NAME

systemd.dnssd − DNS−SD configuration

SYNOPSIS

network_service.dnssd

DESCRIPTION

DNS−SD setup is performed by **systemd-resolved**(8).

The main network service file must have the extension .dnssd; other extensions are ignored.

The .dnssd files are read from the files located in the system network directories /usr/lib/systemd/dnssd and /usr/local/lib/systemd/dnssd, the volatile runtime network directory /run/systemd/dnssd and the local administration network directory /etc/systemd/dnssd. 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.

Along with the network service file foo.dnssd, a "drop−in" directory foo.dnssd.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/dnssd, drop−in ".d" directories can be placed in /usr/lib/systemd/dnssd or /run/systemd/dnssd directories. Drop−in files in /etc take precedence over those in /run which in turn take precedence over those in /usr/lib or /usr/local/lib. Drop−in files under any of these directories take precedence over the main network service file wherever located.

[SERVICE] SECTION OPTIONS

The network service file contains a "[Service]" section, which specifies a discoverable network service announced in a local network with Multicast DNS broadcasts.

Name=

An instance name of the network service as defined in the section 4.1.1 of **RFC 6763**[1], e.g. "webserver".

The option supports simple specifier expansion. The following expansions are understood:

Table 1. Specifiers available

Specifier	Meaning	Details
"%m"	Machine ID	The machine ID of the running system, formatted as string. See **machine-id**(5) for more information.
"%b"	Boot ID	The boot ID of the running system, formatted as string. See **random**(4) for more information.
"%H"	Host name	The hostname of the running system.
"%v"	Kernel release	Identical to **uname −r** output.

Type=

A type of the network service as defined in the section 4.1.2 of **RFC 6763**[1], e.g. "_http._tcp".

Port=

An IP port number of the network service.

Priority=
A priority number set in SRV resource records corresponding to the network service.

Weight=
A weight number set in SRV resource records corresponding to the network service.

TxtText=
A whitespace−separated list of arbitrary key/value pairs conveying additional information about the named service in the corresponding TXT resource record, e.g. "path=/portal/index.html". Keys and values can contain C−style escape sequences which get translated upon reading configuration files.

This option together with *TxtData=* may be specified more than once, in which case multiple TXT resource records will be created for the service. If the empty string is assigned to this option, the list is reset and all prior assignments will have no effect.

TxtData=
A whitespace−separated list of arbitrary key/value pairs conveying additional information about the named service in the corresponding TXT resource record where values are base64−encoded string representing any binary data, e.g. "data=YW55IGJpbmFyeSBkYXRhCg==". Keys can contain C−style escape sequences which get translated upon reading configuration files.

This option together with *TxtText=* may be specified more than once, in which case multiple TXT resource records will be created for the service. If the empty string is assigned to this option, the list is reset and all prior assignments will have no effect.

EXAMPLES

Example 1. HTTP service

```
# /etc/systemd/dnssd/http.dnssd
[Service]
Name=%H
Type=_http._tcp
Port=80
TxtText=path=/stats/index.html t=temperature_sensor
```

This makes the http server running on the host discoverable in the local network given MulticastDNS is enabled on the network interface.

Now the utility "resolvectl" should be able to resolve the service to the host's name:

```
$ resolvectl service meteo._http._tcp.local
meteo._http._tcp.local: meteo.local:80 [priority=0, weight=0]
                        169.254.208.106%senp0s21f0u2u4
                        fe80::213:3bff:fe49:8aa%senp0s21f0u2u4
                        path=/stats/index.html
                        t=temperature_sensor
                        (meteo/_http._tcp/local)

−− Information acquired via protocol mDNS/IPv6 in 4.0ms.
−− Data is authenticated: yes
```

"Avahi" running on a different host in the same local network should see the service as well:

```
$ avahi−browse −a −r
+ enp3s0 IPv6 meteo                                    Web Site             local
+ enp3s0 IPv4 meteo                                    Web Site             local
= enp3s0 IPv6 meteo                                    Web Site             local
  hostname = [meteo.local]
  address = [fe80::213:3bff:fe49:8aa]
```

```
  port = [80]
  txt = ["path=/stats/index.html" "t=temperature_sensor"]
= enp3s0 IPv4 meteo                    Web Site        local
  hostname = [meteo.local]
  address = [169.254.208.106]
  port = [80]
  txt = ["path=/stats/index.html" "t=temperature_sensor"]
```

SEE ALSO

systemd(1), **systemd-resolved.service**(8), **resolvectl**(1)

NOTES

1. RFC 6763
   https://tools.ietf.org/html/rfc6763