCPv6PrefixDelegation=
Takes a boolean value. When enabled, requests prefixes using a DHCPv6 client configured on another link. By default, an address within each delegated prefix will be assigned, and the prefixes will be announced through IPv6 Router Advertisement when *IPv6SendRA=* is enabled. Such default settings can be configured in [DHCPv6PrefixDelegation] section. Defaults to disabled.

IPv6MTUBytes=
Configures IPv6 maximum transmission unit (MTU). An integer greater than or equal to 1280 bytes. When unset, the kernel's default will be used.

BatmanAdvanced=, *Bond=*, *Bridge=*, *VRF=*
The name of the B.A.T.M.A.N. Advanced, bond, bridge, or VRF interface to add the link to. See **systemd.netdev**(5).

IPVLAN=, *IPVTAP=*, *L2TP=*, *MACsec=*, *MACVLAN=*, *MACVTAP=*, *Tunnel=*, *VLAN=*, *VXLAN=*, *Xfrm=*
The name of an IPVLAN, IPVTAP, L2TP, MACsec, MACVLAN, MACVTAP, tunnel, VLAN, VXLAN, or Xfrm to be created on the link. See **systemd.netdev**(5). This option may be specified more than once.

ActiveSlave=
Takes a boolean. Specifies the new active slave. The "ActiveSlave=" option is only valid for following modes: "active−backup", "balance−alb" and "balance−tlb". Defaults to false.

PrimarySlave=
Takes a boolean. Specifies which slave is the primary device. The specified device will always be the active slave while it is available. Only when the primary is off−line will alternate devices be used. This is useful when one slave is preferred over another, e.g. when one slave has higher throughput than another. The "PrimarySlave=" option is only valid for following modes: "active−backup", "balance−alb" and "balance−tlb". Defaults to false.

ConfigureWithoutCarrier=
Takes a boolean. Allows networkd to configure a specific link even if it has no carrier. Defaults to false. If **IgnoreCarrierLoss=** is not explicitly set, it will default to this value.

IgnoreCarrierLoss=
Takes a boolean. Allows networkd to retain both the static and dynamic configuration of the interface even if its carrier is lost. When unset, the value specified with **ConfigureWithoutCarrier=** is used.

When *ActivationPolicy=* is set to "always−up", this is forced to "true".

KeepConfiguration=
Takes a boolean or one of "static", "dhcp−on−stop", "dhcp". When "static", **systemd−networkd** will not drop static addresses and routes on starting up process. When set to "dhcp−on−stop", **systemd−networkd** will not drop addresses and routes on stopping the daemon. When "dhcp", the addresses and routes provided by a DHCP server will never be dropped even if the DHCP lease expires. This is contrary to the DHCP specification, but may be the best choice if, e.g., the root filesystem relies on this connection. The setting "dhcp" implies "dhcp−on−stop", and "yes" implies "dhcp" and "static". Defaults to "no".

[ADDRESS] SECTION OPTIONS

An [Address] section accepts the following keys. Specify several [Address] sections to configure several addresses.

Address=
As in the [Network] section. This key is mandatory. Each [Address] section can contain one *Address=* setting.

Peer=
The peer address in a point−to−point connection. Accepts the same format as the *Address=* key.

Broadcast=
Takes an IPv4 address or boolean value. The address must be in the format described in **inet_pton**(3). If set to true, then the IPv4 broadcast address will be derived from the *Address=* setting. If set to false, then the broadcast address will not be set. Defaults to true, except for wireguard interfaces, where it default to false.

Label=
An address label.

PreferredLifetime=
Allows the default "preferred lifetime" of the address to be overridden. Only three settings are accepted: "forever", "infinity", which is the default and means that the address never expires, and "0", which means that the address is considered immediately "expired" and will not be used, unless explicitly requested. A setting of **PreferredLifetime=0** is useful for addresses which are added to be used only by a specific application, which is then configured to use them explicitly.

Scope=
The scope of the address, which can be "global" (valid everywhere on the network, even through a gateway), "link" (only valid on this device, will not traverse a gateway) or "host" (only valid within the device itself, e.g. 127.0.0.1) or an unsigned integer in the range 0...255. Defaults to "global".

RouteMetric=
The metric of the prefix route, which is pointing to the subnet of the configured IP address, taking the configured prefix length into account. Takes an unsigned integer in the range 0...4294967295. When unset or set to 0, the kernel's default value is used. This setting will be ignored when *AddPrefixRoute=* is false.

HomeAddress=
Takes a boolean. Designates this address the "home address" as defined in **RFC 6275**[10]. Supported only on IPv6. Defaults to false.

DuplicateAddressDetection=
Takes one of "ipv4", "ipv6", "both", "none". When "ipv4", performs IPv4 Address Conflict Detection. See **RFC 5227**[11]. When "ipv6", performs IPv6 Duplicate Address Detection. See **RFC 4862**[12]. Defaults to "ipv4" for IPv4 link−local addresses, "ipv6" for IPv6 addresses, and "none" otherwise.

ManageTemporaryAddress=
Takes a boolean. If true the kernel manage temporary addresses created from this one as template on behalf of Privacy Extensions **RFC 3041**[13]. For this to become active, the use_tempaddr sysctl setting has to be set to a value greater than zero. The given address needs to have a prefix length of 64. This flag allows using privacy extensions in a manually configured network, just like if stateless

auto−configuration was active. Defaults to false.

AddPrefixRoute=
Takes a boolean. When true, the prefix route for the address is automatically added. Defaults to true.

AutoJoin=
Takes a boolean. Joining multicast group on ethernet level via **ip maddr** command would not work if we have an Ethernet switch that does IGMP snooping since the switch would not replicate multicast packets on ports that did not have IGMP reports for the multicast addresses. Linux vxlan interfaces created via **ip link add vxlan** or networkd's netdev kind vxlan have the group option that enables then to do the required join. By extending ip address command with option "autojoin" we can get similar functionality for openvswitch (OVS) vxlan interfaces as well as other tunneling mechanisms that need to receive multicast traffic. Defaults to "no".

[NEIGHBOR] SECTION OPTIONS

A [Neighbor] section accepts the following keys. The neighbor section adds a permanent, static entry to the neighbor table (IPv6) or ARP table (IPv4) for the given hardware address on the links matched for the network. Specify several [Neighbor] sections to configure several static neighbors.

Address=
The IP address of the neighbor.

LinkLayerAddress=
The link layer address (MAC address or IP address) of the neighbor.

[IPV6ADDRESSLABEL] SECTION OPTIONS

An [IPv6AddressLabel] section accepts the following keys. Specify several [IPv6AddressLabel] sections to configure several address labels. IPv6 address labels are used for address selection. See **RFC 3484**[14]. Precedence is managed by userspace, and only the label itself is stored in the kernel.

Label=
The label for the prefix, an unsigned integer in the range 0–4294967294. 0xffffffff is reserved. This setting is mandatory.

Prefix=
IPv6 prefix is an address with a prefix length, separated by a slash "/" character. This key is mandatory.

[ROUTINGPOLICYRULE] SECTION OPTIONS

An [RoutingPolicyRule] section accepts the following keys. Specify several [RoutingPolicyRule] sections to configure several rules.

TypeOfService=
Takes a number between 0 and 255 that specifies the type of service to match.

From=
Specifies the source address prefix to match. Possibly followed by a slash and the prefix length.

To=
Specifies the destination address prefix to match. Possibly followed by a slash and the prefix length.

FirewallMark=
Specifies the iptables firewall mark value to match (a number between 1 and 4294967295). Optionally, the firewall mask (also a number between 1 and 4294967295) can be suffixed with a slash ("/"), e.g., "7/255".

Table=
Specifies the routing table identifier to lookup if the rule selector matches. Takes one of predefined names "default", "main", and "local", and names defined in *RouteTable=* in **networkd.conf**(5), or a number between 1 and 4294967295. Defaults to "main".

Priority=
Specifies the priority of this rule. *Priority=* is an unsigned integer in the range 0...4294967295.

Higher number means lower priority, and rules get processed in order of increasing number. Defaults to unset, and the kernel will pick a value dynamically.

IncomingInterface=
Specifies incoming device to match. If the interface is loopback, the rule only matches packets originating from this host.

OutgoingInterface=
Specifies the outgoing device to match. The outgoing interface is only available for packets originating from local sockets that are bound to a device.

SourcePort=
Specifies the source IP port or IP port range match in forwarding information base (FIB) rules. A port range is specified by the lower and upper port separated by a dash. Defaults to unset.

DestinationPort=
Specifies the destination IP port or IP port range match in forwarding information base (FIB) rules. A port range is specified by the lower and upper port separated by a dash. Defaults to unset.

IPProtocol=
Specifies the IP protocol to match in forwarding information base (FIB) rules. Takes IP protocol name such as "tcp", "udp" or "sctp", or IP protocol number such as "6" for "tcp" or "17" for "udp". Defaults to unset.

InvertRule=
A boolean. Specifies whether the rule is to be inverted. Defaults to false.

Family=
Takes a special value "ipv4", "ipv6", or "both". By default, the address family is determined by the address specified in *To=* or *From=*. If neither *To=* nor *From=* are specified, then defaults to "ipv4".

