NAME

systemd-resolved.service, systemd-resolved − Network Name Resolution manager

SYNOPSIS

systemd−resolved.service

/lib/systemd/systemd−resolved

DESCRIPTION

systemd−resolved is a system service that provides network name resolution to local applications. It implements a caching and validating DNS/DNSSEC stub resolver, as well as an LLMNR and MulticastDNS resolver and responder. Local applications may submit network name resolution requests via three interfaces:

- The native, fully−featured API **systemd−resolved** exposes on the bus, see **org.freedesktop.resolve1**(5) and **org.freedesktop.LogControl1**(5) for details. Usage of this API is generally recommended to clients as it is asynchronous and fully featured (for example, properly returns DNSSEC validation status and interface scope for addresses as necessary for supporting link−local networking).
- The glibc **getaddrinfo**(3) API as defined by **RFC3493**[1] and its related resolver functions, including **gethostbyname**(3). This API is widely supported, including beyond the Linux platform. In its current form it does not expose DNSSEC validation status information however, and is synchronous only. This API is backed by the glibc Name Service Switch (**nss**(5)). Usage of the glibc NSS module **nss-resolve**(8) is required in order to allow glibc's NSS resolver functions to resolve hostnames via **systemd−resolved**.
- Additionally, **systemd−resolved** provides a local DNS stub listener on IP address 127.0.0.53 on the local loopback interface. Programs issuing DNS requests directly, bypassing any local API may be directed to this stub, in order to connect them to **systemd−resolved**. Note however that it is strongly recommended that local programs use the glibc NSS or bus APIs instead (as described above), as various network resolution concepts (such as link−local addressing, or LLMNR Unicode domains) cannot be mapped to the unicast DNS protocol.

The DNS servers contacted are determined from the global settings in /etc/systemd/resolved.conf, the per−link static settings in /etc/systemd/network/*.network files (in case **systemd-networkd.service**(8) is used), the per−link dynamic settings received over DHCP, information provided via **resolvectl**(1), and any DNS server information made available by other system services. See **resolved.conf**(5) and **systemd.network**(5) for details about systemd's own configuration files for DNS servers. To improve compatibility, /etc/resolv.conf is read in order to discover configured system DNS servers, but only if it is not a symlink to /run/systemd/resolve/stub−resolv.conf, /usr/lib/systemd/resolv.conf or /run/systemd/resolve/resolv.conf (see below).

SYNTHETIC RECORDS

systemd−resolved synthesizes DNS resource records (RRs) for the following cases:

- The local, configured hostname is resolved to all locally configured IP addresses ordered by their scope, or — if none are configured — the IPv4 address 127.0.0.2 (which is on the local loopback interface) and the IPv6 address ::1 (which is the local host).
- The hostnames "localhost" and "localhost.localdomain" as well as any hostname ending in ".localhost" or ".localhost.localdomain" are resolved to the IP addresses 127.0.0.1 and ::1.
- The hostname "_gateway" is resolved to all current default routing gateway addresses, ordered by their metric. This assigns a stable hostname to the current gateway, useful for referencing it independently of the current network configuration state.
- The hostname "_outbound" is resolved to the local IPv4 and IPv6 addresses that are most likely used for communication with other hosts. This is determined by requesting a routing decision to the configured default gateways from the kernel and then using the local IP addresses selected by this decision. This hostname is only available if there is at least one local default gateway configured. This assigns a stable hostname to the local outbound IP addresses, useful for referencing them

independently of the current network configuration state.

- The mappings defined in /etc/hosts are resolved to their configured addresses and back, but they will not affect lookups for non−address types (like MX). Support for /etc/hosts may be disabled with *ReadEtcHosts=no*, see **resolved.conf**(5).

PROTOCOLS AND ROUTING

The lookup requests that systemd−resolved.service receives are routed to the available DNS servers, LLMNR, and MulticastDNS interfaces according to the following rules:

- Names for which synthetic records are generated (the local hostname, "localhost" and "localdomain", local gateway, as listed in the previous section) and addresses configured in /etc/hosts are never routed to the network and a reply is sent immediately.
- Single−label names are resolved using LLMNR on all local interfaces where LLMNR is enabled. Lookups for IPv4 addresses are only sent via LLMNR on IPv4, and lookups for IPv6 addresses are only sent via LLMNR on IPv6. Note that lookups for single−label synthesized names are not routed to LLMNR, MulticastDNS or unicast DNS.
- Queries for the address records (A and AAAA) of single−label non−synthesized names are resolved via unicast DNS using search domains. For any interface which defines search domains, such look−ups are routed to the servers defined for that interface, suffixed with each of those search domains. When global search domains are defined, such look−ups are routed to the global servers. For each search domain, queries are performed by suffixing the name with each of the search domains in turn. Additionally, lookup of single−label names via unicast DNS may be enabled with the *ResolveUnicastSingleLabel=yes* setting. The details of which servers are queried and how the final reply is chosen are described below. Note that this means that address queries for single−label names are never sent out to remote DNS servers by default, and resolution is only possible if search domains are defined.
- Multi−label names with the domain suffix ".local" are resolved using MulticastDNS on all local interfaces where MulticastDNS is enabled. As with LLMNR, IPv4 address lookups are sent via IPv4 and IPv6 address lookups are sent via IPv6.
- Queries for multi−label names are routed via unicast DNS on local interfaces that have a DNS server configured, plus the globally configured DNS servers if there are any. Which interfaces are used is determined by the routing logic based on search and route−only domains, described below. Note that by default, lookups for domains with the ".local" suffix are not routed to DNS servers, unless the domain is specified explicitly as routing or search domain for the DNS server and interface. This means that on networks where the ".local" domain is defined in a site−specific DNS server, explicit search or routing domains need to be configured to make lookups work within this DNS domain. Note that these days, it's generally recommended to avoid defining ".local" in a DNS server, as **RFC6762**[2] reserves this domain for exclusive MulticastDNS use.
- Address lookups (reverse lookups) are routed similarly to multi−label names, with the exception that addresses from the link−local address range are never routed to unicast DNS and are only resolved using LLMNR and MulticastDNS (when enabled).

