Network configuration

This article describes how to configure network connections on OSI layer 3 and above. Medium-specifics are handled in the /Ethernet and /Wireless subpages.

Related articles

- Domain name resolution
- Firewalls
- Internet sharing
- Network Debugging
- Router

1 Check the connection

To troubleshoot a network connection, go through the following conditions and ensure that you meet them:

1. Your network interface is listed and enabled. Otherwise, check the device driver – see /Ethernet#Device driver or /Wireless#Device driver.
2. You are connected to the network. The cable is plugged in or you are connected to the wireless LAN.
3. Your network interface has an IP address.
4. Your routing table is correctly set up.
5. You can ping a local IP address (e.g. your default gateway).
6. You can ping a public IP address (e.g. `9.9.9.9`, which is a DNS server operated by the Quad9 Foundation and is a convenient address to test with).
7. Check if you can resolve domain names (e.g. `archlinux.org`).

1.1 Ping

ping is used to test if you can reach a host.

```
$ ping www.example.com
```

```
PING www.example.com (93.184.216.34) 56(84) bytes of data.
64 bytes from 93.184.216.34 (93.184.216.34): icmp_seq=1 ttl=56 time=11.632 ms
64 bytes from 93.184.216.34 (93.184.216.34): icmp_seq=2 ttl=56 time=11.726 ms
64 bytes from 93.184.216.34 (93.184.216.34): icmp_seq=3 ttl=56 time=10.683 ms
...
```

For every reply received, the *ping* utility will print a line like the above until you interrupt (`Ctrl+c`) it interactively. For more information see the `ping(8) (https://man.archlinux.org/man/ping.8)` manual. Note that computers can be configured not to respond to ICMP echo requests. [1] (https://unix.stackexchange.com/questions/412446/how-to-disable-ping-response-icmp-echo-in-linux-all-the-time)

If you receive an error message (see ping error indications) or no reply, this may be related to incomplete configuration, but also your default gateway or your Internet Service Provider (ISP). You can run a traceroute to further diagnose the route to the host.

2 Network management

To set up a network connection, go through the following steps:

1. Ensure your network interface is listed and enabled.
2. Connect to the network. Plug in the Ethernet cable or connect to the wireless LAN.
3. Configure your network connection:
   - Most networks use the Dynamic Host Configuration Protocol for network configuration. Clients can automatically obtain a dynamic or static IP address from the DHCP server via a standalone DHCP client or using a network manager.
   - If the network does not have a DHCP server, you can configure a static IP address, routing table and DNS servers manually for each client. See #Static IP address for details.

> **Note**
>
> The installation image uses
>
> - systemd-networkd, which is configured as a DHCP client for Ethernet (https://gitlab.archlinux.org/archlinux/archiso/-/blob/master/configs/releng/airootfs/etc/systemd/network/20-ethernet.network), WLAN (https://gitlab.archlinux.org/archlinux/archiso/-/blob/master/configs/releng/airootfs/etc/systemd/network/20-wlan.network) and WWAN (https://gitlab.archlinux.org/archlinux/archiso/-/blob/master/configs/releng/airootfs/etc/systemd/network/20-wwan.network) network interfaces, and
> - systemd-resolved configured for system-wide DNS, see systemd-resolved#DNS.

2.1 Manual

2.1.1 iproute2

iproute2 is a dependency of the `base (https://archlinux.org/packages/?name=base)` meta package and provides the `ip(8) (https://man.archlinux.org/man/ip.8)` command-line interface, used to manage network interfaces, IP addresses and the routing table. Be aware that configuration made using `ip` will be lost after a reboot. For persistent configuration, you can automate *ip* commands using scripts and systemd units. Also note that `ip` commands can generally be abbreviated, for clarity they are however spelled out in this article.

> **Note**
>
> Arch Linux has deprecated `net-tools (https://archlinux.org/packages/?name=net-tools)` in favor of `iproute2 (https://archlinux.org/packages/?name=iproute2)`.[2] (https://archlinux.org/news/deprecation-of-net-tools/) See also Deprecated Linux networking commands and their replacements (https://dougvitale.wordpress.com/2011/12/21/deprecated-linux-networking-commands-and-their-replacements/).

