NAME

resolved.conf, resolved.conf.d − Network Name Resolution configuration files

SYNOPSIS

/etc/systemd/resolved.conf

/etc/systemd/resolved.conf.d/*.conf

/run/systemd/resolved.conf.d/*.conf

/usr/lib/systemd/resolved.conf.d/*.conf

DESCRIPTION

These configuration files control local DNS and LLMNR name resolution.

CONFIGURATION DIRECTORIES AND PRECEDENCE

The default configuration is defined during compilation, so a configuration file is only needed when it is necessary to deviate from those defaults. By default, the configuration file in /etc/systemd/ contains commented out entries showing the defaults as a guide to the administrator. This file can be edited to create local overrides.

When packages need to customize the configuration, they can install configuration snippets in /usr/lib/systemd/*.conf.d/ or /usr/local/lib/systemd/*.conf.d/. Files in /etc/ are reserved for the local administrator, who may use this logic to override the configuration files installed by vendor packages. The main configuration file is read before any of the configuration directories, and has the lowest precedence; entries in a file in any configuration directory override entries in the single configuration file. Files in the *.conf.d/ configuration subdirectories are sorted by their filename in lexicographic order, regardless of which of the subdirectories they reside in. When multiple files specify the same option, for options which accept just a single value, the entry in the file with the lexicographically latest name takes precedence. For options which accept a list of values, entries are collected as they occur in files sorted lexicographically. It is recommended to prefix all filenames in those subdirectories with a two−digit number and a dash, to simplify the ordering of the files.

To disable a configuration file supplied by the vendor, the recommended way is to place a symlink to /dev/null in the configuration directory in /etc/, with the same filename as the vendor configuration file.

OPTIONS

The following options are available in the "[Resolve]" section:

DNS=

A space−separated list of IPv4 and IPv6 addresses to use as system DNS servers. DNS requests are sent to one of the listed DNS servers in parallel to suitable per−link DNS servers acquired from **systemd-networkd.service**(8) or set at runtime by external applications. For compatibility reasons, if this setting is not specified, the DNS servers listed in /etc/resolv.conf are used instead, if that file exists and any servers are configured in it. This setting defaults to the empty list.

FallbackDNS=

A space−separated list of IPv4 and IPv6 addresses to use as the fallback DNS servers. Any per−link DNS servers obtained from **systemd-networkd.service**(8) take precedence over this setting, as do any servers set via *DNS*= above or /etc/resolv.conf. This setting is hence only used if no other DNS server information is known. If this option is not given, a compiled−in list of DNS servers is used instead.

Domains=

A space−separated list of domains. These domains are used as search suffixes when resolving single−label host names (domain names which contain no dot), in order to qualify them into fully−qualified domain names (FQDNs). Search domains are strictly processed in the order they are specified, until the name with the suffix appended is found. For compatibility reasons, if this setting is not specified, the search domains listed in /etc/resolv.conf are used instead, if that file exists and any domains are configured in it. This setting defaults to the empty list.

Specified domain names may optionally be prefixed with "~". In this case they do not define a search path, but preferably direct DNS queries for the indicated domains to the DNS servers configured with

the system *DNS*= setting (see above), in case additional, suitable per−link DNS servers are known. If no per−link DNS servers are known using the "~" syntax has no effect. Use the construct "~." (which is composed of "~" to indicate a routing domain and "." to indicate the DNS root domain that is the implied suffix of all DNS domains) to use the system DNS server defined with *DNS*= preferably for all domains.

LLMNR=
Takes a boolean argument or "resolve". Controls Link−Local Multicast Name Resolution support (**RFC 4795**[1]) on the local host. If true, enables full LLMNR responder and resolver support. If false, disables both. If set to "resolve", only resolution support is enabled, but responding is disabled. Note that **systemd-networkd.service**(8) also maintains per−link LLMNR settings. LLMNR will be enabled on a link only if the per−link and the global setting is on.

MulticastDNS=
Takes a boolean argument or "resolve". Controls Multicast DNS support (**RFC 6762**[2]) on the local host. If true, enables full Multicast DNS responder and resolver support. If false, disables both. If set to "resolve", only resolution support is enabled, but responding is disabled. Note that **systemd-networkd.service**(8) also maintains per−link Multicast DNS settings. Multicast DNS will be enabled on a link only if the per−link and the global setting is on.

DNSSEC=
Takes a boolean argument or "allow−downgrade". If true all DNS lookups are DNSSEC−validated locally (excluding LLMNR and Multicast DNS). If the response to a lookup request is detected to be invalid a lookup failure is returned to applications. Note that this mode requires a DNS server that supports DNSSEC. If the DNS server does not properly support DNSSEC all validations will fail. If set to "allow−downgrade" DNSSEC validation is attempted, but if the server does not support DNSSEC properly, DNSSEC mode is automatically disabled. Note that this mode makes DNSSEC validation vulnerable to "downgrade" attacks, where an attacker might be able to trigger a downgrade to non−DNSSEC mode by synthesizing a DNS response that suggests DNSSEC was not supported. If set to false, DNS lookups are not DNSSEC validated.

