NETWORKCTL

NAME

networkctl − Query or modify the status of network links

SYNOPSIS

networkctl [OPTIONS...] COMMAND [LINK...]

DESCRIPTION

networkctl may be used to query or modify the state of the network links as seen by **systemd−networkd**. Please refer to **systemd-networkd.service**(8) for an introduction to the basic concepts, functionality, and configuration syntax.

COMMANDS

The following commands are understood:

list [***PATTERN...***]

Show a list of existing links and their status. If one or more ***PATTERN***s are specified, only links matching one of them are shown. If no further arguments are specified shows all links, otherwise just the specified links.

Produces output similar to:

IDX LINK TYPE OPERATIONAL SETUP 1 lo loopback carrier unmanaged 2 eth0 ether routable configured 3 virbr0 ether no−carrier unmanaged 4 virbr0−nic ether off unmanaged
4 links listed.

The operational status is one of the following:

missing
: The device is missing.

off
: The device is powered down.

no−carrier
: The device is powered up, but does not yet have a carrier.

dormant
: The device has a carrier, but is not yet ready for normal traffic.

degraded−carrier
: One of the bonding or bridge slave network interfaces is in off, no−carrier, or dormant state, and the master interface has no address.

carrier
: The link has carrier, or for bond or bridge master, all bonding or bridge slave network interfaces are enslaved to the master.

degraded
: The link has carrier and addresses valid on the local link configured. For bond or bridge master this means that not all slave network interfaces have carrier but at least one does.

enslaved
: The link has carrier and is enslaved to bond or bridge master network interface.

routable
: The link has carrier and routable address configured. For bond or bridge master it is not necessary for all slave network interfaces to have carrier, but at least one must.

The setup status is one of the following:

pending
: **systemd-udevd**(8) is still processing the link, we do not yet know if we will manage it.

initialized
: **systemd-udevd**(8) has processed the link, but we do not yet know if we will manage it.

configuring
: Configuration for the link is being retrieved or the link is being configured.

configured
: Link has been configured successfully.

unmanaged
: **systemd−networkd** is not handling the link.

failed
: **systemd−networkd** failed to configure the link.

linger
: The link is gone, but has not yet been dropped by **systemd−networkd**.

status [***PATTERN...***]

Show information about the specified links: type, state, kernel module driver, hardware and IP address, configured DNS servers, etc. If one or more ***PATTERN***s are specified, only links matching one of them are shown.

When no links are specified, an overall network status is shown. Also see the option **−−all**.

Produces output similar to:

● State: routable Online state: online Address: 10.193.76.5 on eth0 192.168.122.1 on virbr0 169.254.190.105 on eth0 fe80::5054:aa:bbbb:cccc on eth0 Gateway: 10.193.11.1 (CISCO SYSTEMS, INC.) on eth0 DNS: 8.8.8.8 8.8.4.4

In the overall network status, the online state depends on the individual online state of all required links. Managed links are required for online by default. In this case, the online state is one of the following:

unknown
: All links have unknown online status (i.e. there are no required links).

offline
: All required links are offline.

partial
: Some, but not all, required links are online.

online
: All required links are online.

lldp [***PATTERN...***]

Show discovered LLDP (Link Layer Discovery Protocol) neighbors. If one or more ***PATTERN***s are specified only neighbors on those interfaces are shown. Otherwise, shows discovered neighbors on all interfaces. Note that for this feature to work, ***LLDP=*** must be turned on for the specific interface, see **systemd.network**(5) for details.

Produces output similar to:

LINK SYSTEM−NAME SYSTEM−DESCRIPTION CHASSIS−ID PORT−ID PORT−DESCRIPTION CAPS
enp0s25 GS1900 − 00:e0:4c:00:00:00 2 Port #2 ..b.....
Capability Flags: o − Other; p − Repeater; b − Bridge; w − WLAN Access Point; r − Router; t − Telephone; d − DOCSIS cable device; a − Station; c − Customer VLAN; s − Service VLAN, m − Two−port MAC Relay (TPMR)
1 neighbor(s) listed.

label

Show numerical address labels that can be used for address selection. This is the same information that **ip-addrlabel**(8) shows. See **RFC 3484** for a discussion of address labels.

Produces output similar to:

Prefix/Prefixlen Label ::/0 1 fc00::/7 5 fec0::/10 11 2002::/16 2 3ffe::/16 12 2001:10::/28 7 2001::/32 6 ::ffff:0.0.0.0/96 4 ::/96 3 ::1/128 0

delete ***DEVICE...***

Deletes virtual netdevs. Takes interface name or index number.

up ***DEVICE...***

Bring devices up. Takes interface name or index number.

down ***DEVICE...***

Bring devices down. Takes interface name or index number.

renew ***DEVICE...***

Renew dynamic configurations e.g. addresses received from DHCP server. Takes interface name or index number.

forcerenew ***DEVICE...***

Send a FORCERENEW message to all connected clients, triggering DHCP reconfiguration. Takes interface name or index number.

reconfigure ***DEVICE...***

Reconfigure network interfaces. Takes interface name or index number. Note that this does not reload .netdev or .network corresponding to the specified interface. So, if you edit config files, it is necessary to call **networkctl reload** first to apply new settings.

reload

Reload .netdev and .network files.

If a new or modified .netdev file is found, then the corresponding netdev is created or updated, respectively. Note, if the corresponding interface already exists, then some of new settings may not be applied. E.g., VLAN ID cannot be changed after the interface was created, so changing [VLAN] ***Id=*** will not take effect if the matching VLAN interface already exists. To apply such settings, the interfaces need to be removed manually before reload. Also note that even if a .netdev file is removed, **systemd-networkd**(8) does not remove the existing netdev corresponding to the file.

If a new, modified, or removed .network file is found, then all interfaces that matched the file are reconfigured.

edit ***FILE|@DEVICE...***

Edit network configuration files, which include .network, .netdev, and .link files. If no network config file matching the given name is found, a new one will be created under /etc/ or /run/, depending on whether **−−runtime** is specified. Specially, if the name is prefixed by "@", it will be treated as a network interface, and editing will be performed on the network config files associated with it. Additionally, the interface name can be suffixed with ":network" (default), ":link", or ":netdev", in order to choose the type of network config to operate on.

If **−−drop−in=** is specified, edit the drop−in file instead of the main configuration file. Unless **−−no−reload** is specified, **systemd-networkd**(8) will be reloaded after the edit of the .network or .netdev files finishes. The same applies for .link files and **systemd-udevd**(8). Note that the changed link settings are not automatically applied after reloading. To achieve that, trigger uevents for the corresponding interface. Refer to **systemd.link**(5) for more information.

If **−−stdin** is specified, the new content will be read from standard input. In this mode, the old content of the file is discarded.

cat [***FILE|@DEVICE...***]

Show network configuration files. This command honors the "@" prefix in a similar way as **edit**, with support for an additional suffix ":all" for showing all types of configuration files associated with the interface at once. When no argument is specified, **networkd.conf**(5) and its drop−in files will be shown.

mask ***FILE...***

Mask network configuration files, which include .network, .netdev, and .link files. A symlink of the given name will be created under /etc/ or /run/, depending on whether **−−runtime** is specified, that points to /dev/null. If a non−empty config file with the specified name exists under the target directory or a directory with higher priority (e.g. **−−runtime** is used while an existing config resides in /etc/), the operation is aborted.

This command honors **−−no−reload** in the same way as **edit**.

unmask ***FILE...***

Unmask network configuration files, i.e. reverting the effect of **mask**. Note that this command operates regardless of the scope of the directory, i.e. **−−runtime** is of no effect.

This command honors **−−no−reload** in the same way as **edit** and **mask**.

persistent−storage ***BOOL***

Notify **systemd-networkd.service**(8) that the persistent storage for the service is ready. This is called by systemd−networkd−persistent−storage.service. Usually, this command should not be called manually by users or administrators.

OPTIONS

The following options are understood:

−a −−all
Show all links with **status**.
−s −−stats
Show link statistics with **status**.
−l, **−−full**
Do not ellipsize the output.
−n, **−−lines=**
When used with **status**, controls the number of journal lines to show, counting from the most recent ones. Takes a positive integer argument. Defaults to 10.
−−drop−in=*NAME*
When used with **edit**, edit the drop−in file ***NAME*** instead of the main configuration file.
−−no−reload
When used with **edit**, **mask**, or **unmask**, **systemd-networkd.service**(8) or **systemd-udevd.service**(8) will not be reloaded after the operation finishes.
−−runtime
When used with **edit** or **mask**, operate on the file under /run/ instead of /etc/.
−−stdin
When used with **edit**, the contents of the file will be read from standard input and the editor will not be launched. In this mode, the old contents of the file are automatically replaced. This is useful to "edit" configuration from scripts, especially so that drop−in directories are created and populated in one go. Multiple drop−ins may be "edited" in this mode with **−−drop−in=**, and the same contents will be written to all of them. Otherwise, exactly one main configuration file is expected.
−−no−ask−password
Do not query the user for authentication for privileged operations.
−−json=*MODE*
Shows output formatted as JSON. Expects one of "short" (for the shortest possible output without any redundant whitespace or line breaks), "pretty" (for a pretty version of the same, with indentation and line breaks) or "off" (to turn off JSON output, the default).
−h, **−−help**
Print a short help text and exit.
−−version
Print a short version string and exit.
−−no−legend
Do not print the legend, i.e. column headers and the footer with hints.
−−no−pager
Do not pipe output into a pager.

EXIT STATUS

On success, 0 is returned, a non−zero failure code otherwise.

SEE ALSO

systemd-networkd.service(8), **systemd.network**(5), **systemd.netdev**(5), **ip**(8)

SYSTEMD−NETWORKD.SERVICE

NAME

systemd-networkd.service, systemd-networkd − Network manager

SYNOPSIS

systemd−networkd.service
/usr/lib/systemd/systemd−networkd

DESCRIPTION

systemd−networkd is a system service that manages networks. It detects and configures network devices as they appear, as well as creating virtual network devices.

Certain low−level settings of physical network devices (e.g. device names and altnames) as well as the creation of SR−IOV virtual functions on physical network interfaces may be managed by **systemd-udevd**(8) according to the contents of **systemd.link**(5) files.

systemd−networkd will create "virtual" network devices (e.g. bridges and tunnels) based on the configuration in **systemd.netdev**(5) files, respecting the [Match] sections in those files.

systemd−networkd will manage network addresses and routes for any link for which it finds a .network file with an appropriate [Match] section, see **systemd.network**(5). For those links, it will flush existing network addresses and routes when bringing up the device (except when directed not to). Any links not matched by one of the .network files will be ignored. It is also possible to explicitly tell systemd−networkd to ignore a link by using the ***Unmanaged=yes*** option, see **systemd.network**(5).

When systemd−networkd exits, it generally leaves existing network devices and configuration intact. This makes it possible to transition from the initrd and to restart the service without breaking connectivity. This also means that when configuration is updated and systemd−networkd is restarted, netdev interfaces for which configuration was removed will not be dropped, and may need to be cleaned up manually.

systemd−networkd may be introspected and controlled at runtime using **networkctl**(1).

See **org.freedesktop.network1**(5) and **org.freedesktop.LogControl1**(5) for a description of the D−Bus API.

CONFIGURATION FILES

The configuration files are read from the files located in the system network directory /usr/lib/systemd/network, the volatile runtime network directory /run/systemd/network and the local administration network directory /etc/systemd/network.

Networks are configured in .network files, see **systemd.network**(5), and virtual network devices are configured in .netdev files, see **systemd.netdev**(5).

SEE ALSO

networkctl(1), **systemd**(1), **systemd.link**(5), **systemd.network**(5), **systemd.netdev**(5), **systemd-networkd-wait-online.service**(8), **systemd-network-generator.service**(8), **org.freedesktop.network1**(5)

SYSTEMD.LINK

NAME

systemd.link − Network device configuration

SYNOPSIS

link.link

DESCRIPTION

A plain ini−style text file that encodes configuration for matching network devices, used by **systemd-udevd**(8) and in particular its **net_setup_link** builtin. See **systemd.syntax**(7) for a general description of the syntax.

Note that some distributions may incorporate .link files in their early boot facilities (e.g. by including copies of the .link files in initramfs). As such it may be necessary to take manual steps to ensure that any local changes are consistent with early−boot storage facilities. The relevant distribution−specific documentation should be consulted.

The .link files are read from the files located in the system network directory /usr/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 alphanumeric order, regardless of the directories in which they live. However, files with identical filenames replace each other. It is recommended that each filename is prefixed with a number smaller than "70" (e.g. 10−eth0.link). Otherwise, the default .link files or those generated by **systemd-network-generator.service**(8) may take precedence over user configured files. Files in /etc/ have the highest priority, files in /run/ take precedence over files with the same name in /usr/lib/. This can be used to override a system−supplied link 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 link file foo.link, a "drop−in" directory foo.link.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 /usr/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 /usr/lib/. Drop−in files under any of these directories take precedence over the main link file wherever located.

The link file contains a [Match] section, which determines if a given link file may be applied to a given device, as well as a [Link] section specifying how the device should be configured. The first (in lexical order) of the link files that matches a given device is applied. Note that a default file 99−default.link is shipped by the system. Any user−supplied .link should hence have a lexically earlier name to be considered at all.

See **udevadm**(8) for diagnosing problems with .link files.

[MATCH] SECTION OPTIONS

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

OriginalName=*

The first (in alphanumeric order) of the link files that matches a given interface is applied, all later files are ignored, even if they match as well. The following keys are accepted:

MACAddress=
A whitespace−separated list of hardware addresses. The acceptable formats are:
colon−delimited hexadecimal
Each field must be one byte. E.g. "12:34:56:78:90:ab" or "AA:BB:CC:DD:EE:FF".
hyphen−delimited hexadecimal
Each field must be one byte. E.g. "12−34−56−78−90−ab" or "AA−BB−CC−DD−EE−FF".
dot−delimited hexadecimal
Each field must be two bytes. E.g. "1234.5678.90ab" or "AABB.CCDD.EEFF".
IPv4 address format
E.g. "127.0.0.1" or "192.168.0.1".
IPv6 address format
E.g. "2001:0db8:85a3::8a2e:0370:7334" or "::1".
The total length of each MAC address must be 4 (for IPv4 tunnel), 6 (for Ethernet), 16 (for IPv6 tunnel), or 20 (for InfiniBand). 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. Defaults to unset.

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, or IPv4 or IPv6 address format. 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. Defaults to unset.

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.

Kind=
A whitespace−separated list of shell−style globs matching the device kind, as exposed by **networkctl status *INTERFACE*** or **ip −d link show *INTERFACE***. If the list is prefixed with a "!", the test is inverted. Some valid values are "bond", "bridge", "gre", "tun", "veth". Valid kinds are given by netlink's "IFLA_INFO_KIND" attribute, 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.

OriginalName=
A whitespace−separated list of shell−style globs matching the device name, as exposed by the udev property "INTERFACE". This cannot be used to match on names that have already been changed from userspace. Caution is advised when matching on kernel−assigned names, as they are known to be unstable between reboots.

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, the 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, the 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, the 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, the previously assigned value is cleared.

Credential=
Checks whether the specified credential was passed to the systemd−udevd.service service. See **System and Service Credentials** for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, the 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, the 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, the previously assigned value is cleared.

[LINK] SECTION OPTIONS

The [Link] section accepts the following keys:

Description=
A description of the device.

Property=
Set specified udev properties. This takes space separated list of key−value pairs concatenated with equal sign ("="). Example:

```
Property=HOGE=foo BAR=baz
```

This option supports simple specifier expansion, see the Specifiers section below. This option can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.
This setting is useful to configure the "ID_NET_MANAGED_BY=" property which declares which network management service shall manage the interface, which is respected by **systemd-networkd**(8) and others. Use

```
Property=ID_NET_MANAGED_BY=io.systemd.Network
```

to declare explicitly that **systemd−networkd** shall manage the interface, or set the property to something else to declare explicitly it shall not do so. See **systemd.network**(5) for details how this property is used to match interface names.

ImportProperty=
Import specified udev properties from the saved database. This takes space separated list of property names. Example:

```
ImportProperty=HOGE BAR
```

This option supports simple specifier expansion, see the Specifiers section below. This option can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.
If the same property is also set in ***Property=*** in the above, then the imported property value will be overridden by the value specified in ***Property=***.

UnsetProperty=
Unset specified udev properties. This takes space separated list of property names. Example:

```
UnsetProperty=HOGE BAR
```

This option supports simple specifier expansion, see the Specifiers section below. This option can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.
This setting is applied after ***ImportProperty=*** and ***Property=*** are applied. Hence, if the same property is specified in ***ImportProperty=*** or ***Property=***, then the imported or specified property value will be ignored, and the property will be unset.

Alias=
The ***ifalias*** interface property is set to this value.

MACAddressPolicy=
The policy by which the MAC address should be set. The available policies are:

persistent
If the hardware has a persistent MAC address, as most hardware should, and if it is used by the kernel, nothing is done. Otherwise, a new MAC address is generated which is guaranteed to be the same on every boot for the given machine and the given device, but which is otherwise random. This feature depends on ID_NET_NAME_* properties to exist for the link. On hardware where these properties are not set, the generation of a persistent MAC address will fail.

random
If the kernel is using a random MAC address, nothing is done. Otherwise, a new address is randomly generated each time the device appears, typically at boot. Either way, the random address will have the "unicast" and "locally administered" bits set.

none
Keeps the MAC address assigned by the kernel. Or use the MAC address specified in ***MACAddress=***.

An empty string assignment is equivalent to setting "none".

MACAddress=
The interface MAC address to use. For this setting to take effect, ***MACAddressPolicy=*** must either be unset, empty, or "none".

NamePolicy=
An ordered, space−separated list of policies by which the interface name should be set. ***NamePolicy=*** may be disabled by specifying **net.ifnames=0** on the kernel command line. Each of the policies may fail, and the first successful one is used. The name is not set directly, but is exported to udev as the property **ID_NET_NAME**, which is, by default, used by a **udev**(7), rule to set ***NAME***. The available policies are:

kernel
If the kernel claims that the name it has set for a device is predictable, then no renaming is performed.

database
The name is set based on entries in the udev's Hardware Database with the key ***ID_NET_NAME_FROM_DATABASE***.

onboard
The name is set based on information given by the firmware for on−board devices, as exported by the udev property ***ID_NET_NAME_ONBOARD***. See **systemd.net-naming-scheme**(7).

slot
The name is set based on information given by the firmware for hot−plug devices, as exported by the udev property ***ID_NET_NAME_SLOT***. See **systemd.net-naming-scheme**(7).

path
The name is set based on the device's physical location, as exported by the udev property ***ID_NET_NAME_PATH***. See **systemd.net-naming-scheme**(7).

mac
The name is set based on the device's persistent MAC address, as exported by the udev property ***ID_NET_NAME_MAC***. See **systemd.net-naming-scheme**(7).

keep
If the device already had a name given by userspace (as part of creation of the device or a rename), keep it.