2.1.2 Static IP address

A static IP address can be configured with most standard network managers and also dhcpcd.

To manually configure a static IP address, add an IP address as described in #IP addresses, set up your routing table and configure your DNS servers.

2.1.3 IP addresses

IP addresses are managed using `ip-address(8) (https://man.archlinux.org/man/ip-address.8)`.

List IP addresses:

```
$ ip address show
```

Add an IP address to an interface:

```
# ip address add address/prefix_len broadcast + dev interface
```

Note that:

- the address is given in CIDR notation to also supply a subnet mask
- `+` is a special symbol that makes `ip` derive the broadcast address from the IP address and the subnet mask

> **Note**
>
> Make sure manually assigned IP addresses do not conflict with DHCP assigned ones.

Delete an IP address from an interface:

```
# ip address del address/prefix_len dev interface
```

Delete all addresses matching a criteria, e.g. of a specific interface:

```
# ip address flush dev interface
```

> **Tip**
>
> IPv4 addresses can be calculated with ipcalc (https://jodies.de/ipcalc) (`ipcalc (https://archlinux.org/packages/?name=ipcalc)`).

2.1.4 Routing table

The routing table is used to determine if you can reach an IP address directly or what gateway (router) you should use. If no other route matches the IP address, the default gateway is used.

The routing table is managed using `ip-route(8) (https://man.archlinux.org/man/ip-route.8)`.

PREFIX is either a CIDR notation or `default` for the default gateway.

List IPv4 routes:

```
$ ip route show
```

List IPv6 routes:

```
$ ip -6 route show
```

Add a route:

```
# ip route add PREFIX via address dev interface
```

Delete a route:

```
# ip route del PREFIX via address dev interface
```

2.2 Automatic

Automatic network configuration is accomplished using **Dynamic Host Configuration Protocol** (DHCP). The network's DHCP server provides IP address(es), the default gateway IP address(es) and optionally also DNS name servers upon request from the DHCP client.

See **Router#DNS and DHCP** for a DHCP server comparison table.

2.2.1 Network managers

A network manager lets you manage network connection settings in so called network profiles to facilitate switching networks.

Tip

You can check if a DHCPv4 server is running with `dhcping (https://archlinux.org/packages/?name=dhcping)`.

Note

Each network interface should be managed by only one DHCP client or network manager, so it is advised to run only one DHCP client or network manager on the system.