If lookups are routed to multiple interfaces, the first successful response is returned (thus effectively merging the lookup zones on all matching interfaces). If the lookup failed on all interfaces, the last failing response is returned.

Routing of lookups is determined by the per−interface routing domains (search and route−only) and global search domains. See **systemd.network**(5) and **resolvectl**(1) for a description how those settings are set dynamically and the discussion of *Domains=* in **resolved.conf**(5) for a description of globally configured DNS settings.

The following query routing logic applies for unicast DNS lookups initiated by systemd−resolved.service:

- If a name to look up matches (that is: is equal to or has as suffix) any of the configured routing domains (search or route−only) of any link, or the globally configured DNS settings, "best matching" routing domain is determined: the matching one with the most labels. The query is then

sent to all DNS servers of any links or the globally configured DNS servers associated with this "best matching" routing domain. (Note that more than one link might have this same "best matching" routing domain configured, in which case the query is sent to all of them in parallel).

In case of single−label names, when search domains are defined, the same logic applies, except that the name is first suffixed by each of the search domains in turn. Note that this search logic doesn't apply to any names with at least one dot. Also see the discussion about compatibility with the traditional glibc resolver below.

- If a query does not match any configured routing domain (either per−link or global), it is sent to all DNS servers that are configured on links with the *DefaultRoute*= option set, as well as the globally configured DNS server.
- If there is no link configured as *DefaultRoute*= and no global DNS server configured, one of the compiled−in fallback DNS servers is used.
- Otherwise the unicast DNS query fails, as no suitable DNS servers can be determined.

The *DefaultRoute*= option is a boolean setting configurable with **resolvectl** or in .network files. If not set, it is implicitly determined based on the configured DNS domains for a link: if there's a route−only domain other than "~.", it defaults to false, otherwise to true.

Effectively this means: in order to support single−label non−synthesized names, define appropriate search domains. In order to preferably route all DNS queries not explicitly matched by routing domain configuration to a specific link, configure a "~." route−only domain on it. This will ensure that other links will not be considered for these queries (unless they too carry such a routing domain). In order to route all such DNS queries to a specific link only if no other link is preferred, set the *DefaultRoute*= option for the link to true and do not configure a "~." route−only domain on it. Finally, in order to ensure that a specific link never receives any DNS traffic not matching any of its configured routing domains, set the *DefaultRoute*= option for it to false.

See **org.freedesktop.resolve1**(5) for information about the D−Bus APIs systemd−resolved provides.

COMPATIBILITY WITH THE TRADITIONAL GLIBC STUB RESOLVER

This section provides a short summary of differences in the stub resolver implemented by **nss-resolve**(8) together with **systemd−resolved** and the traditional stub resolver implemented in nss−dns.

- Some names are always resolved internally (see Synthetic Records above). Traditionally they would be resolved by nss−files if provided in /etc/hosts. But note that the details of how a query is constructed are under the control of the client library. nss−dns will first try to resolve names using search domains and even if those queries are routed to systemd−resolved, it will send them out over the network using the usual rules for multi−label name routing [3].
- Single−label names are not resolved for A and AAAA records using unicast DNS (unless overridden with *ResolveUnicastSingleLabel*=, see **resolved.conf**(5)). This is similar to the **no−tld−query** option being set in **resolv.conf**(5).
- Search domains are not used for *suffixing* of multi−label names. (Search domains are nevertheless used for lookup *routing*, for names that were originally specified as single−label or multi−label.) Any name with at least one dot is always interpreted as a FQDN. nss−dns would resolve names both as relative (using search domains) and absolute FQDN names. Some names would be resolved as relative first, and after that query has failed, as absolute, while other names would be resolved in opposite order. The *ndots* option in /etc/resolv.conf was used to control how many dots the name needs to have to be resolved as relative first. This stub resolver does not implement this at all: multi−label names are only resolved as FQDNs.[4]
- This resolver has a notion of the special ".local" domain used for MulticastDNS, and will not route queries with that suffix to unicast DNS servers unless explicitly configured, see above. Also, reverse lookups for link−local addresses are not sent to unicast DNS servers.
- This resolver reads and caches /etc/hosts internally. (In other words, nss−resolve replaces nss−files in addition to nss−dns). Entries in /etc/hosts have highest priority.

