IPv6

In Arch Linux, IPv6 is enabled by default.

Contents

Related articles

IPv6 tunnel broker setup

Neighbor discovery

Pinging the multicast address `ff02::1` results in all hosts in link-local scope responding. An interface has to be specified:

```
$ ping ff02::1%eth0
```

After that, you can get a list of all the neighbors in the local network with:

```
$ ip -6 neigh
```

With a ping to the multicast address `ff02::2` only routers will respond.

If you add an option `-I your-global-ipv6`, link-local hosts will respond with their link-global scope addresses. The interface can be omitted in this case:

```
$ ping -I 2001:4f8:fff6::21 ff02::1
```

Stateless autoconfiguration (SLAAC)

The easiest way to acquire an IPv6 address as long as your network is configured is through *Stateless address autoconfiguration* (SLAAC for short). The address is automatically inferred from the prefix that your router advertises and requires neither further configuration nor specialized software such as a DHCP client.

For clients

If you are using **netctl** you just need to add the following line to your ethernet or wireless configuration.

```
IP6=stateless
```

If you are using **NetworkManager** then it automatically enables IPv6 addresses if there are advertisements for them in the network.

Please note that stateless autoconfiguration works on the condition that IPv6 icmp packets are allowed throughout the network. So for the client side the `ipv6-icmp` packets must be accepted. If you are using the **Simple stateful firewall**/**iptables** you only need to add:

```
-A INPUT -p ipv6-icmp -j ACCEPT
```

If you are using an other firewall frontend (ufw, shorewall, etc) consult their documentation on how to enable the `ipv6-icmp` packets.

For gateways

To properly hand out IPv6s to the network clients we will need to use an advertising daemon. The standard tool for this job is **radvd (https://www.archlinux.org/packages/?name=radvd)** and is available in **official repositories**. Configuration of radvd is fairly simple. Edit `/etc/radvd.conf` to include

```
# replace LAN with your LAN facing interface
interface LAN {
  AdvSendAdvert on;
  MinRtrAdvInterval 3;
  MaxRtrAdvInterval 10;
  prefix ::/64 {
    AdvOnLink on;
    AdvAutonomous on;
    AdvRouterAddr on;
  };
};
```

The above configuration will tell clients to autoconfigure themselves using addresses from the advertised /64 block. Please note that the above configuration advertises *all available prefixes* assigned to the LAN facing interface. If you want to limit the advertised prefixes instead of `::/64` use the desired prefix, eg `2001:DB8::/64`. The `prefix` block can be repeated many times for more prefixes.

The gateway must also allow the traffic of `ipv6-icmp` packets on all basic chains. For the **Simple stateful firewall**/**iptables** add:

```
-A INPUT -p ipv6-icmp -j ACCEPT
-A OUTPUT -p ipv6-icmp -j ACCEPT
-A FORWARD -p ipv6-icmp -j ACCEPT
```

Adjust accordingly for other firewall frontends and do not forget to enable `radvd.service`.

Privacy extensions

When a client acquires an address through SLAAC its IPv6 address is derived from the advertised prefix and the MAC address of the network interface of the client. This may raise security concerns as the MAC address of the computer can be easily derived by the IPv6 address. In order to tackle this problem the *IPv6 Privacy Extensions* standard (**RFC 4941 (https://tools.ietf.org/html/rfc4941)**) has been developed. With privacy extensions the kernel

generates a *temporary* address that is mangled from the original autoconfigured address. Private addresses are preferred when connecting to a remote server so the original address is hidden. To enable Privacy Extensions reproduce the following steps:

Add these lines to `/etc/sysctl.d/40-ipv6.conf`:

```
# Enable IPv6 Privacy Extensions
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
net.ipv6.conf.nic0.use_tempaddr = 2
...
net.ipv6.conf.nicN.use_tempaddr = 2
```

Where `nic0` to `nicN` are your **N**etwork **I**nterface **C**ards. You can find their names using the instructions in **Network configuration#Get current interface names**. The `all.use_tempaddr` or `default.use_tempaddr` parameters are not applied to nic's that already exist when the **sysctl** settings are executed.