Software	Connection type			Wireless authentication	IP address, route and DNS management			Interface		
	Ethernet	PPPoE	Mobile broadband		Static IP	DHCP client	Domain name resolution	CLI	TUI	
dhclient (https://archlinux.org/packages/?name=dhclient)[1]	Yes	No	No	No[2]	Yes	internal	Yes (writes /etc/resolv.conf)	No	No	
dhcpcd	Yes	No	No	Launches wpa_supplicant[3]	Yes	internal	Yes (uses resolvconf or writes /etc/resolv.conf)	No	No	dhcpc s://aur org/pac d-
ConnMan	Yes	No (https://web.archive.org/web/20131231103723/https://01.org/jira/browse/CM-63)	Yes (via ofono (https://aur.archlinux.org/packages/ofono/)AUR)	Yes (via wpa_supplicant (https://archlinux.org/packages/?name=wpa_supplicant) or iwd)	Yes	internal	Yes (runs a builtin resolver and writes /etc/resolv.conf)	connmanctl(1) (https://man.archlinux.org/man/connmanctl.1)	Yes	
netctl	Yes	Yes (via ppp (https://archlinux.org/packages/?name=ppp))	Yes (via ppp (https://archlinux.org/packages/?name=ppp))	Yes (via wpa_supplicant (https://archlinux.org/packages/?name=wpa_supplicant))	Yes	dhcpcd (https://archlinux.org/packages/?name=dhcpcd) or dhclient (https://archlinux.org/packages/?name=dhclient)	Yes (uses resolvconf)	netctl(1) (https://man.archlinux.org/man/netctl.1)	wifi-menu(1) (https://man.archlinux.org/man/wifi-menu.1)[4]	
NetworkManager	Yes	Yes (via ppp (https://archlinux.org/packages/?name=ppp))	Yes (via modemmanager (https://archlinux.org/packages/?name=modemmanager))	Yes (via wpa_supplicant (https://archlinux.org/packages/?name=wpa_supplicant) or iwd)	Yes	internal, dhclient (https://archlinux.org/packages/?name=dhclient) or dhcpcd (https://archlinux.org/packages/?name=dhcpcd)[5]	Yes (uses systemd-resolved, resolvconf or writes /etc/resolv.conf)	nmcli(1) (https://man.archlinux.org/man/nmcli.1)	nmtui(1) (https://man.archlinux.org/man/nmtui.1)	
systemd-networkd	Yes	No (https://github.com/systemd/systemd/issue s/481)	No (https://github.com/systemd/systemd/issues/20370)	No (https://github.com/systemd/systemd/issues/32468)[2]	Yes	internal	Yes (uses systemd-resolved)	networkctl(1) (https://man.archlinux.org/man/networkctl.1)	No	
wpa_supplicant	IEEE 802.1X	No	No	Yes	No			wpa_cli(8) (https://man.archlinux.org/man/wpa_cli.8)	No	wpa_sup (https: inux.or wpa_sup i
iwd	IEEE 802.1X	No	No	Yes	Yes	internal	Yes (uses systemd-resolved or resolvconf)	iwctl(1) (https://man.archlinux.org/man/iwctl.1)	impala (https://archlinux.org/packages/?name=impala)	iwgtk (r.archl ckages

1. No longer maintained as of early 2022. ISC advises no longer using it in production.
2. Wireless authentication can be configured separately with wpa_supplicant or iwd.
3. Wireless authentication must be configured separately with wpa_supplicant.
4. Only Wi-Fi connections can be managed.
5. NetworkManager does not use dhcpcd for DHCPv6, see NetworkManager#DHCP client.

3 Network interfaces

Network interfaces are managed by udev and configured by `systemd.link(5) (https://man.archlinux.org/man/systemd.link.5)` files. The default configuration assigns names to your network interface controllers using Predictable Network Interface Names (https://systemd.io/PREDICTABLE_INTERFACE_NAMES/), which prefixes interfaces names with `en` (wired/Ethernet), `wl` (wireless/WLAN), or `ww` (mobile broadband/WWAN). See `systemd.net-naming-scheme(7) (https://man.archlinux.org/man/systemd.net-naming-scheme.7)`.

Tip

- The system `/usr/lib/systemd/network/99-default.link` is generally sufficient for most cases.
- To change interface names, see #Change interface name and #Revert to traditional interface names.
- You can run `udevadm test-builtin net_setup_link /sys/path/to/network/device` as the root user to diagnose problems with *.link* files.

Note

- The predictable network interface names can change after adding or removing a PCIe device if the system firmware decides to renumber the devices. See systemd issue 33347 (https://github.com/systemd/systemd/issues/33347).
- The `iwd (https://archlinux.org/packages/?name=iwd)` package contains a *.link* file that disables predictable network interface names. Merely having it installed will prevent all network interfaces from being renamed to predictable names. See iwd#Wireless device is not renamed by udev.

3.1 Listing network interfaces

Both wired and wireless interface names can be found via `ls /sys/class/net` or `ip link`. Note that `lo` is the virtual loopback interface and not used in making network connections.

Wireless device names can also be retrieved using `iw dev`. See also /Wireless#Get the name of the interface.

If your network interface is not listed, make sure your device driver was loaded successfully. See /Ethernet#Device driver or /Wireless#Device driver.