Name=
The interface name to use. This option has lower precedence than ***NamePolicy=***, so for this setting to take effect, ***NamePolicy=*** must either be unset, empty, disabled, or all policies configured there must fail. Also see the example below with "Name=dmz0".
Note that specifying a name that the kernel might use for another interface (for example "eth0") is dangerous because the name assignment done by udev will race with the assignment done by the kernel, and only one interface may use the name. Depending on the order of operations, either udev or the kernel will win, making the naming unpredictable. It is best to use some different prefix, for example "internal0"/"external0" or "lan0"/"lan1"/"lan3".
Interface names must have a minimum length of 1 character and a maximum length of 15 characters, and may contain any 7bit ASCII character, with the exception of control characters, ":", "/" and "%". While "." is an allowed character, it is recommended to avoid it when naming interfaces as various tools (such as **resolvconf**(1)) use it as separator character. Also, fully numeric interface names are not allowed (in order to avoid ambiguity with interface specification by numeric indexes), nor are the special strings ".", "..", "all" and "default".

AlternativeNamesPolicy=
A space−separated list of policies by which the interface's alternative names should be set. Each of the policies may fail, and all successful policies are used. The available policies are "database", "onboard", "slot", "path", and "mac". If the kernel does not support the alternative names, then this setting will be ignored.

AlternativeName=
The alternative interface name to use. This option can be specified multiple times. If the empty string is assigned to this option, the list is reset, and all prior assignments have no effect. If the kernel does not support the alternative names, then this setting will be ignored.
Alternative interface names may be used to identify interfaces in various tools. In contrast to the primary name (as configured with ***Name=*** above) there may be multiple alternative names referring to the same interface. Alternative names may have a maximum length of 127 characters, in contrast to the 15 allowed for the primary interface name, but otherwise are subject to the same naming constraints.

TransmitQueues=
Specifies the device's number of transmit queues. An integer in the range 1…4096. When unset, the kernel's default will be used.

ReceiveQueues=
Specifies the device's number of receive queues. An integer in the range 1…4096. When unset, the kernel's default will be used.

TransmitQueueLength=
Specifies the transmit queue length of the device in number of packets. An unsigned integer in the range 0...4294967294. When unset, the kernel's default will be used.

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.

BitsPerSecond=
The speed to set for the device, the value is rounded down to the nearest Mbps. The usual suffixes K, M, G are supported and are understood to the base of 1000.

Duplex=
The duplex mode to set for the device. The accepted values are **half** and **full**.

AutoNegotiation=
Takes a boolean. If set to yes, automatic negotiation of transmission parameters is enabled. Autonegotiation is a procedure by which two connected ethernet devices choose common transmission parameters, such as speed, duplex mode, and flow control. When unset, the kernel's default will be used.
Note that if autonegotiation is enabled, speed and duplex settings are read−only. If autonegotiation is disabled, speed and duplex settings are writable if the driver supports multiple link modes.

WakeOnLan=
The Wake−on−LAN policy to set for the device. Takes the special value "off" which disables Wake−on−LAN, or space separated list of the following words:

phy
Wake on PHY activity.

unicast
Wake on unicast messages.

multicast
Wake on multicast messages.

broadcast
Wake on broadcast messages.

arp
Wake on ARP.

magic
Wake on receipt of a magic packet.

secureon
Enable SecureOn password for MagicPacket. Implied when ***WakeOnLanPassword=*** is specified. If specified without ***WakeOnLanPassword=*** option, then the password is read from the credential "***LINK***.link.wol.password" (e.g., "60−foo.link.wol.password"), and if the credential not found, then read from "wol.password". See ***ImportCredential=/LoadCredential=/SetCredential=*** in **systemd.exec**(5) for details. The password in the credential, must be 6 bytes in hex format with each byte separated by a colon (":") like an Ethernet MAC address, e.g., "aa:bb:cc:dd:ee:ff".

Defaults to unset, and the device's default will be used. This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.

WakeOnLanPassword=
Specifies the SecureOn password for MagicPacket. Takes an absolute path to a regular file or an **AF_UNIX** stream socket, or the plain password. When a path to a regular file is specified, the password is read from it. When an **AF_UNIX** stream socket is specified, a connection is made to it and the password is read from it. The password must be 6 bytes in hex format with each byte separated by a colon (":") like an Ethernet MAC address, e.g., "aa:bb:cc:dd:ee:ff". This implies ***WakeOnLan=secureon***. Defaults to unset, and the current value will not be changed.

Port=
The port option is used to select the device port. The supported values are:

tp
An Ethernet interface using Twisted−Pair cable as the medium.

aui
Attachment Unit Interface (AUI). Normally used with hubs.

bnc
An Ethernet interface using BNC connectors and co−axial cable.

mii
An Ethernet interface using a Media Independent Interface (MII).

fibre
An Ethernet interface using Optical Fibre as the medium.

Advertise=
This sets what speeds and duplex modes of operation are advertised for auto−negotiation. This implies "AutoNegotiation=yes". The supported values are:

Table 1. Supported advertise values By default, this is unset, i.e. all possible modes will be advertised. This option may be specified more than once, in which case all specified speeds and modes are advertised. If the empty string is assigned to this option, the list is reset, and all prior assignments have no effect.

ReceiveChecksumOffload=
Takes a boolean. If set to true, hardware offload for checksumming of ingress network packets is enabled. When unset, the kernel's default will be used.

TransmitChecksumOffload=
Takes a boolean. If set to true, hardware offload for checksumming of egress network packets is enabled. When unset, the kernel's default will be used.

TCPSegmentationOffload=
Takes a boolean. If set to true, TCP Segmentation Offload (TSO) is enabled. When unset, the kernel's default will be used.

TCP6SegmentationOffload=
Takes a boolean. If set to true, TCP6 Segmentation Offload (tx−tcp6−segmentation) is enabled. When unset, the kernel's default will be used.

GenericSegmentationOffload=
Takes a boolean. If set to true, Generic Segmentation Offload (GSO) is enabled. When unset, the kernel's default will be used.

GenericReceiveOffload=
Takes a boolean. If set to true, Generic Receive Offload (GRO) is enabled. When unset, the kernel's default will be used.

GenericReceiveOffloadHardware=
Takes a boolean. If set to true, hardware accelerated Generic Receive Offload (GRO) is enabled. When unset, the kernel's default will be used.

LargeReceiveOffload=
Takes a boolean. If set to true, Large Receive Offload (LRO) is enabled. When unset, the kernel's default will be used.

ReceivePacketSteeringCPUMask=
Configures Receive Packet Steering (RPS) list of CPUs to which RPS may forward traffic. Takes a list of CPU indices or ranges separated by either whitespace or commas. Alternatively, takes the special value "all", which will include all available CPUs in the mask. CPU ranges are specified by the lower and upper CPU indices separated by a dash (e.g. "2−6"). This option may be specified more than once, in which case the specified list of CPU ranges are merged. If an empty string is assigned, the list is reset, all assignments prior to this will have no effect. Defaults to unset and RPS CPU list is unchanged. To disable RPS when it was previously enabled, use the special value "disable".

ReceiveVLANCTAGHardwareAcceleration=
Takes a boolean. If set to true, receive VLAN CTAG hardware acceleration is enabled. When unset, the kernel's default will be used.

TransmitVLANCTAGHardwareAcceleration=
Takes a boolean. If set to true, transmit VLAN CTAG hardware acceleration is enabled. When unset, the kernel's default will be used.

ReceiveVLANCTAGFilter=
Takes a boolean. If set to true, receive filtering on VLAN CTAGs is enabled. When unset, the kernel's default will be used.

TransmitVLANSTAGHardwareAcceleration=
Takes a boolean. If set to true, transmit VLAN STAG hardware acceleration is enabled. When unset, the kernel's default will be used.

NTupleFilter=
Takes a boolean. If set to true, receive N−tuple filters and actions are enabled. When unset, the kernel's default will be used.

RxChannels=, ***TxChannels=***, ***OtherChannels=***, ***CombinedChannels=***
Specifies the number of receive, transmit, other, or combined channels, respectively. Takes an unsigned integer in the range 1...4294967295 or "max". If set to "max", the advertised maximum value of the hardware will be used. When unset, the number will not be changed. Defaults to unset.

RxBufferSize=, ***RxMiniBufferSize=***, ***RxJumboBufferSize=***, ***TxBufferSize=***
Specifies the maximum number of pending packets in the NIC receive buffer, mini receive buffer, jumbo receive buffer, or transmit buffer, respectively. Takes an unsigned integer in the range 1...4294967295 or "max". If set to "max", the advertised maximum value of the hardware will be used. When unset, the number will not be changed. Defaults to unset.

RxFlowControl=
Takes a boolean. When set, enables receive flow control, also known as the ethernet receive PAUSE message (generate and send ethernet PAUSE frames). When unset, the kernel's default will be used.

TxFlowControl=
Takes a boolean. When set, enables transmit flow control, also known as the ethernet transmit PAUSE message (respond to received ethernet PAUSE frames). When unset, the kernel's default will be used.

AutoNegotiationFlowControl=
Takes a boolean. When set, auto negotiation enables the interface to exchange state advertisements with the connected peer so that the two devices can agree on the ethernet PAUSE configuration. When unset, the kernel's default will be used.

GenericSegmentOffloadMaxBytes=
Specifies the maximum size of a Generic Segment Offload (GSO) packet the device should accept. The usual suffixes K, M, G are supported and are understood to the base of 1024. An unsigned integer in the range 1...65536. Defaults to unset.

GenericSegmentOffloadMaxSegments=
Specifies the maximum number of Generic Segment Offload (GSO) segments the device should accept. An unsigned integer in the range 1...65535. Defaults to unset.

UseAdaptiveRxCoalesce=, ***UseAdaptiveTxCoalesce=***
Boolean properties that, when set, enable/disable adaptive Rx/Tx coalescing if the hardware supports it. When unset, the kernel's default will be used.

RxCoalesceSec=, ***RxCoalesceIrqSec=***, ***RxCoalesceLowSec=***, ***RxCoalesceHighSec=***, ***TxCoalesceSec=***, ***TxCoalesceIrqSec=***, ***TxCoalesceLowSec=***, ***TxCoalesceHighSec=***
These properties configure the delay before Rx/Tx interrupts are generated after a packet is sent/received. The "Irq" properties come into effect when the host is servicing an IRQ. The "Low" and "High" properties come into effect when the packet rate drops below the low packet rate threshold or exceeds the high packet rate threshold respectively if adaptive Rx/Tx coalescing is enabled. When unset, the kernel's defaults will be used.

RxMaxCoalescedFrames=, ***RxMaxCoalescedIrqFrames=***, ***RxMaxCoalescedLowFrames=***, ***RxMaxCoalescedHighFrames=***, ***TxMaxCoalescedFrames=***, ***TxMaxCoalescedIrqFrames=***, ***TxMaxCoalescedLowFrames=***, ***TxMaxCoalescedHighFrames=***
These properties configure the maximum number of frames that are sent/received before a Rx/Tx interrupt is generated. The "Irq" properties come into effect when the host is servicing an IRQ. The "Low" and "High" properties come into effect when the packet rate drops below the low packet rate threshold or exceeds the high packet rate threshold respectively if adaptive Rx/Tx coalescing is enabled. When unset, the kernel's defaults will be used.

CoalescePacketRateLow=, ***CoalescePacketRateHigh=***
These properties configure the low and high packet rate (expressed in packets per second) threshold respectively and are used to determine when the corresponding coalescing settings for low and high packet

rates come into effect if adaptive Rx/Tx coalescing is enabled. If unset, the kernel's defaults will be used.

CoalescePacketRateSampleIntervalSec=
Configures how often to sample the packet rate used for adaptive Rx/Tx coalescing. This property cannot be zero. This lowest time granularity supported by this property is seconds. Partial seconds will be rounded up before being passed to the kernel. If unset, the kernel's default will be used.

StatisticsBlockCoalesceSec=
How long to delay driver in−memory statistics block updates. If the driver does not have an in−memory statistic block, this property is ignored. This property cannot be zero. If unset, the kernel's default will be used.

MDI=
Specifies the medium dependent interface (MDI) mode for the interface. A MDI describes the interface from a physical layer implementation to the physical medium used to carry the transmission. Takes one of the following words: "straight" (or equivalently: "mdi"), "crossover" (or equivalently: "mdi−x", "mdix"), and "auto". When "straight", the MDI straight through mode will be used. When "crossover", the MDI crossover (MDI−X) mode will be used. When "auto", the MDI status is automatically detected. Defaults to unset, and the kernel's default will be used.

SR−IOVVirtualFunctions=
Specifies the number of SR−IOV virtual functions. Takes an integer in the range 0...2147483647. Defaults to unset, and automatically determined from the values specified in the ***VirtualFunction=*** settings in the [SR−IOV] sections.

[SR−IOV] SECTION OPTIONS

SR−IOV provides the ability to partition a single physical PCI resource into virtual PCI functions which can then be e.g. injected into a VM. In the case of network VFs, SR−IOV reduces latency and CPU utilisation for north−south network traffic (that is, traffic with endpoints outside the host machine), by allowing traffic to bypass the host machine's network stack.

The presence of an [SR−IOV] section in a .link file will cause the creation and configuration of the specified virtual function. Within a .network file, the specified virtual function will be configured, but must already exist. Specify several [SR−IOV] sections to configure several SR−IOVs.

The [SR−IOV] section accepts the following keys.

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

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

QualityOfService=
Specifies quality of service of the virtual function. Takes an 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 one 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 one 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.

SPECIFIERS

Some settings resolve specifiers which may be used to write generic unit files referring to runtime or unit parameters that are replaced when the unit files are loaded. Specifiers must be known and resolvable for the setting to be valid. The following specifiers are understood:

Table 2. Specifiers available in unit files

EXAMPLES

Example 1. /usr/lib/systemd/network/99−default.link

The link file 99−default.link that is shipped with systemd defines the default policies for the interface name, alternative names, and MAC address of links.

```
[Match] OriginalName=*
[Link] NamePolicy=keep kernel database onboard slot path AlternativeNamesPolicy=database onboard slot path MACAddressPolicy=persistent
```

Example 2. /etc/systemd/network/10−dmz.link

This example assigns the fixed name "dmz0" to the interface with the MAC address 00:a0:de:63:7a:e6:

```
[Match] MACAddress=00:a0:de:63:7a:e6
[Link] Name=dmz0
```

NamePolicy= is not set, so ***Name=*** takes effect. We use the "10−" prefix to order this file early in the list. Note that it needs to be before 99−default.link, i.e. it needs a numerical prefix, to have any effect at all.

Example 3. (Re−)applying a .link file to an interface

After a new .link file has been created, or an existing .link file modified, the new settings may be applied to the matching interface with the following commands:

```
$ sudo udevadm control −−reload $ sudo ip link set eth0 down $ sudo udevadm trigger −−verbose −−settle −−action add /sys/class/net/eth0
```

You may also need to stop the service that manages the network interface, e.g. **systemd-networkd.service**(8) or NetworkManager.service before the above operation, and then restart the service after that. For more details about **udevadm** command, see **udevadm**(8).

Example 4. Debugging *NamePolicy=* assignments

```
$ sudo SYSTEMD_LOG_LEVEL=debug udevadm test−builtin net_setup_link /sys/class/net/hub0 ... Parsed configuration file /usr/lib/systemd/network/99−default.link Parsed configuration file /etc/systemd/network/10−eth0.link ID_NET_DRIVER=cdc_ether Config file /etc/systemd/network/10−eth0.link applies to device hub0 link_config: autonegotiation is unset or enabled, the speed and duplex are not writable. hub0: Device has name_assign_type=4 Using default interface naming scheme 'v240'. hub0: Policies did not yield a name, using specified Name=hub0. ID_NET_LINK_FILE=/etc/systemd/network/10−eth0.link ID_NET_NAME=hub0 ...
```

Explicit ***Name=*** configuration wins in this case.

```
sudo SYSTEMD_LOG_LEVEL=debug udevadm test−builtin net_setup_link /sys/class/net/enp0s31f6 ... Parsed configuration file /usr/lib/systemd/network/99−default.link Parsed configuration file /etc/systemd/network/10−eth0.link Created link configuration context. ID_NET_DRIVER=e1000e Config file /usr/lib/systemd/network/99−default.link applies to device enp0s31f6 link_config: autonegotiation is unset or enabled, the speed and duplex are not writable. enp0s31f6: Device has name_assign_type=4 Using default interface naming scheme 'v240'. enp0s31f6: Policy *keep*: keeping existing userspace name enp0s31f6: Device has addr_assign_type=0 enp0s31f6: MAC on the device already matches policy *persistent* ID_NET_LINK_FILE=/usr/lib/systemd/network/99−default.link ...
```

In this case, the interface was already renamed, so the **keep** policy specified as the first option in 99−default.link means that the existing name is preserved. If **keep** was removed, or if were in boot before the renaming has happened, we might get the following instead:

```
enp0s31f6: Policy *path* yields "enp0s31f6". enp0s31f6: Device has addr_assign_type=0 enp0s31f6: MAC on the device already matches policy *persistent* ID_NET_LINK_FILE=/usr/lib/systemd/network/99−default.link ID_NET_NAME=enp0s31f6 ...
```

Please note that the details of output are subject to change.