User=
Takes a username, a user ID, or a range of user IDs separated by a dash. Defaults to unset.

SuppressPrefixLength=
Takes a number *N* in the range 0...128 and rejects routing decisions that have a prefix length of *N* or less. Defaults to unset.

Type=
Specifies Routing Policy Database (RPDB) rule type. Takes one of "blackhole", "unreachable" or "prohibit".

[NEXTHOP] SECTION OPTIONS

The [NextHop] section is used to manipulate entries in the kernel's "nexthop" tables. The [NextHop] section accepts the following keys. Specify several [NextHop] sections to configure several hops.

Id=
The id of the next hop. Takes an unsigned integer in the range 1...4294967295. If left unspecified, then automatically chosen by kernel.

Gateway=
As in the [Network] section.

Family=
Takes one of the special values "ipv4" or "ipv6". By default, the family is determined by the address specified in *Gateway=*. If *Gateway=* is not specified, then defaults to "ipv4".

OnLink=
Takes a boolean. If set to true, the kernel does not have to check if the gateway is reachable directly by the current machine (i.e., attached to the local network), so that we can insert the nexthop in the kernel table without it being complained about. Defaults to "no".

Blackhole=
Takes a boolean. If enabled, packets to the corresponding routes are discarded silently, and *Gateway=*

cannot be specified. Defaults to "no".

Group=
Takes a whitespace separated list of nexthop IDs. Each ID must be in the range 1...4294967295. Optionally, each nexthop ID can take a weight after a colon ("*id*[:*weight*]"). The weight must be in the range 1...255. If the weight is not specified, then it is assumed that the weight is 1. This setting cannot be specified with *Gateway=*, *Family=*, *Blackhole=*. This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared. Defaults to unset.

[ROUTE] SECTION OPTIONS

The [Route] section accepts the following keys. Specify several [Route] sections to configure several routes.

Gateway=
Takes the gateway address or the special values "_dhcp4" and "_ipv6ra". If "_dhcp4" or "_ipv6ra" is set, then the gateway address provided by DHCPv4 or IPv6 RA is used.

GatewayOnLink=
Takes a boolean. If set to true, the kernel does not have to check if the gateway is reachable directly by the current machine (i.e., attached to the local network), so that we can insert the route in the kernel table without it being complained about. Defaults to "no".

Destination=
The destination prefix of the route. Possibly followed by a slash and the prefix length. If omitted, a full−length host route is assumed.

Source=
The source prefix of the route. Possibly followed by a slash and the prefix length. If omitted, a full−length host route is assumed.

Metric=
The metric of the route. Takes an unsigned integer in the range 0...4294967295. Defaults to unset, and the kernel's default will be used.

IPv6Preference=
Specifies the route preference as defined in **RFC 4191**[15] for Router Discovery messages. Which can be one of "low" the route has a lowest priority, "medium" the route has a default priority or "high" the route has a highest priority.

Scope=
The scope of the IPv4 route, which can be "global", "site", "link", "host", or "nowhere":

- "global" means the route can reach hosts more than one hop away.
- "site" means an interior route in the local autonomous system.
- "link" means the route can only reach hosts on the local network (one hop away).
- "host" means the route will not leave the local machine (used for internal addresses like 127.0.0.1).
- "nowhere" means the destination doesn't exist.

For IPv4 route, defaults to "host" if *Type=* is "local" or "nat", and "link" if *Type=* is "broadcast", "multicast", or "anycast". In other cases, defaults to "global". The value is not used for IPv6.

PreferredSource=
The preferred source address of the route. The address must be in the format described in **inet_pton**(3).

Table=
The table identifier for the route. Takes one of predefined names "default", "main", and "local", and names defined in *RouteTable=* in **networkd.conf**(5), or a number between 1 and 4294967295. The table can be retrieved using **ip route show table** *num*. If unset and *Type=* is "local", "broadcast", "anycast", or "nat", then "local" is used. In other cases, defaults to "main".

Protocol=
The protocol identifier for the route. Takes a number between 0 and 255 or the special values "kernel", "boot", "static", "ra" and "dhcp". Defaults to "static".

Type=
Specifies the type for the route. Takes one of "unicast", "local", "broadcast", "anycast", "multicast", "blackhole", "unreachable", "prohibit", "throw", "nat", and "xresolve". If "unicast", a regular route is defined, i.e. a route indicating the path to take to a destination network address. If "blackhole", packets to the defined route are discarded silently. If "unreachable", packets to the defined route are discarded and the ICMP message "Host Unreachable" is generated. If "prohibit", packets to the defined route are discarded and the ICMP message "Communication Administratively Prohibited" is generated. If "throw", route lookup in the current routing table will fail and the route selection process will return to Routing Policy Database (RPDB). Defaults to "unicast".

InitialCongestionWindow=
The TCP initial congestion window is used during the start of a TCP connection. During the start of a TCP session, when a client requests a resource, the server's initial congestion window determines how many packets will be sent during the initial burst of data without waiting for acknowledgement. Takes a number between 1 and 1023. Note that 100 is considered an extremely large value for this option. When unset, the kernel's default (typically 10) will be used.

InitialAdvertisedReceiveWindow=
The TCP initial advertised receive window is the amount of receive data (in bytes) that can initially be buffered at one time on a connection. The sending host can send only that amount of data before waiting for an acknowledgment and window update from the receiving host. Takes a number between 1 and 1023. Note that 100 is considered an extremely large value for this option. When unset, the kernel's default will be used.

QuickAck=
Takes a boolean. When true enables TCP quick ack mode for the route. When unset, the kernel's default will be used.

FastOpenNoCookie=
Takes a boolean. When true enables TCP fastopen without a cookie on a per−route basis. When unset, the kernel's default will be used.

TTLPropagate=
Takes a boolean. When true enables TTL propagation at Label Switched Path (LSP) egress. When unset, the kernel's default will be used.

MTUBytes=
The maximum transmission unit in bytes to set for the route. The usual suffixes K, M, G, are supported and are understood to the base of 1024.

Note that if IPv6 is enabled on the interface, and the MTU is chosen below 1280 (the minimum MTU for IPv6) it will automatically be increased to this value.

IPServiceType=
Takes string; "CS6" or "CS4". Used to set IP service type to CS6 (network control) or CS4 (Realtime). Defaults to CS6.

TCPAdvertisedMaximumSegmentSize=
Specifies the Path MSS (in bytes) hints given on TCP layer. The usual suffixes K, M, G, are supported and are understood to the base of 1024. An unsigned integer in the range 1–4294967294. When unset, the kernel's default will be used.

MultiPathRoute=address[@name] [weight]
Configures multipath route. Multipath routing is the technique of using multiple alternative paths through a network. Takes gateway address. Optionally, takes a network interface name or index separated with "@", and a weight in 1..256 for this multipath route separated with whitespace. This

setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.

NextHop=
Specifies the nexthop id. Takes an unsigned integer in the range 1...4294967295. If set, the corresponding [NextHop] section must be configured. Defaults to unset.

[DHCPV4] SECTION OPTIONS

The [DHCPv4] section configures the DHCPv4 client, if it is enabled with the *DHCP=* setting described above:

SendHostname=
When true (the default), the machine's hostname (or the value specified with *Hostname=*, described below) will be sent to the DHCP server. Note that the hostname must consist only of 7−bit ASCII lower−case characters and no spaces or dots, and be formatted as a valid DNS domain name. Otherwise, the hostname is not sent even if this option is true.

Hostname=
Use this value for the hostname which is sent to the DHCP server, instead of machine's hostname. Note that the specified hostname must consist only of 7−bit ASCII lower−case characters and no spaces or dots, and be formatted as a valid DNS domain name.

MUDURL=
When configured, the specified Manufacturer Usage Description (MUD) URL will be sent to the DHCPv4 server. Takes a URL of length up to 255 characters. A superficial verification that the string is a valid URL will be performed. DHCPv4 clients are intended to have at most one MUD URL associated with them. See **RFC 8520**[16].

MUD is an embedded software standard defined by the IETF that allows IoT device makers to advertise device specifications, including the intended communication patterns for their device when it connects to the network. The network can then use this to author a context−specific access policy, so the device functions only within those parameters.

ClientIdentifier=
The DHCPv4 client identifier to use. Takes one of **mac**, **duid** or **duid−only**. If set to **mac**, the MAC address of the link is used. If set to **duid**, an RFC4361−compliant Client ID, which is the combination of IAID and DUID (see below), is used. If set to **duid−only**, only DUID is used, this may not be RFC compliant, but some setups may require to use this. Defaults to **duid**.

VendorClassIdentifier=
The vendor class identifier used to identify vendor type and configuration.

UserClass=
A DHCPv4 client can use UserClass option to identify the type or category of user or applications it represents. The information contained in this option is a string that represents the user class of which the client is a member. Each class sets an identifying string of information to be used by the DHCP service to classify clients. Takes a whitespace−separated list of strings.

DUIDType=
Override the global *DUIDType=* setting for this network. See **networkd.conf**(5) for a description of possible values.