3.2 Enabling and disabling network interfaces

Network interfaces can be enabled or disabled using `ip link set` *`interface`* `up|down`, see `ip-link(8) (https://man.archlinux.org/man/ip-link.8)`.

To check the status of the interface `enp2s0`:

```
$ ip link show dev enp2s0
```

```
2: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master br0 state DOWN mode DEFAULT qlen 1000
...
```

The `UP` in `<BROADCAST,MULTICAST,UP,LOWER_UP>` is what indicates the interface is up, not the later `state DOWN`.

Note

If your default route is through interface `enp2s0`, taking it down will also remove the route, and bringing it back up will not automatically re-establish the default route. See #Routing table for re-establishing it.

3.3 Change interface name

Note

When changing the naming scheme, do not forget to update all network-related configuration files and custom systemd unit files to reflect the change.

You can change the device name by defining the name manually with a `systemd.link(5) (https://man.archlinux.org/man/systemd.link.5)` file. The file must be ordered lexicographically before `99-default.link`, for example:

```
/etc/systemd/network/10-net0.link
```

```
[Match]
PermanentMACAddress=aa:bb:cc:dd:ee:ff

[Link]
Name=net0
```

Alternatively, a udev rule can be used:

```
/etc/udev/rules.d/10-network.rules
```

```
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="aa:bb:cc:dd:ee:ff", NAME="net0"
```

These rules will be applied automatically at boot. To apply the change immediately, do a manual trigger of the udev rule on the `net` subsystem:

```
# udevadm trigger --verbose --subsystem-match=net --action=add
```

If you want to run a **test** on the changes made, `udevadm --debug test /sys/class/net/*` can be of help.

Note

- The priority of `Name` is lower than `NamePolicy`, so make sure the latter is unset/empty or the name will not be changed.
- The network interface must be down before changing its name. **[3] (https://github.com/systemd/systemd/issues/26601)**
- To get the MAC address of each card, run `ip link`.
- Make sure to use the lower-case hex values in your udev rules. It does not like upper-case.

If the network card has a dynamic MAC, you can use `Path` (which can be checked using `networkctl status` *`interface_name`*):

```
/etc/systemd/network/10-net1.link
```

```
[Match]
Path=pci-0000:01:00.0

[Link]
Name=net1
```

Or, use a udev rule with `DEVPATH`:

```
/etc/udev/rules.d/10-network.rules
```

```
SUBSYSTEM=="net", DEVPATH=="/devices/pci*/*1c.0/*/net/*", NAME="net1"
```

To get the `DEVPATH` of all currently-connected devices, see where the symlinks in `/sys/class/net/` lead. For example:

```
$ file /sys/class/net/*
```

```
/sys/class/net/enp0s20f0u4u1: symbolic link to ../../devices/pci0000:00/0000:00:14.0/usb2/2-4/2-4.1/2-4.1:1.0/net/enp0s20f0u4u1
/sys/class/net/enp0s31f6:     symbolic link to ../../devices/pci0000:00/0000:00:1f.6/net/enp0s31f6
/sys/class/net/lo:            symbolic link to ../../devices/virtual/net/lo
/sys/class/net/wlp4s0:        symbolic link to ../../devices/pci0000:00/0000:00:1c.6/0000:04:00.0/net/wlp4s0
```

The device path should match both the new and old device name, since the rule may be executed more than once on bootup. For example, in the given rule, `"/devices/pci*/*1c.0/*/net/en*"` would be wrong since it will stop matching once the name is changed to `net1`. Only the system-default rule will fire the second time around, causing the name to be changed back.

