ConnMan

From ArchWiki

ConnMan (https://01.org/connman) is a command-line network manager designed for use with embedded devices and fast resolve times. It is modular through a **plugin architecture (http://git.kernel.org/cgit/network/connman/connman.git/tree/plugins)**, but has native **DHCP (http://git.kernel.org/cgit/network/connman/connman.git/tree/src/dhcp.c)** and **NTP (http://git.kernel.org/cgit/network/connman/connman.git/tree/src/ntp.c)** support.

Related articles

Network configuration

Wireless network configuration

Category:Network configuration

Contents

Installation

Install the **`connman (https://www.archlinux.org/packages/?name=connman)`** package. **`wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)`** and **`bluez (https://www.archlinux.org/packages/?name=bluez)`** are optional dependencies required for Wi-Fi and Bluetooth functionality respectively.

Before **enabling** `connman.service`, ensure any existing **network configuration** is disabled.

Desktop clients

- **cmst** — Qt GUI for ConnMan.

https://github.com/andrew-bibb/cmst || `cmst (https://aur.archlinux.org/packages/cmst/)`[AUR]

- **connman-ncurses** — Simple ncurses UI for ConnMan; not all of connman functionality is implemented, but usable (with X or from terminal without X), see the **wiki (https://github.com/eurogiciel-oss/connman-json-client/wiki)**.

  https://github.com/eurogiciel-oss/connman-json-client || `connman-ncurses-git (https://aur.archlinux.org/packages/connman-ncurses-git/)`[AUR]

- **connman-notify** — Connman event notification client

  https://github.com/wavexx/connman-notify || `connman-notify (https://aur.archlinux.org/packages/connman-notify/)`[AUR][**broken link**: archived in **aur-mirror (https://github.com/felixonmars/aur3-mirror/tree/master/connman-notify)**]

- **ConnMan-UI** — GTK3 client applet.

  https://github.com/tbursztyka/connman-ui || `connman-ui-git (https://aur.archlinux.org/packages/connman-ui-git/)`[AUR]

- **connman_dmenu** — Client/frontend for dmenu.

  https://github.com/taylorchu/connman_dmenu || `connman_dmenu-git (https://aur.archlinux.org/packages/connman_dmenu-git/)`[AUR]

- **Econnman** — Enlightenment desktop panel applet.

  http://www.enlightenment.org || `econnman (https://aur.archlinux.org/packages/econnman/)`[AUR]

- **LXQt-Connman-Applet** — LXQt desktop panel applet.

  https://github.com/surlykke/lxqt-connman-applet || `lxqt-connman-applet-git (https://aur.archlinux.org/packages/lxqt-connman-applet-git/)`[AUR]

- **qconnman-ui** — Qt management interface used on O.S. Systems products

  https://github.com/OSSystems/qconnman-ui || `qconnman-ui-git (https://aur.archlinux.org/packages/qconnman-ui-git/)`[AUR][**broken link**: archived in **aur-mirror (https://github.com/felixonmars/aur3-mirror/tree/master/qconnman-ui-git)**]

- **connman-gtk** — GTK client.

  https://github.com/jgke/connman-gtk || `connman-gtk (https://aur.archlinux.org/packages/connman-gtk/)`[AUR]

- **gnome-extension-connman** — Gnome3 extension for connman; it contains only some of the functionality without installing connman-gtk.

  https://github.com/jgke/gnome-extension-connman ||
  https://extensions.gnome.org/extension/981/connman-extension/

Usage

ConnMan has a standard command line client `connmanctl`. It can run in 2 modes:

- In **command mode** commands are entered as arguments to `connmanctl` command, just like **systemctl**.

- **Interactive mode** is started by typing `connmanctl` without arguments. Prompt will change to `connmanctl>` to indicate it is waiting for user commands, just like **python** interactive mode. The interactive mode supports tab completion, which makes finding the correct connection easy.

Wired

ConnMan will automatically handle wired connections.

Wi-Fi

Enabling and disabling wifi

To check if wifi is enabled you can run `connmanctl technologies` and check for the line that says `Powered: True/False`. To power the wifi on you can run `connmanctl enable wifi` or if you need to disable it you can run `connmanctl disable wifi`. Other ways to enable wifi could include using the `Fn` keys on the laptop to turn it on or running `ip link set <interface> up`.

Connecting to an open access point

The commands in this section show how to run `connmanctl` in command mode.

