## 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 the **API Documentation**[1] 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**[2] 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 host names 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, user request made 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) 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 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).

## PROTOCOLS AND ROUTING

Lookup requests are routed to the available DNS servers, LLMNR and MulticastDNS interfaces according to the following rules:

- Lookups for the special hostname "localhost" are never routed to the network. (A few other, special domains are handled the same way.)
- Single−label names are routed to all local interfaces capable of IP multicasting, using the LLMNR protocol. Lookups for IPv4 addresses are only sent via LLMNR on IPv4, and lookups for IPv6 addresses are only sent via LLMNR on IPv6. Lookups for the locally configured host name and the "_gateway" host name are never routed to LLMNR.
- Multi−label names with the domain suffix ".local" are routed to all local interfaces capable of IP multicasting, using the MulticastDNS protocol. As with LLMNR IPv4 address lookups are sent via IPv4 and IPv6 address lookups are sent via IPv6.
- Other multi−label names are routed to all local interfaces that have a DNS server configured, plus the globally configured DNS server if there is one. Address lookups from the link−local address range are never routed to DNS. 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 within this DNS domain work. Note that today it's generally recommended to avoid defining ".local" in a DNS server, as **RFC6762**[3] reserves this domain for exclusive MulticastDNS use.

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 may be influenced by configuring per−interface domain names and other settings. See **systemd.network**(5) and **resolvectl**(1) for details. The following query routing logic applies for unicast DNS traffic:

- If a name to look up matches (that is: is equal to or has as suffix) any of the configured search or route−only domains of any link (or the globally configured DNS settings), the "best matching" search/route−only 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" search/route−only domain. (Note that more than one link might have this same "best matching" search/route−only domain configured, in which case the query is sent to all of them in parallel).
- If a query does not match any configured search/route−only domain (neither per−link nor global), it is sent to all DNS servers that are configured on links with the "DNS default route" option set, as well as the globally configured DNS server.
- If there is no link configured as "DNS default route" and no global DNS server configured, the compiled−in fallback DNS server is used.
- Otherwise the query is failed as no suitable DNS servers could be determined.

The "DNS default route" 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 any route−only domain (not matching "~.") it defaults to false, otherwise to true.

Effectively this means: in order to preferably route all DNS queries not explicitly matched by search/route−only domain configuration to a specific link, configure a "~." route−only domain on it. This will ensure that other links will not be considered for the queries (unless they too carry such a route−only domain). In order to route all such DNS queries to a specific link only in case no other link is preferable, then set the "DNS default route" 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 search/route−only domains, set the "DNS default route" option for it to false.

See the **resolved D−Bus API Documentation**[1] for information about the APIs systemd−resolved provides.

## /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. API Documentation
   https://www.freedesktop.org/wiki/Software/systemd/resolved
2. RFC3493
   https://tools.ietf.org/html/rfc3493
3. RFC6762
   https://tools.ietf.org/html/rfc6762