If you are using a USB network device (e.g. Android phone tethering) that has a dynamic MAC address and you want to be able to use different USB ports, you could use a rule that matched depending on vendor and model ID instead:

```
/etc/systemd/network/20-net2.link
```

```
[Match]
Property=ID_VENDOR_ID=12ab ID_MODEL_ID=3cd4

[Link]
Name=net2
```

or

```
/etc/udev/rules.d/10-network.rules
```

```
SUBSYSTEM=="net", ACTION=="add", ATTRS{idVendor}=="12ab", ATTRS{idProduct}=="3cd4", NAME="net2"
```

Note

When choosing the static names **it should be avoided to use names in the format of "eth*X*" and "wlan*X*"**, because this may lead to race conditions between the kernel and udev during boot. Instead, it is better to use interface names that are not used by the kernel as default, e.g.: `net0`, `net1`, `wifi0`, `wifi1`. For further details please see the **systemd (https://systemd.io/PREDICTABLE_INTERFACE_NAMES/)** documentation.

3.4 Revert to traditional interface names

If you would prefer to retain traditional interface names such as `eth0`, **Predictable Network Interface Names (https://systemd.io/PREDICTABLE_INTERFACE_NAMES/)** can be disabled by changing the default `NamePolicy` for udev's `net_setup_link` built-in:

```
/etc/systemd/network/99-default.link.d/traditional-naming.conf
```

```
[Link]
NamePolicy=keep kernel
```

Alternatively, `net_setup_link` can be completely disabled by masking the corresponding udev rule:

```
# ln -s /dev/null /etc/udev/rules.d/80-net-setup-link.rules
```

or by adding `net.ifnames=0` to the **kernel parameters**.

Note

systemd.link(5) (https://man.archlinux.org/man/systemd.link.5) relies on `net_setup_link` to work. Prefer to use the first approach unless you fully understand what you are doing.

3.5 Set device MTU and queue length

You can change the device **MTU** and queue length by defining manually with a systemd.link(5) (https://man.archlinux.org/man/systemd.link.5) config. For example:

```
/etc/systemd/network/30-mtu.link

[Match]
Type=wlan

[Link]
MTUBytes=1500
TransmitQueueLength=2000
```

Or through a udev rule:

```
/etc/udev/rules.d/10-network.rules

ACTION=="add", SUBSYSTEM=="net", KERNEL=="wl*", ATTR{mtu}="1500", ATTR{tx_queue_len}="2000"
```

`MTUBytes` : Using a value larger than 1500 (so called **jumbo frames**) can significantly speed up your network transfers. Note that all network interfaces, including switches in the local network, must support the same MTU in order to use jumbo frames. For PPPoE, the MTU should not be larger than 1492. You can also set MTU via systemd.netdev(5) (https://man.archlinux.org/man/systemd.netdev.5).

`TransmitQueueLength` : Small value for slower devices with a high latency like modem links and ISDN. High value is recommended for server connected over the high-speed internet connections that perform large data transfers.

4 Set the hostname

A **hostname** is a unique name created to identify a machine on a network, configured in `/etc/hostname` —see hostname(5) (https://man.archlinux.org/man/hostname.5) and hostname(7) (https://man.archlinux.org/man/hostname.7) for details. The file can contain the system's domain name, if any. To set the hostname, **edit** `/etc/hostname` to include a single line with *`yourhostname`* :

```
/etc/hostname

yourhostname
```

Tip

For advice on choosing a hostname, see **RFC 1178**.

Alternatively, using hostnamectl(1) (https://man.archlinux.org/man/hostnamectl.1):

```
# hostnamectl hostname yourhostname
```

To temporarily set the hostname (until reboot), use hostname(1) (https://man.archlinux.org/man/hostname.1) from inetutils (https://archlinux.org/packages/?name=inetutils):

```
# hostname yourhostname
```

To set the "pretty" hostname and other machine metadata, see machine-info(5) (https://man.archlinux.org/man/machine-info.5).

4.1 Local network hostname resolution

To make your machine accessible in your LAN via its hostname you can:

- edit the `/etc/hosts` file for every device in your LAN, see hosts(5) (https://man.archlinux.org/man/hosts.5)
- set up a **DNS server** to resolve your hostname and make the LAN devices use it (e.g. via **DHCP**)
- or the easy way: use a **Zero-configuration networking** service:

- Hostname resolution via Microsoft's **NetBIOS**. Provided by **Samba** on Linux. It only requires the `nmb.service`. Computers running Windows, macOS, or Linux with `nmb` running, will be able to find your machine.
- Hostname resolution via **mDNS**. Provided by either `nss_mdns` with **Avahi** (see **Avahi#Hostname resolution** for setup details) or **systemd-resolved**. Computers running macOS, or Linux with Avahi or systemd-resolved running, will be able to find your machine. The older Win32 API does not support mDNS, which may prevent some older Windows applications from accessing your device.