After a reboot, at the latest, Privacy Extensions should be enabled.

dhcpcd

dhcpcd includes in its default configuration file since version 6.4.0 the option `slaac private`, which enables "Stable Private IPv6 Addresses instead of hardware based ones", implementing **RFC 7217 (https://tools.ietf.org/html/rfc7217)** (**commit (http://roy.marples.name/projects/dhcpcd/info/8aa9dab00dc72c453aeccbde885ecce27a3d81ff)**).

Therefore, it is not necessary to change anything, except if it is desired to change of IPv6 address more often than each time the system is connected to a new network. Set it to `slaac hwaddr` for a stable address.

NetworkManager

NetworkManager does not honour the settings placed in `/etc/sysctl.d/40-ipv6.conf`. This can be verified by running `$ ip -6 addr show interface` after rebooting: no `scope global temporary` address appears besides the regular one.

To enable IPv6 Privacy Extensions by default, add these lines to `/etc/NetworkManager/NetworkManager.conf`

/etc/NetworkManager/NetworkManager.conf

```
...
[connection]
ipv6.ip6-privacy=2
```

In order to enable IPv6 Privacy Extensions for individual NetworkManager-managed connections, edit the desired connection keyfile in `/etc/NetworkManager/system-connections/` and append to its `[ipv6]` section the key-value pair `ip6-privacy=2`:

/etc/NetworkManager/system-connections/*example_connection*

```
...
[ipv6]
method=auto
ip6-privacy=2
```

Note: Although it may seem the `scope global temporary` IPv6 address created by enabling Privacy Extensions never gets renewed (it never shifts to `deprecated` status at the term of its `valid_lft` lifetime), it is to be verified over a longer period of time that this address **does** indeed change.

systemd-networkd

Systemd-networkd also does not honor the settings `net.ipv6.conf.xxx.use_tempaddr` placed in `/etc/sysctl.d/40-ipv6.conf` unless the option `IPv6PrivacyExtensions` is set with the value `kernel` in the .network file(s). Other options for the IPv6 Privacy Extensions like

```
net.ipv6.conf.xxx.temp_prefered_lft
net.ipv6.conf.xxx.temp_valid_lft
```

are honored, however.

See systemd-networkd and `systemd.network(5)` (https://jlk.fjfi.cvut.cz/arch/manpages/man/systemd.network.5) for details.

connman

Add to `/var/lib/connman/settings` under the global section:

```
[global]
IPv6.privacy=preferred
```

Static address

Sometime using static address can improve security. For example, if your local router uses Neighbor Discovery or radvd (**RFC 2461 (http://www.apps.ietf.org/rfc/rfc2461.html)**), your interface will automatically be assigned an address based its MAC address (using IPv6's Stateless Autoconfiguration). This may be less than ideal for security since it allows a system to be tracked even if the network portion of the IP address changes.

To assign a static IP address using **netctl**, look at the example profile in `/etc/netctl/examples/ethernet-static`. The following lines are important:

```
...
# For IPv6 static address configuration
IP6=static
Address6=('1234:5678:9abc:def::1/64' '1234:3456::123/96')
Routes6=('abcd::1234')
Gateway6='1234:0:123::abcd'
```

Note: If you are connected IPv6-only, than you need to determine ipv6 dns server. For example

```
DNS=('6666:6666::1' '6666:6666::2')
```

If your provider did not give you ipv6 dns and you are not running your own, you can choose from **resolv.conf** article.

IPv6 and PPPoE

The standard tool for PPPoE, `pppd`, provides support for IPv6 on PPPoE as long as your ISP and your modem support it. Just add the following to `/etc/ppp/pppoe.conf`

```
+ipv6
```

If you are using **netctl** for PPPoE then just add the following to your netctl configuration instead

```
PPPoEIP6=yes
```

Prefix delegation (DHCPv6-PD)

Note: This section is targeted towards custom gateway configuration, not client machines. For standard market routers please consult the documentation of your router on how to enable prefix delegation.

Prefix delegation is a common IPv6 deployment technique used by many ISPs. It is a method of assigning a network prefix to a user site (ie. local network). A router can be configured to assign different network prefixes to various subnetworks. The ISP handles out a network prefix using DHCPv6 (usually a `/56` or `/64`) and a dhcp client assigns the prefixes to the local network. For a simple two interface gateway it practically assigns an IPv6 prefix to the interface connected to to the local network from an address acquired through the interface connected to WAN (or a pseudo-interface such as ppp).