Example 5. /etc/systemd/network/10−internet.link

This example assigns the fixed name "internet0" to the interface with the device path "pci−0000:00:1a.0−*":

```
[Match] Path=pci−0000:00:1a.0−*
[Link] Name=internet0
```

Example 6. /etc/systemd/network/25−wireless.link

Here's an overly complex example that shows the use of a large number of [Match] and [Link] settings.

```
[Match] MACAddress=12:34:56:78:9a:bc Driver=brcmsmac Path=pci−0000:02:00.0−* Type=wlan Virtualization=no Host=my−laptop Architecture=x86−64
[Link] Name=wireless0 MTUBytes=1450 BitsPerSecond=10M WakeOnLan=magic MACAddress=cb:a9:87:65:43:21
```

SEE ALSO

systemd-udevd.service(8), **udevadm**(8), **systemd.netdev**(5), **systemd.network**(5), **systemd-network-generator.service**(8)

SYSTEMD.NETDEV

NAME

systemd.netdev − Virtual Network Device configuration

SYNOPSIS

netdev.netdev

DESCRIPTION

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

The main Virtual Network Device file must have the extension .netdev; other extensions are ignored. Virtual network devices are created as soon as **systemd−networkd** is started if possible. If a netdev with the specified name already exists, **systemd−networkd** will try to update the config if the kind of the existing netdev is equivalent to the requested one, otherwise (e.g. when bridge device foo exists but bonding device with the same name is configured in a .netdev file) use the existing netdev as−is rather than replacing with the requested netdev. Note, several settings (e.g. vlan ID) cannot be changed after the netdev is created. To change such settings, it is necessary to first remove the existing netdev, and then run **networkctl reload** command or restart **systemd−networkd**. See also **networkctl**(1).

The .netdev files are read from the files located in the system network directory /usr/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 alphanumeric order, regardless of the directories in which they live. However, files with identical filenames replace each other. It is recommended that each filename is prefixed with a number smaller than "70" (e.g. 10−vlan.netdev). Otherwise, .netdev files generated by **systemd-network-generator.service**(8) may take precedence over user configured files. Files in /etc/ have the highest priority, files in /run/ take precedence over files with the same name in /usr/lib/. 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 netdev file foo.netdev, a "drop−in" directory foo.netdev.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 /usr/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 /usr/lib/. Drop−in files under any of these directories take precedence over the main netdev file wherever located. (Of course, since /run/ is temporary and /usr/lib/ is for vendors, it is unlikely drop−ins should be used in either of those places.)

SUPPORTED NETDEV KINDS

The following kinds of virtual network devices may be configured in .netdev files:

Table 1. Supported kinds of virtual network devices

[MATCH] SECTION OPTIONS

A virtual network device is only created if the [Match] section matches the current environment, or if the section is empty. The following keys are accepted:

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, the 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, the 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, the 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, the previously assigned value is cleared.

Credential=
Checks whether the specified credential was passed to the systemd−udevd.service service. See **System and Service Credentials** for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, the 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, the 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, the previously assigned value is cleared.

[NETDEV] SECTION OPTIONS

The [NetDev] section accepts the following keys:

Description=
A free−form description of the netdev.

Name=
The interface name used when creating the netdev. This setting is compulsory.

Kind=
The netdev kind. This setting is compulsory. See the "Supported netdev kinds" section for the valid keys.

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. For "tun" or "tap" devices, ***MTUBytes=*** setting is not currently supported in [NetDev] section. Please specify it in [Link] section of corresponding **systemd.network**(5) files.

MACAddress=
Specifies the MAC address to use for the device, or takes the special value "none". When "none", **systemd−networkd** does not request the MAC address for the device, and the kernel will assign a random MAC address. For "tun", "tap", or "l2tp" devices, the ***MACAddress=*** setting in the [NetDev] section is not supported and will be ignored. Please specify it in the [Link] section of the corresponding **systemd.network**(5) file. If this option is not set, "vlan" device inherits the MAC address of the master interface. For other kind of netdevs, if this option is not set, then the MAC address is generated based on the interface name and the **machine-id**(5).
Note, even if "none" is specified, **systemd−udevd** will assign the persistent MAC address for the device, as 99−default.link has ***MACAddressPolicy=persistent***. So, it is also necessary to create a custom .link file for the device, if the MAC address assignment is not desired.

[BRIDGE] SECTION OPTIONS

The [Bridge] section only applies for netdevs of kind "bridge", and accepts the following keys:

HelloTimeSec=
HelloTimeSec specifies the number of seconds between two hello packets sent out by the root bridge and the designated bridges. Hello packets are used to communicate information about the topology throughout the entire bridged local area network.

MaxAgeSec=
MaxAgeSec specifies the number of seconds of maximum message age. If the last seen (received) hello packet is more than this number of seconds old, the bridge in question will start the takeover procedure in attempt to become the Root Bridge itself.

ForwardDelaySec=
ForwardDelaySec specifies the number of seconds spent in each of the Listening and Learning states before the Forwarding state is entered.

AgeingTimeSec=
This specifies the number of seconds a MAC Address will be kept in the forwarding database after having a packet received from this MAC Address.

Priority=
The priority of the bridge. An integer between 0 and 65535. A lower value means higher priority. The bridge having the lowest priority will be elected as root bridge.

GroupForwardMask=
A 16−bit bitmask represented as an integer which allows forwarding of link local frames with 802.1D reserved addresses (01:80:C2:00:00:0X). A logical AND is performed between the specified bitmask and the exponentiation of 2^X, the lower nibble of the last octet of the MAC address. For example, a value of 8 would allow forwarding of frames addressed to 01:80:C2:00:00:03 (802.1X PAE).

DefaultPVID=
This specifies the default port VLAN ID of a newly attached bridge port. Set this to an integer in the range 1...4094 or "none" to disable the PVID.

MulticastQuerier=
Takes a boolean. This setting controls the IFLA_BR_MCAST_QUERIER option in the kernel. If enabled, the kernel will send general ICMP queries from a zero source address. This feature should allow faster convergence on startup, but it causes some multicast−aware switches to misbehave and disrupt forwarding of multicast packets. When unset, the kernel's default will be used.

MulticastSnooping=
Takes a boolean. This setting controls the IFLA_BR_MCAST_SNOOPING option in the kernel. If enabled, IGMP snooping monitors the Internet Group Management Protocol (IGMP) traffic between hosts and multicast routers. When unset, the kernel's default will be used.

VLANFiltering=
Takes a boolean. This setting controls the IFLA_BR_VLAN_FILTERING option in the kernel. If enabled, the bridge will be started in VLAN−filtering mode. When unset, the kernel's default will be used.

VLANProtocol=
Allows setting the protocol used for VLAN filtering. Takes **802.1q** or, **802.1ad**, and defaults to unset and kernel's default is used.

STP=
Takes a boolean. This enables the bridge's Spanning Tree Protocol (STP). When unset, the kernel's default will be used.

MulticastIGMPVersion=
Allows changing bridge's multicast Internet Group Management Protocol (IGMP) version. Takes an integer 2 or 3. When unset, the kernel's default will be used.

FDBMaxLearned=
Specifies the maximum number of learned Ethernet addresses for the bridge. When the limit is reached, no more addresses are learned. When unset, the kernel's default will be used. 0 disables the limit.

[VLAN] SECTION OPTIONS

The [VLAN] section only applies for netdevs of kind "vlan", and accepts the following key:

Id=
The VLAN ID to use. An integer in the range 0...4094. This setting is compulsory.

Protocol=
Allows setting the protocol used for the VLAN interface. Takes "802.1q" or, "802.1ad", and defaults to unset and kernel's default is used.

GVRP=
Takes a boolean. The Generic VLAN Registration Protocol (GVRP) is a protocol that allows automatic learning of VLANs on a network. When unset, the kernel's default will be used.

MVRP=
Takes a boolean. Multiple VLAN Registration Protocol (MVRP) formerly known as GARP VLAN Registration Protocol (GVRP) is a standards−based Layer 2 network protocol, for automatic configuration of VLAN information on switches. It was defined in the 802.1ak amendment to 802.1Q−2005. When unset, the kernel's default will be used.

LooseBinding=
Takes a boolean. The VLAN loose binding mode, in which only the operational state is passed from the parent to the associated VLANs, but the VLAN device state is not changed. When unset, the kernel's default will be used.

ReorderHeader=
Takes a boolean. When enabled, the VLAN reorder header is used and VLAN interfaces behave like physical interfaces. When unset, the kernel's default will be used.

EgressQOSMaps=, ***IngressQOSMaps=***
Defines a mapping of Linux internal packet priority (**SO_PRIORITY**) to VLAN header PCP field for outgoing and incoming frames, respectively. Takes a whitespace−separated list of integer pairs, where each integer must be in the range 1...4294967294, in the format "from"−"to", e.g., "21−7 45−5". Note that "from" must be greater than or equal to "to". When unset, the kernel's default will be used.

[MACVLAN] SECTION OPTIONS

The [MACVLAN] section only applies for netdevs of kind "macvlan", and accepts the following key:

Mode=
The MACVLAN mode to use. The supported options are "private", "vepa", "bridge", "passthru", and "source".

SourceMACAddress=
A whitespace−separated list of remote hardware addresses allowed on the MACVLAN. This option only has an effect in source mode. 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. Defaults to unset.

BroadcastMulticastQueueLength=
Specifies the length of the receive queue for broadcast/multicast packets. An unsigned integer in the range 0...4294967294. Defaults to unset.

BroadcastQueueThreshold=
Controls the threshold for broadcast queueing of the macvlan device. Takes the special value "no", or an integer in the range 0...2147483647. When "no" is specified, the broadcast queueing is disabled altogether. When an integer is specified, a multicast address will be queued as broadcast if the number of devices using the macvlan is greater than the given value. Defaults to unset, and the kernel default will be used.

[MACVTAP] SECTION OPTIONS

The [MACVTAP] section applies for netdevs of kind "macvtap" and accepts the same keys as [MACVLAN].

[IPVLAN] SECTION OPTIONS

The [IPVLAN] section only applies for netdevs of kind "ipvlan", and accepts the following key:

Mode=

The IPVLAN mode to use. The supported options are "L2","L3" and "L3S".

Flags=
The IPVLAN flags to use. The supported options are "bridge","private" and "vepa".

[IPVTAP] SECTION OPTIONS

The [IPVTAP] section only applies for netdevs of kind "ipvtap" and accepts the same keys as [IPVLAN].

[VXLAN] SECTION OPTIONS

The [VXLAN] section only applies for netdevs of kind "vxlan", and accepts the following keys:

VNI=
The VXLAN Network Identifier (or VXLAN Segment ID). Takes a number in the range 1...16777215.

Remote=
Configures destination IP address.

Local=
Configures local IP address. It must be an address on the underlying interface of the VXLAN interface, or one of the special values "ipv4_link_local", "ipv6_link_local", "dhcp4", "dhcp6", and "slaac". If one of the special values is specified, an address which matches the corresponding type on the underlying interface will be used. Defaults to unset.

Group=
Configures VXLAN multicast group IP address. All members of a VXLAN must use the same multicast group address.

TOS=
The Type Of Service byte value for a vxlan interface.

TTL=
A fixed Time To Live N on Virtual eXtensible Local Area Network packets. Takes "inherit" or a number in the range 0...255. 0 is a special value meaning inherit the inner protocol's TTL value. "inherit" means that it will inherit the outer protocol's TTL value.

MacLearning=
Takes a boolean. When true, enables dynamic MAC learning to discover remote MAC addresses.

FDBAgeingSec=
The lifetime of Forwarding Database entry learnt by the kernel, in seconds.

MaximumFDBEntries=
Configures maximum number of FDB entries.

ReduceARPProxy=
Takes a boolean. When true, bridge−connected VXLAN tunnel endpoint answers ARP requests from the local bridge on behalf of remote **Distributed Overlay Virtual Ethernet (DOVE)** clients. Defaults to false.

L2MissNotification=
Takes a boolean. When true, enables netlink LLADDR miss notifications.

L3MissNotification=
Takes a boolean. When true, enables netlink IP address miss notifications.

RouteShortCircuit=
Takes a boolean. When true, route short circuiting is turned on.

UDPChecksum=
Takes a boolean. When true, transmitting UDP checksums when doing VXLAN/IPv4 is turned on.

UDP6ZeroChecksumTx=
Takes a boolean. When true, sending zero checksums in VXLAN/IPv6 is turned on.

UDP6ZeroChecksumRx=
Takes a boolean. When true, receiving zero checksums in VXLAN/IPv6 is turned on.

RemoteChecksumTx=
Takes a boolean. When true, remote transmit checksum offload of VXLAN is turned on.

RemoteChecksumRx=
Takes a boolean. When true, remote receive checksum offload in VXLAN is turned on.

GroupPolicyExtension=
Takes a boolean. When true, it enables Group Policy VXLAN extension security label mechanism across network peers based on VXLAN. For details about the Group Policy VXLAN, see the **VXLAN Group Policy** document. Defaults to false.

GenericProtocolExtension=
Takes a boolean. When true, Generic Protocol Extension extends the existing VXLAN protocol to provide protocol typing, OAM, and versioning capabilities. For details about the VXLAN GPE Header, see the **Generic Protocol Extension for VXLAN** document. If destination port is not specified and Generic Protocol Extension is set, the default port of 4790 is used. Defaults to false.

DestinationPort=
Configures the default destination UDP port. If the destination port is not specified, the Linux kernel default will be used. Set to 4789 to get the IANA assigned value.

PortRange=
Configures the source port range for the VXLAN. The kernel assigns the source UDP port based on the flow to help the receiver to do load balancing. When this option is not set, the normal range of local UDP ports is used.

FlowLabel=
Specifies the flow label to use in outgoing packets. The valid range is 0−1048575.

IPDoNotFragment=
Allows setting the IPv4 Do not Fragment (DF) bit in outgoing packets, or to inherit its value from the IPv4 inner header. Takes a boolean value, or "inherit". Set to "inherit" if the encapsulated protocol is IPv6. When unset, the kernel's default will be used.

Independent=
Takes a boolean. When true, the vxlan interface is created without any underlying network interface. Defaults to false, which means that a .network file that requests this VXLAN interface using ***VXLAN=*** is required for the VXLAN to be created.

[GENEVE] SECTION OPTIONS

The [GENEVE] section only applies for netdevs of kind "geneve", and accepts the following keys:

Id=
Specifies the Virtual Network Identifier (VNI) to use, a number between 0 and 16777215. This field is mandatory.

Remote=
Specifies the unicast destination IP address to use in outgoing packets.

TOS=
Specifies the TOS value to use in outgoing packets. Takes a number between 1 and 255.

TTL=
Accepts the same values as in the [VXLAN] section, except that when unset or set to 0, the kernel's default will be used, meaning that packet TTL will be set from /proc/sys/net/ipv4/ip_default_ttl.

UDPChecksum=
Takes a boolean. When true, specifies that UDP checksum is calculated for transmitted packets over IPv4.

UDP6ZeroChecksumTx=
Takes a boolean. When true, skip UDP checksum calculation for transmitted packets over IPv6.

UDP6ZeroChecksumRx=
Takes a boolean. When true, allows incoming UDP packets over IPv6 with zero checksum field.

DestinationPort=
Specifies destination port. Defaults to 6081. If not set or assigned the empty string, the default port of 6081 is used.

FlowLabel=
Specifies the flow label to use in outgoing packets.

IPDoNotFragment=
Accepts the same key as in [VXLAN] section.

InheritInnerProtocol=
Takes a boolean. When true, inner Layer 3 protocol is set as Protocol Type in the GENEVE header instead of Ethernet. Defaults to false.

[BAREUDP] SECTION OPTIONS

The [BareUDP] section only applies for netdevs of kind "bareudp", and accepts the following keys:

DestinationPort=
Specifies the destination UDP port (in range 1...65535). This is mandatory.

EtherType=
Specifies the L3 protocol. Takes one of "ipv4", "ipv6", "mpls−uc" or "mpls−mc". This is mandatory.

MinSourcePort=
Specifies the lowest value of the UDP tunnel source port (in range 1...65535). Defaults to unset.

[L2TP] SECTION OPTIONS

The [L2TP] section only applies for netdevs of kind "l2tp", and accepts the following keys:

TunnelId=
Specifies the tunnel identifier. Takes an number in the range 1...4294967295. The value used must match the "PeerTunnelId=" value being used at the peer. This setting is compulsory.

PeerTunnelId=
Specifies the peer tunnel id. Takes a number in the range 1...4294967295. The value used must match the "TunnelId=" value being used at the peer. This setting is compulsory.

Remote=
Specifies the IP address of the remote peer. This setting is compulsory.

Local=
Specifies the IP address of a local interface. Takes an IP address, or the special values "auto", "static", or "dynamic". Optionally a name of a local interface can be specified after "@", e.g. "192.168.0.1@eth0" or "auto@eth0". When an address is specified, then a local or specified interface must have the address, and the remote address must be accessible through the local address. If "auto", then one of the addresses on a local or specified interface which is accessible to the remote address will be used. Similarly, if "static" or "dynamic" is set, then one of the static or dynamic addresses will be used. Defaults to "auto".

EncapsulationType=
Specifies the encapsulation type of the tunnel. Takes one of "udp" or "ip".

UDPSourcePort=
Specifies the UDP source port to be used for the tunnel. When UDP encapsulation is selected it is mandatory. Ignored when IP encapsulation is selected.

UDPDestinationPort=
Specifies destination port. When UDP encapsulation is selected it is mandatory. Ignored when IP encapsulation is selected.

UDPChecksum=
Takes a boolean. When true, specifies that UDP checksum is calculated for transmitted packets over IPv4.

UDP6ZeroChecksumTx=
Takes a boolean. When true, skip UDP checksum calculation for transmitted packets over IPv6.

UDP6ZeroChecksumRx=
Takes a boolean. When true, allows incoming UDP packets over IPv6 with zero checksum field.

[L2TPSESSION] SECTION OPTIONS

The [L2TPSession] section only applies for netdevs of kind "l2tp", and accepts the following keys:

Name=
Specifies the name of the session. This setting is compulsory.

SessionId=
Specifies the session identifier. Takes an number in the range 1...4294967295. The value used must match the "SessionId=" value being used at the peer. This setting is compulsory.

PeerSessionId=
Specifies the peer session identifier. Takes an number in the range 1...4294967295. The value used must match the "PeerSessionId=" value being used at the peer. This setting is compulsory.

Layer2SpecificHeader=
Specifies layer2specific header type of the session. One of "none" or "default". Defaults to "default".

[MACSEC] SECTION OPTIONS

The [MACsec] section only applies for network devices of kind "macsec", and accepts the following keys:

Port=
Specifies the port to be used for the MACsec transmit channel. The port is used to make secure channel identifier (SCI). Takes a value between 1 and 65535. Defaults to unset.

Encrypt=
Takes a boolean. When true, enable encryption. Defaults to unset.

[MACSECRECEIVECHANNEL] SECTION OPTIONS

The [MACsecReceiveChannel] section only applies for network devices of kind "macsec", and accepts the following keys:

Port=
Specifies the port to be used for the MACsec receive channel. The port is used to make secure channel identifier (SCI). Takes a value between 1 and 65535. This option is compulsory, and is not set by default.