- This resolver also implements LLMNR and MulticastDNS in addition to the classic unicast DNS protocol, and will resolve single−label names using LLMNR (when enabled) and names ending in ".local" using MulticastDNS (when enabled).
- Environment variables *$LOCALDOMAIN* and *$RES_OPTIONS* described in **resolv.conf**(5) are not supported currently.

/ETC/RESOLV.CONF

Four modes of handling /etc/resolv.conf (see **resolv.conf**(5)) are supported:

- **systemd−resolved** maintains the /run/systemd/resolve/stub−resolv.conf file for compatibility with traditional Linux programs. This file may be symlinked from /etc/resolv.conf. This file lists the 127.0.0.53 DNS stub (see above) as the only DNS server. It also contains a list of search domains that are in use by systemd−resolved. The list of search domains is always kept up−to−date. Note that /run/systemd/resolve/stub−resolv.conf should not be used directly by applications, but only through a symlink from /etc/resolv.conf. This file may be symlinked from /etc/resolv.conf in order to connect all local clients that bypass local DNS APIs to **systemd−resolved** with correct search domains settings. This mode of operation is recommended.
- A static file /usr/lib/systemd/resolv.conf is provided that lists the 127.0.0.53 DNS stub (see above) as only DNS server. This file may be symlinked from /etc/resolv.conf in order to connect all local clients that bypass local DNS APIs to **systemd−resolved**. This file does not contain any search domains.
- **systemd−resolved** maintains the /run/systemd/resolve/resolv.conf file for compatibility with traditional Linux programs. This file may be symlinked from /etc/resolv.conf and is always kept up−to−date, containing information about all known DNS servers. Note the file format's limitations: it does not know a concept of per−interface DNS servers and hence only contains system−wide DNS server definitions. Note that /run/systemd/resolve/resolv.conf should not be used directly by applications, but only through a symlink from /etc/resolv.conf. If this mode of operation is used local clients that bypass any local DNS API will also bypass **systemd−resolved** and will talk directly to the known DNS servers.
- Alternatively, /etc/resolv.conf may be managed by other packages, in which case **systemd−resolved** will read it for DNS configuration data. In this mode of operation **systemd−resolved** is consumer rather than provider of this configuration file.

Note that the selected mode of operation for this file is detected fully automatically, depending on whether /etc/resolv.conf is a symlink to /run/systemd/resolve/resolv.conf or lists 127.0.0.53 as DNS server.

SIGNALS

SIGUSR1

Upon reception of the **SIGUSR1** process signal **systemd−resolved** will dump the contents of all DNS resource record caches it maintains, as well as all feature level information it learnt about configured DNS servers into the system logs.

SIGUSR2

Upon reception of the **SIGUSR2** process signal **systemd−resolved** will flush all caches it maintains. Note that it should normally not be necessary to request this explicitly – except for debugging purposes – as **systemd−resolved** flushes the caches automatically anyway any time the host's network configuration changes. Sending this signal to **systemd−resolved** is equivalent to the **resolvectl flush−caches** command, however the latter is recommended since it operates in a synchronous way.

SIGRTMIN+1

Upon reception of the **SIGRTMIN+1** process signal **systemd−resolved** will forget everything it learnt about the configured DNS servers. Specifically any information about server feature support is flushed out, and the server feature probing logic is restarted on the next request, starting with the most fully featured level. Note that it should normally not be necessary to request this explicitly – except for debugging purposes – as **systemd−resolved** automatically forgets learnt information any time the DNS server configuration changes. Sending this signal to **systemd−resolved** is equivalent to the

resolvectl reset−server−features command, however the latter is recommended since it operates in a synchronous way.

SEE ALSO

systemd(1), **resolved.conf**(5), **dnssec-trust-anchors.d**(5), **nss-resolve**(8), **resolvectl**(1), **resolv.conf**(5), **hosts**(5), **systemd.network**(5), **systemd-networkd.service**(8)

NOTES

1. RFC3493
   https://tools.ietf.org/html/rfc3493
2. RFC6762
   https://tools.ietf.org/html/rfc6762
3. For example, if /etc/resolv.conf has

   nameserver 127.0.0.53
   search foobar.com barbar.com

   and we look up "localhost", nss-dns will send the following queries to systemd-resolved listening on 127.0.0.53:53: first "localhost.foobar.com", then "localhost.barbar.com", and finally "localhost". If (hopefully) the first two queries fail, systemd-resolved will synthesize an answer for the third query.

   When using nss-dns with any search domains, it is thus crucial to always configure nss-files with higher priority and provide mappings for names that should not be resolved using search domains.
4. There are currently more than 1500 top-level domain names defined, and new ones are added regularly, often using "attractive" names that are also likely to be used locally. Not looking up multi-label names in this fashion avoids fragility in both directions: a valid global name could be obscured by a local name, and resolution of a relative local name could suddenly break when a new top-level domain is created, or when a new subdomain of a top-level domain in registered. Resolving any given name as either relative or absolute avoids this ambiguity.