DUIDRawData=
Override the global *DUIDRawData=* setting for this network. See **networkd.conf**(5) for a description of possible values.

IAID=
The DHCP Identity Association Identifier (IAID) for the interface, a 32−bit unsigned integer.

Anonymize=
Takes a boolean. When true, the options sent to the DHCP server will follow the **RFC 7844**[17] (Anonymity Profiles for DHCP Clients) to minimize disclosure of identifying information. Defaults to

false.

This option should only be set to true when *MACAddressPolicy=* is set to **random** (see **systemd.link**(5)).

When true, *SendHostname=*, *ClientIdentifier=*, *VendorClassIdentifier=*, *UserClass=*, *RequestOptions=*, *SendOption=*, *SendVendorOption=*, and *MUDURL=* are ignored.

With this option enabled DHCP requests will mimic those generated by Microsoft Windows, in order to reduce the ability to fingerprint and recognize installations. This means DHCP request sizes will grow and lease data will be more comprehensive than normally, though most of the requested data is not actually used.

RequestOptions=
Sets request options to be sent to the server in the DHCPv4 request options list. A whitespace−separated list of integers in the range 1...254. Defaults to unset.

SendOption=
Send an arbitrary raw option in the DHCPv4 request. Takes a DHCP option number, data type and data separated with a colon ("*option*:*type*:*value*"). The option number must be an integer in the range 1...254. The type takes one of "uint8", "uint16", "uint32", "ipv4address", or "string". Special characters in the data string may be escaped using **C−style escapes**[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

SendVendorOption=
Send an arbitrary vendor option in the DHCPv4 request. Takes a DHCP option number, data type and data separated with a colon ("*option*:*type*:*value*"). The option number must be an integer in the range 1...254. The type takes one of "uint8", "uint16", "uint32", "ipv4address", or "string". Special characters in the data string may be escaped using **C−style escapes**[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

UseDNS=
When true (the default), the DNS servers received from the DHCP server will be used.

This corresponds to the **nameserver** option in **resolv.conf**(5).

RoutesToDNS=
When true, the routes to the DNS servers received from the DHCP server will be configured. When *UseDNS=* is disabled, this setting is ignored. Defaults to true.

UseNTP=
When true (the default), the NTP servers received from the DHCP server will be used by systemd−timesyncd.service.

RoutesToNTP=
When true, the routes to the NTP servers received from the DHCP server will be configured. When *UseNTP=* is disabled, this setting is ignored. Defaults to true.

UseSIP=
When true (the default), the SIP servers received from the DHCP server will be collected and made available to client programs.

UseMTU=
When true, the interface maximum transmission unit from the DHCP server will be used on the current link. If *MTUBytes=* is set, then this setting is ignored. Defaults to false.

UseHostname=
When true (the default), the hostname received from the DHCP server will be set as the transient

hostname of the system.

UseDomains=
Takes a boolean, or the special value **route**. When true, the domain name received from the DHCP server will be used as DNS search domain over this link, similar to the effect of the **Domains=** setting. If set to **route**, the domain name received from the DHCP server will be used for routing DNS queries only, but not for searching, similar to the effect of the **Domains=** setting when the argument is prefixed with "~". Defaults to true on Ubuntu.

It is recommended to enable this option only on trusted networks, as setting this affects resolution of all hostnames, in particular of single−label names. It is generally safer to use the supplied domain only as routing domain, rather than as search domain, in order to not have it affect local resolution of single−label names.

When set to true, this setting corresponds to the **domain** option in **resolv.conf**(5).

UseRoutes=
When true (the default), the static routes will be requested from the DHCP server and added to the routing table with a metric of 1024, and a scope of **global**, **link** or **host**, depending on the route's destination and gateway. If the destination is on the local host, e.g., 127.x.x.x, or the same as the link's own address, the scope will be set to **host**. Otherwise if the gateway is null (a direct route), a **link** scope will be used. For anything else, scope defaults to **global**.

RouteMetric=
Set the routing metric for routes specified by the DHCP server. Takes an unsigned integer in the range 0...4294967295. Defaults to 1024.

RouteTable=num
The table identifier for DHCP routes (a number between 1 and 4294967295, or 0 to unset). The table can be retrieved using **ip route show table** *num*.

When used in combination with *VRF=*, the VRF's routing table is used when this parameter is not specified.

RouteMTUBytes=
Specifies the MTU for the DHCP routes. Please see the [Route] section for further details.

UseGateway=
When true, the gateway will be requested from the DHCP server and added to the routing table with a metric of 1024, and a scope of **link**. When unset, the value specified with *UseRoutes=* is used.

UseTimezone=
When true, the timezone received from the DHCP server will be set as timezone of the local system. Defaults to false.

FallbackLeaseLifetimeSec=
Allows to set DHCPv4 lease lifetime when DHCPv4 server does not send the lease lifetime. Takes one of "forever" or "infinity". The latter means that the address never expires. Defaults to unset.

RequestBroadcast=
Request the server to use broadcast messages before the IP address has been configured. This is necessary for devices that cannot receive RAW packets, or that cannot receive packets at all before an IP address has been configured. On the other hand, this must not be enabled on networks where broadcasts are filtered out.

MaxAttempts=
Specifies how many times the DHCPv4 client configuration should be attempted. Takes a number or "infinity". Defaults to "infinity". Note that the time between retries is increased exponentially, up to approximately one per minute, so the network will not be overloaded even if this number is high. The default is suitable in most circumstances.

ListenPort=
Set the port from which the DHCP client packets originate.

DenyList=
A whitespace−separated list of IPv4 addresses. DHCP offers from servers in the list are rejected. Note that if *AllowList=* is configured then *DenyList=* is ignored.

AllowList=
A whitespace−separated list of IPv4 addresses. DHCP offers from servers in the list are accepted.

SendRelease=
When true, the DHCPv4 client sends a DHCP release packet when it stops. Defaults to true.

SendDecline=
A boolean. When "true", the DHCPv4 client receives the IP address from the DHCP server. After a new IP is received, the DHCPv4 client performs IPv4 Duplicate Address Detection. If duplicate use is detected, the DHCPv4 client rejects the IP by sending a **DHCPDECLINE** packet and tries to obtain an IP address again. See **RFC 5224**[11]. Defaults to "unset".

[DHCPV6] SECTION OPTIONS

The [DHCPv6] section configures the DHCPv6 client, if it is enabled with the *DHCP=* setting described above, or invoked by the IPv6 Router Advertisement:

MUDURL=, *IAID=*, *DUIDType=*, *DUIDRawData=*, *RequestOptions=*
As in the [DHCPv4] section.

SendOption=
As in the [DHCPv4] section, however because DHCPv6 uses 16−bit fields to store option numbers, the option number is an integer in the range 1...65536.

SendVendorOption=
Send an arbitrary vendor option in the DHCPv6 request. Takes an enterprise identifier, DHCP option number, data type, and data separated with a colon ("*enterprise identifier*:*option*:*type*:*value*"). Enterprise identifier is an unsigned integer in the range 1...4294967294. The option number must be an integer in the range 1...254. Data type takes one of "uint8", "uint16", "uint32", "ipv4address", "ipv6address", or "string". Special characters in the data string may be escaped using **C−style escapes**[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

UserClass=
A DHCPv6 client can use User Class option to identify the type or category of user or applications it represents. The information contained in this option is a string that represents the user class of which the client is a member. Each class sets an identifying string of information to be used by the DHCP service to classify clients. Special characters in the data string may be escaped using **C−style escapes**[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Takes a whitespace−separated list of strings. Note that currently **NUL** bytes are not allowed.

VendorClass=
A DHCPv6 client can use VendorClass option to identify the vendor that manufactured the hardware on which the client is running. The information contained in the data area of this option is contained in one or more opaque fields that identify details of the hardware configuration. Takes a whitespace−separated list of strings.

PrefixDelegationHint=
Takes an IPv6 address with prefix length in the same format as the *Address=* in the [Network] section. The DHCPv6 client will include a prefix hint in the DHCPv6 solicitation sent to the server. The prefix length must be in the range 1–128. Defaults to unset.

UseAddress=
When true (the default), the IP addresses provided by the DHCPv6 server will be assigned.

UseDNS=, *UseNTP=*, *UseHostname=*, *UseDomains=*
As in the [DHCPv4] section.

ForceDHCPv6PDOtherInformation=
Takes a boolean that enforces DHCPv6 stateful mode when the 'Other information' bit is set in Router Advertisement messages. By default setting only the 'O' bit in Router Advertisements makes DHCPv6 request network information in a stateless manner using a two−message Information Request and Information Reply message exchange. **RFC 7084**[19], requirement WPD−4, updates this behavior for a Customer Edge router so that stateful DHCPv6 Prefix Delegation is also requested when only the 'O' bit is set in Router Advertisements. This option enables such a CE behavior as it is impossible to automatically distinguish the intention of the 'O' bit otherwise. By default this option is set to false, enable it if no prefixes are delegated when the device should be acting as a CE router.

WithoutRA=
Allows DHCPv6 client to start without router advertisements's managed or other address configuration flag. Takes one of "solicit" or "information−request". Defaults to unset.