To scan the network `connmanctl` accepts simple names called *technologies*. To scan for nearby Wi-Fi networks:

```
$ connmanctl scan wifi
```

To list the available networks found after a scan run (example output):

```
$ connmanctl services
```

```
*AO MyNetwork               wifi_dc85de828967_68756773616d_managed_psk
    OtherNET                wifi_dc85de828967_38303944616e69656c73_managed_psk
    AnotherOne              wifi_dc85de828967_3257495245363836_managed_wep
    FourthNetwork           wifi_dc85de828967_4d7572706879_managed_wep
    AnOpenNetwork           wifi_dc85de828967_4d6568657272696e_managed_none
```

To connect to an open network, use the second field beginning with `wifi_`:

```
$ connmanctl connect wifi_dc85de828967_4d6568657272696e_managed_none
```

Tip: Network names can be tab-completed.

You should now be connected to the network. Check using `ip addr` or `connmanctl state`.

Connecting to a protected access point

For protected access points you will need to provide some information to the ConnMan daemon, at the very least a password or a passphrase.

The commands in this section show how to run `connmanctl` in interactive mode, it is required for running the `agent` command. To start interactive mode simply type:

```
$ connmanctl
```

You then proceed almost as above, first scan for any Wi-Fi *technologies*:

```
connmanctl> scan wifi
```

To list services:

```
connmanctl> services
```

Now you need to register the agent to handle user requests. The command is:

```
connmanctl> agent on
```

You now need to connect to one of the protected services. To do this easily, just use tab completion for the wifi_ service. If you were connecting to OtherNET in the example above you would type:

```
connmanctl> connect wifi_dc85de828967_38303944616e69656c73_managed_psk
```

The agent will then ask you to provide any information the daemon needs to complete the connection. The information requested will vary depending on the type of network you are connecting to. The agent will also print additional data about the information it needs as shown in the example below.

```
Agent RequestInput wifi_dc85de828967_38303944616e69656c73_managed_psk
  Passphrase = [ Type=psk, Requirement=mandatory ]
  Passphrase?
```

Provide the information requested, in this example the passphrase, and then type:

```
connmanctl> quit
```

If the information you provided is correct you should now be connected to the protected access point.

Settings

Settings and profiles are automatically created for networks the user connects to often. They contain fields for the passphrase, essid and other information. Profile settings are stored in directories under `/var/lib/connman/` by their service name. To view all network profiles run this command from **root shell**:

```
# cat /var/lib/connman/*/settings
```

Note: VPN settings can be found in `/var/lib/connman-vpn/` .

Technologies

Various hardware interfaces are referred to as *Technologies* by ConnMan.

To list available *technologies* run:

```
$ connmanctl technologies
```

To get just the types by their name one can use this one liner:

```
$ connmanctl technologies | awk '/Type/ { print $NF }'
```

Note: The field `Type = tech_name` provides the technology type used with `connmanctl` commands

To interact with them one must refer to the technology by type. *Technologies* can be toggled on/off with:

```
$ connmanctl enable technology_type
```

and:

```
$ connmanctl disable technology_type
```

For example to toggle off wifi:

```
$ connmanctl disable wifi
```

Warning: connman grabs rfkill events. It is most likely impossible to use `rfkill` or `bluetoothctl` to (un)block devices, yet hardware keys may still work.**[1] (https://git.kernel.org/cgit/network/connman/connman.git/tree/doc/overview-api.txt#n406)** Always use `connmanctl enable|disable`

Tips and tricks

Avoid changing the hostname

By default, ConnMan changes the **transient hostname (http://www.freedesktop.org/software/systemd/man/hostnamectl.html)** on a per network basis. This can create problems with X authority: If ConnMan changes your hostname to something else than the one used to generate the xauth magic cookie, then it will become impossible to create new windows. Symptoms are error messages like "No protocol specified" and "Can't open display: :0.0". Manually resetting the host name fixes this, but a permanent solution is to prevent ConnMan from changing your host name in the first place. This can be accomplished by adding the following to `/etc/connman/main.conf`:

```
[General]
AllowHostnameUpdates=false
```

Make sure to **restart** the `connman.service` after changing this file.

For testing purposes it is recommended to watch the **journal** and plug the network cable a few times to see the action.

Prefer ethernet to wireless

By default ConnMan does not prefer ethernet over wireless, which can lead to it deciding to stick with a slow wireless network even when ethernet is available. You can tell connman to prefer ethernet adding the following to `/etc/connman/main.conf`:

```
[General]
PreferredTechnologies=ethernet,wifi
```

Exclusive connection

ConnMan allows you to be connected to both ethernet and wireless at the same time. This can be useful as it allows programs that established a connection over wifi to stay connected even after you connect to ethernet. But some people prefer to have only a single unambiguous connection active at a time. That behavior can be activated by adding the following to `/etc/connman/main.conf`:

```
[General]
SingleConnectedTechnology=true
```

Connecting to eduroam (802.1X)

WPA2 Enterprise networks such as eduroam require a separate configuration file before **connecting** to the network. For example, create `/var/lib/connman/eduroam.config`:

eduroam.config

```
[service_eduroam]
Type=wifi
Name=eduroam
EAP=peap
CACertFile=/etc/ssl/certs/certificate.cer
Phase2=MSCHAPV2
Identity=user@foo.edu
AnonymousIdentity=anonymous@foo.edu
Passphrase=password
```

Restart `wpa_supplicant.service` and `connman.service` to connect to the new network.