MACAddress=
Specifies the MAC address to be used for the MACsec receive channel. The MAC address used to make secure channel identifier (SCI). This setting is compulsory, and is not set by default.

[MACSECTRANSMITASSOCIATION] SECTION OPTIONS

The [MACsecTransmitAssociation] section only applies for network devices of kind "macsec", and accepts the following keys:

PacketNumber=
Specifies the packet number to be used for replay protection and the construction of the initialization vector (along with the secure channel identifier [SCI]). Takes a value between 1−4,294,967,295. Defaults to unset.

KeyId=
Specifies the identification for the key. Takes a number between 0−255. This option is compulsory, and is not set by default.

Key=
Specifies the encryption key used in the transmission channel. The same key must be configured on the peer's matching receive channel. This setting is compulsory, and is not set by default. Takes a 128−bit key encoded in a hexadecimal string, for example "dffafc8d7b9a43d5b9a3dfbbf6a30c16".

KeyFile=
Takes an absolute path to a file which contains a 128−bit key encoded in a hexadecimal string, which will be used in the transmission channel. When this option is specified, ***Key=*** is ignored. Note that the file must be readable by the user "systemd−network", so it should be, e.g., owned by "root:systemd−network" with a "0640" file mode. If the path refers to an **AF_UNIX** stream socket in the file system a connection is made to it and the key read from it.

Activate=
Takes a boolean. If enabled, then the security association is activated. Defaults to unset.

UseForEncoding=
Takes a boolean. If enabled, then the security association is used for encoding. Only one [MACsecTransmitAssociation] section can enable this option. When enabled, ***Activate=yes*** is implied. Defaults to unset.

[MACSECRECEIVEASSOCIATION] SECTION OPTIONS

The [MACsecReceiveAssociation] section only applies for network devices of kind "macsec", and accepts the following keys:

Port=
Accepts the same key as in [MACsecReceiveChannel] section.

MACAddress=
Accepts the same key as in [MACsecReceiveChannel] section.

PacketNumber=
Accepts the same key as in [MACsecTransmitAssociation] section.

KeyId=
Accepts the same key as in [MACsecTransmitAssociation] section.

Key=
Accepts the same key as in [MACsecTransmitAssociation] section.

KeyFile=
Accepts the same key as in [MACsecTransmitAssociation] section.

Activate=
Accepts the same key as in [MACsecTransmitAssociation] section.

[TUNNEL] SECTION OPTIONS

The [Tunnel] section only applies for netdevs of kind "ipip", "sit", "gre", "gretap", "ip6gre", "ip6gretap", "vti", "vti6", "ip6tnl", and "erspan" and accepts the following keys:

External=
Takes a boolean value. When true, then the tunnel is externally controlled, which is also known as collect metadata mode, and most settings below like ***Local=*** or ***Remote=*** are ignored. This implies ***Independent=***. Defaults to false.

Local=
A static local address for tunneled packets. It must be an address on another interface of this host, or one of the special values "any", "ipv4_link_local", "ipv6_link_local", "dhcp4", "dhcp6", and "slaac". If one of the special values except for "any" is specified, an address which matches the corresponding type on the underlying interface will be used. Defaults to "any".

Remote=
The remote endpoint of the tunnel. Takes an IP address or the special value "any".

TOS=
The Type Of Service byte value for a tunnel interface. For details about the TOS, see the **Type of Service in the Internet Protocol Suite** document.

TTL=
A fixed Time To Live N on tunneled packets. N is a number in the range 1...255. 0 is a special value meaning that packets inherit the TTL value. The default value for IPv4 tunnels is 0 (inherit). The default value for IPv6 tunnels is 64.

DiscoverPathMTU=
Takes a boolean. When true, enables Path MTU Discovery on the tunnel. When ***IgnoreDontFragment=*** is enabled, defaults to false. Otherwise, defaults to true.

IgnoreDontFragment=
Takes a boolean. When true, enables IPv4 Don't Fragment (DF) suppression on the tunnel. Defaults to false. Note that if ***IgnoreDontFragment=*** is set to true, ***DiscoverPathMTU=*** cannot be set to true. Only applicable to GRE, GRETAP, and ERSPAN tunnels.

IPv6FlowLabel=
Configures the 20−bit flow label (see **RFC 6437**) field in the IPv6 header (see **RFC 2460**), which is used by a node to label packets of a flow. It is only used for IPv6 tunnels. A flow label of zero is used to indicate packets that have not been labeled. It can be configured to a value in the range 0...0xFFFFF, or be set to "inherit", in which case the original flowlabel is used.

CopyDSCP=
Takes a boolean. When true, the Differentiated Service Code Point (DSCP) field will be copied to the inner header from outer header during the decapsulation of an IPv6 tunnel packet. DSCP is a field in an IP packet that enables different levels of service to be assigned to network traffic. Defaults to "no".

EncapsulationLimit=
The Tunnel Encapsulation Limit option specifies how many additional levels of encapsulation are permitted to be prepended to the packet. For example, a Tunnel Encapsulation Limit option containing a limit value of zero means that a packet carrying that option may not enter another tunnel before exiting the current tunnel. (see **RFC 2473**). The valid range is 0...255 and "none". Defaults to 4.

Key=
The ***Key=*** parameter specifies the same key to use in both directions (***InputKey=*** and ***OutputKey=***). The ***Key=*** is either a number or an IPv4 address−like dotted quad. It is used as mark−configured SAD/SPD entry as part of the lookup key (both in data and control path) in IP XFRM (framework used to implement IPsec protocol). See **ip−xfrm — transform configuration** for details. It is only used for VTI/VTI6, GRE, GRETAP, and ERSPAN tunnels.

InputKey=
The ***InputKey=*** parameter specifies the key to use for input. The format is same as ***Key=***. It is only used for VTI/VTI6, GRE, GRETAP, and ERSPAN tunnels.

OutputKey=
The ***OutputKey=*** parameter specifies the key to use for output. The format is same as ***Key=***. It is only used for VTI/VTI6, GRE, GRETAP, and ERSPAN tunnels.

Mode=
An "ip6tnl" tunnel can be in one of three modes "ip6ip6" for IPv6 over IPv6, "ipip6" for IPv4 over IPv6 or "any" for either.

Independent=
Takes a boolean. When false (the default), the tunnel is always created over some network device, and a .network file that requests this tunnel using ***Tunnel=*** is required for the tunnel to be created. When true, the tunnel is created independently of any network as "tunnel@NONE".

AssignToLoopback=
Takes a boolean. If set to "yes", the loopback interface "lo" is used as the underlying device of the tunnel interface. Defaults to "no".

AllowLocalRemote=
Takes a boolean. When true allows tunnel traffic on ***ip6tnl*** devices where the remote endpoint is a local host address. When unset, the kernel's default will be used.

FooOverUDP=
Takes a boolean. Specifies whether ***FooOverUDP=*** tunnel is to be configured. Defaults to false. This takes effects only for IPIP, SIT, GRE, and GRETAP tunnels. For more detail information see **Foo over UDP**

FOUDestinationPort=
This setting specifies the UDP destination port for encapsulation. This field is mandatory when ***FooOverUDP=yes***, and is not set by default.

FOUSourcePort=
This setting specifies the UDP source port for encapsulation. Defaults to **0** — that is, the source port for packets is left to the network stack to decide.

Encapsulation=
Accepts the same key as in the [FooOverUDP] section.

IPv6RapidDeploymentPrefix=
Reconfigure the tunnel for **IPv6 Rapid Deployment**, also known as 6rd. The value is an ISP−specific IPv6 prefix with a non−zero length. Only applicable to SIT tunnels.

ISATAP=
Takes a boolean. If set, configures the tunnel as Intra−Site Automatic Tunnel Addressing Protocol (ISATAP) tunnel. Only applicable to SIT tunnels. When unset, the kernel's default will be used.

SerializeTunneledPackets=
Takes a boolean. If set to yes, then packets are serialized. Only applies for GRE, GRETAP, and ERSPAN tunnels. When unset, the kernel's default will be used.

ERSPANVersion=
Specifies the ERSPAN version number. Takes 0 for version 0 (a.k.a. type I), 1 for version 1 (a.k.a. type II), or 2 for version 2 (a.k.a. type III). Defaults to 1.

ERSPANIndex=
Specifies the ERSPAN v1 index field for the interface. Takes an integer in the range 0...1048575, which is associated with the ERSPAN traffic's source port and direction. Only used when ***ERSPANVersion=1***. Defaults to 0.

ERSPANDirection=
Specifies the ERSPAN v2 mirrored traffic's direction. Takes "ingress" or "egress". Only used when ***ERSPANVersion=2***. Defaults to "ingress".

ERSPANHardwareId=
Specifies an unique identifier of the ERSPAN v2 engine. Takes an integer in the range 0...63. Only used when ***ERSPANVersion=2***. Defaults to 0.

[FOOOVERUDP] SECTION OPTIONS

The [FooOverUDP] section only applies for netdevs of kind "fou" and accepts the following keys:

Encapsulation=
Specifies the encapsulation mechanism used to store networking packets of various protocols inside the UDP packets. Supports the following values: "FooOverUDP" provides the simplest no−frills model of UDP encapsulation, it simply encapsulates packets directly in the UDP payload. "GenericUDPEncapsulation" is a generic and extensible encapsulation, it allows encapsulation of packets for any IP protocol and optional data as part of the encapsulation. For more detailed information see **Generic UDP Encapsulation**. Defaults to "FooOverUDP".

Port=
Specifies the port number where the encapsulated packets will arrive. Those packets will be removed and manually fed back into the network stack with the encapsulation removed to be sent to the real destination. This option is mandatory.

PeerPort=
Specifies the peer port number. Defaults to unset. Note that when peer port is set "Peer=" address is mandatory.

Protocol=
The ***Protocol=*** specifies the protocol number of the packets arriving at the UDP port. When ***Encapsulation=FooOverUDP***, this field is mandatory and is not set by default. Takes an IP protocol name such as "gre" or "ipip", or an integer within the range 1…255. When ***Encapsulation=GenericUDPEncapsulation***, this must not be specified.

Peer=
Configures peer IP address. Note that when peer address is set "PeerPort=" is mandatory.

Local=
Configures local IP address.

[PEER] SECTION OPTIONS

The [Peer] section only applies for netdevs of kind "veth" and accepts the following keys:

Name=
The interface name used when creating the netdev. This setting is compulsory.

MACAddress=
The peer MACAddress, if not set, it is generated in the same way as the MAC address of the main interface.

[VXCAN] SECTION OPTIONS

The [VXCAN] section only applies for netdevs of kind "vxcan" and accepts the following key:

Peer=
The peer interface name used when creating the netdev. This setting is compulsory.

[TUN] SECTION OPTIONS

The [Tun] section only applies for netdevs of kind "tun", and accepts the following keys:

MultiQueue=
Takes a boolean. Configures whether to use multiple file descriptors (queues) to parallelize packets sending and receiving. Defaults to "no".

PacketInfo=
Takes a boolean. Configures whether packets should be prepended with four extra bytes (two flag bytes and two protocol bytes). If disabled, it indicates that the packets will be pure IP packets. Defaults to "no".

VNetHeader=
Takes a boolean. Configures IFF_VNET_HDR flag for a tun or tap device. It allows sending and receiving larger Generic Segmentation Offload (GSO) packets. This may increase throughput significantly. Defaults to "no".

User=
User to grant access to the /dev/net/tun device.

Group=
Group to grant access to the /dev/net/tun device.

KeepCarrier=
Takes a boolean. If enabled, to make the interface maintain its carrier status, the file descriptor of the interface is kept open. This may be useful to keep the interface in running state, for example while the backing process is temporarily shutdown. Defaults to "no".

[TAP] SECTION OPTIONS

The [Tap] section only applies for netdevs of kind "tap", and accepts the same keys as the [Tun] section.

[WIREGUARD] SECTION OPTIONS

The [WireGuard] section accepts the following keys:

PrivateKey=
The Base64 encoded private key for the interface. It can be generated using the **wg genkey** command (see **wg**(8)). Specially, if the specified key is prefixed with "@", it is interpreted as the name of the credential from which the actual key shall be read. **systemd-networkd.service**(8) automatically imports credentials matching "network.wireguard.*". For more details on credentials, refer to **systemd.exec**(5). A private key is mandatory to use WireGuard. If not set, the credential "network.wireguard.private.***netdev***" is used if exists. I.e. for 50−foobar.netdev, "network.wireguard.private.50−foobar" is tried.
Note that because this information is secret, it is strongly recommended to use an (encrypted) credential. Alternatively, you may want to set the permissions of the .netdev file to be owned by "root:systemd−network" with a "0640" file mode.

PrivateKeyFile=
Takes an absolute path to a file which contains the Base64 encoded private key for the interface. When this option is specified, then ***PrivateKey=*** is ignored. Note that the file must be readable by the user "systemd−network", so it should be, e.g., owned by "root:systemd−network" with a "0640" file mode. If the path refers to an **AF_UNIX** stream socket in the file system a connection is made to it and the key read from it.

ListenPort=
Sets UDP port for listening. Takes either value between 1 and 65535 or "auto". If "auto" is specified, the port is automatically generated based on interface name. Defaults to "auto".

FirewallMark=
Sets a firewall mark on outgoing WireGuard packets from this interface. Takes a number between 1 and 4294967295.

RouteTable=
The table identifier for the routes to the addresses specified in the ***AllowedIPs=***. Takes a negative boolean value, one of the predefined names "default", "main", and "local", names defined in ***RouteTable=*** in **networkd.conf**(5), or a number in the range 1…4294967295. When "off" the routes to the addresses specified in the ***AllowedIPs=*** setting will not be configured. Defaults to false. This setting will be ignored when the same setting is specified in the [WireGuardPeer] section.

RouteMetric=
The priority of the routes to the addresses specified in the ***AllowedIPs=***. Takes an integer in the range 0…4294967295. Defaults to 0 for IPv4 addresses, and 1024 for IPv6 addresses. This setting will be ignored when the same setting is specified in the [WireGuardPeer] section.

[WIREGUARDPEER] SECTION OPTIONS

The [WireGuardPeer] section accepts the following keys:

PublicKey=
Sets a Base64 encoded public key calculated by **wg pubkey** (see **wg**(8)) from a private key, and usually transmitted out of band to the author of the configuration file. This option honors the "@" prefix in the same way as the **PrivateKey=** setting of the **[WireGuard]** section. This option is mandatory for this section.

PublicKeyFile=
Takes an absolute path to a file which contains the Base64 encoded public key for the peer. When this option is specified, then ***PublicKey=*** will be ignored. Note that the file must be readable by the user "systemd−network", so it should be, e.g., owned by "root:systemd−network" with a "0640" file mode. If the path refers to an **AF_UNIX** stream socket in the file system a connection is made to it and the key read from it.

PresharedKey=
Optional preshared key for the interface. It can be generated by the **wg genpsk** command. This option adds an additional layer of symmetric−key cryptography to be mixed into the already existing public−key cryptography, for post−quantum resistance. This option honors the "@" prefix in the same way as the **PrivateKey=** setting of the **[WireGuard]** section.
Note that because this information is secret, it is strongly recommended to use an (encrypted) credential. Alternatively, you may want to set the permissions of the .netdev file to be owned by "root:systemd−network" with a "0640" file mode.

PresharedKeyFile=
Takes an absolute path to a file which contains the Base64 encoded preshared key for the peer. When this option is specified, then ***PresharedKey=*** is ignored. Note that the file must be readable by the user "systemd−network", so it should be, e.g., owned by "root:systemd−network" with a "0640" file mode. If the path refers to an **AF_UNIX** stream socket in the file system a connection is made to it and the key read from it.

AllowedIPs=
Sets a comma−separated list of IP (v4 or v6) addresses with CIDR masks from which this peer is allowed to send incoming traffic and to which outgoing traffic for this peer is directed. This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.
The catch−all 0.0.0.0/0 may be specified for matching all IPv4 addresses, and ::/0 may be specified for matching all IPv6 addresses.
Note that this only affects ***routing inside the network interface itself***, i.e. the packets that pass through the tunnel itself. To cause packets to be sent via the tunnel in the first place, an appropriate route needs to be added as well — either in the "[Routes]" section on the ".network" matching the wireguard interface, or externally to **systemd−networkd**.

Endpoint=
Sets an endpoint IP address or hostname, followed by a colon, and then a port number. IPv6 address must be in the square brackets. For example, "111.222.333.444:51820" for IPv4 and "[1111:2222::3333]:51820" for IPv6 address. This endpoint will be updated automatically once to the most recent source IP address and port of correctly authenticated packets from the peer at configuration time.
This option honors the "@" prefix in the same way as the **PrivateKey=** setting of the **[WireGuard]** section.

PersistentKeepalive=
Sets a seconds interval, between 1 and 65535 inclusive, of how often to send an authenticated empty packet to the peer for the purpose of keeping a stateful firewall or NAT mapping valid persistently. For example, if the interface very rarely sends traffic, but it might at anytime receive traffic from a peer, and it is behind NAT, the interface might benefit from having a persistent keepalive interval of 25 seconds. If set to 0 or "off", this option is disabled. By default or when unspecified, this option is off. Most users will not need this.

RouteTable=
The table identifier for the routes to the addresses specified in the ***AllowedIPs=***. Takes a negative boolean value, one of the predefined names "default", "main", and "local", names defined in ***RouteTable=*** in **networkd.conf**(5), or a number in the range 1…4294967295. Defaults to unset, and the value specified in the same setting in the [WireGuard] section will be used.

RouteMetric=
The priority of the routes to the addresses specified in the ***AllowedIPs=***. Takes an integer in the range 0…4294967295. Defaults to unset, and the value specified in the same setting in the [WireGuard] section will be used.

[BOND] SECTION OPTIONS

The [Bond] section accepts the following key:

Mode=
Specifies one of the bonding policies. The default is "balance−rr" (round robin). Possible values are "balance−rr", "active−backup", "balance−xor", "broadcast", "802.3ad", "balance−tlb", and "balance−alb".

TransmitHashPolicy=
Selects the transmit hash policy to use for slave selection in balance−xor, 802.3ad, and tlb modes. Possible values are "layer2", "layer3+4", "layer2+3", "encap2+3", and "encap3+4".

LACPTransmitRate=
Specifies the rate with which link partner transmits Link Aggregation Control Protocol Data Unit packets in

802.3ad mode. Possible values are "slow", which requests partner to transmit LACPDUs every 30 seconds, and "fast", which requests partner to transmit LACPDUs every second. The default value is "slow".