5 Tips and tricks

5.1 Bonding or LAG

See **netctl** or **systemd-networkd**, or **Wireless bonding**.

5.2 IP address aliasing

IP aliasing is the process of adding more than one IP address to a network interface. With this, one node on a network can have multiple connections to a network, each serving a different purpose. Typical uses are virtual hosting of Web and FTP servers, or reorganizing servers without having to update any other machines (this is especially useful for nameservers).

5.2.1 Example

To manually set an alias, for some NIC, use **iproute2 (https://archlinux.org/packages/?name=iproute2)** to execute

```
# ip addr add 192.168.2.101/24 dev enp2s0 label enp2s0:1
```

To remove a given alias execute

```
# ip addr del 192.168.2.101/24 dev enp2s0:1
```

Packets destined for a subnet will use the primary alias by default. If the destination IP is within a subnet of a secondary alias, then the source IP is set respectively. Consider the case where there is more than one NIC, the default routes can be listed with `ip route`.

5.3 Promiscuous mode

Toggling **promiscuous mode** will make a (wireless) NIC forward all traffic it receives to the OS for further processing. This is opposite to "normal mode" where a NIC will drop frames it is not intended to receive. It is most often used for advanced network troubleshooting and **packet sniffing**.

/etc/systemd/system/promiscuous@.service

```
[Unit]
Description=Set %i interface in promiscuous mode
After=network.target

[Service]
Type=oneshot
ExecStart=/usr/bin/ip link set dev %i promisc on
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

If you want to enable promiscuous mode on interface `enp2s0`, **enable** `promiscuous@enp2s0.service`.

5.4 Investigate sockets

ss is a utility to investigate network ports and is part of the **iproute2 (https://archlinux.org/packages/?name=iproute2)** package. It has a similar functionality to the **deprecated (https://archlinux.org/news/deprecation-of-net-tools/)** netstat utility.

Common usage includes:

Display all TCP Sockets with service names:

```
$ ss -at
```

Display all TCP Sockets with port numbers:

```
$ ss -atn
```

Display all UDP Sockets:

```
$ ss -au
```

For more information see **ss(8) (https://man.archlinux.org/man/ss.8)**.

6 Troubleshooting

6.1 The TCP window scaling problem

TCP packets contain a "window" value in their headers indicating how much data the other host may send in return. This value is represented with only 16 bits, hence the window size is at most 64KiB. TCP packets are cached for a while (they have to be reordered), and as memory is (or used to be) limited, one host could easily run out of it.

Back in 1992, as more and more memory became available, **RFC:1323** was written to improve the situation: Window Scaling. The "window" value, provided in all packets, will be modified by a Scale Factor defined once, at the very beginning of the connection. That 8-bit Scale Factor allows the Window to be up to 32 times higher than the initial 64KiB.

It appears that some broken routers and firewalls on the Internet are rewriting the Scale Factor to 0 which causes misunderstandings between hosts. The Linux kernel 2.6.17 introduced a new calculation scheme generating higher Scale Factors, virtually making the aftermaths of the broken routers and firewalls more visible.

The resulting connection is at best very slow or broken.

6.1.1 How to diagnose the problem

First of all, let us make it clear: this problem is odd. In some cases, you will not be able to use TCP connections (HTTP, FTP, ...) at all and in others, you will be able to communicate with some hosts (very few).

When you have this problem, the output from **dmesg** is okay, logs are clean and `ip addr` will report normal status... and actually everything appears normal.

If you cannot browse any website, but you can ping some random hosts, chances are great that you are experiencing this problem: ping uses ICMP and is not affected by TCP problems.

You can try to use **Wireshark**. You might see successful UDP and ICMP communications but unsuccessful TCP communications (only to foreign hosts).

6.1.2 Ways of fixing it

6.1.2.1 Bad

To fix it the bad way, you can change the `tcp_rmem` value, on which Scale Factor calculation is based. Although it should work for most hosts, it is not guaranteed, especially for very distant ones.

