NAME

systemd.network − Network configuration

SYNOPSIS

network.network

DESCRIPTION

Network setup is performed by **systemd-networkd**(8).

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

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

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

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

Note that an interface without any static IPv6 addresses configured, and neither DHCPv6 nor IPv6LL enabled, shall be considered to have no IPv6 support. IPv6 will be automatically disabled for that interface by writing "1" to /proc/sys/net/ipv6/conf/*ifname*/disable_ipv6.

[MATCH] SECTION OPTIONS

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

A network file is said to match a device if each of the entries in the "[Match]" section matches, or if the section is empty. The following keys are accepted:

MACAddress=

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

Example:

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

Path=

A whitespace−separated list of shell−style globs matching the persistent path, as exposed by the udev property "ID_PATH". If the list is prefixed with a "!", the test is inverted; i.e. it is true when "ID_PATH" does not match any item in the list.

Driver=

A whitespace−separated list of shell−style globs matching the driver currently bound to the device, as exposed by the udev property "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 the udev property "DEVTYPE". If the list is prefixed with a "!", the test is inverted.

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

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

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

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

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

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

[LINK] SECTION OPTIONS

The "[Link]" section accepts the following keys:

MACAddress=
: The hardware address to set for the device.

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

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

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

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

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

AllMulticast=
: Takes a boolean. If set to true, the driver retrieves all multicast packets from the network. This happens

when multicast routing is enabled.

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.

RequiredForOnline=

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

The network will be brought up normally in all cases, 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".

[NETWORK] SECTION OPTIONS

The "[Network]" section accepts the following keys:

Description=

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

DHCP=

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

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

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

See the "[DHCP]" section below for further configuration options for the DHCP client support.

DHCPServer=

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

LinkLocalAddressing=

Enables link−local address autoconfiguration. Accepts "yes", "no", "ipv4", or "ipv6". If *Bridge=* is set, defaults to "no", and if not, defaults to "ipv6".

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.

IPv6Token=

An IPv6 address with the top 64 bits unset. When set, indicates the 64−bit interface part of SLAAC IPv6 addresses for this link. Note that the token is only ever used for SLAAC, and not for DHCPv6 addresses, even in the case DHCP is requested by router advertisement. By default, the token is autogenerated.

LLMNR=

Takes a boolean or "resolve". When true, enables **Link−Local Multicast Name Resolution**[1] 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**[2] 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 false or "opportunistic". When set to "opportunistic", enables **DNS−over−TLS**[3] support on the link. This option defines a per−interface setting for **resolved.conf**(5)'s global *DNSOverTLS=* option. Defaults to false. This setting is read by **systemd-resolved.service**(8).

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

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

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

EmitLLDP=
: Controls support for Ethernet LLDP packet emission. Accepts a boolean parameter or the special values "nearest−bridge", "non−tpmr−bridge" and "customer−bridge". Defaults to false, which turns off LLDP packet emission. If not false, a short LLDP packet with information about the local system is sent out in regular intervals on the link. The LLDP packet will contain information about the local host name, 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**[5]. 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.

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

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

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

DNS=
A DNS server address, which must be in the format described in **inet_pton**(3). This option may be specified more than once. This setting is read by **systemd-resolved.service**(8).

Domains=
A 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 host names (host names containing no dots) to become fully qualified domain names (FQDNs). If a single−label host name 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 host names 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. This option may be specified more than once. This setting is read by **systemd-**

timesyncd.service(8).

IPForward=

Configures IP packet forwarding for the system. If enabled, incoming packets on any network interface will be forwarded to any other interfaces according to the routing table. Takes a boolean, or the values "ipv4" or "ipv6", which only enable IP packet forwarding for the specified address family. This controls the net.ipv4.ip_forward and net.ipv6.conf.all.forwarding sysctl options of the network interface (see **ip−sysctl.txt**[6] for details about sysctl options). Defaults to "no".

