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 directory /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 in /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 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. (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.)

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=
: The hardware address of the interface (use full colon-delimited hexadecimal, e.g., 01:23:45:67:89:ab).

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 "DRIVER" of its parent device, or if that is not set the driver as exposed by "ethtool -i" of the device itself.

Type=
: A whitespace-separated list of shell-style globs matching the device type, as exposed by the udev property "DEVTYPE".

Name=
: A whitespace-separated list of shell-style globs matching the device name, as exposed by the udev

property "INTERFACE".

Host=
: Matches against the hostname or machine ID of the host. See "ConditionHost=" in **systemd.unit(5)** for details.

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.

KernelCommandLine=
: Checks whether a specific kernel command line option is set (or if prefixed with the exclamation mark unset). See "ConditionKernelCommandLine=" in **systemd.unit(5)** for details.

Architecture=
: Checks whether the system is running on a specific architecture. See "ConditionArchitecture=" in **systemd.unit(5)** for details.

[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=
: A boolean. Enables or disables the ARP (low-level Address Resolution Protocol) for this interface. Defaults to unset, which means that the kernel 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.

[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".

  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 is not used for name resolution. See option **UseDomains=** below.

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

DHCPServer=
: A boolean. Enables DHCPv4 server support. 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". Defaults to "ipv6".

IPv4LLRoute=
: A boolean. When 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=
: 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=
: 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)**.

DNSSEC=
: A boolean or "allow-downgrade". When true, enables **DNSSEC**[3] DNS validation support on the link. When set to "allow-downgrade", compatibility with non-DNSSEC capable networks is increased, by automatically turning off DNSEC 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-2009**[4]. 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. 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 fc00::/7 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.

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 either a boolean argument, 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**[5] 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**[6], 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=
: Enable or disable IPv6 Router Advertisement (RA) reception support for the interface. Takes a boolean parameter. If true, RAs are accepted; if false, RAs are ignored, independently of the local forwarding state. When not set, the kernel 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**[5] 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**.

IPv6DuplicateAddressDetection=
: Configures the amount of IPv6 Duplicate Address Detection (DAD) probes to send. Defaults to unset.

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. Defaults to unset.

ProxyARP=
: A boolean. Configures proxy ARP. 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**[7]. Defaults to unset.

Bridge=
: The name of the bridge to add the link to.

Bond=
: The name of the bond to add the link to.

VRF=
: The name of the VRF to add the link to.

VLAN=
: The name of a VLAN to create on the link. This option may be specified more than once.

MACVLAN=

The name of a MACVLAN to create on the link. This option may be specified more than once.

VXLAN=
The name of a VXLAN to create on the link. This option may be specified more than once.

Tunnel=
The name of a Tunnel to create on the link. This option may be specified more than once.

[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.

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.

HomeAddress=
Takes a boolean argument. Designates this address the "home address" as defined in **RFC 6275**[8]. Supported only on IPv6. Defaults to false.

DuplicateAddressDetection=
Takes a boolean argument. Do not perform Duplicate Address Detection **RFC 4862**[9] when adding this address. Supported only on IPv6. Defaults to false.

ManageTemporaryAddress=
Takes a boolean argument. If true the kernel manage temporary addresses created from this one as template on behalf of Privacy Extensions **RFC 3041**[10]. 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 argument. 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 argument. 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".

[ROUTE] SECTION OPTIONS

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

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

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).

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*.

[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. Defaults to false.

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

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.

UseDomains=
: Takes a boolean argument, 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 false.

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.

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

CriticalConnection=
: When true, the connection will never be torn down even if the DHCP lease expires. This is contrary to the DHCP specification, but may be the best choice if, say, the root filesystem relies on this connection. Defaults to false.

ClientIdentifier=
: The DHCPv4 client identifier to use. Either "mac" to use the MAC address of the link or "duid" (the default, see below) to use an RFC4361-compliant Client ID.

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

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*.

[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 argument, 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 false.

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*.

[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=*
: Configures whether the DHCP leases handed out to clients shall contain DNS server information. The *EmitDNS=* setting takes a boolean argument and 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=*
: Configures whether the DHCP leases handed out to clients shall contain timezone information. The *EmitTimezone=* setting takes a boolean argument and 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.

[BRIDGE] SECTION OPTIONS

The "[Bridge]" section accepts the following keys.

UnicastFlood=
: A boolean. Controls whether the bridge should flood traffic for which an FDB entry is missing and the destination is unknown through this port. Defaults to on.

HairPin=
: A boolean. Configures whether traffic may be sent back out of the port on which it was received. By default, this flag is false, and the bridge will not forward traffic back out of the receiving port.

UseBPDU=
: A boolean. Configures whether STP Bridge Protocol Data Units will be processed by the bridge port. Defaults to yes.

FastLeave=
: 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. Defaults to off.

AllowPortToBeRoot=
: A boolean. Configures whether a given port is allowed to become a root port. Only used when STP is enabled on the bridge. Defaults to on.

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.

[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 info is appended to the new static MAC table entry.

[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.

EXAMPLE

Example 1. /etc/systemd/network/50-static.network

```
[Match]
Name=enp2s0

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

Example 2. /etc/systemd/network/80-dhcp.network

```
[Match]
Name=en*

[Network]
DHCP=yes
```

Example 3. /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
```

Example 4. /etc/systemd/network/25-bridge-slave-interface.network

```
[Match]
Name=enp2s0

[Network]
Bridge=bridge0
```

Example 5. /etc/systemd/network/25-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
```

Example 6. /etc/systemd/network/25-ipip.network

```
[Match]
Name=em1

[Network]
Tunnel=ipip-tun
```

Example 7. /etc/systemd/network/25-sit.network

```
[Match]
```

```
Name=em1

[Network]
Tunnel=sit-tun
```

Example 8. /etc/systemd/network/25-gre.network

```
[Match]
Name=em1

[Network]
Tunnel=gre-tun
```

Example 9. /etc/systemd/network/25-vti.network

```
[Match]
Name=em1

[Network]
Tunnel=vti-tun
```

Example 10. /etc/systemd/network/25-bond.network

```
[Match]
Name=bond1

[Network]
DHCP=yes
```

Example 11. /etc/systemd/network/25-vrf.network

Add the bond1 interface to the VRF master interface vrf-test. This will redirect routes generated on this interface to be within the routing table defined during VRF creation. Traffic won't be redirected towards the VRFs routing table unless specific ip-rules are added.

```
[Match]
Name=bond1

[Network]
VRF=vrf-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. DNSSEC
   https://tools.ietf.org/html/rfc4033

4. IEEE 802.1AB-2009
   http://standards.ieee.org/getieee802/download/802.1AB-2009.pdf

5. ip-sysctl.txt
   https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt

6. RFC 4941
   https://tools.ietf.org/html/rfc4941

7. RFC 1027
   https://tools.ietf.org/html/rfc1027

8. RFC 6275
   https://tools.ietf.org/html/rfc6275
9. RFC 4862
   https://tools.ietf.org/html/rfc4862
10. RFC 3041
    https://tools.ietf.org/html/rfc3041