RapidCommit=
Takes a boolean. The DHCPv6 client can obtain configuration parameters from a DHCPv6 server through a rapid two−message exchange (solicit and reply). When the rapid commit option is enabled by both the DHCPv6 client and the DHCPv6 server, the two−message exchange is used, rather than the default four−message exchange (solicit, advertise, request, and reply). The two−message exchange provides faster client configuration and is beneficial in environments in which networks are under a heavy load. See **RFC 3315**[20] for details. Defaults to true.

[DHCPV6PREFIXDELEGATION] SECTION OPTIONS

The [DHCPv6PrefixDelegation] section configures delegated prefixes assigned by DHCPv6 server. The settings in this section are used only when *DHCPv6PrefixDelegation=* setting is enabled.

SubnetId=
Configure a specific subnet ID on the interface from a (previously) received prefix delegation. You can either set "auto" (the default) or a specific subnet ID (as defined in **RFC 4291**[21], section 2.5.4), in which case the allowed value is hexadecimal, from 0 to 0x7fffffffffffffff inclusive.

Announce=
Takes a boolean. When enabled, and *IPv6SendRA=* in [Network] section is enabled, the delegated prefixes are distributed through the IPv6 Router Advertisement. Defaults to yes.

Assign=
Takes a boolean. Specifies whether to add an address from the delegated prefixes which are received from the WAN interface by the DHCPv6 Prefix Delegation. When true (on LAN interfce), the EUI−64 algorithm will be used by default to form an interface identifier from the delegated prefixes. See also *Token=* setting below. Defaults to yes.

Token=
Specifies an optional address generation mode for assigning an address in each delegated prefix. Takes an IPv6 address. When set, the lower bits of the supplied address is combined with the upper bits of each delegatad prefix received from the WAN interface by the DHCPv6 Prefix Delegation to form a complete address. When *Assign=* is disabled, this setting is ignored. When unset, the EUI−64 algorithm will be used to form addresses. Defaults to unset.

ManageTemporaryAddress=
As in the [Address] section, but defaults to true.

RouteMetric=
The metric of the route to the delegated prefix subnet. Takes an unsigned integer in the range 0...4294967295. When unset or set to 0, the kernel's default value is used.

[IPV6ACCEPTRA] SECTION OPTIONS

The [IPv6AcceptRA] section configures the IPv6 Router Advertisement (RA) client, if it is enabled with the *IPv6AcceptRA=* setting described above:

UseDNS=
When true (the default), the DNS servers received in the Router Advertisement will be used.

This corresponds to the **nameserver** option in **resolv.conf**(5).

UseDomains=
Takes a boolean, or the special value "route". When true, the domain name received via IPv6 Router Advertisement (RA) will be used as DNS search domain over this link, similar to the effect of the **Domains=** setting. If set to "route", the domain name received via IPv6 RA will be used for routing DNS queries only, but not for searching, similar to the effect of the **Domains=** setting when the argument is prefixed with "~". Defaults to true on Ubuntu.

It is recommended to enable this option only on trusted networks, as setting this affects resolution of all hostnames, in particular of single−label names. It is generally safer to use the supplied domain only as routing domain, rather than as search domain, in order to not have it affect local resolution of single−label names.

When set to true, this setting corresponds to the **domain** option in **resolv.conf**(5).

RouteTable=num
The table identifier for the routes received in the Router Advertisement (a number between 1 and 4294967295, or 0 to unset). The table can be retrieved using **ip route show table** *num*.

RouteMetric=
Set the routing metric for the routes received in the Router Advertisement. Takes an unsigned integer in the range 0...4294967295. Defaults to 1024.

UseAutonomousPrefix=
When true (the default), the autonomous prefix received in the Router Advertisement will be used and take precedence over any statically configured ones.

UseOnLinkPrefix=
When true (the default), the onlink prefix received in the Router Advertisement will be used and takes precedence over any statically configured ones.

RouterDenyList=
A whitespace−separated list of IPv6 router addresses. Any information advertised by the listed router is ignored.

RouterAllowList=
A whitespace−separated list of IPv6 router addresses. Only information advertised by the listed router is accepted. Note that if *RouterAllowList=* is configured then *RouterDenyList=* is ignored.

PrefixDenyList=
A whitespace−separated list of IPv6 prefixes. IPv6 prefixes supplied via router advertisements in the list are ignored.

PrefixAllowList=
A whitespace−separated list of IPv6 prefixes. IPv6 prefixes supplied via router advertisements in the list are allowed. Note that if *PrefixAllowList=* is configured then *PrefixDenyList=* is ignored.

RouteDenyList=
A whitespace−separated list of IPv6 route prefixes. IPv6 route prefixes supplied via router advertisements in the list are ignored.

RouteAllowList=
A whitespace−separated list of IPv6 route prefixes. IPv6 route prefixes supplied via router advertisements in the list are allowed. Note that if *RouteAllowList=* is configured then *RouteDenyList=* is ignored.

DHCPv6Client=
Takes a boolean, or the special value "always". When true or "always", the DHCPv6 client will be

started when the RA has the managed or other information flag. If set to "always", the DHCPv6 client will also be started in managed mode when neither managed nor other information flag is set in the RA. Defaults to true.

[DHCPSERVER] SECTION OPTIONS

The [DHCPServer] section contains settings for the DHCP server, if enabled via the *DHCPServer=* option described above:

ServerAddress=
: Specifies server address for the DHCP server. Takes an IPv4 address with prefix length, for example "192.168.0.1/24". This setting may be useful when the link on which the DHCP server is running has multiple static addresses. When unset, one of static addresses in the link will be automatically selected. Defaults to unset.

PoolOffset=, *PoolSize=*
: Configures the pool of addresses to hand out. The pool is a contiguous sequence of IP addresses in the subnet configured for the server address, which does not include the subnet nor the broadcast address. *PoolOffset=* takes the offset of the pool from the start of subnet, or zero to use the default value. *PoolSize=* takes the number of IP addresses in the pool or zero to use the default value. By default, the pool starts at the first address after the subnet address and takes up the rest of the subnet, excluding the broadcast address. If the pool includes the server address (the default), this is reserved and not handed out to clients.

DefaultLeaseTimeSec=, *MaxLeaseTimeSec=*
: Control the default and maximum DHCP lease time to pass to clients. These settings take time values in seconds or another common time unit, depending on the suffix. The default lease time is used for clients that did not ask for a specific lease time. If a client asks for a lease time longer than the maximum lease time, it is automatically shortened to the specified time. The default lease time defaults to 1h, the maximum lease time to 12h. Shorter lease times are beneficial if the configuration data in DHCP leases changes frequently and clients shall learn the new settings with shorter latencies. Longer lease times reduce the generated DHCP network traffic.

UplinkInterface=
: Specifies name or index of uplink interface, or one of the special values ":none" and ":auto". When emitting DNS, NTP, or SIP servers are enabled but no servers are specified, the servers configured in the uplink interface will be emitted. When ":auto", the link which has default gateway with higher priority will be automatically selected. When ":none", no uplink interface will be selected. Defaults to ":auto".

EmitDNS=, *DNS=*
: *EmitDNS=* takes a boolean. Configures whether the DHCP leases handed out to clients shall contain DNS server information. Defaults to "yes". The DNS servers to pass to clients may be configured with the *DNS=* option, which takes a list of IPv4 addresses. If the *EmitDNS=* option is enabled but no servers configured, the servers are automatically propagated from an "uplink" interface that has appropriate servers set. The "uplink" interface is determined by the default route of the system with the highest priority. Note that this information is acquired at the time the lease is handed out, and does not take uplink interfaces into account that acquire DNS server information at a later point. If no suitable uplink interface is found the DNS server data from /etc/resolv.conf is used. Also, note that the leases are not refreshed if the uplink network configuration changes. To ensure clients regularly acquire the most current uplink DNS server information, it is thus advisable to shorten the DHCP lease time via *MaxLeaseTimeSec=* described above.

EmitNTP=, *NTP=*, *EmitSIP=*, *SIP=*, *EmitPOP3=*, *POP3=*, *EmitSMTP=*, *SMTP=*, *EmitLPR=*, *LPR=*
: Similar to the *EmitDNS=* and *DNS=* settings described above, these settings configure whether and what server information for the indicate protocol shall be emitted as part of the DHCP lease. The same syntax, propagation semantics and defaults apply as for *EmitDNS=* and *DNS=*.

EmitRouter=
: Similar to the *EmitDNS=* setting described above, this setting configures whether the DHCP lease

should contain the router option. The same syntax, propagation semantics and defaults apply as for *EmitDNS=*.

EmitTimezone=, *Timezone=*
Takes a boolean. Configures whether the DHCP leases handed out to clients shall contain timezone information. Defaults to "yes". The *Timezone=* setting takes a timezone string (such as "Europe/Berlin" or "UTC") to pass to clients. If no explicit timezone is set, the system timezone of the local host is propagated, as determined by the /etc/localtime symlink.