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

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

IPMasquerade=

Configures IP masquerading for the network interface. If enabled, packets forwarded from the network interface will be appear as coming from the local host. Takes a boolean argument. Implies *IPForward=ipv4*. Defaults to "no".

IPv6PrivacyExtensions=

Configures use of stateless temporary addresses that change over time (see **RFC 4941**[7], Privacy Extensions for Stateless Address Autoconfiguration in IPv6). Takes a boolean or the special values "prefer−public" and "kernel". When true, enables the privacy extensions and prefers temporary addresses over public addresses. When "prefer−public", enables the privacy extensions, but prefers public addresses over temporary addresses. When false, the privacy extensions remain disabled. When "kernel", the kernel's default setting will be left in place. Defaults to "no".

IPv6AcceptRA=

Takes a boolean. Controls IPv6 Router Advertisement (RA) reception support for the interface. If true, RAs are accepted; if false, RAs are ignored, independently of the local forwarding state. If unset, the kernel's default is used, and RAs are accepted only when local forwarding is disabled for that interface. 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.

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

Also see **ip−sysctl.txt**[6] 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 if this option is enabled a userspace implementation of the IPv6 RA protocol is used, and the kernel's own implementation remains disabled, since ‘networkd‘ needs to know all details supplied in the advertisements, and these are not available from the kernel if the kernel's own implemenation is used.

IPv6DuplicateAddressDetection=

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

IPv6HopLimit=

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

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

IPv6ProxyNDP=
: Takes a boolean. Configures proxy NDP for IPv6. Proxy NDP (Neighbor Discovery Protocol) is a technique for IPv6 to allow routing of addresses to a different destination when peers expect them to be present on a certain physical link. In this case a router answers Neighbour Advertisement messages intended for another machine by offering its own MAC address as destination. Unlike proxy ARP for IPv4, it is not enabled globally, but will only send Neighbour Advertisement messages for addresses in the IPv6 neighbor proxy table, which can also be shown by **ip −6 neighbour show proxy**. systemd−networkd will control the per−interface ‘proxy_ndp‘ switch for each configured interface depending on this option. When unset, the kernel's default will be used.

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

IPv6PrefixDelegation=
: Whether to enable or disable Router Advertisement sending on a link. Allowed values are "static" which distributes prefixes as defined in the "[IPv6PrefixDelegation]" and any "[IPv6Prefix]" sections, "dhcpv6" which requests prefixes using a DHCPv6 client configured for another link and any values configured in the "[IPv6PrefixDelegation]" section while ignoring all static prefix configuration sections, "yes" which uses both static configuration and DHCPv6, and "false" which turns off IPv6 prefix delegation altogether. Defaults to "false". See the "[IPv6PrefixDelegation]" and the "[IPv6Prefix]" sections for more configuration options.

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.

Bridge=
: The name of the bridge to add the link to. See **systemd.netdev**(5).

Bond=
: The name of the bond to add the link to. See **systemd.netdev**(5).

VRF=
: The name of the VRF to add the link to. See **systemd.netdev**(5).

VLAN=
: The name of a VLAN to create on the link. See **systemd.netdev**(5). This option may be specified more than once.

IPVLAN=
: The name of a IPVLAN to create on the link. See **systemd.netdev**(5). This option may be specified more than once.

MACVLAN=
: The name of a MACVLAN to create on the link. See **systemd.netdev**(5). This option may be specified more than once.

VXLAN=
: The name of a VXLAN to create on the link. See **systemd.netdev**(5). This option may be specified more than once.

Tunnel=
: The name of a Tunnel to create on the link. See **systemd.netdev**(5). This option may be specified more than once.

ActiveSlave=
: Takes a boolean. Specifies the new active slave. The "ActiveSlave=" option is only valid for following

modes: "active−backup", "balance−alb" and "balance−tlb". Defaults to false.