With dibbler

Dibbler (http://klub.com.pl/dhcpv6/) is a portable DHCPv6 client a server which can be used for Prefix delegation. It is available in the **AUR** as **`dibbler (https://aur.archlinux.org/packages/dibbler/)`**[AUR].

If you are using `dibbler` edit `/etc/dibbler/client.conf`

```
log-mode short
log-level 7
# use the interface connected to your WAN
iface "WAN" {
  ia
  pd
}
```

Tip: Read `dibbler-client(8)` for more information.

With dhcpcd

dhcpcd apart from IPv4 dhcp support also provides a fairly complete implementation of the DHCPv6 client standard which includes DHCPv6-PD. If you are using `dhcpcd` edit `/etc/dhcpcd.conf`. You might already be using dhcpcd for IPv4 so just update your existing configuration.

```
duid
noipv6rs
waitip 6
# Uncomment this line if you are running dhcpcd for IPv6 only.
#ipv6only

# use the interface connected to WAN
interface WAN
ipv6rs
iaid 1
# use the interface connected to your LAN
ia_pd 1 LAN
#ia_pd 1/::/64 LAN/0/64
```

This configuration will ask for a prefix from WAN interface (`WAN`) and delegate it to the internal interface (`LAN`). In the event that a `/64` range is issued, you will need to use the 2nd `ia_pd instruction` that is commented out instead. It will also disable router solicitations on all interfaces except for the WAN interface (`WAN`).

Tip: Also read **dhcpcd(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/dhcpcd.8)** and **dhcpcd.conf(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/dhcpcd.conf.5)**.

With WIDE-DHCPv6

WIDE-DHCPv6 (http://wide-dhcpv6.sourceforge.net/) is an open-source implementation of Dynamic Host Configuration Protocol for IPv6 (DHCPv6) originally developed by the KAME project. It is available in the **AUR** as **`wide-dhcpv6 (https://aur.archlinux.org/packages/wide-dhcpv6/)`**[AUR].

If you are using `wide-dhcpv6` edit `/etc/wide-dhcpv6/dhcp6c.conf`

```
# use the interface connected to your WAN
interface WAN {
  send ia-pd 0;
};

id-assoc pd 0 {
  # use the interface connected to your LAN
  prefix-interface LAN {
    sla-id 1;
    sla-len 8;
  };
};
```

Note: `sla-len` should be set so that `(WAN-prefix) + (sla-len) = 64` . In this case it is set up for a `/56` prefix 56+8=64. For a `/64` prefix `sla-len` should be `0` .

To enable/start wide-dhcpv6 client use the following command. Change `WAN` with the interface that is connected to your WAN.

```
# systemctl enable/start dhcp6c@WAN.service
```

Tip: Read `dhcp6c(8)` and `dhcp6c.conf(5)` for more information.

IPv6 on Comcast

`dhcpcd -4` or `dhcpcd -6` worked using a Motorola SURFBoard 6141 and a Realtek RTL8168d/8111d. Either would work, but would not run dual stack: both protocols and addresses on one interface. (The `-6` command would not work if `-4` ran first, even after resetting the interface. And when it did, it gave the NIC a /128 address.) Try these commands:

```
# dhclient -4 enp3s0
# dhclient -P -v enp3s0
```

The `-P` argument grabs a lease of the IPv6 prefix only. `-v` writes to `stdout` what is also written to `/var/lib/dhclient/dhclient6.leases`:

```
Bound to *:546
Listening on Socket/enp3s0
Sending on   Socket/enp3s0
PRC: Confirming active lease (INIT-REBOOT).
XMT: Forming Rebind, 0 ms elapsed.
XMT:  X-- IA_PD a1:b2:cd:e2
XMT:  | X-- Requested renew  +3600
XMT:  | X-- Requested rebind +5400
XMT:  | | X-- IAPREFIX 1234:5:6700:890::/64
```

`IAPREFIX` is the necessary value. Substitute `::1` before the CIDR slash to make the prefix a real address:

```
# ip -6 addr add 1234:5:6700:890::1/64 dev enp3s0
```

Disable IPv6

Note: The Arch kernel has IPv6 support built in directly, therefore a module cannot be blacklisted.