MIIMonitorSec=
Specifies the frequency that Media Independent Interface link monitoring will occur. A value of zero disables MII link monitoring. This value is rounded down to the nearest millisecond. The default value is 0.

PeerNotifyDelaySec=
Specifies the number of seconds the delay between each peer notification (gratuitous ARP and unsolicited IPv6 Neighbor Advertisement) when they are issued after a failover event. This delay should be a multiple of the MII link monitor interval (miimon). The valid range is 0...300s. The default value is 0, which means to match the value of the ***MIIMonitorSec=***.

UpDelaySec=
Specifies the delay before a link is enabled after a link up status has been detected. This value is rounded down to a multiple of ***MIIMonitorSec=***. The default value is 0.

DownDelaySec=
Specifies the delay before a link is disabled after a link down status has been detected. This value is rounded down to a multiple of ***MIIMonitorSec=***. The default value is 0.

LearnPacketIntervalSec=
Specifies the number of seconds between instances where the bonding driver sends learning packets to each slave peer switch. The valid range is 1...0x7fffffff; the default value is 1. This option has an effect only for the balance−tlb and balance−alb modes.

AdSelect=
Specifies the 802.3ad aggregation selection logic to use. Possible values are "stable", "bandwidth" and "count".

AdActorSystemPriority=
Specifies the 802.3ad actor system priority. Takes a number in the range 1...65535.

AdUserPortKey=
Specifies the 802.3ad user defined portion of the port key. Takes a number in the range 0...1023.

AdActorSystem=
Specifies the 802.3ad system MAC address. This cannot be a null or multicast address.

FailOverMACPolicy=
Specifies whether the active−backup mode should set all slaves to the same MAC address at the time of enslavement or, when enabled, to perform special handling of the bond's MAC address in accordance with the selected policy. The default policy is none. Possible values are "none", "active" and "follow".

ARPValidate=
Specifies whether or not ARP probes and replies should be validated in any mode that supports ARP monitoring, or whether non−ARP traffic should be filtered (disregarded) for link monitoring purposes. Possible values are "none", "active", "backup" and "all".

ARPIntervalSec=
Specifies the ARP link monitoring frequency. A value of 0 disables ARP monitoring. The default value is 0, and the default unit seconds.

ARPIPTargets=
Specifies the IP addresses to use as ARP monitoring peers when ***ARPIntervalSec=*** is greater than 0. These are the targets of the ARP request sent to determine the health of the link to the targets. Specify these values in IPv4 dotted decimal format. At least one IP address must be given for ARP monitoring to function. The maximum number of targets that can be specified is 16. The default value is no IP addresses.

ARPAllTargets=
Specifies the quantity of ***ARPIPTargets=*** that must be reachable in order for the ARP monitor to consider a slave as being up. This option affects only active−backup mode for slaves with ARPValidate enabled. Possible values are "any" and "all".

PrimaryReselectPolicy=
Specifies the reselection policy for the primary slave. This affects how the primary slave is chosen to become the active slave when failure of the active slave or recovery of the primary slave occurs. This option is designed to prevent flip−flopping between the primary slave and other slaves. Possible values are "always", "better" and "failure".

ResendIGMP=
Specifies the number of IGMP membership reports to be issued after a failover event. One membership report is issued immediately after the failover, subsequent packets are sent in each 200ms interval. The valid range is 0...255. Defaults to 1. A value of 0 prevents the IGMP membership report from being issued in response to the failover event.

PacketsPerSlave=
Specify the number of packets to transmit through a slave before moving to the next one. When set to 0, then a slave is chosen at random. The valid range is 0...65535. Defaults to 1. This option only has effect when in balance−rr mode.

GratuitousARP=
Specify the number of peer notifications (gratuitous ARPs and unsolicited IPv6 Neighbor Advertisements) to be issued after a failover event. As soon as the link is up on the new slave, a peer notification is sent on the bonding device and each VLAN sub−device. This is repeated at each link monitor interval (ARPIntervalSec or MIIMonitorSec, whichever is active) if the number is greater than 1. The valid range is 0...255. The default value is 1. These options affect only the active−backup mode.

AllSlavesActive=
Takes a boolean. Specifies that duplicate frames (received on inactive ports) should be dropped when false, or delivered when true. Normally, bonding will drop duplicate frames (received on inactive ports), which is desirable for most users. But there are some times it is nice to allow duplicate frames to be delivered. The default value is false (drop duplicate frames received on inactive ports).

DynamicTransmitLoadBalancing=
Takes a boolean. Specifies if dynamic shuffling of flows is enabled. Applies only for balance−tlb mode. Defaults to unset.

MinLinks=
Specifies the minimum number of links that must be active before asserting carrier. The default value is 0.

ARPMissedMax=
Specify the maximum number of arp interval monitor cycle for missed ARP replies. If this number is exceeded, link is reported as down. Defaults to unset.

For more detail information see **Linux Ethernet Bonding Driver HOWTO**

[XFRM] SECTION OPTIONS

The [Xfrm] section accepts the following keys:

InterfaceId=
Sets the ID/key of the xfrm interface which needs to be associated with a SA/policy. Can be decimal or hexadecimal, valid range is 1−0xffffffff. This is mandatory.

Independent=
Takes a boolean. If false (the default), the xfrm interface must have an underlying device which can be used for hardware offloading.

For more detail information see **Virtual XFRM Interfaces**.

[VRF] SECTION OPTIONS

The [VRF] section only applies for netdevs of kind "vrf" and accepts the following key:

Table=
The numeric routing table identifier. This setting is compulsory.

[BATMANADVANCED] SECTION OPTIONS

The [BatmanAdvanced] section only applies for netdevs of kind "batadv" and accepts the following keys:

GatewayMode=
Takes one of "off", "server", or "client". A batman−adv node can either run in server mode (sharing its internet connection with the mesh) or in client mode (searching for the most suitable internet connection in the mesh) or having the gateway support turned off entirely (which is the default setting).

Aggregation=
Takes a boolean value. Enables or disables aggregation of originator messages. Defaults to true.

BridgeLoopAvoidance=
Takes a boolean value. Enables or disables avoidance of loops on bridges. Defaults to true.

DistributedArpTable=
Takes a boolean value. Enables or disables the distributed ARP table. Defaults to true.

Fragmentation=
Takes a boolean value. Enables or disables fragmentation. Defaults to true.

HopPenalty=
The hop penalty setting allows one to modify **batctl**(8) preference for multihop routes vs. short routes. This integer value is applied to the TQ (Transmit Quality) of each forwarded OGM (Originator Message), thereby propagating the cost of an extra hop (the packet has to be received and retransmitted which costs airtime). A higher hop penalty will make it more unlikely that other nodes will choose this node as intermediate hop towards any given destination. The default hop penalty of '15' is a reasonable value for most setups and probably does not need to be changed. However, mobile nodes could choose a value of 255 (maximum value) to avoid being chosen as a router by other nodes. The minimum value is 0.

OriginatorIntervalSec=
The value specifies the interval in seconds, unless another time unit is specified in which batman−adv floods the network with its protocol information. See **systemd.time**(7) for more information.

GatewayBandwidthDown=
If the node is a server, this parameter is used to inform other nodes in the network about this node's internet connection download bandwidth in bits per second. Just enter any number suffixed with K, M, G or T (base 1000) and the batman−adv module will propagate the entered value in the mesh.

GatewayBandwidthUp=
If the node is a server, this parameter is used to inform other nodes in the network about this node's internet connection upload bandwidth in bits per second. Just enter any number suffixed with K, M, G or T (base 1000) and the batman−adv module will propagate the entered value in the mesh.

RoutingAlgorithm=
This can be either "batman−v" or "batman−iv" and describes which routing_algo of **batctl**(8) to use. The algorithm cannot be changed after interface creation. Defaults to "batman−v".

[IPOIB] SECTION OPTIONS

The [IPoIB] section only applies for netdevs of kind "ipoib" and accepts the following keys:

PartitionKey=
Takes an integer in the range 1...0xffff, except for 0x8000. Defaults to unset, and the kernel's default is used.

Mode=
Takes one of the special values "datagram" or "connected". Defaults to unset, and the kernel's default is used. When "datagram", the Infiniband unreliable datagram (UD) transport is used, and so the interface MTU is equal to the IB L2 MTU minus the IPoIB encapsulation header (4 bytes). For example, in a typical IB fabric with a 2K MTU, the IPoIB MTU will be 2048 − 4 = 2044 bytes.
When "connected", the Infiniband reliable connected (RC) transport is used. Connected mode takes advantage of the connected nature of the IB transport and allows an MTU up to the maximal IP packet size of 64K, which reduces the number of IP packets needed for handling large UDP datagrams, TCP segments, etc and increases the performance for large messages.

IgnoreUserspaceMulticastGroup=
Takes an boolean value. When true, the kernel ignores multicast groups handled by userspace. Defaults to unset, and the kernel's default is used.

[WLAN] SECTION OPTIONS

The [WLAN] section only applies to WLAN interfaces, and accepts the following keys:

PhysicalDevice=
Specifies the name or index of the physical WLAN device (e.g. "0" or "phy0"). The list of the physical WLAN devices that exist on the host can be obtained by **iw phy** command. This option is mandatory.

Type=
Specifies the type of the interface. Takes one of the "ad−hoc", "station", "ap", "ap−vlan", "wds", "monitor", "mesh−point", "p2p−client", "p2p−go", "p2p−device", "ocb", and "nan". This option is mandatory.

WDS=
Enables the Wireless Distribution System (WDS) mode on the interface. The mode is also known as the "4

address mode". Takes a boolean value. Defaults to unset, and the kernel's default will be used.

EXAMPLES

Example 1. /etc/systemd/network/25−bridge.netdev

```
[NetDev] Name=bridge0 Kind=bridge
```

Example 2. /etc/systemd/network/25−vlan1.netdev

```
[Match] Virtualization=no
[NetDev] Name=vlan1 Kind=vlan
[VLAN] Id=1
```

Example 3. /etc/systemd/network/25−ipip.netdev

```
[NetDev] Name=ipip−tun Kind=ipip MTUBytes=1480
[Tunnel] Local=192.168.223.238 Remote=192.169.224.239 TTL=64
```

Example 4. /etc/systemd/network/1−fou−tunnel.netdev

```
[NetDev] Name=fou−tun Kind=fou
[FooOverUDP] Port=5555 Protocol=4
```

Example 5. /etc/systemd/network/25−fou−ipip.netdev

```
[NetDev] Name=ipip−tun Kind=ipip
[Tunnel] Independent=yes Local=10.65.208.212 Remote=10.65.208.211 FooOverUDP=yes
FOUDestinationPort=5555
```

Example 6. /etc/systemd/network/25−tap.netdev

```
[NetDev] Name=tap−test Kind=tap
[Tap] MultiQueue=yes PacketInfo=yes
```

Example 7. /etc/systemd/network/25−sit.netdev

```
[NetDev] Name=sit−tun Kind=sit MTUBytes=1480
[Tunnel] Local=10.65.223.238 Remote=10.65.223.239
```

Example 8. /etc/systemd/network/25−6rd.netdev

```
[NetDev] Name=6rd−tun Kind=sit MTUBytes=1480
[Tunnel] Local=10.65.223.238 IPv6RapidDeploymentPrefix=2602::/24
```

Example 9. /etc/systemd/network/25−gre.netdev

```
[NetDev] Name=gre−tun Kind=gre MTUBytes=1480
[Tunnel] Local=10.65.223.238 Remote=10.65.223.239
```

Example 10. /etc/systemd/network/25−ip6gre.netdev

```
[NetDev] Name=ip6gre−tun Kind=ip6gre
[Tunnel] Key=123
```

Example 11. /etc/systemd/network/25−vti.netdev

```
[NetDev] Name=vti−tun Kind=vti MTUBytes=1480
[Tunnel] Local=10.65.223.238 Remote=10.65.223.239
```

Example 12. /etc/systemd/network/25−veth.netdev

```
[NetDev] Name=veth−test Kind=veth
[Peer] Name=veth−peer
```

Example 13. /etc/systemd/network/25−bond.netdev

```
[NetDev] Name=bond1 Kind=bond
[Bond] Mode=802.3ad TransmitHashPolicy=layer3+4 MIIMonitorSec=1s LACPTransmitRate=fast
```

Example 14. /etc/systemd/network/25−dummy.netdev

```
[NetDev] Name=dummy−test Kind=dummy MACAddress=12:34:56:78:9a:bc
```

Example 15. /etc/systemd/network/25−vrf.netdev

Create a VRF interface with table 42.

```
[NetDev] Name=vrf−test Kind=vrf
[VRF] Table=42
```

Example 16. /etc/systemd/network/25−macvtap.netdev

Create a MacVTap device.

```
[NetDev] Name=macvtap−test Kind=macvtap
```

Example 17. /etc/systemd/network/25−wireguard.netdev

```
[NetDev] Name=wg0 Kind=wireguard
[WireGuard] PrivateKey=EEGlnEPYJV//kbvvIqxKkQwOiS+UENyPncC4bF46ong= ListenPort=51820
[WireGuardPeer] PublicKey=RDf+LSpeEre7YEIKaxg+wbpsNV7du+ktR99uBEtIiCA=
AllowedIPs=fd31:bf08:57cb::/48,192.168.26.0/24 Endpoint=wireguard.example.com:51820
```

Example 18. /etc/systemd/network/27−xfrm.netdev

```
[NetDev] Name=xfrm0 Kind=xfrm
[Xfrm] Independent=yes
```

SEE ALSO

systemd(1), **systemd-networkd.service**(8), **systemd.link**(5), **systemd.network**(5), **systemd-network-generator.service**(8)

SYSTEMD.NETWORK

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.

Note that not all settings and configurations can be made with .network files, and that it may be necessary to use **systemd.link**(5)) or **systemd.netdev**(5)) files in conjuction with .network files when working with physical and virtual network devices respectively.

The .network files are read from the files located in the system network directories /usr/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 alphanumeric order, regardless of the directories in which they live. However, files with identical filenames replace each other. It is recommended that each filename is prefixed with a number smaller than "70" (e.g. 10−eth0.network). Otherwise, the default .network files or those generated by **systemd-network-generator.service**(8) may take precedence over user configured files. 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 /usr/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 /usr/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 interface; and a [Network] section specifying how the interface should be configured. The first (in alphanumeric order) of the network files that matches a given interface is applied, all later files are ignored, even if they match as well.

Note that any network interfaces that have the ***ID_NET_MANAGED_BY=*** udev property set will never be matched by any .network files – unless the property's value is the string "io.systemd.Network" – even if the [Match] section would otherwise match. This may be used to exclude specific network interfaces from **systemd-networkd**(8)'s management, while keeping the [Match] section generic. The ***ID_NET_MANAGED_BY=*** property thus declares intended ***ownership*** of the device, and permits ensuring that concurrent network management implementations do not compete for management of specific devices.

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. The acceptable formats are:

colon−delimited hexadecimal
: Each field must be one byte. E.g. "12:34:56:78:90:ab" or "AA:BB:CC:DD:EE:FF".

hyphen−delimited hexadecimal
: Each field must be one byte. E.g. "12−34−56−78−90−ab" or "AA−BB−CC−DD−EE−FF".

dot−delimited hexadecimal
: Each field must be two bytes. E.g. "1234.5678.90ab" or "AABB.CCDD.EEFF".

IPv4 address format
: E.g. "127.0.0.1" or "192.168.0.1".

IPv6 address format
: E.g. "2001:0db8:85a3::8a2e:0370:7334" or "::1".

The total length of each MAC address must be 4 (for IPv4 tunnel), 6 (for Ethernet), 16 (for IPv6 tunnel), or 20 (for InfiniBand). 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. Defaults to unset.

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, or IPv4 or IPv6 address format. 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. Defaults to unset.

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.

Kind=

A whitespace−separated list of shell−style globs matching the device kind, as exposed by **networkctl status** ***INTERFACE*** or **ip −d link show** ***INTERFACE***. If the list is prefixed with a "!", the test is inverted. Some valid values are "bond", "bridge", "gre", "tun", "veth". Valid kinds are given by netlink's "IFLA_INFO_KIND" attribute, 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, the 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, the 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, the 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, the previously assigned value is cleared.

Credential=
Checks whether the specified credential was passed to the systemd−udevd.service service. See **System and Service Credentials** for details. When prefixed with an exclamation mark ("!"), the result is negated. If an empty string is assigned, the 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, the 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, the 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 IPv4 ARP (low−level Address Resolution Protocol) and IPv6 NDP (Neighbor Discovery Protocol) for this interface are 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.
If this is set to false for the underlying link of a "passthru" mode MACVLAN/MACVTAP, the virtual interface will be created with the "nopromisc" flag set.

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...2147483647. Defaults to unset.

RequiredForOnline=
Takes a boolean, a minimum operational state (e.g., "carrier"), or a range of operational state separated with a colon (e.g., "degraded:routable"). 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, **systemd−networkd−wait−online** deems that the interface is online when the operational state is in the specified range.
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".
The boolean value "yes" is translated as follows;

CAN devices
"carrier",

Master devices, e.g. bond or bridge
"degraded−carrier" with ***RequiredFamilyForOnline=any***,

Bonding port interfaces
"enslaved",

Other interfaces
"degraded".

This setting can be overridden by the command line option for **systemd−networkd−wait−online**. See **systemd-networkd-wait-online.service**(8) for more details.

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 "no". Note that this option has no effect if "RequiredForOnline=no".

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

SR−IOV provides the ability to partition a single physical PCI resource into virtual PCI functions which can then be e.g. injected into a VM. In the case of network VFs, SR−IOV reduces latency and CPU utilisation for north−south network traffic (that is, traffic with endpoints outside the host machine), by allowing traffic to bypass the host machine’s network stack.
The presence of an [SR−IOV] section in a .link file will cause the creation and configuration of the specified virtual function. Within a .network file, the specified virtual function will be configured, but must already exist. Specify several [SR−IOV] sections to configure several SR−IOVs.
The [SR−IOV] section accepts the following keys.

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

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

QualityOfService=
Specifies quality of service of the virtual function. Takes an 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 one 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 one 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 Advertisements, if reception is enabled, regardless of this parameter. By explicitly enabling DHCPv6 support here, the DHCPv6 client will be started in the mode specified by the ***WithoutRA=*** setting in the [DHCPv6] section, 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 is not 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.
Even if this is enabled, the DHCP server will not be started automatically and wait for the persistent storage being ready to load/save leases in the storage, unless ***RelayTarget=*** or ***PersistLeases=no*** are specified in the [DHCPServer] section. It will be started after systemd−networkd−persistent−storage.service is started, which calls **networkctl persistent−storage yes**. See **networkctl**(1) for more details.