SendOption=
Send a raw option with value via DHCPv4 server. Takes a DHCP option number, data type and data ("*option*:*type*:*value*"). The option number is an integer in the range 1...254. The type takes one of "uint8", "uint16", "uint32", "ipv4address", "ipv6address", or "string". Special characters in the data string may be escaped using **C-style escapes**[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

SendVendorOption=
Send a vendor option with value via DHCPv4 server. Takes a DHCP option number, data type and data ("*option*:*type*:*value*"). The option number is an integer in the range 1...254. The type takes one of "uint8", "uint16", "uint32", "ipv4address", or "string". Special characters in the data string may be escaped using **C-style escapes**[18]. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

BindToInterface=
Takes a boolean value. When "yes", DHCP server socket will be bound to its network interface and all socket communication will be restricted to this interface. Defaults to "yes", except if *RelayTarget=* is used (see below), in which case it defaults defaults to "no".

RelayTarget=
Takes an IPv4 address, which must be in the format described in **inet_pton**(3). Turns this DHCP server into a DHCP relay agent. See **RFC 1542**[22]. The address is the address of DHCP server or another relay agent to forward DHCP messages to and from.

RelayAgentCircuitId=
Specifies value for Agent Circuit ID suboption of Relay Agent Information option. Takes a string, which must be in the format "string:*value*", where "*value*" should be replaced with the value of the suboption. Defaults to unset (means no Agent Circuit ID suboption is generated). Ignored if *RelayTarget=* is not specified.

RelayAgentRemoteId=
Specifies value for Agent Remote ID suboption of Relay Agent Information option. Takes a string, which must be in the format "string:*value*", where "*value*" should be replaced with the value of the suboption. Defaults to unset (means no Agent Remote ID suboption is generated). Ignored if *RelayTarget=* is not specified.

[DHCPSERVERSTATICLEASE] SECTION OPTIONS

The "[DHCPServerStaticLease]" section configures a static DHCP lease to assign a fixed IPv4 address to a specific device based on its MAC address. This section can be specified multiple times.

MACAddress=
The hardware address of a device to match. This key is mandatory.

Address=
The IPv4 address that should be assigned to the device that was matched with *MACAddress=*. This key is mandatory.

[IPV6SENDRA] SECTION OPTIONS

The [IPv6SendRA] section contains settings for sending IPv6 Router Advertisements and whether to act as a router, if enabled via the *IPv6SendRA=* option described above. IPv6 network prefixes or routes are defined with one or more [IPv6Prefix] or [IPv6RoutePrefix] sections.

Managed=, *OtherInformation=*

Takes a boolean. Controls whether a DHCPv6 server is used to acquire IPv6 addresses on the network link when *Managed=* is set to "true" or if only additional network information can be obtained via DHCPv6 for the network link when *OtherInformation=* is set to "true". Both settings default to "false", which means that a DHCPv6 server is not being used.

RouterLifetimeSec=
Takes a timespan. Configures the IPv6 router lifetime in seconds. When set to 0, the host is not acting as a router. Defaults to 30 minutes.

RouterPreference=
Configures IPv6 router preference if *RouterLifetimeSec=* is non−zero. Valid values are "high", "medium" and "low", with "normal" and "default" added as synonyms for "medium" just to make configuration easier. See **RFC 4191**[15] for details. Defaults to "medium".

EmitDNS=, *DNS=*
DNS= specifies a list of recursive DNS server IPv6 addresses that are distributed via Router Advertisement messages when *EmitDNS=* is true. *DNS=* also takes special value "_link_local"; in that case the IPv6 link local address is distributed. If *DNS=* is empty, DNS servers are read from the [Network] section. If the [Network] section does not contain any DNS servers either, DNS servers from the uplink with the highest priority default route are used. When *EmitDNS=* is false, no DNS server information is sent in Router Advertisement messages. *EmitDNS=* defaults to true.

EmitDomains=, *Domains=*
A list of DNS search domains distributed via Router Advertisement messages when *EmitDomains=* is true. If *Domains=* is empty, DNS search domains are read from the [Network] section. If the [Network] section does not contain any DNS search domains either, DNS search domains from the uplink with the highest priority default route are used. When *EmitDomains=* is false, no DNS search domain information is sent in Router Advertisement messages. *EmitDomains=* defaults to true.

DNSLifetimeSec=
Lifetime in seconds for the DNS server addresses listed in *DNS=* and search domains listed in *Domains=*.

[IPV6PREFIX] SECTION OPTIONS

One or more [IPv6Prefix] sections contain the IPv6 prefixes that are announced via Router Advertisements. See **RFC 4861**[23] for further details.

AddressAutoconfiguration=, *OnLink=*
Takes a boolean to specify whether IPv6 addresses can be autoconfigured with this prefix and whether the prefix can be used for onlink determination. Both settings default to "true" in order to ease configuration.

Prefix=
The IPv6 prefix that is to be distributed to hosts. Similarly to configuring static IPv6 addresses, the setting is configured as an IPv6 prefix and its prefix length, separated by a "/" character. Use multiple [IPv6Prefix] sections to configure multiple IPv6 prefixes since prefix lifetimes, address autoconfiguration and onlink status may differ from one prefix to another.

PreferredLifetimeSec=, *ValidLifetimeSec=*
Preferred and valid lifetimes for the prefix measured in seconds. *PreferredLifetimeSec=* defaults to 604800 seconds (one week) and *ValidLifetimeSec=* defaults to 2592000 seconds (30 days).

Assign=
Takes a boolean. When true, adds an address from the prefix. Default to false.

RouteMetric=
The metric of the prefix route. Takes an unsigned integer in the range 0...4294967295. When unset or set to 0, the kernel's default value is used. This setting is ignored when *Assign=* is false.

[IPV6ROUTEPREFIX] SECTION OPTIONS

One or more [IPv6RoutePrefix] sections contain the IPv6 prefix routes that are announced via Router Advertisements. See **RFC 4191**[15] for further details.

Route=
The IPv6 route that is to be distributed to hosts. Similarly to configuring static IPv6 routes, the setting is configured as an IPv6 prefix routes and its prefix route length, separated by a "/" character. Use multiple [IPv6PrefixRoutes] sections to configure multiple IPv6 prefix routes.

LifetimeSec=
Lifetime for the route prefix measured in seconds. *LifetimeSec=* defaults to 604800 seconds (one week).

[BRIDGE] SECTION OPTIONS

The [Bridge] section accepts the following keys:

UnicastFlood=
Takes a boolean. Controls whether the bridge should flood traffic for which an FDB entry is missing and the destination is unknown through this port. When unset, the kernel's default will be used.

MulticastFlood=
Takes a boolean. Controls whether the bridge should flood traffic for which an MDB entry is missing and the destination is unknown through this port. When unset, the kernel's default will be used.

MulticastToUnicast=
Takes a boolean. Multicast to unicast works on top of the multicast snooping feature of the bridge. Which means unicast copies are only delivered to hosts which are interested in it. When unset, the kernel's default will be used.

NeighborSuppression=
Takes a boolean. Configures whether ARP and ND neighbor suppression is enabled for this port. When unset, the kernel's default will be used.

Learning=
Takes a boolean. Configures whether MAC address learning is enabled for this port. When unset, the kernel's default will be used.

HairPin=
Takes a boolean. Configures whether traffic may be sent back out of the port on which it was received. When this flag is false, then the bridge will not forward traffic back out of the receiving port. When unset, the kernel's default will be used.

UseBPDU=
Takes a boolean. Configures whether STP Bridge Protocol Data Units will be processed by the bridge port. When unset, the kernel's default will be used.

FastLeave=
Takes a boolean. This flag allows the bridge to immediately stop multicast traffic on a port that receives an IGMP Leave message. It is only used with IGMP snooping if enabled on the bridge. When unset, the kernel's default will be used.

AllowPortToBeRoot=
Takes a boolean. Configures whether a given port is allowed to become a root port. Only used when STP is enabled on the bridge. When unset, the kernel's default will be used.

ProxyARP=
Takes a boolean. Configures whether proxy ARP to be enabled on this port. When unset, the kernel's default will be used.

ProxyARPWiFi=
Takes a boolean. Configures whether proxy ARP to be enabled on this port which meets extended requirements by IEEE 802.11 and Hotspot 2.0 specifications. When unset, the kernel's default will be used.

MulticastRouter=
Configures this port for having multicast routers attached. A port with a multicast router will receive all multicast traffic. Takes one of "no" to disable multicast routers on this port, "query" to let the

system detect the presence of routers, "permanent" to permanently enable multicast traffic forwarding on this port, or "temporary" to enable multicast routers temporarily on this port, not depending on incoming queries. When unset, the kernel's default will be used.

Cost=
Sets the "cost" of sending packets of this interface. Each port in a bridge may have a different speed and the cost is used to decide which link to use. Faster interfaces should have lower costs. It is an integer value between 1 and 65535.

Priority=
Sets the "priority" of sending packets on this interface. Each port in a bridge may have a different priority which is used to decide which link to use. Lower value means higher priority. It is an integer value between 0 to 63. Networkd does not set any default, meaning the kernel default value of 32 is used.

[BRIDGEFDB] SECTION OPTIONS

The [BridgeFDB] section manages the forwarding database table of a port and accepts the following keys. Specify several [BridgeFDB] sections to configure several static MAC table entries.