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

ConfigureWithoutCarrier=
: Takes a boolean. Allows networkd to configure a specific link even if it has no carrier. Defaults to false.

IgnoreCarrierLoss=
: A boolean. Allows networkd to retain both the static and dynamic configuration of the interface even if its carrier is lost. Defaults to false.

KeepConfiguration=
: Takes a boolean or one of "static", "dhcp−on−stop", "dhcp". When "static", **systemd−networkd** will not drop static addresses and routes on starting up process. When set to "dhcp−on−stop", **systemd−networkd** will not drop addresses and routes on stopping the daemon. When "dhcp", the addresses and routes provided by a DHCP server will never be dropped even if the DHCP lease expires. This is contrary to the DHCP specification, but may be the best choice if, e.g., the root filesystem relies on this connection. The setting "dhcp" implies "dhcp−on−stop", and "yes" implies "dhcp" and "static". Defaults to "dhcp−on−stop".

[ADDRESS] SECTION OPTIONS

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

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

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

Broadcast=
: The broadcast address, which must be in the format described in **inet_pton**(3). This key only applies to IPv4 addresses. If it is not given, it is derived from the *Address=* key.

Label=
: An address label.

PreferredLifetime=
: Allows the default "preferred lifetime" of the address to be overridden. Only three settings are accepted: "forever" or "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", "link" or "host" or an unsigned integer ranges 0 to 255. Defaults to "global".

HomeAddress=
: Takes a boolean. Designates this address the "home address" as defined in **RFC 6275**[9]. Supported only on IPv6. Defaults to false.

DuplicateAddressDetection=
: Takes a boolean. Do not perform Duplicate Address Detection **RFC 4862**[10] when adding this address. Supported only on IPv6. Defaults to false.

ManageTemporaryAddress=
: Takes a boolean. If true the kernel manage temporary addresses created from this one as template on behalf of Privacy Extensions **RFC 3041**[11]. 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 to use privacy extensions in a manually configured network, just like if stateless auto−configuration was active. Defaults to false.

PrefixRoute=
: Takes a boolean. When adding or modifying an IPv6 address, the userspace application needs a way to suppress adding a prefix route. This is for example relevant together with IFA_F_MANAGERTEMPADDR, where userspace creates autoconf generated addresses, but depending on on−link, no route for the prefix should be added. Defaults to false.

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

[NEIGHBOR] SECTION OPTIONS

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

Address=
: The IP address of the neighbor.

MACAddress=
: The hardware 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**[12]. Precedence is managed by userspace, and only the label itself is stored in the kernel

Label=
: The label for the prefix (an unsigned integer) ranges 0 to 4294967294. 0xffffffff is reserved. This key is mandatory.

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

[ROUTINGPOLICYRULE] SECTION OPTIONS

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

TypeOfService=
: Specifies the type of service to match a number between 0 to 255.

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

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

FirewallMark=
: Specifies the iptables firewall mark value to match (a number between 1 and 4294967295).

Table=

Specifies the routing table identifier to lookup if the rule selector matches. The table identifier for a route (a number between 1 and 4294967295).

Priority=
: Specifies the priority of this rule. *Priority=* is an unsigned integer. Higher number means lower priority, and rules get processed in order of increasing number.

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

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

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

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

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

InvertRule=
: A boolean. Specifies wheather the rule to be inverted. Defaults to false.

[ROUTE] SECTION OPTIONS

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

Gateway=
: As in the "[Network]" section.

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., the kernel does not need to check if the gateway is 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 (an unsigned integer).

IPv6Preference=
: Specifies the route preference as defined in **RFC4191**[13] 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 route, which can be "global", "link" or "host". Defaults to "global".

PreferredSource=

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

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

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

Type=
Specifies the type for the route. 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 data bytes will be sent during the initial burst of data. Takes a size in bytes between 1 and 4294967295 (2^32 − 1). The usual suffixes K, M, G are supported and are understood to the base of 1024. When unset, the kernel's default will be used.