LinkLocalAddressing=
Enables link−local address autoconfiguration. Accepts a boolean, **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 ***KeepMaster=*** or ***Bridge=*** is set or when the specified ***MACVLAN=***/***MACVTAP=*** has ***Mode=passthru***, or **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.

IPv4LLStartAddress=
Specifies the first IPv4 link−local address to try. Takes an IPv4 address for example 169.254.1.2, from the link−local address range: 169.254.0.0/16 except for 169.254.0.0/24 and 169.254.255.0/24. This setting may be useful if the device should always have the same address as long as there is no address conflict. When unset, a random address will be automatically selected. Defaults to unset.

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 IPv4 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
```

If you'd like to create an IPv6 default route bound to the interface, please use the following:

```
[Route] Gateway=:: Table=1234
```

LLMNR=
Takes a boolean or "resolve". When true, enables **Link−Local Multicast Name Resolution** 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** 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** 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 unset, and the global setting will be used. This setting is read by **systemd-resolved.service**(8).

DNSSEC=
Takes a boolean or "allow−downgrade". When true, enables **DNSSEC** 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 unset, and the global setting will be used. 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**. 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.
If an IPv4 link−local address (169.254.0.0/16) is specified, IPv4 Address Conflict Detection (**RFC 5227**) is enabled for the address. To assign an IPv4 link−local address without IPv4 Address Conflict Detection, please use [Address] section to configure the address and disable ***DuplicateAddressDetection=***.

```
[Address] Address=169.254.10.1/24 DuplicateAddressDetection=none
```

If an empty string is specified, then the all previous assignments in both [Network] and [Address] sections are cleared.

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).

UseDomains=
Specifies the protocol−independent default value for the same settings in [IPv6AcceptRA], [DHCPv4], and [DHCPv6] sections below. Takes a boolean, or the special value **route**. See also the same setting in [DHCPv4] below. Defaults to unset.

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).

IPv4Forwarding=
Configures IPv4 packet forwarding for the interface. Takes a boolean value. This controls the net.ipv4.conf.***INTERFACE***.forwarding sysctl option of the network interface. See **IP Sysctl** for more details about the sysctl option. Defaults to true if ***IPMasquerade=*** is enabled for IPv4, otherwise the value specified to the same setting in **networkd.conf**(5) will be used. If none of them are specified, the sysctl option will not be changed.
To control the global setting, use the same setting in **networkd.conf**(5).

IPv6Forwarding=
Configures interface−specific host/router behaviour. Takes a boolean value. This controls the net.ipv6.conf.***INTERFACE***.forwarding sysctl option of the network interface. See **IP Sysctl** for more details about the sysctl option. Defaults to true if ***IPMasquerade=*** is enabled for IPv6 or ***IPv6SendRA=*** is enabled, otherwise the value specified to the same setting in **networkd.conf**(5) will be used. If none of them are specified, the sysctl option will not be changed.
To control the global setting, use the same setting in **networkd.conf**(5).
Note, unlike ***IPv4Forwarding=***, enabling per−interface ***IPv6Forwarding=*** on two or more interfaces ***DOES NOT*** make IPv6 packets forwarded within the interfaces. This setting just controls the per−interface sysctl value, and the sysctl value is not directly correlated to whether packets are forwarded. To ensure IPv6 packets forwarded, the global setting in **networkd.conf**(5) needs to be enabled.

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. Typically, this should be enabled on the downstream interface of routers. Takes one of "ipv4", "ipv6", "both", or "no". Defaults to "no". Note that any positive boolean values such as "yes" or "true" are now deprecated. Please use one of the values above. Specifying "ipv4" or "both" implies ***IPv4Forwarding=*** settings in both .network file for this interface and the global **networkd.conf**(5) unless they are explicitly specified. Similarly for ***IPv6Forwarding=*** when "ipv6" or "both" is specified. See ***IPv4Forwarding=***/***IPv6Forwarding=*** in the above for the per−link settings, and **networkd.conf**(5) for the global settings.

IPv6PrivacyExtensions=
Configures use of stateless temporary addresses that change over time (see **RFC 4941**, 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. When unspecified, the value specified in the same setting in **networkd.conf**(5), which defaults to "no", will be used.

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. Defaults to false for bridge devices, when ***IPv6Forwarding=***, ***IPv6SendRA=***, or ***KeepMaster=*** is enabled. Otherwise, enabled by default. Cannot be enabled on devices aggregated in a bond device or 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** 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. Takes an integer in the range 1...255. 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.

IPv6RetransmissionTimeSec=
Configures IPv6 Retransmission Time. The time between retransmitted Neighbor Solicitation messages. Used by address resolution and the Neighbor Unreachability Detection algorithm. A value of zero is ignored and the kernel's current value will be used. Defaults to unset, and the kernel's current value will be used.

IPv4ReversePathFilter=
Configure IPv4 Reverse Path Filtering. If enabled, when an IPv4 packet is received, the machine will first check whether the ***source*** of the packet would be routed through the interface it came in. If there is no route to the source on that interface, the machine will drop the packet. Takes one of "no", "strict", or "loose". When "no", no source validation will be done. When "strict", each incoming packet is tested against the FIB and if the incoming interface is not the best reverse path, the packet check will fail. By default, failed packets are discarded. When "loose", each incoming packet's source address is tested against the FIB. The packet is dropped only if the source address is not reachable via any interface on that router. See **RFC 3704**. When unset, the kernel's default will be used.

MulticastIGMPVersion=
Configures IPv4 Multicast IGMP Version to be used, and controls the value of /proc/sys/net/ipv4/conf/***INTERFACE***/force_igmp_version. Takes one of "no", "v1", "v2", or "v3". When "no", no enforcement of an IGMP version is applied. IGMPv1/v2 fallbacks are allowed, and **systemd−networkd** will return to IGMPv3 mode after all IGMPv1/v2 Querier Present timers have expired. When "v1", use of IGMP version 1 is enforced. An IGMPv1 report will be returned even if IGMPv2/v3 queries are received. When "v2", use of IGMP version 2 is enforced. An IGMPv2 report will be returned if an IGMPv2/v3 query is received. **systemd−networkd** will fall back to IGMPv1 if an IGMPv1 query is received. When "v3", use of IGMP version 3 is enforced, and the response is the same as with "no". Defaults to unset — the sysctl is not set.

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**. When unset, the kernel's default will be used.

IPv4ProxyARPPrivateVLAN=
Takes a boolean. Configures proxy ARP private VLAN for IPv4, also known as VLAN aggregation, private VLAN, source−port filtering, port−isolation, or MAC−forced forwarding.
This variant of the ARP proxy technique will allow the ARP proxy to reply back to the same interface.
See **RFC 3069**. 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 setting 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 the [IPv6RoutePrefix] sections are distributed as defined in the [IPv6SendRA] section. If ***DHCPPrefixDelegation=*** is enabled, then the delegated prefixes are also distributed. See ***DHCPPrefixDelegation=*** setting and the [IPv6SendRA], [IPv6Prefix], [IPv6RoutePrefix], and [DHCPPrefixDelegation] sections for more configuration options.
If enabled, ***IPv6Forwarding=*** on this interface is also enabled, unless the setting is explicitly specified. See ***IPv6Forwarding=*** in the above for more details.

DHCPPrefixDelegation=
Takes a boolean value. When enabled, requests subnet prefixes on another link via the DHCPv6 protocol or via the 6RD option in the DHCPv4 protocol. An address within each delegated prefix will be assigned, and the prefixes will be announced through IPv6 Router Advertisement if ***IPv6SendRA=*** is enabled. This behaviour can be configured in the [DHCPPrefixDelegation] 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.

KeepMaster=
Takes a boolean value. When enabled, the current master interface index will not be changed, and ***BatmanAdvanced=***, ***Bond=***, ***Bridge=***, and ***VRF=*** settings are ignored. This may be useful when a netdev with a master interface is created by another program, e.g. **systemd-nspawn**(1). Defaults to false.

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).

IPoIB=, ***IPVLAN=***, ***IPVTAP=***, ***MACsec=***, ***MACVLAN=***, ***MACVTAP=***, ***Tunnel=***, ***VLAN=***, ***VXLAN=***, ***Xfrm=***
The name of an IPoIB, IPVLAN, IPVTAP, 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 **systemd−networkd** to configure a specific link even if it has no carrier. Defaults to false. If enabled, and the ***IgnoreCarrierLoss=*** setting is not explicitly set, then it is enabled as well.
With this enabled, to make the interface enter the "configured" state, which is required to make **systemd−networkd−wait−online** work properly for the interface, all dynamic address configuration mechanisms like ***DHCP=*** and ***IPv6AcceptRA=*** (which is enabled by default in most cases) need to be disabled. Also, ***DuplicateAddressDetection=*** (which is enabled by default for IPv4 link−local addresses and all IPv6 addresses) needs to be disabled for all static address configurations. Otherwise, without carrier, the interface will be stuck in the "configuring" state, and **systemd−networkd−wait−online** for the interface will timeout. Also, it is recommended to set ***RequiredForOnline=no−carrier*** to make **systemd−networkd−wait−online** work for the interface.

IgnoreCarrierLoss=
Takes a boolean or a timespan. When true, **systemd−networkd** retains both the static and dynamic configuration of the interface even if its carrier is lost. When false, **systemd−networkd** drops both the static and dynamic configuration of the interface. When a timespan is specified, **systemd−networkd** waits for the specified timespan, and ignores the carrier loss if the link regain its carrier within the timespan. Setting 0 seconds is equivalent to "no", and "infinite" is equivalent to "yes".
Setting a finite timespan may be useful when e.g. in the following cases:
- A wireless interface connecting to a network which has multiple access points with the same SSID.
- Enslaving a wireless interface to a bond interface, which may disconnect from the connected access point and causes its carrier to be lost.
- The driver of the interface resets when the MTU is changed.

When ***Bond=*** is specified to a wireless interface, defaults to 3 seconds. When the DHCPv4 client is enabled and ***UseMTU=*** in the [DHCPv4] section enabled, defaults to 5 seconds. Otherwise, defaults to the value specified with ***ConfigureWithoutCarrier=***. When ***ActivationPolicy=*** is set to "always−up", this is forced to "yes", and ignored any user specified values.

KeepConfiguration=
Takes a boolean or one of "static", "dynamic−on−stop", and "dynamic". When "static", **systemd−networkd** will not drop statically configured addresses and routes on starting up process. When "dynamic−on−stop", the dynamically configurad addresses and routes, such as DHCPv4, DHCPv6, SLAAC, and IPv4 link−local address, will not be dropped when **systemd−networkd** is being stopped. When "dynamic", the dynamically configured addresses and routes will never be dropped, and the lifetime of DHCPv4 leases will be ignored. 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 "dynamic" implies "dynamic−on−stop", and "yes" implies "dynamic" and "static". Defaults to "dynamic−on−stop" when **systemd−networkd** is running in initrd, "yes" when the root filesystem is a network filesystem, and "no" otherwise.

[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 setting 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=*** setting.

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=
Specifies the label for the IPv4 address. The label must be a 7−bit ASCII string with a length of 1...15 characters. Defaults to unset.

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 integer in the range 0...255. Defaults to "global". IPv4 only − IPv6 scope is automatically assigned by the kernel and cannot be set manually.

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**. Supported only on IPv6. Defaults to false.

DuplicateAddressDetection=
Takes one of "ipv4", "ipv6", "both", or "none". When "ipv4", performs IPv4 Address Conflict Detection. See **RFC 5227**. When "ipv6", performs IPv6 Duplicate Address Detection. See **RFC 4862**. Defaults to "ipv4" for IPv4 link−local addresses (169.254.0.0/16), "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**. 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 **systemd−networkd**'s netdev kind vxlan have the group option that enables them 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".

NetLabel=label
This setting provides a method for integrating static and dynamic network configuration into Linux **NetLabel** subsystem rules, used by **Linux Security Modules (LSMs)** for network access control. The label, with suitable LSM rules, can be used to control connectivity of (for example) a service with peers in the local network. At least with SELinux, only the ingress can be controlled but not egress. The benefit of using this setting is that it may be possible to apply interface independent part of NetLabel configuration at very early stage of system boot sequence, at the time when the network interfaces are not available yet, with **netlabelctl**(8), and the per−interface configuration with **systemd−networkd** once the interfaces appear later. Currently this feature is only implemented for SELinux.
The option expects a single NetLabel label. The label must conform to lexical restrictions of LSM labels. When an interface is configured with IP addresses, the addresses and subnetwork masks will be appended to the **NetLabel Fallback Peer Labeling** rules. They will be removed when the interface is deconfigured. Failures to manage the labels will be ignored.

Warning Once labeling is enabled for network traffic, a lot of LSM access control points in Linux networking stack go from dormant to active. Care should be taken to avoid getting into a situation where for example remote connectivity is broken, when the security policy has not been updated to consider LSM per−packet access controls and no rules would allow any network traffic. Also note that additional configuration with **netlabelctl**(8) is needed.

Example:
```
[Address] NetLabel=system_u:object_r:localnet_peer_t:s0
```
With the example rules applying for interface "eth0", when the interface is configured with an IPv4 address of 10.0.0.123/8, **systemd−networkd** performs the equivalent of **netlabelctl** operation
```
netlabelctl unlbl add interface eth0 address:10.0.0.0/8 label:system_u:object_r:localnet_peer_t:s0
```
and the reverse operation when the IPv4 address is deconfigured. The configuration can be used with LSM rules; in case of SELinux to allow a SELinux domain to receive data from objects of SELinux "peer" class. For example:
```
type localnet_peer_t; allow my_server_t localnet_peer_t:peer recv;
```
The effect of the above configuration and rules (in absence of other rules as may be the case) is to only allow "my_server_t" (and nothing else) to receive data from local subnet 10.0.0.0/8 of interface "eth0".

NFTSet=source:family:table:set
This setting provides a method for integrating network configuration into firewall rules with **NFT** sets. The benefit of using the setting is that static network configuration (or dynamically obtained network addresses, see similar directives in other sections) can be used in firewall rules with the indirection of NFT set types. For example, access could be granted for hosts in the local subnetwork only. Firewall rules using IP address of an interface are also instantly updated when the network configuration changes, for example via DHCP.
This option expects a whitespace separated list of NFT set definitions. Each definition consists of a colon−separated tuple of source type (one of "address", "prefix" or "ifindex"), NFT address family (one of "arp", "bridge", "inet", "ip", "ip6", or "netdev"), table name and set name. The names of tables and sets must conform to lexical restrictions of NFT table names. The type of the element used in the NFT filter must match the type implied by the directive ("address", "prefix" or "ifindex") and address type (IPv4 or IPv6) as shown in the table below.
Table 1. Defined *source type* values When an interface is configured with IP addresses, the addresses, subnetwork masks or interface index will be appended to the NFT sets. The information will be removed when the interface is deconfigured. **systemd−networkd** only inserts elements to (or removes from) the sets, so the related NFT rules, tables and sets must be prepared elsewhere in advance. Failures to manage the sets will be ignored.
Example:
```
[Address] NFTSet=prefix:netdev:filter:eth_ipv4_prefix
```
Corresponding NFT rules:
```
table netdev filter { set eth_ipv4_prefix { type ipv4_addr flags interval } chain eth_ingress { type filter hook ingress device "eth0" priority filter; policy drop; ip daddr != @eth_ipv4_prefix drop accept } }
```

[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**. Precedence is managed by userspace, and only the label itself is stored in the kernel.

Label=
The label for the prefix. Takes an unsigned integer in the range 0...4294967294 (0xfffffffe). 4294967295

(0xffffffff) is reserved. This setting is mandatory.

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

[ROUTINGPOLICYRULE] SECTION OPTIONS

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

TypeOfService=
This specifies the Type of Service (ToS) field of packets to match; it takes an unsigned integer in the range 0...255. The field can be used to specify precedence (the first 3 bits) and ToS (the next 3 bits). The field can be also used to specify Differentiated Services Code Point (DSCP) (the first 6 bits) and Explicit Congestion Notification (ECN) (the last 2 bits). See **Type of Service** and **Differentiated services** for more details.

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 in the range 0...4294967295). Optionally, the firewall mask (also a number between 0...4294967295) can be suffixed with a slash ("/"), e.g., "7/255". When the mark value is non−zero and no mask is explicitly specified, all bits of the mark are compared.

Table=
Specifies the routing table identifier to look up 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". Ignored if ***L3MasterDevice=*** is true.

Priority=
Specifies the priority of this rule. ***Priority=*** is an 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.

GoTo=
Specifies the target priority used by the "goto" type of rule. Takes an integer in the range 1...4294967295. This must be larger than the priority of the rule specified in ***Priority=***. When specified, ***Type=goto*** is implied. This is mandatory when ***Type=goto***.

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.

L3MasterDevice=
Takes a boolean. Specifies whether the rule is to direct lookups to the tables associated with level 3 master devices (also known as Virtual Routing and Forwarding or VRF devices). For further details see **Virtual Routing and Forwarding (VRF)**. Defaults to false.

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.

SuppressInterfaceGroup=
Takes an integer in the range 0...2147483647 and rejects routing decisions that have an interface with the same group id. It has the same meaning as **suppress_ifgroup** in **ip rule**. Defaults to unset.

Type=
Specifies Routing Policy Database (RPDB) rule type. Takes one of "table", "goto", "nop", "blackhole", "unreachable", or "prohibit". When "goto", the target priority must be specified in ***GoTo=***. Defaults to "table".

[NEXTHOP] SECTION OPTIONS

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

Id=
The id of the next hop. Takes an integer in the range 1...4294967295. This is mandatory if ***ManageForeignNextHops=no*** is specified in **networkd.conf**(5). Otherwise, if unspecified, an unused ID will be automatically picked.

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 settings. 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** 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 does not exist.

For IPv4 route, defaults to "host" if ***Type=*** is "local" or "nat", and "link" if ***Type=*** is "broadcast", "multicast", "anycast", or "unicast". 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".

HopLimit=
Configures per route hop limit. Takes an integer in the range 1...255. See also ***IPv6HopLimit=***.

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, the TCP quick ACK mode for the route is enabled. 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.

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.

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.

TCPCongestionControlAlgorithm=
Specifies the TCP congestion control algorithm for the route. Takes a name of the algorithm, e.g. "bbr", "dctcp", or "vegas". When unset, the kernel's default will be used.

TCPRetransmissionTimeoutSec=
Specifies the TCP Retransmission Timeout (RTO) for the route. Takes time values in seconds. This value specifies the timeout of an alive TCP connection, when retransmissions remain unacknowledged. 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:

RequestAddress=
Takes an IPv4 address. When specified, the Requested IP Address option (option code 50) is added with it to the initial DHCPDISCOVER message sent by the DHCP client. Defaults to unset, and an already assigned dynamic address to the interface is automatically picked.

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**. 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** or **duid**. 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, is used. IAID can be configured by ***IAID=***. DUID can be configured by ***DUIDType=*** and ***DUIDRawData=***. 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.

RapidCommit=
Takes a boolean. The DHCPv4 client can obtain configuration parameters from a DHCPv4 server through a rapid two−message exchange (discover and ack). When the rapid commit option is set by both the DHCPv4 client and the DHCPv4 server, the two−message exchange is used. Otherwise, the four−message exchange (discover, offer, request, and ack) is used. The two−message exchange provides faster client configuration. See **RFC 4039** for details. Defaults to true when ***Anonymize=no*** and neither ***AllowList=*** nor ***DenyList=*** is specified, and false otherwise.

Anonymize=
Takes a boolean. When true, the options sent to the DHCP server will follow the **RFC 7844** (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, ***ClientIdentifier=mac***, ***RapidCommit=no***, ***SendHostname=no***, ***Use6RD=no***, ***UseCaptivePortal=no***, ***UseMTU=no***, ***UseNTP=no***, ***UseSIP=no***, and ***UseTimezone=no*** are implied and these settings in the .network file are silently ignored. Also, ***Hostname=***, ***MUDURL=***, ***RequestAddress=***, ***RequestOptions=***, ***SendOption=***, ***SendVendorOption=***, ***UserClass=***, and ***VendorClassIdentifier=*** are silently 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**. 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**. This setting can be specified multiple times. If an empty string is specified, then all options specified earlier are cleared. Defaults to unset.

IPServiceType=
Takes one of the special values "none", "CS6", or "CS4". When "none" no IP service type is set to the packet sent from the DHCPv4 client. When "CS6" (network control) or "CS4" (realtime), the corresponding service type will be set. Defaults to "CS6".

SocketPriority=
The Linux socket option **SO_PRIORITY** applied to the raw IP socket used for initial DHCPv4 messages. Unset by default. Usual values range from 0 to 6. More details about **SO_PRIORITY** socket option in **socket**(7). Can be used in conjunction with [VLAN] section ***EgressQOSMaps=*** setting of .netdev file to set the 802.1Q VLAN ethernet tagged header priority, see **systemd.netdev**(5).

Label=
Specifies the label for the IPv4 address received from the DHCP server. The label must be a 7−bit ASCII string with a length of 1...15 characters. 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.

UseCaptivePortal=
When true (the default), the captive portal advertised by the DHCP server will be recorded and made available to client programs and displayed in the **networkctl**(1) status output per−link.

UseDNR=
When true, designated resolvers advertised by the DHCP server will be used as encrypted DNS servers. See **RFC 9463**.
Defaults to unset, and the value for ***UseDNS=*** will be used.

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.
Note, some drivers will reset the interfaces if the MTU is changed. For such interfaces, please try to use ***IgnoreCarrierLoss=*** with a short timespan, e.g. "3 seconds".

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, similarly 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, similarly to the effect of the **Domains=** setting when the argument is prefixed with "˜".
When unspecified, the value specified in the same setting in the [Network] section will be used. When it is unspecified, the value specified in the same setting in the [DHCPv4] section in **networkd.conf**(5) will be used. When it is unspecified, the value specified in the same setting in the [Network] section in **networkd.conf**(5) will be used. When none of them are specified, defaults to "no".
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 (including the prefix route added for the specified prefix). Takes an unsigned integer in the range 0...4294967295. Defaults to 1024.

RouteTable=num
The table identifier for DHCP routes. Takes one of predefined names "default", "main", and "local", and names defined in ***RouteTable=*** in **networkd.conf**(5), or a number between 1...4294967295.
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.

QuickAck=
Takes a boolean. When true, the TCP quick ACK mode is enabled for the routes configured by the acquired DHCPv4 lease. When unset, the kernel's default will be used.

InitialCongestionWindow=
As in the [Route] section.

InitialAdvertisedReceiveWindow=
As in the [Route] section.

UseGateway=
When true, and the DHCP server provides a Router option, the default gateway based on the router address will be configured. Defaults to unset, and the value specified with ***UseRoutes=*** will be used.
Note, when the server provides both the Router and Classless Static Routes option, and ***UseRoutes=*** is enabled, the Router option is always ignored regardless of this setting. See **RFC 3442**.

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

Use6RD=
When true, subnets of the received IPv6 prefix are assigned to downstream interfaces which enables ***DHCPPrefixDelegation=***. See also ***DHCPPrefixDelegation=*** in the [Network] section, the [DHCPPrefixDelegation] section, and **RFC 5969**. Defaults to false.

UnassignedSubnetPolicy=
Takes "none", or one of the reject types: "unreachable", "prohibit", "blackhole", or "throw". If a reject type is specified, the reject route corresponding to the acquired 6RD prefix will be configured. For example, when "unreachable",