MACAddress=
As in the [Network] section. This key is mandatory.

Destination=
Takes an IP address of the destination VXLAN tunnel endpoint.

VLANId=
The VLAN ID for the new static MAC table entry. If omitted, no VLAN ID information is appended to the new static MAC table entry.

VNI=
The VXLAN Network Identifier (or VXLAN Segment ID) to use to connect to the remote VXLAN tunnel endpoint. Takes a number in the range 1...16777215. Defaults to unset.

AssociatedWith=
Specifies where the address is associated with. Takes one of "use", "self", "master" or "router". "use" means the address is in use. User space can use this option to indicate to the kernel that the fdb entry is in use. "self" means the address is associated with the port drivers fdb. Usually hardware. "master" means the address is associated with master devices fdb. "router" means the destination address is associated with a router. Note that it's valid if the referenced device is a VXLAN type device and has route shortcircuit enabled. Defaults to "self".

OutgoingInterface=
Specifies the name or index of the outgoing interface for the VXLAN device driver to reach the remote VXLAN tunnel endpoint. Defaults to unset.

[BRIDGEMDB] SECTION OPTIONS

The [BridgeMDB] section manages the multicast membership entries forwarding database table of a port and accepts the following keys. Specify several [BridgeMDB] sections to configure several permanent multicast membership entries.

MulticastGroupAddress=
Specifies the IPv4 or IPv6 multicast group address to add. This setting is mandatory.

VLANId=
The VLAN ID for the new entry. Valid ranges are 0 (no VLAN) to 4094. Optional, defaults to 0.

[LLDP] SECTION OPTIONS

The [LLDP] section manages the Link Layer Discovery Protocol (LLDP) and accepts the following keys:

MUDURL=
When configured, the specified Manufacturer Usage Descriptions (MUD) URL will be sent in LLDP packets. The syntax and semantics are the same as for *MUDURL=* in the [DHCPv4] section described above.

The MUD URLs received via LLDP packets are saved and can be read using the **sd_lldp_neighbor_get_mud_url()** function.

[CAN] SECTION OPTIONS

The [CAN] section manages the Controller Area Network (CAN bus) and accepts the following keys:

BitRate=
The bitrate of CAN device in bits per second. The usual SI prefixes (K, M) with the base of 1000 can be used here. Takes a number in the range 1...4294967295.

SamplePoint=
Optional sample point in percent with one decimal (e.g. "75%", "87.5%") or permille (e.g. "875‰").

DataBitRate=, *DataSamplePoint=*
The bitrate and sample point for the data phase, if CAN−FD is used. These settings are analogous to the *BitRate=* and *SamplePoint=* keys.

FDMode=
Takes a boolean. When "yes", CAN−FD mode is enabled for the interface. Note, that a bitrate and optional sample point should also be set for the CAN−FD data phase using the *DataBitRate=* and *DataSamplePoint=* keys.

FDNonISO=
Takes a boolean. When "yes", non−ISO CAN−FD mode is enabled for the interface. When unset, the kernel's default will be used.

RestartSec=
Automatic restart delay time. If set to a non−zero value, a restart of the CAN controller will be triggered automatically in case of a bus−off condition after the specified delay time. Subsecond delays can be specified using decimals (e.g. "0.1s") or a "ms" or "us" postfix. Using "infinity" or "0" will turn the automatic restart off. By default automatic restart is disabled.

Termination=
Takes a boolean. When "yes", the termination resistor will be selected for the bias network. When unset, the kernel's default will be used.

TripleSampling=
Takes a boolean. When "yes", three samples (instead of one) are used to determine the value of a received bit by majority rule. When unset, the kernel's default will be used.

BusErrorReporting=
Takes a boolean. When "yes", reporting of CAN bus errors is activated (those include single bit, frame format, and bit stuffing errors, unable to send dominant bit, unable to send recessive bit, bus overload, active error announcement, error occurred on transmission). When unset, the kernel's default will be used. Note: in case of a CAN bus with a single CAN device, sending a CAN frame may result in a huge number of CAN bus errors.

ListenOnly=
Takes a boolean. When "yes", listen−only mode is enabled. When the interface is in listen−only mode, the interface neither transmit CAN frames nor send ACK bit. Listen−only mode is important to debug CAN networks without interfering with the communication or acknowledge the CAN frame. When unset, the kernel's default will be used.

[QDISC] SECTION OPTIONS

The [QDisc] section manages the traffic control queueing discipline (qdisc).

Parent=
Specifies the parent Queueing Discipline (qdisc). Takes one of "clsact" or "ingress". This is mandatory.

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

[NETWORKEMULATOR] SECTION OPTIONS

The [NetworkEmulator] section manages the queueing discipline (qdisc) of the network emulator. It can be used to configure the kernel packet scheduler and simulate packet delay and loss for UDP or TCP applications, or limit the bandwidth usage of a particular service to simulate internet connections.

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

DelaySec=

Specifies the fixed amount of delay to be added to all packets going out of the interface. Defaults to unset.

DelayJitterSec=

Specifies the chosen delay to be added to the packets outgoing to the network interface. Defaults to unset.

PacketLimit=

Specifies the maximum number of packets the qdisc may hold queued at a time. An unsigned integer in the range 0–4294967294. Defaults to 1000.

LossRate=

Specifies an independent loss probability to be added to the packets outgoing from the network interface. Takes a percentage value, suffixed with "%". Defaults to unset.

DuplicateRate=

Specifies that the chosen percent of packets is duplicated before queuing them. Takes a percentage value, suffixed with "%". Defaults to unset.

[TOKENBUCKETFILTER] SECTION OPTIONS

The [TokenBucketFilter] section manages the queueing discipline (qdisc) of token bucket filter (tbf).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

LatencySec=

Specifies the latency parameter, which specifies the maximum amount of time a packet can sit in the Token Bucket Filter (TBF). Defaults to unset.

LimitBytes=

Takes the number of bytes that can be queued waiting for tokens to become available. When the size is suffixed with K, M, or G, it is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset.

BurstBytes=

Specifies the size of the bucket. This is the maximum amount of bytes that tokens can be available for instantaneous transfer. When the size is suffixed with K, M, or G, it is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset.

Rate=

Specifies the device specific bandwidth. When suffixed with K, M, or G, the specified bandwidth is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. Defaults to unset.

MPUBytes=
The Minimum Packet Unit (MPU) determines the minimal token usage (specified in bytes) for a packet. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to zero.

PeakRate=
Takes the maximum depletion rate of the bucket. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. Defaults to unset.

MTUBytes=
Specifies the size of the peakrate bucket. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset.

[PIE] SECTION OPTIONS

The [PIE] section manages the queueing discipline (qdisc) of Proportional Integral controller−Enhanced (PIE).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=
Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming packets are dropped. An unsigned integer in the range 1...4294967294. Defaults to unset and kernel's default is used.

[FLOWQUEUEPIE] SECTION OPTIONS

The "[FlowQueuePIE]" section manages the queueing discipline (qdisc) of Flow Queue Proportional Integral controller−Enhanced (fq_pie).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=
Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming packets are dropped. An unsigned integer ranges 1 to 4294967294. Defaults to unset and kernel's default is used.

[STOCHASTICFAIRBLUE] SECTION OPTIONS

The [StochasticFairBlue] section manages the queueing discipline (qdisc) of stochastic fair blue (sfb).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=
Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming

packets are dropped. An unsigned integer in the range 0–4294967294. Defaults to unset and kernel's default is used.

[STOCHASTICFAIRNESSQUEUEING] SECTION OPTIONS

The [StochasticFairnessQueueing] section manages the queueing discipline (qdisc) of stochastic fairness queueing (sfq).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PerturbPeriodSec=

Specifies the interval in seconds for queue algorithm perturbation. Defaults to unset.

[BFIFO] SECTION OPTIONS

The [BFIFO] section manages the queueing discipline (qdisc) of Byte limited Packet First In First Out (bfifo).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

LimitBytes=

Specifies the hard limit in bytes on the FIFO buffer size. The size limit prevents overflow in case the kernel is unable to dequeue packets as quickly as it receives them. When this limit is reached, incoming packets are dropped. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel default is used.

[PFIFO] SECTION OPTIONS

The [PFIFO] section manages the queueing discipline (qdisc) of Packet First In First Out (pfifo).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=

Specifies the hard limit on the number of packets in the FIFO queue. The size limit prevents overflow in case the kernel is unable to dequeue packets as quickly as it receives them. When this limit is reached, incoming packets are dropped. An unsigned integer in the range 0–4294967294. Defaults to unset and kernel's default is used.

[PFIFOHEADDROP] SECTION OPTIONS

The [PFIFOHeadDrop] section manages the queueing discipline (qdisc) of Packet First In First Out Head Drop (pfifo_head_drop).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class

identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=
As in [PFIFO] section.