Disable functionality

Warning: Disabling the IPv6 stack can break certain programs which expect it to be enabled. **FS#46297 (https://bugs.archlinux.org/task/46297)**

Adding `ipv6.disable=1` to the kernel line disables the whole IPv6 stack, which is likely what you want if you are experiencing issues. See **Kernel parameters** for more information.

Alternatively, adding `ipv6.disable_ipv6=1` instead will keep the IPv6 stack functional but will not assign IPv6 addresses to any of your network devices.

One can also avoid assigning IPv6 addresses to specific network interfaces by adding the following sysctl config to `/etc/sysctl.d/40-ipv6.conf`:

```
# Disable IPv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.nic0.disable_ipv6 = 1
...
net.ipv6.conf.nicN.disable_ipv6 = 1
```

Note that you must list all of the targeted interfaces explicitly, as disabling `all.disable_ipv6` does not apply to interfaces that are already "up" when sysctl settings are applied.

Note 2, if disabling IPv6 by sysctl, you should comment out the IPv6 hosts in your `/etc/hosts`:

```
#<ip-address> <hostname.domain.org> <hostname>
127.0.0.1 localhost.localdomain localhost
#::1 localhost.localdomain localhost
```

otherwise there could be some connection errors because hosts are resolved to their IPv6 address which is not reachable.

Other programs

Disabling IPv6 functionality in the kernel does not prevent other programs from trying to use IPv6. In most cases, this is completely harmless, but if you find yourself having issues with that program, you should consult the program's manual pages for a way to disable that functionality.

dhcpcd

dhcpcd will continue to harmlessly attempt to perform IPv6 router solicitation. To disable this, as stated in the **dhcpcd.conf(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/dhcpcd.conf.5)man page**, add the following to `/etc/dhcpcd.conf`:

```
noipv6rs
noipv6
```

NetworkManager

To disable IPv6 in NetworkManager, right click the network status icon, and select *Edit Connections* > *Wired* > Network name > *Edit* > *IPv6 Settings* > *Method* > *Ignore/Disabled*

Then click "Save".

ntpd

Following advice in **systemd#Drop-in files**, change how systemd starts `ntpd.service`:

```
# systemctl edit ntpd.service
```

This will create a drop-in snippet that will be run instead of the default `ntpd.service`. The `-4` flag prevents IPv6 from being used by the ntp daemon. Put the following into the drop-in snippet:

```
[Service]
ExecStart=
ExecStart=/usr/bin/ntpd -4 -g -u ntp:ntp
```

which first clears the previous `ExecStart`, and then replaces it with one that includes the `-4` flag.

systemd-networkd

networkd supports disabling IPv6 on a per-interface basis. When a network unit's `[Network]` section has either `LinkLocalAddressing=ipv4` or `LinkLocalAddressing=no`, networkd will not try to configure IPv6 on the matching interfaces.

Note however that even when using the above option, networkd will still be expecting to receive router advertisements if IPv6 is not disabled globally. If IPv6 traffic is not being received by the interface (e.g. due to sysctl or ip6tables settings), it will remain in the configuring state and potentially cause timeouts for services waiting for the network to be fully configured. To avoid this, the `IPv6AcceptRA=no` option should also be set in the `[Network]` section.

See also

- **IPv6 (https://www.kernel.org/doc/Documentation/networking/ipv6.txt)** - kernel.org documentation

- **IPv6 temporary addresses (http://www.ipsidixit.net/2012/08/09/ipv6-temporary-addresses-and-privacy-extensions/)** - a summary about temporary addresses and privacy extensions
- **IPv6 prefixes (http://tldp.org/HOWTO/Linux+IPv6-HOWTO/ch03s02.html)** - a summary of prefix types
- **net.ipv6 options (http://tldp.org/HOWTO/Linux+IPv6-HOWTO/ch11s02.html)** - documentation of kernel parameters

Retrieved from "https://wiki.archlinux.org/index.php?title=IPv6&oldid=507356"