```
unreachable 2001:db8::/56 dev lo proto dhcp metric 1024 pref medium
```

will be configured. See **RFC 7084**. If "none" is specified, such route will not be configured. This may be useful when custom firewall rules that handle packets for unassigned subnets will be configured. Defaults to "unreachable".

IPv6OnlyMode=
When true, the DHCPv4 configuration will be delayed by the timespan provided by the DHCP server and skip to configure dynamic IPv4 network connectivity if IPv6 connectivity is provided within the timespan. See **RFC 8925**. Defaults to false.

FallbackLeaseLifetimeSec=
Allows one to set DHCPv4 lease lifetime when DHCPv4 server does not send the lease lifetime. Takes one of "forever" or "infinity". If specified, the acquired 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.

ServerPort=
Set the port on which the DHCP server is listening.

DenyList=
A whitespace−separated list of IPv4 addresses. Each address can optionally take a prefix length after "/". DHCP offers from servers in the list are rejected. Note that if ***AllowList=*** is configured then ***DenyList=*** is ignored. Note that this filters only DHCP offers, so the filtering might not work when ***RapidCommit=*** is enabled. See also ***RapidCommit=*** above.

AllowList=
A whitespace−separated list of IPv4 addresses. Each address can optionally take a prefix length after "/". DHCP offers from servers in the list are accepted.
Note that this filters only DHCP offers, so the filtering might not work when ***RapidCommit=*** is enabled. See also ***RapidCommit=*** above.

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

SendDecline=
A boolean. When true, **systemd−networkd** performs IPv4 Duplicate Address Detection to the acquired address by the DHCPv4 client. If duplicate is detected, the DHCPv4 client rejects the address by sending a **DHCPDECLINE** packet to the DHCP server, and tries to obtain an IP address again. See **RFC 5227**. Defaults to false.

NetLabel=
This applies the NetLabel for the addresses received with DHCP, like ***NetLabel=*** in [Address] section applies it to statically configured addresses. See ***NetLabel=*** in [Address] section for more details.

NFTSet=
This applies the NFT set for the network configuration received with DHCP, like ***NFTSet=*** in [Address] section applies it to static configuration. See ***NFTSet=*** in [Address] section for more details. For "address" or "prefix" source types, the type of the element used in the NFT filter must be "ipv4_addr".

[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**. 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**. 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.

UnassignedSubnetPolicy=
Takes "none" or one of the reject types: "unreachable", "prohibit", "blackhole", or "throw". If a reject type is specified, the reject route corresponding to the delegated prefix will be configured. For example, when "unreachable",

```
unreachable 2001:db8::/56 dev lo proto dhcp metric 1024 pref medium
```

will be configured. See **RFC 7084**. If "none" is specified, such route will not be configured. This may be useful when custom firewall rules that handle packets for unassigned subnets will be configured. Defaults to "unreachable".

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 set by both the DHCPv6 client and the DHCPv6 server, the two−message exchange is used. Otherwise, the four−message exchange (solicit, advertise, request, and reply) is used. The two−message exchange provides faster client configuration. See **RFC 3315** for details. Defaults to true, and the two−message exchange will be used if the server support it.

SendHostname=
When true (the default), the machine's hostname (or the value specified with ***Hostname=***, described below) will be sent to the DHCPv6 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 DHCPv6 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.

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

UseCaptivePortal=
When true (the default), the captive portal advertised by the DHCPv6 server will be recorded and made available to client programs and displayed in the **networkctl**(1) status output per−link.

UseDelegatedPrefix=
When true (the default), the client will request the DHCPv6 server to delegate prefixes. If the server provides prefixes to be delegated, then subnets of the prefixes are assigned to the interfaces that have ***DHCPPrefixDelegation=yes***. See also the ***DHCPPrefixDelegation=*** setting in the [Network] section, settings in the [DHCPPrefixDelegation] section, and **RFC 8415**.

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

NFTSet=
This applies the NFT set for the network configuration received with DHCP, like ***NFTSet=*** in [Address] section applies it to static configuration. See ***NFTSet=*** in [Address] section for more details. For "address" or "prefix" source types, the type of the element used in the NFT filter must be "ipv6_addr".

WithoutRA=
Allows DHCPv6 client to start without router advertisements's "managed" or "other configuration" flag. Takes one of "no", "solicit", or "information−request". If this is not specified, "solicit" is used when ***DHCPPrefixDelegation=*** is enabled and ***UplinkInterface=:self*** is specified in the [DHCPPrefixDelegation] section. Otherwise, defaults to "no", and the DHCPv6 client will be started when an RA is received. See also the ***DHCPv6Client=*** setting in the [IPv6AcceptRA] section.

[DHCPPREFIXDELEGATION] SECTION OPTIONS

The [DHCPPrefixDelegation] section configures subnet prefixes of the delegated prefixes acquired by a DHCPv6 client or by a DHCPv4 client through the 6RD option on another interface. The settings in this section are used only when the ***DHCPPrefixDelegation=*** setting in the [Network] section is enabled.

UplinkInterface=
Specifies the name or the index of the uplink interface, or one of the special values ":self" and ":auto". When ":self", the interface itself is considered the uplink interface, and ***WithoutRA=solicit*** is implied if the setting is not explicitly specified. When ":auto", the first link which acquired prefixes to be delegated from the DHCPv6 or DHCPv4 server is selected. Defaults to ":auto".

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**, 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. This setting will be ignored when the ***DHCPPrefixDelegation=*** setting is enabled on the upstream interface. 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 interface), 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. This accepts the same syntax as ***Token=*** in the [IPv6AcceptRA] section. If ***Assign=*** is set to false, then this setting will be ignored. Defaults to unset, which means the EUI−64 algorithm will be used.

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 set to 0, the kernel's default value is used. Defaults to 256.

NetLabel=
This applies the NetLabel for the addresses received with DHCP, like ***NetLabel=*** in [Address] section applies it to statically configured addresses. See ***NetLabel=*** in [Address] section for more details.

NFTSet=
This applies the NFT set for the network configuration received with DHCP, like ***NFTSet=*** in [Address] section applies it to static configuration. See ***NFTSet=*** in [Address] section for more details. For "address" or "prefix" source types, the type of the element used in the NFT filter must be "ipv6_addr".

[IPV6ACCEPTRA] SECTION OPTIONS

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

UseRedirect=
When true (the default), Redirect message sent by the current first−hop router will be accepted, and routes to redirected nodes will be configured.

Token=
Specifies an optional address generation mode for the Stateless Address Autoconfiguration (SLAAC). The following values are supported:

eui64
The EUI−64 algorithm will be used to generate an address for that prefix. Only supported by Ethernet or InfiniBand interfaces.

static:*ADDRESS*
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.

prefixstable[:*ADDRESS*][,*UUID*]
The algorithm specified in **RFC 7217** will be used to generate interface identifiers. 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.
This mode can also optionally take a non−null UUID in the format which **sd_id128_from_string()** accepts, e.g. "86b123b969ba4b7eb8b3d8605123525a" or "86b123b9−69ba−4b7e−b8b3−d8605123525a". If a UUID is specified, the value is used as the secret key to generate interface identifiers. If not specified, then an application specific ID generated with the system's machine−ID will be used as the secret key. See **sd-id128**(3), **sd_id128_from_string**(3), and **sd_id128_get_machine**(3).
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 be changed, even if the prefix received in the RA message has not been changed.

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 for Ethernet or InfiniBand interfaces, otherwise "prefixstable" will be used to form an interface identifier for that prefix.
This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.
Examples:
Token=eui64 Token=::1a:2b:3c:4d Token=static:::1a:2b:3c:4d Token=prefixstable
Token=prefixstable:2002:da8:1::

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).

UseDNR=
When true, the DNR servers received in the Router Advertisement will be used. Defaults to the value of **UseDNS=**.

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, similarly 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, similarly to the effect of the **Domains=** setting when the argument is prefixed with "~". Defaults to false.
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. Takes one of predefined names "default", "main", and "local", and names defined in ***RouteTable=*** in **networkd.conf**(5), or a number between 1...4294967295.
When used in combination with ***VRF=***, the VRF's routing table is used when this parameter is not specified.

RouteMetric=
Set the routing metric for the routes received in the Router Advertisement. Takes an unsigned integer in the range 0...4294967295, or three unsigned integer separated with ":", in that case the first one is used when the router preference is high, the second is for medium preference, and the last is for low preference ("***high***:***medium***:***low***"). Defaults to "512:1024:2048".

QuickAck=
Takes a boolean. When true, the TCP quick ACK mode is enabled for the routes configured by the received RAs. When unset, the kernel's default will be used.

UseMTU=
Takes a boolean. When true, the MTU received in the Router Advertisement will be used. Defaults to true.

UseHopLimit=
Takes a boolean. When true, the hop limit received in the Router Advertisement will be set to routes configured based on the advertisement. See also ***IPv6HopLimit=***. Defaults to true.

UseReachableTime=
Takes a boolean. When true, the reachable time received in the Router Advertisement will be set on the interface receiving the advertisement. It is used as the base timespan of the validity of a neighbor entry. Defaults to true.

UseRetransmissionTime=
Takes a boolean. When true, the retransmission time received in the Router Advertisement will be set on the interface receiving the advertisement. It is used as the time between retransmissions of Neighbor Solicitation messages to a neighbor when resolving the address or when probing the reachability of a neighbor. Defaults to true.

UseGateway=
When true (the default), the router address will be configured as the default gateway.

UseRoutePrefix=
When true (the default), the routes corresponding to the route prefixes received in the Router Advertisement will be configured.

UseCaptivePortal=
When true (the default), the captive portal received in the Router Advertisement will be recorded and made available to client programs and displayed in the **networkctl**(1) status output per−link.

UsePREF64=
When true, the IPv6 PREF64 (or NAT64) prefixes received in the Router Advertisement will be recorded and made available to client programs and displayed in the **networkctl**(1) status output per−link. See **RFC 8781**. Defaults to false.

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. Each address can optionally take a prefix length after "/". Any information advertised by the listed router is ignored.

RouterAllowList=
A whitespace−separated list of IPv6 router addresses. Each address can optionally take a prefix length after "/". 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. Each prefix can optionally take its prefix length after "/". IPv6 prefixes supplied via router advertisements in the list are ignored.

PrefixAllowList=
A whitespace−separated list of IPv6 prefixes. Each prefix can optionally take its prefix length after "/". 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. Each prefix can optionally take its prefix length after "/". IPv6 route prefixes supplied via router advertisements in the list are ignored.

RouteAllowList=

A whitespace−separated list of IPv6 route prefixes. Each prefix can optionally take its prefix length after "/". 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, the DHCPv6 client will be started in "solicit" mode if the RA has the "managed" flag or "information−request" mode if the RA lacks the "managed" flag but has the "other configuration" flag. If set to "always", the DHCPv6 client will be started in "solicit" mode when an RA is received, even if neither the "managed" nor the "other configuration" flag is set in the RA. This will be ignored when ***WithoutRA=*** in the [DHCPv6] section is enabled, or ***UplinkInterface=:self*** in the [DHCPPrefixDelegation] section is specified. Defaults to true.

NetLabel=

This applies the NetLabel for the addresses received with RA, like ***NetLabel=*** in [Address] section applies it to statically configured addresses. See ***NetLabel=*** in [Address] section for more details.

NFTSet=

This applies the NFT set for the network configuration received with RA, like ***NFTSet=*** in [Address] section applies it to static configuration. See ***NFTSet=*** in [Address] section for more details. For "address" or "prefix" source types, the type of the element used in the NFT filter must be "ipv6_addr".

[DHCPSERVER] SECTION OPTIONS

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

ServerAddress=

Specifies the server address for the DHCP server. Takes an IPv4 address with prefix length separated with a slash, e.g. "192.168.0.1/24". Defaults to unset, and one of static IPv4 addresses configured in [Network] or [Address] section will be automatically selected. This setting may be useful when the interface on which the DHCP server is running has multiple static IPv4 addresses.

This implies ***Address=*** in [Network] or [Address] section with the same address and prefix length. That is,

```
[Network] DHCPServer=yes Address=192.168.0.1/24 Address=192.168.0.2/24 [DHCPServer] ServerAddress=192.168.0.1/24
```

or

```
[Network] DHCPServer=yes [Address] Address=192.168.0.1/24 [Address] Address=192.168.0.2/24 [DHCPServer] ServerAddress=192.168.0.1/24
```

are equivalent to the following:

```
[Network] DHCPServer=yes Address=192.168.0.2/24 [DHCPServer] ServerAddress=192.168.0.1/24
```

Since version 255, like the ***Address=*** setting in [Network] or [Address] section, this also supports a null address, e.g. "0.0.0.0/24", and an unused address will be automatically selected. For more details about the automatic address selection, see ***Address=*** setting in [Network] section in the above.

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 the name or the index of the uplink interface, or one of the special values ":none" and ":auto". When emitting DNS, NTP, or SIP servers is enabled but no servers are specified, the servers configured in the uplink interface will be emitted. When ":auto", the link which has a default gateway with the highest 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, or special value "_server_address" which will be converted to the address used by the DHCP server.

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.

This setting can be specified multiple times. If an empty string is specified, then all DNS servers specified earlier are cleared.

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=, ***Router=***

The ***EmitRouter=*** setting takes a boolean value, and configures whether the DHCP lease should contain the router option. The ***Router=*** setting takes an IPv4 address, and configures the router address to be emitted. When the ***Router=*** setting is not specified, then the server address will be used for the router option. When the ***EmitRouter=*** setting is disabled, the ***Router=*** setting will be ignored. The ***EmitRouter=*** setting defaults to true, and the ***Router=*** setting defaults to unset.

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.

BootServerAddress=

Takes an IPv4 address of the boot server used by e.g. PXE boot systems. When specified, this address is sent in the **siaddr** field of the DHCP message header. See **RFC 2131** for more details. Defaults to unset.

BootServerName=

Takes a name of the boot server used by e.g. PXE boot systems. When specified, this name is sent in the DHCP option 66 ("TFTP server name"). See **RFC 2132** for more details. Defaults to unset.

Note that typically setting one of ***BootServerName=*** or ***BootServerAddress=*** is sufficient, but both can be set too, if desired.

BootFilename=

Takes a path or URL to a file loaded by e.g. a PXE boot loader. When specified, this path is sent in the DHCP option 67 ("Bootfile name"). See **RFC 2132** for more details. Defaults to unset.

IPv6OnlyPreferredSec=

Takes a timespan. Controls the **RFC 8925** IPv6−Only Preferred option. Specifies the DHCPv4 option to indicate that a host supports an IPv6−only mode and is willing to forgo obtaining an IPv4 address if the network provides IPv6 connectivity. Defaults to unset, and not send the option. The minimum allowed value is 300 seconds.

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**. 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**. 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 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**. 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.

RapidCommit=

Takes a boolean. When true, the server supports **RFC 4039**. When a client sends a DHCPDISCOVER message with the Rapid Commit option to the server, then the server will reply with a DHCPACK message to the client, instead of DHCPOFFER. Defaults to true.

PersistLeases=

Takes a boolean. When true, the DHCP server will load and save leases in the persistent storage. When false, the DHCP server will neither load nor save leases in the persistent storage. Hence, bound leases will be lost when the interface is reconfigured e.g. by **networkctl reconfigure**, or **systemd-networkd.service**(8) is restarted. That may cause address conflict on the network. So, please take an extra care when disable this setting. When unspecified, the value specified in the same setting in **networkd.conf**(5), which defaults to "yes", will be used.

[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. The value must be 0 seconds, or between 4 seconds and 9000 seconds. When set to 0, the host is not acting as a router. Defaults to 1800 seconds (30 minutes).

ReachableTimeSec=
Configures the time, used in the Neighbor Unreachability Detection algorithm, for which clients can assume a neighbor is reachable after having received a reachability confirmation. Takes a time span in the range 0...4294967295 ms. When 0, clients will handle it as if the value was not specified. Defaults to 0.

RetransmitSec=
Configures the time, used in the Neighbor Unreachability Detection algorithm, for which clients can use as retransmit time on address resolution and the Neighbor Unreachability Detection algorithm. Takes a time span in the range 0...4294967295 ms. When 0, clients will handle it as if the value wasn't specified. Defaults to 0.

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** for details. Defaults to "medium".

HopLimit=
Configures hop limit. Takes an integer in the range 0...255. See also ***IPv6HopLimit=***.

UplinkInterface=
Specifies the name or the index of the uplink interface, or one of the special values ":none" and ":auto". When emitting DNS servers or search domains is enabled but no servers are specified, the servers configured in the uplink interface will be emitted. When ":auto", the value specified to the same setting in the [DHCPPrefixDelegation] section will be used if ***DHCPPrefixDelegation=*** is enabled, otherwise the link which has a default gateway with the highest priority will be automatically selected. When ":none", no uplink interface will be selected. Defaults to ":auto".

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 interface specified in ***UplinkInterface=*** will be 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 interface specified in ***UplinkInterface=*** will be 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=***. Defaults to 3600 seconds (one hour).

HomeAgent=
Takes a boolean. Specifies that IPv6 router advertisements indicate to hosts that the router acts as a Home Agent and includes a Home Agent option. Defaults to false. See **RFC 6275** for further details.

HomeAgentLifetimeSec=
Takes a timespan. Specifies the lifetime of the Home Agent. An integer, the default unit is seconds, in the range 1...65535. Defaults to the value set to ***RouterLifetimeSec=***.

HomeAgentPreference=
Configures IPv6 Home Agent preference. Takes an integer in the range 0...65535. Defaults to 0.

[IPV6PREFIX] SECTION OPTIONS

One or more [IPv6Prefix] sections contain the IPv6 prefixes that are announced via Router Advertisements. See **RFC 4861** 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 1800 seconds (30 minutes) and ***ValidLifetimeSec=*** defaults to 3600 seconds (one hour).

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

Token=
Specifies an optional address generation mode for assigning an address in each prefix. This accepts the same syntax as ***Token=*** in the [IPv6AcceptRA] section. If ***Assign=*** is set to false, then this setting will be ignored. Defaults to unset, which means the EUI−64 algorithm will be used.

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** 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 [IPv6RoutePrefix] sections to configure multiple IPv6 prefix routes.

LifetimeSec=
Lifetime for the route prefix measured in seconds. ***LifetimeSec=*** defaults to 3600 seconds (one hour).

[IPV6PREF64PREFIX] SECTION OPTIONS

One or more [IPv6PREF64Prefix] sections contain the IPv6 PREF64 (or NAT64) prefixes that are announced via Router Advertisements. See **RFC 8781** for further details.

Prefix=
The IPv6 PREF64 (or NAT64) prefix that is to be distributed to hosts. The setting holds an IPv6 prefix that should be set up for NAT64 translation (PLAT) to allow 464XLAT on the network segment. Use multiple [IPv6PREF64Prefix] sections to configure multiple IPv6 prefixes since prefix lifetime may differ from one prefix to another. The prefix is an address with a prefix length, separated by a slash "/" character. Valid NAT64 prefix length are 96, 64, 56, 48, 40, and 32 bits.

LifetimeSec=
Lifetime for the prefix measured in seconds. Should be greater than or equal to ***RouterLifetimeSec=***. ***LifetimeSec=*** defaults to 1800 seconds.

[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.

Isolated=
Takes a boolean. Configures whether this port is isolated or not. Within a bridge, isolated ports can only communicate with non−isolated ports. When set to true, this port can only communicate with other ports whose Isolated setting is false. When set to false, this port can communicate with any other ports. 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. **systemd−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 is 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, IPv6, or L2 MAC 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‰"). This will be ignored when ***BitRate=*** is unspecified.

TimeQuantaNSec=, ***PropagationSegment=***, ***PhaseBufferSegment1=***, ***PhaseBufferSegment2=***, ***SyncJumpWidth=***
Specifies the time quanta, propagation segment, phase buffer segment 1 and 2, and the synchronization jump width, which allow one to define the CAN bit−timing in a hardware independent format as proposed by the Bosch CAN 2.0 Specification. ***TimeQuantaNSec=*** takes a timespan in nanoseconds. ***PropagationSegment=***, ***PhaseBufferSegment1=***, ***PhaseBufferSegment2=***, and ***SyncJumpWidth=*** take number of time quantum specified in ***TimeQuantaNSec=*** and must be an unsigned integer in the range 0...4294967295. These settings except for ***SyncJumpWidth=*** will be ignored when ***BitRate=*** is specified.

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.

DataTimeQuantaNSec=, ***DataPropagationSegment=***, ***DataPhaseBufferSegment1=***, ***DataPhaseBufferSegment2=***, ***DataSyncJumpWidth=***
Specifies the time quanta, propagation segment, phase buffer segment 1 and 2, and the synchronization jump width for the data phase, if CAN−FD is used. These settings are analogous to the ***TimeQuantaNSec=*** or related settings.

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, or ***DataTimeQuanta=*** and related settings.

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 or a termination resistor value in ohm in the range 0...65535. When "yes", the termination resistor is set to 120 ohm. When "no" or "0" is set, the termination resistor is disabled. 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.

Loopback=
Takes a boolean. When "yes", loopback mode is enabled. When the loopback mode is enabled, the interface treats messages transmitted by itself as received messages. The loopback mode is important to debug CAN networks. When unset, the kernel's default will be used.

OneShot=
Takes a boolean. When "yes", one−shot mode is enabled. When unset, the kernel's default will be used.

PresumeAck=
Takes a boolean. When "yes", the interface will ignore missing CAN ACKs. When unset, the kernel's default will be used.

ClassicDataLengthCode=
Takes a boolean. When "yes", the interface will handle the 4bit data length code (DLC). When unset, the kernel's default will be used.

[IPOIB] SECTION OPTIONS

The [IPoIB] section manages the IP over Infiniband and accepts the following keys:

Mode=
Takes one of the special values "datagram" or "connected". Defaults to unset, and the kernel's default is used. When "datagram", the Infiniband unreliable datagram (UD) transport is used, and so the interface MTU is equal to the IB L2 MTU minus the IPoIB encapsulation header (4 bytes). For example, in a typical IB fabric with a 2K MTU, the IPoIB MTU will be 2048 − 4 = 2044 bytes.
When "connected", the Infiniband reliable connected (RC) transport is used. Connected mode takes advantage of the connected nature of the IB transport and allows an MTU up to the maximal IP packet size of 64K, which reduces the number of IP packets needed for handling large UDP datagrams, TCP segments, etc and increases the performance for large messages.

IgnoreUserspaceMulticastGroup=
Takes an boolean value. When true, the kernel ignores multicast groups handled by userspace. Defaults to unset, and the kernel's default is 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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.

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.

AutoRateIngress=
Takes a boolean value. Enables automatic capacity estimation based on traffic arriving at this qdisc. This is most likely to be useful with cellular links, which tend to change quality randomly. If this setting is enabled, the ***Bandwidth=*** setting is used as an initial estimate. Defaults to unset, and the kernel's default is used.

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

MPUBytes=
Rounds each packet (including overhead) up to the specified bytes. Takes an integer in the range 1...256. Defaults to unset and kernel's default is used.

CompensationMode=
Takes one of "none", "atm", or "ptm". Specifies the compensation mode for overhead calculation. When "none", no compensation is taken into account. When "atm", enables the compensation for ATM cell framing, which is normally found on ADSL links. When "ptm", enables the compensation for PTM encoding, which is normally found on VDSL2 links and uses a 64b/65b encoding scheme. Defaults to unset and the kernel's default is used.

UseRawPacketSize=
Takes a boolean value. When true, the packet size reported by the Linux kernel will be used, instead of the underlying IP packet size. Defaults to unset, and the kernel's default is used.

FlowIsolationMode=
CAKE places packets from different flows into different queues, then packets from each queue are delivered

fairly. This specifies whether the fairness is based on source address, destination address, individual flows, or any combination of those. The available values are:

none
: The flow isolation is disabled, and all traffic passes through a single queue.

src−host
: Flows are defined only by source address. Equivalent to the "srchost" option for **tc qdisc** command. See also **tc-cake**(8).

dst−host
: Flows are defined only by destination address. Equivalent to the "dsthost" option for **tc qdisc** command. See also **tc-cake**(8).

hosts
: Flows are defined by source−destination host pairs. Equivalent to the same option for **tc qdisc** command. See also **tc-cake**(8).

flows
: Flows are defined by the entire 5−tuple of source address, destination address, transport protocol, source port and destination port. Equivalent to the same option for **tc qdisc** command. See also **tc-cake**(8).

dual−src−host
: Flows are defined by the 5−tuple (see "flows" in the above), and fairness is applied first over source addresses, then over individual flows. Equivalent to the "dual−srchost" option for **tc qdisc** command. See also **tc-cake**(8).

dual−dst−host
: Flows are defined by the 5−tuple (see "flows" in the above), and fairness is applied first over destination addresses, then over individual flows. Equivalent to the "dual−dsthost" option for **tc qdisc** command. See also **tc-cake**(8).

triple
: Flows are defined by the 5−tuple (see "flows"), and fairness is applied over source and destination addresses, and also over individual flows. Equivalent to the "triple−isolate" option for **tc qdisc** command. See also **tc-cake**(8).

Defaults to unset and the kernel's default is used.

NAT=
: Takes a boolean value. When true, CAKE performs a NAT lookup before applying flow−isolation rules, to determine the true addresses and port numbers of the packet, to improve fairness between hosts inside the NAT. This has no practical effect when ***FlowIsolationMode=*** is "none" or "flows", or if NAT is performed on a different host. Defaults to unset, and the kernel's default is used.

PriorityQueueingPreset=
: CAKE divides traffic into "tins", and each tin has its own independent set of flow−isolation queues, bandwidth threshold, and priority. This specifies the preset of tin profiles. The available values are:

besteffort
: Disables priority queueing by placing all traffic in one tin.

precedence
: Enables priority queueing based on the legacy interpretation of TOS "Precedence" field. Use of this preset on the modern Internet is firmly discouraged.

diffserv8
: Enables priority queueing based on the Differentiated Service ("DiffServ") field with eight tins: Background Traffic, High Throughput, Best Effort, Video Streaming, Low Latency Transactions, Interactive Shell, Minimum Latency, and Network Control.

diffserv4
: Enables priority queueing based on the Differentiated Service ("DiffServ") field with four tins: Background Traffic, Best Effort, Streaming Media, and Latency Sensitive.

diffserv3
: Enables priority queueing based on the Differentiated Service ("DiffServ") field with three tins: Background Traffic, Best Effort, and Latency Sensitive.

Defaults to unset, and the kernel's default is used.

FirewallMark=
: Takes an integer in the range 1...4294967295. When specified, firewall−mark−based overriding of CAKE's tin selection is enabled. Defaults to unset, and the kernel's default is used.

Wash=
: Takes a boolean value. When true, CAKE clears the DSCP fields, except for ECN bits, of any packet passing through CAKE. Defaults to unset, and the kernel's default is used.

SplitGSO=
: Takes a boolean value. When true, CAKE will split General Segmentation Offload (GSO) super−packets into their on−the−wire components and dequeue them individually. Defaults to unset, and the kernel's default is used.

RTTSec=
: Specifies the RTT for the filter. Takes a timespan. Typical values are e.g. 100us for extremely high−performance 10GigE+ networks like datacentre, 1ms for non−WiFi LAN connections, 100ms for typical internet connections. Defaults to unset, and the kernel's default will be used.

AckFilter=
: Takes a boolean value, or special value "aggressive". If enabled, ACKs in each flow are queued and redundant ACKs to the upstream are dropped. If yes, the filter will always keep at least two redundant ACKs in the queue, while in "aggressive" mode, it will filter down to a single ACK. This may improve download throughput on links with very asymmetrical rate limits. Defaults to unset, and the kernel's default will be 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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.

[CLASSFULMULTIQUEUEING] SECTION OPTIONS

The [ClassfulMultiQueueing] section manages the queueing discipline (qdisc) of Classful Multi Queueing (mq).

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 0x1–0xffff 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.

[BANDMULTIQUEUEING] SECTION OPTIONS

The [BandMultiQueueing] section manages the queueing discipline (qdisc) of Band Multi Queueing (multiq).

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 0x1–0xffff 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.

[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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 0x1–0xffff 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 configurations of a bridge master or port, and accepts the following keys. To make the settings in this section take an effect, ***VLANFiltering=*** option has to be enabled on the bridge master, see the [Bridge] section in **systemd.netdev**(5). If at least one valid settings specified in this section in a .network file for an interface, all assigned VLAN IDs on the interface that are not configured in the .network file will be removed. If VLAN IDs on an interface need to be managed by other tools, then the settings in this section cannot be used in the matching .network file.

VLAN=

The VLAN ID allowed on the port. This can be either a single ID or a range M−N. Takes an integer in the range 1…4094. This setting can be specified multiple times. If an empty string is assigned, then the all previous assignments are cleared.

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

PVID=

The port VLAN ID specified here is assigned to all untagged frames at ingress. Takes an VLAN ID or negative boolean value (e.g. "no"). When false, the currently assigned port VLAN ID will be dropped. Configuring ***PVID=*** implicates the use of ***VLAN=*** setting in the above and will enable the VLAN ID for ingress as well. Defaults to unset, and will keep the assigned port VLAN ID if exists.

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 (DHCPv6 PD)

```
# /etc/systemd/network/55−dhcpv6−pd−upstream.network [Match] Name=enp1s0
[Network] DHCP=ipv6
# The lines below are optional, to also assign an address in the delegated prefix # to the upstream interface. Uncomment the lines below if necessary. #[Network] #DHCPPrefixDelegation=yes #[DHCPPrefixDelegation] #UplinkInterface=:self #SubnetId=0 #Announce=no
# If the upstream network does not provides any Router Advertisement (RA) messages, # then uncomment the lines below to make the DHCPv6 client forcibly started in the # managed mode. #[Network] #IPv6AcceptRA=no #[DHCPv6] #WithoutRA=solicit
# If the upstream network provides Router Advertisement (RA) messages with the # Managed bit unset, then uncomment the lines below to make the DHCPv6 client # forcibly started in the managed mode when an RA is received. #[DHCPv6] #UseAddress=no #[IPv6AcceptRA] #DHCPv6Client=always
# /etc/systemd/network/55−dhcpv6−pd−downstream.network [Match] Name=enp2s0
[Network] DHCPPrefixDelegation=yes IPv6SendRA=yes
# It is expected that the host is acting as a router. So, usually it is not # necessary to receive Router Advertisement from other hosts in the downstream network. IPv6AcceptRA=no
[DHCPPrefixDelegation] UplinkInterface=enp1s0 SubnetId=1 Announce=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. IPv6 Prefix Delegation (DHCPv4 6RD)

```
# /etc/systemd/network/55−dhcpv4−6rd−upstream.network [Match] Name=enp1s0
[Network] DHCP=ipv4
# When DHCPv4−6RD is used, the upstream network does not support IPv6. # Hence, it is not necessary to wait for Router Advertisement, which is enabled by default. IPv6AcceptRA=no
[DHCPv4] Use6RD=yes
# /etc/systemd/network/55−dhcpv4−6rd−downstream.network [Match] Name=enp2s0
[Network] DHCPPrefixDelegation=yes IPv6SendRA=yes
# It is expected that the host is acting as a router. So, usually it is not # necessary to receive Router Advertisement from other hosts in the downstream network. IPv6AcceptRA=no
[DHCPPrefixDelegation] UplinkInterface=enp1s0 SubnetId=1 Announce=yes
```

This will enable DHCPv4−6RD on the interface enp1s0 as an upstream interface where the DHCPv4 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 5. A bridge with two enslaved links

```
# /etc/systemd/network/25−bridge−static.netdev [NetDev] Name=bridge0 Kind=bridge
# /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 6. 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=100−299
[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 7. 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 8. 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 9. 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 will not 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 10. MacVTap

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

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

Example 11. 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-network-generator.service**(8), **systemd-resolved.service**(8)