[PFIFOFAST] SECTION OPTIONS

The [PFIFOFast] section manages the queueing discipline (qdisc) of Packet First In First Out Fast (pfifo_fast).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

[CAKE] SECTION OPTIONS

The [CAKE] section manages the queueing discipline (qdisc) of Common Applications Kept Enhanced (CAKE).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

OverheadBytes=
Specifies that bytes to be addeded to the size of each packet. Bytes may be negative. Takes an integer in the range from −64 to 256. Defaults to unset and kernel's default is used.

Bandwidth=
Specifies the shaper bandwidth. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. Defaults to unset and kernel's default is used.

[CONTROLLEDDELAY] SECTION OPTIONS

The [ControlledDelay] section manages the queueing discipline (qdisc) of controlled delay (CoDel).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=
Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming packets are dropped. An unsigned integer in the range 0–4294967294. Defaults to unset and kernel's default is used.

TargetSec=

Takes a timespan. Specifies the acceptable minimum standing/persistent queue delay. Defaults to unset and kernel's default is used.

IntervalSec=
Takes a timespan. This is used to ensure that the measured minimum delay does not become too stale. Defaults to unset and kernel's default is used.

ECN=
Takes a boolean. This can be used to mark packets instead of dropping them. Defaults to unset and kernel's default is used.

CEThresholdSec=
Takes a timespan. This sets a threshold above which all packets are marked with ECN Congestion Experienced (CE). Defaults to unset and kernel's default is used.

[DEFICITROUNDROBINSCHEDULER] SECTION OPTIONS

The [DeficitRoundRobinScheduler] section manages the queueing discipline (qdisc) of Deficit Round Robin Scheduler (DRR).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

[DEFICITROUNDROBINSCHEDULERCLASS] SECTION OPTIONS

The [DeficitRoundRobinSchedulerClass] section manages the traffic control class of Deficit Round Robin Scheduler (DRR).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", or a qdisc identifier. The qdisc identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

ClassId=
Configures the unique identifier of the class. It is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to unset.

QuantumBytes=
Specifies the amount of bytes a flow is allowed to dequeue before the scheduler moves to the next class. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to the MTU of the interface.

[ENHANCEDTRANSMISSIONSELECTION] SECTION OPTIONS

The [EnhancedTransmissionSelection] section manages the queueing discipline (qdisc) of Enhanced Transmission Selection (ETS).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

Bands=
Specifies the number of bands. An unsigned integer in the range 1–16. This value has to be at least large enough to cover the strict bands specified through the *StrictBands=* and bandwidth−sharing bands specified in *QuantumBytes=*.

StrictBands=
Specifies the number of bands that should be created in strict mode. An unsigned integer in the range 1–16.

QuantumBytes=
Specifies the white−space separated list of quantum used in band−sharing bands. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.

PriorityMap=
The priority map maps the priority of a packet to a band. The argument is a whitespace separated list of numbers. The first number indicates which band the packets with priority 0 should be put to, the second is for priority 1, and so on. There can be up to 16 numbers in the list. If there are fewer, the default band that traffic with one of the unmentioned priorities goes to is the last one. Each band number must be in the range 0...255. This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.

[GENERICRANDOMEARLYDETECTION] SECTION OPTIONS

The [GenericRandomEarlyDetection] section manages the queueing discipline (qdisc) of Generic Random Early Detection (GRED).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

VirtualQueues=
Specifies the number of virtual queues. Takes an integer in the range 1...16. Defaults to unset and kernel's default is used.

DefaultVirtualQueue=
Specifies the number of default virtual queue. This must be less than *VirtualQueue=*. Defaults to unset and kernel's default is used.

GenericRIO=
Takes a boolean. It turns on the RIO−like buffering scheme. Defaults to unset and kernel's default is used.

[FAIRQUEUEINGCONTROLLEDDELAY] SECTION OPTIONS

The [FairQueueingControlledDelay] section manages the queueing discipline (qdisc) of fair queuing controlled delay (FQ−CoDel).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=
Specifies the hard limit on the real queue size. When this limit is reached, incoming packets are dropped. Defaults to unset and kernel's default is used.

MemoryLimitBytes=
Specifies the limit on the total number of bytes that can be queued in this FQ−CoDel instance. When

suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel's default is used.

Flows=
Specifies the number of flows into which the incoming packets are classified. Defaults to unset and kernel's default is used.

TargetSec=
Takes a timespan. Specifies the acceptable minimum standing/persistent queue delay. Defaults to unset and kernel's default is used.

IntervalSec=
Takes a timespan. This is used to ensure that the measured minimum delay does not become too stale. Defaults to unset and kernel's default is used.

QuantumBytes=
Specifies the number of bytes used as the "deficit" in the fair queuing algorithm timespan. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel's default is used.

ECN=
Takes a boolean. This can be used to mark packets instead of dropping them. Defaults to unset and kernel's default is used.

CEThresholdSec=
Takes a timespan. This sets a threshold above which all packets are marked with ECN Congestion Experienced (CE). Defaults to unset and kernel's default is used.

[FAIRQUEUEING] SECTION OPTIONS

The [FairQueueing] section manages the queueing discipline (qdisc) of fair queue traffic policing (FQ).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=
Specifies the hard limit on the real queue size. When this limit is reached, incoming packets are dropped. Defaults to unset and kernel's default is used.

FlowLimit=
Specifies the hard limit on the maximum number of packets queued per flow. Defaults to unset and kernel's default is used.

QuantumBytes=
Specifies the credit per dequeue RR round, i.e. the amount of bytes a flow is allowed to dequeue at once. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel's default is used.

InitialQuantumBytes=
Specifies the initial sending rate credit, i.e. the amount of bytes a new flow is allowed to dequeue initially. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. Defaults to unset and kernel's default is used.

MaximumRate=
Specifies the maximum sending rate of a flow. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. Defaults to unset and kernel's default is used.

Buckets=
Specifies the size of the hash table used for flow lookups. Defaults to unset and kernel's default is used.

OrphanMask=
Takes an unsigned integer. For packets not owned by a socket, fq is able to mask a part of hash and reduce number of buckets associated with the traffic. Defaults to unset and kernel's default is used.

Pacing=
Takes a boolean, and enables or disables flow pacing. Defaults to unset and kernel's default is used.

CEThresholdSec=
Takes a timespan. This sets a threshold above which all packets are marked with ECN Congestion Experienced (CE). Defaults to unset and kernel's default is used.

[TRIVIALLINKEQUALIZER] SECTION OPTIONS

The [TrivialLinkEqualizer] section manages the queueing discipline (qdisc) of trivial link equalizer (teql).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

Id=
Specifies the interface ID "N" of teql. Defaults to "0". Note that when teql is used, currently, the module **sch_teql** with **max_equalizers=N+1** option must be loaded before **systemd−networkd** is started.

[HIERARCHYTOKENBUCKET] SECTION OPTIONS

The [HierarchyTokenBucket] section manages the queueing discipline (qdisc) of hierarchy token bucket (htb).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

DefaultClass=
Takes the minor id in hexadecimal of the default class. Unclassified traffic gets sent to the class. Defaults to unset.

RateToQuantum=
Takes an unsigned integer. The DRR quantums are calculated by dividing the value configured in *Rate=* by *RateToQuantum=*.

[HIERARCHYTOKENBUCKETCLASS] SECTION OPTIONS

The [HierarchyTokenBucketClass] section manages the traffic control class of hierarchy token bucket (htb).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", or a qdisc identifier. The qdisc identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

ClassId=
Configures the unique identifier of the class. It is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to unset.

Priority=
Specifies the priority of the class. In the round−robin process, classes with the lowest priority field are tried for packets first.

QuantumBytes=
Specifies how many bytes to serve from leaf at once. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

MTUBytes=
Specifies the maximum packet size we create. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

OverheadBytes=
Takes an unsigned integer which specifies per−packet size overhead used in rate computations. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

Rate=
Specifies the maximum rate this class and all its children are guaranteed. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. This setting is mandatory.

CeilRate=
Specifies the maximum rate at which a class can send, if its parent has bandwidth to spare. When suffixed with K, M, or G, the specified size is parsed as Kilobits, Megabits, or Gigabits, respectively, to the base of 1000. When unset, the value specified with *Rate=* is used.

BufferBytes=
Specifies the maximum bytes burst which can be accumulated during idle period. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

CeilBufferBytes=
Specifies the maximum bytes burst for ceil which can be accumulated during idle period. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024.

[HEAVYHITTERFILTER] SECTION OPTIONS

The [HeavyHitterFilter] section manages the queueing discipline (qdisc) of Heavy Hitter Filter (hhf).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=
Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

PacketLimit=
Specifies the hard limit on the queue size in number of packets. When this limit is reached, incoming packets are dropped. An unsigned integer in the range 0–4294967294. Defaults to unset and kernel's default is used.

[QUICKFAIRQUEUEING] SECTION OPTIONS

The [QuickFairQueueing] section manages the queueing discipline (qdisc) of Quick Fair Queueing (QFQ).

Parent=
Configures the parent Queueing Discipline (qdisc). Takes one of "root", "clsact", "ingress" or a class identifier. The class identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

Handle=

Configures the major number of unique identifier of the qdisc, known as the handle. Takes a hexadecimal number in the range 0x1–0xffff. Defaults to unset.