Note that DNSSEC validation requires retrieval of additional DNS data, and thus results in a small DNS look−up time penalty.

DNSSEC requires knowledge of "trust anchors" to prove data integrity. The trust anchor for the Internet root domain is built into the resolver, additional trust anchors may be defined with **dnssec-trust-anchors.d**(5). Trust anchors may change at regular intervals, and old trust anchors may be revoked. In such a case DNSSEC validation is not possible until new trust anchors are configured locally or the resolver software package is updated with the new root trust anchor. In effect, when the built−in trust anchor is revoked and *DNSSEC*= is true, all further lookups will fail, as it cannot be proved anymore whether lookups are correctly signed, or validly unsigned. If *DNSSEC*= is set to "allow−downgrade" the resolver will automatically turn off DNSSEC validation in such a case.

Client programs looking up DNS data will be informed whether lookups could be verified using DNSSEC, or whether the returned data could not be verified (either because the data was found unsigned in the DNS, or the DNS server did not support DNSSEC or no appropriate trust anchors were known). In the latter case it is assumed that client programs employ a secondary scheme to validate the returned DNS data, should this be required.

It is recommended to set *DNSSEC*= to true on systems where it is known that the DNS server supports DNSSEC correctly, and where software or trust anchor updates happen regularly. On other systems it is recommended to set *DNSSEC*= to "allow−downgrade".

In addition to this global DNSSEC setting **systemd-networkd.service**(8) also maintains per−link DNSSEC settings. For system DNS servers (see above), only the global DNSSEC setting is in effect. For per−link DNS servers the per−link setting is in effect, unless it is unset in which case the global

setting is used instead.

Site−private DNS zones generally conflict with DNSSEC operation, unless a negative (if the private zone is not signed) or positive (if the private zone is signed) trust anchor is configured for them. If "allow−downgrade" mode is selected, it is attempted to detect site−private DNS zones using top−level domains (TLDs) that are not known by the DNS root server. This logic does not work in all private zone setups.

Defaults to "allow−downgrade"

DNSOverTLS=

Takes false or "opportunistic". When set to "opportunistic" DNS request are attempted to send encrypted with DNS−over−TLS. If the DNS server does not support TLS, DNS−over−TLS is disabled. Note that this mode makes DNS−over−TLS vulnerable to "downgrade" attacks, where an attacker might be able to trigger a downgrade to non−encrypted mode by synthesizing a response that suggests DNS−over−TLS was not supported. If set to false, DNS lookups are send over UDP.

Note that DNS−over−TLS requires additional data to be send for setting up an encrypted connection, and thus results in a small DNS look−up time penalty.

Note as the resolver is not capable of authenticating the server, it is vulnerable for "man−in−the−middle" attacks.

In addition to this global DNSOverTLS setting **systemd-networkd.service**(8) also maintains per−link DNSOverTLS settings. For system DNS servers (see above), only the global DNSOverTLS setting is in effect. For per−link DNS servers the per−link setting is in effect, unless it is unset in which case the global setting is used instead.

Defaults to off.

Cache=

Takes a boolean or "no−negative" as argument. If "yes" (the default), resolving a domain name which already got queried earlier will return the previous result as long as it is still valid, and thus does not result in a new network request. Be aware that turning off caching comes at a performance penalty, which is particularly high when DNSSEC is used.

If "no−negative", only positive answers are cached.

Note that caching is turned off implicitly if the configured DNS server is on a host−local IP address (such as 127.0.0.1 or ::1), in order to avoid duplicate local caching.

DNSStubListener=

Takes a boolean argument or one of "udp" and "tcp". If "udp", a DNS stub resolver will listen for UDP requests on address 127.0.0.53 port 53. If "tcp", the stub will listen for TCP requests on the same address and port. If "yes" (the default), the stub listens for both UDP and TCP requests. If "no", the stub listener is disabled.

Note that the DNS stub listener is turned off implicitly when its listening address and port are already in use.

ReadEtcHosts=

Takes a boolean argument. If "yes" (the default), the DNS stub resolver will read /etc/hosts, and try to resolve hosts or address by using the entries in the file before sending query to DNS servers.

SEE ALSO

systemd(1), **systemd-resolved.service**(8), **systemd-networkd.service**(8), **dnssec-trust-anchors.d**(5), **resolv.conf**(4)

NOTES

1. RFC 4795
   https://tools.ietf.org/html/rfc4795
2. RFC 6762
   https://tools.ietf.org/html/rfc6762