InitialAdvertisedReceiveWindow=
The TCP initial advertised receive window is the amount of receive data (in bytes) that can initally 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 size in bytes between 1 and 4294967295 (2^32 − 1). The usual suffixes K, M, G are supported and are understood to the base of 1024. When unset, the kernel's default will be used.

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

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

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

[DHCP] SECTION OPTIONS

The "[DHCP]" section configures the DHCPv4 and DHCP6 client, if it is enabled with the *DHCP=* setting described above:

UseDNS=
When true (the default), the DNS servers received from the DHCP server will be used and take precedence over any statically configured ones.

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

UseNTP=
When true (the default), the NTP servers received from the DHCP server will be used by systemd−timesyncd and take precedence over any statically configured ones.

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.

Anonymize=
Takes a boolean. When true, the options sent to the DHCP server will follow the **RFC 7844**[14] (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)).

Note that this configuration will overwrite others. In concrete, the following variables will be ignored: *SendHostname=*, *ClientIdentifier=*, *UseRoutes=*, *SendHostname=*, *UseMTU=*, *VendorClassIdentifier=*, *UseTimezone=*.

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.

SendHostname=
When true (the default), the machine's hostname will be sent to the DHCP server. Note that the machine's 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 is set to true.

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

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.

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

It is recommended to enable this option only on trusted networks, as setting this affects resolution of all host names, 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".

UseTimezone=
When true, the timezone received from the DHCP server will be set as timezone of the local system. Defaults to "no".

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

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

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

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

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

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

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.

RouteMetric=
: Set the routing metric for routes specified by the DHCP server.

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

  When used in combination with *VRF=* the VRF's routing table is used unless this parameter is specified.

ListenPort=
: Allow setting custom port for the DHCP client to listen on.

RapidCommit=
: Takes a boolean. The DHCPv6 client can obtain configuration parameters from a DHCPv6 server through a rapid two−message exchange (solicit and reply). When the rapid commit option is enabled by both the DHCPv6 client and the DHCPv6 server, the two−message exchange is used, rather than the default four−method exchange (solicit, advertise, request, and reply). The two−message exchange provides faster client configuration and is beneficial in environments in which networks are under a heavy load. See **RFC 3315**[15] for details. Defaults to true.

ForceDHCPv6PDOtherInformation=
: Takes a boolean that enforces DHCPv6 stateful mode when the 'Other information' bit is set in Router Advertisement messages. By default setting only the 'O' bit in Router Advertisements makes DHCPv6 request network information in a stateless manner using a two−message Information Request and Information Reply message exchange. **RFC 7084**[16], requirement WPD−4, updates this behavior for a Customer Edge router so that stateful DHCPv6 Prefix Delegation is also requested when only the 'O' bit is set in Router Advertisements. This option enables such a CE behavior as it is impossible to automatically distinguish the intention of the 'O' bit otherwise. By default this option is set to 'false',

enable it if no prefixes are delegated when the device should be acting as a CE router.

[IPV6ACCEPTRA] SECTION OPTIONS

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

UseDNS=

When true (the default), the DNS servers received in the Router Advertisement will be used and take precedence over any statically configured ones.

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

UseDomains=

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

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

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

RouteTable=num

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

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 take precedence over any statically configured ones.

[DHCPSERVER] SECTION OPTIONS

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

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

EmitNTP=, *NTP=*
Similar to the *EmitDNS=* and *DNS=* settings described above, these settings configure whether and what NTP server information shall be emitted as part of the DHCP lease. The same syntax, propagation semantics and defaults apply as for *EmitDNS=* and *DNS=*.

EmitRouter=
Similar to the *EmitDNS=* setting described above, this setting configures whether the DHCP lease should contain the router option. The same syntax, propagation semantics and defaults apply as for *EmitDNS=*.