[QUICKFAIRQUEUEINGCLASS] SECTION OPTIONS

The [QuickFairQueueingClass] section manages the traffic control class of Quick Fair Queueing (qfq).

Parent=

Configures the parent Queueing Discipline (qdisc). Takes one of "root", or a qdisc identifier. The qdisc identifier is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to "root".

ClassId=

Configures the unique identifier of the class. It is specified as the major and minor numbers in hexadecimal in the range 0x1–Oxffff separated with a colon ("major:minor"). Defaults to unset.

Weight=

Specifies the weight of the class. Takes an integer in the range 1...1023. Defaults to unset in which case the kernel default is used.

MaxPacketBytes=

Specifies the maximum packet size in bytes for the class. When suffixed with K, M, or G, the specified size is parsed as Kilobytes, Megabytes, or Gigabytes, respectively, to the base of 1024. When unset, the kernel default is used.

[BRIDGEVLAN] SECTION OPTIONS

The [BridgeVLAN] section manages the VLAN ID configuration of a bridge port and accepts the following keys. Specify several [BridgeVLAN] sections to configure several VLAN entries. The *VLANFiltering=* option has to be enabled, see the [Bridge] section in **systemd.netdev**(5).

VLAN=

The VLAN ID allowed on the port. This can be either a single ID or a range M−N. VLAN IDs are valid from 1 to 4094.

EgressUntagged=

The VLAN ID specified here will be used to untag frames on egress. Configuring *EgressUntagged=* implicates the use of *VLAN=* above and will enable the VLAN ID for ingress as well. This can be either a single ID or a range M−N.

PVID=

The Port VLAN ID specified here is assigned to all untagged frames at ingress. *PVID=* can be used only once. Configuring *PVID=* implicates the use of *VLAN=* above and will enable the VLAN ID for ingress as well.

EXAMPLES

Example 1. Static network configuration

```
# /etc/systemd/network/50−static.network
[Match]
Name=enp2s0

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
```

This brings interface "enp2s0" up with a static address. The specified gateway will be used for a default route.

Example 2. DHCP on ethernet links

```
# /etc/systemd/network/80−dhcp.network
[Match]
Name=en*
```

```
[Network]
DHCP=yes
```

This will enable DHCPv4 and DHCPv6 on all interfaces with names starting with "en" (i.e. ethernet interfaces).

Example 3. IPv6 Prefix Delegation

```
# /etc/systemd/network/55−ipv6−pd−upstream.network
[Match]
Name=enp1s0

[Network]
DHCP=ipv6

# /etc/systemd/network/56−ipv6−pd−downstream.network
[Match]
Name=enp2s0

[Network]
IPv6SendRA=yes
DHCPv6PrefixDelegation=yes
```

This will enable DHCPv6−PD on the interface enp1s0 as an upstream interface where the DHCPv6 client is running and enp2s0 as a downstream interface where the prefix is delegated to. The delegated prefixes are distributed by IPv6 Router Advertisement on the downstream network.

Example 4. A bridge with two enslaved links

```
# /etc/systemd/network/25−bridge−static.network
[Match]
Name=bridge0

[Network]
Address=192.168.0.15/24
Gateway=192.168.0.1
DNS=192.168.0.1

# /etc/systemd/network/25−bridge−slave−interface−1.network
[Match]
Name=enp2s0

[Network]
Bridge=bridge0

# /etc/systemd/network/25−bridge−slave−interface−2.network
[Match]
Name=wlp3s0

[Network]
Bridge=bridge0
```

This creates a bridge and attaches devices "enp2s0" and "wlp3s0" to it. The bridge will have the specified static address and network assigned, and a default route via the specified gateway will be added. The specified DNS server will be added to the global list of DNS resolvers.

Example 5. Bridge port with VLAN forwarding

```
# /etc/systemd/network/25−bridge−slave−interface−1.network
[Match]
Name=enp2s0

[Network]
Bridge=bridge0

[BridgeVLAN]
VLAN=1−32
PVID=42
EgressUntagged=42

[BridgeVLAN]
VLAN=100−200

[BridgeVLAN]
EgressUntagged=300−400
```

This overrides the configuration specified in the previous example for the interface "enp2s0", and enables VLAN on that bridge port. VLAN IDs 1−32, 42, 100−400 will be allowed. Packets tagged with VLAN IDs 42, 300−400 will be untagged when they leave on this interface. Untagged packets which arrive on this interface will be assigned VLAN ID 42.

Example 6. Various tunnels

```
/etc/systemd/network/25−tunnels.network
[Match]
Name=ens1

[Network]
Tunnel=ipip−tun
Tunnel=sit−tun
Tunnel=gre−tun
Tunnel=vti−tun


/etc/systemd/network/25−tunnel−ipip.netdev
[NetDev]
Name=ipip−tun
Kind=ipip


/etc/systemd/network/25−tunnel−sit.netdev
[NetDev]
Name=sit−tun
Kind=sit


/etc/systemd/network/25−tunnel−gre.netdev
[NetDev]
Name=gre−tun
Kind=gre


/etc/systemd/network/25−tunnel−vti.netdev
[NetDev]
```

```
Name=vti−tun
Kind=vti
```

This will bring interface "ens1" up and create an IPIP tunnel, a SIT tunnel, a GRE tunnel, and a VTI tunnel using it.

Example 7. A bond device

```
# /etc/systemd/network/30−bond1.network
[Match]
Name=bond1

[Network]
DHCP=ipv6

# /etc/systemd/network/30−bond1.netdev
[NetDev]
Name=bond1
Kind=bond

# /etc/systemd/network/30−bond1−dev1.network
[Match]
MACAddress=52:54:00:e9:64:41

[Network]
Bond=bond1

# /etc/systemd/network/30−bond1−dev2.network
[Match]
MACAddress=52:54:00:e9:64:42

[Network]
Bond=bond1
```

This will create a bond device "bond1" and enslave the two devices with MAC addresses 52:54:00:e9:64:41 and 52:54:00:e9:64:42 to it. IPv6 DHCP will be used to acquire an address.

Example 8. Virtual Routing and Forwarding (VRF)

Add the "bond1" interface to the VRF master interface "vrf1". This will redirect routes generated on this interface to be within the routing table defined during VRF creation. For kernels before 4.8 traffic won't be redirected towards the VRFs routing table unless specific ip−rules are added.

```
# /etc/systemd/network/25−vrf.network
[Match]
Name=bond1

[Network]
VRF=vrf1
```

Example 9. MacVTap

This brings up a network interface "macvtap−test" and attaches it to "enp0s25".

```
# /lib/systemd/network/25−macvtap.network
[Match]
Name=enp0s25
```

```
[Network]
MACVTAP=macvtap−test
```

Example 10. A Xfrm interface with physical underlying device.

```
# /etc/systemd/network/27−xfrm.netdev
[NetDev]
Name=xfrm0
Kind=xfrm

[Xfrm]
InterfaceId=7

# /etc/systemd/network/27−eth0.network
[Match]
Name=eth0

[Network]
Xfrm=xfrm0
```

This creates a "xfrm0" interface and binds it to the "eth0" device. This allows hardware based ipsec offloading to the "eth0" nic. If offloading is not needed, xfrm interfaces can be assigned to the "lo" device.

SEE ALSO

systemd(1), **systemd-networkd.service**(8), **systemd.link**(5), **systemd.netdev**(5), **systemd-resolved.service**(8)

NOTES

1. RFC 7217
   https://tools.ietf.org/html/rfc7217
2. Link-Local Multicast Name Resolution
   https://tools.ietf.org/html/rfc4795
3. Multicast DNS
   https://tools.ietf.org/html/rfc6762
4. DNS-over-TLS
   https://tools.ietf.org/html/rfc7858
5. DNSSEC
   https://tools.ietf.org/html/rfc4033
6. IEEE 802.1AB-2016
   https://standards.ieee.org/findstds/standard/802.1AB-2016.html
7. ip-sysctl.txt
   https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt
8. RFC 4941
   https://tools.ietf.org/html/rfc4941
9. RFC 1027
   https://tools.ietf.org/html/rfc1027
10. RFC 6275
    https://tools.ietf.org/html/rfc6275
11. RFC 5227
    https://tools.ietf.org/html/rfc5227
12. RFC 4862
    https://tools.ietf.org/html/rfc4862

13. RFC 3041
https://tools.ietf.org/html/rfc3041
14. RFC 3484
https://tools.ietf.org/html/rfc3484
15. RFC 4191
https://tools.ietf.org/html/rfc4191
16. RFC 8520
https://tools.ietf.org/html/rfc8520
17. RFC 7844
https://tools.ietf.org/html/rfc7844
18. C-style escapes
https://en.wikipedia.org/wiki/Escape_sequences_in_C#Table_of_escape_sequences
19. RFC 7084
https://tools.ietf.org/html/rfc7084
20. RFC 3315
https://tools.ietf.org/html/rfc3315#section-17.2.1
21. RFC 4291
https://tools.ietf.org/html/rfc4291#section-2.5.4
22. RFC 1542
https://tools.ietf.org/html/rfc1542
23. RFC 4861
https://tools.ietf.org/html/rfc4861