```
# sysctl -w net.ipv4.tcp_rmem="4096 87380 174760"
```

6.1.2.2 Good

Simply disable Window Scaling. Since Window Scaling is a nice TCP feature, it may be uncomfortable to disable it, especially if you cannot fix the broken router. There are several ways to disable Window Scaling, and it seems that the most bulletproof way (which will work with most kernels) is to add the following line to `/etc/sysctl.d/99-disable_window_scaling.conf` (see also **sysctl**):

```
net.ipv4.tcp_window_scaling = 0
```

6.1.2.3 Best

This problem is caused by broken routers/firewalls, so let us change them. Some users have reported that the broken router was their very own DSL router.

6.1.3 More about it

This section is based on the LWN article **TCP window scaling and broken routers (https://lwn.net/Articles/92727/)** and an archived Kernel Trap article: **Window Scaling on the Internet (https://web.archive.org/web/20120426135627/http://kerneltrap.org:80/node/6723)**.

There are also several relevant threads on the LKML.

6.2 local hostname is resolved over the network

nss-myhostname(8) (https://man.archlinux.org/man/nss-myhostname.8) (an **NSS** module provided by **systemd** and enabled by default in `/etc/nsswitch.conf`) provides `localhost` and the local **hostname** resolution to an IP address. Some software may, however, still instead read `/etc/hosts` directly; see **[4] (https://lists.debian.org/debian-devel/2013/07/msg00809.html)** **[5] (https://bugzilla.mozilla.org/show_bug.cgi?id=87717#c55)** for examples.

To prevent such software from unsafely resolving the local hostname over the network, add an entry for it to the **hosts(5) (https://man.archlinux.org/man/hosts.5)** file:

```
/etc/hosts
```

```
127.0.0.1        localhost
::1              localhost
127.0.1.1        yourhostname
```

For a system with a permanent IP address, replace `127.0.1.1` with that permanent IP address. For a system with a **fully qualified domain name**, insert the fully qualified domain name before the hostname (see the following link for **the reasoning (https://www.debian.org/doc/manuals/debian-reference/ch05.en.html#_the_hostname_resolution)**). For example:

```
/etc/hosts
```

```
127.0.0.1        localhost
::1              localhost
203.0.113.45     host1.fqdomain.example host1
```

> **Note**
>
> The order of hostnames/aliases that follow the IP address in `/etc/hosts` is significant. The first string is considered the canonical hostname and may be appended with parent domains, where domain components are separated by a dot. All following strings on the same line are considered aliases. See `hosts(5) (https://man.archlinux.org/man/hosts.5)` for more information.

7 See also

- **Linux Network Administrators Guide (https://www.tldp.org/LDP/nag2/index.html)**
- **Debian Reference: Network setup (https://www.debian.org/doc/manuals/debian-reference/ch05.en.html)**
- **RHEL7: Networking Guide (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/)**
- **Monitoring and tuning the Linux Networking Stack: Receiving data (https://blog.packagecloud.io/eng/2016/06/22/monitoring-tuning-linux-networking-stack-receiving-data/)**
- **Monitoring and tuning the Linux Networking Stack: Sending data (https://blog.packagecloud.io/eng/2017/02/06/monitoring-tuning-linux-networking-stack-sending-data/)**
- **Tracing a packet journey using tracepoints, perf and eBPF (https://blog.yadutaf.fr/2017/07/28/tracing-a-packet-journey-using-linux-tracepoints-perf-ebpf/)**

Retrieved from "https://wiki.archlinux.org/index.php?title=Network_configuration&oldid=843054"