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

[IPV6PREFIXDELEGATION] SECTION OPTIONS

The "[IPv6PrefixDelegation]" section contains settings for sending IPv6 Router Advertisements and whether to act as a router, if enabled via the *IPv6PrefixDelegation=* option described above. IPv6 network prefixes are defined with one or more "[IPv6Prefix]" 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. If set, this host also announces itself in Router Advertisements as an IPv6 router for the network link. When unset, the host is not acting as a router.

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**[13] for details. Defaults to "medium".

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

EmitDomains=, *Domains=*
A list of DNS search domains distributed via Router Advertisement messages when *EmitDomains=* is true. If *Domains=* is empty, DNS search domains are read from the "[Network]" section. If the

"[Network]" section does not contain any DNS search domains either, DNS search domains from the uplink with the highest priority default route are used. When *EmitDomains=* is false, no DNS search domain information is sent in Router Advertisement messages. *EmitDomains=* defaults to true.

DNSLifetimeSec=

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

[IPV6PREFIX] SECTION OPTIONS

One or more "[IPv6Prefix]" sections contain the IPv6 prefixes that are announced via Router Advertisements. See **RFC 4861**[17] for further details.

AddressAutoconfiguration=, *OnLink=*

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

Prefix=

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

PreferredLifetimeSec=, *ValidLifetimeSec=*

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

[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, and the bridge will not forward traffic back out of the receiving port. When unset, the kernel's default will be used.

UseBPDU=

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

FastLeave=

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

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

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

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

[BRIDGEFDB] SECTION OPTIONS

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

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

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.

[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.

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

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.

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.

[BRIDGEVLAN] SECTION OPTIONS

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

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

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

PVID=

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

EXAMPLES

Example 1. Static network configuration

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

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

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

Example 2. DHCP on ethernet links

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

[Network]
DHCP=yes
```

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

Example 3. A bridge with two enslaved links

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

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

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

[Network]
Bridge=bridge0

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

[Network]
Bridge=bridge0
```

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

Example 4.

```
# /etc/systemd/network/20-bridge-slave-interface-vlan.network
[Match]
Name=enp2s0

[Network]
Bridge=bridge0

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

[BridgeVLAN]
VLAN=100-200

[BridgeVLAN]
EgressUntagged=300-400
```

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

Example 5. 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 6. 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 7. Virtual Routing and Forwarding (VRF)

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

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

[Network]
VRF=vrf1
```

Example 8. MacVTap

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

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

```
Name=enp0s25

[Network]
MACVTAP=macvtap−test
```

SEE ALSO

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

NOTES

1. Link-Local Multicast Name Resolution
   https://tools.ietf.org/html/rfc4795
2. Multicast DNS
   https://tools.ietf.org/html/rfc6762
3. DNS-over-TLS
   https://tools.ietf.org/html/rfc7858
4. DNSSEC
   https://tools.ietf.org/html/rfc4033
5. IEEE 802.1AB-2016
   https://standards.ieee.org/findstds/standard/802.1AB-2016.html
6. ip-sysctl.txt
   https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt
7. RFC 4941
   https://tools.ietf.org/html/rfc4941
8. RFC 1027
   https://tools.ietf.org/html/rfc1027
9. RFC 6275
   https://tools.ietf.org/html/rfc6275
10. RFC 4862
    https://tools.ietf.org/html/rfc4862
11. RFC 3041
    https://tools.ietf.org/html/rfc3041
12. RFC 3484
    https://tools.ietf.org/html/rfc3484
13. RFC4191
    https://tools.ietf.org/html/rfc4191
14. RFC 7844
    https://tools.ietf.org/html/rfc7844
15. RFC 3315
    https://tools.ietf.org/html/rfc3315#section-17.2.1
16. RFC 7084
    https://tools.ietf.org/html/rfc7084
17. RFC 4861
    https://tools.ietf.org/html/rfc4861