Note:

- Options are case-sensitive. **[2] (https://together.jolla.com/question/55969/connman-fails-due-to-case-sensitive-settings/)**
- Consult the institution hosting the eduroam network for various settings such as username, password, `EAP`, `Phase2output`, and needed certificates.

For more information, see **`connman-service.config(5)` (http://jlk.fjfi.cvut.cz/arch/manpages/man/connman-service.config.5)** and **Wireless network configuration#eduroam**.

Avoiding conflicts with local DNS server

If you are running a local DNS server, it will likely have problems binding to port 53 (TCP and/or UDP) after installing Connman. This is because Connman includes its own DNS proxy which also tries to bind to those ports. If you see log messages from **BIND** or **dnsmasq** like

```
"named[529]: could not listen on UDP socket: address in use"
```

this could be the problem. To verify which application is listening on the ports, you can execute `ss -tulpn` as root.

To fix this connmand can be started with the options `-r` or `--nodnsproxy` by **overriding** the systemd service file. Create the folder `/etc/systemd/system/connman.service.d/` and add the file `disable_dns_proxy.conf`:

```
[Service]
ExecStart=
ExecStart=/usr/bin/connmand -n --nodnsproxy
```

Make sure to **reload** the systemd daemon and **restart** the `connman.service`, and your DNS proxy, after adding this file.

Blacklist interfaces

If something like **Docker** is creating virtual interfaces Connman may attempt to connect to one of these instead of your physical adapter if the connection drops. A simple way of avoiding this is to blacklist the interfaces you do not want to use. Connman will by default blacklist interfaces starting with `vmnet`, `vboxnet`, `virbr` and `ifb`, so those need to be included in the new blacklist as well.

Blacklisting interface names is also useful to avoid a race condition where connman may access `eth#` or `wlan#` before systemd/udev can change it to use a **predictable interface name** like `enp4s0`. Blacklisting the conventional (and unpredictable) interface prefixes makes connman wait until they are renamed.

If it does not already exist, create `/etc/connman/main.conf`:

```
[General]
NetworkInterfaceBlacklist=vmnet,vboxnet,virbr,ifb,docker,veth,eth,wlan
```

Once `connman.service` has been **restarted** this will also hide all the `veth#######` interfaces from GUI tools like Econnman.

Troubleshooting

Error /net/connman/technology/wifi: Not supported

You need to install **`wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)`** and then **restart** `connman.service`.

Error /net/connman/technology/wifi: No carrier

If wireless scanning leads to above error, this may be due to an unresolved bug.**[3] (https://01.org/jira/browse/CM-670)** If it does not resolve even though wireless **preconditions (https://lists.01.org/pipermail/connman/2014-December/019203.html)** are met, try again after disabling competing network managers and rebooting.

This may also simply be caused by the wireless interface being blocked by **rfkill**, which can occur after restarting wpa_supplicant. Use `rfkill list` to check.

Error Failed to set hostname/domainname

connman can failed to set hostname or domainname due to lack of CAP_SYS_ADMIN.

You will need to edit connman.service (and other like connman-vpn.service , etc ...) to modify the CapabilityBoundingSet line to add CAP_SYS_ADMIN.

See EPERM error of sethostname(2)/setdomainname(2) manpages for more details.

Unknown route on connection

A log entry for an unknown route appears each time a connect is done. For example:

```
...
connmand[473]: wlp2s0 {add} route 82.165.8.211 gw 10.20.30.4 scope 0 <UNIVERSE>
```

```
connmand[473]: wlp2s0 {del} route 82.165.8.211 gw 10.20.30.4 scope 0 <UNIVERSE>
...
```

It likely is Connman performing a connectivity check to the ipv4.connman.net host (which resolves to the IP address `82.165.8.211` at current).**[4] (https://01.org/jira/browse/CM-657)** See the **Connman README (http://git.kernel.org/cgit/network/connman/connman.git/tree/README#n358)** for more information on why and what - apart from the connecting IP - it transmits.

While there is no option to configure the destination host of the check, the connection itself is functional (unless behind a captive portal) if the check is blocked by a firewall rule:

```
# ip6tables -A OUTPUT -d ipv6.connman.net -j REJECT
# iptables -A OUTPUT -d ipv4.connman.net,ipv6.connman.net -j REJECT
```

See also

- **git repo documentation (https://git.kernel.org/cgit/network/connman/connman.git/tree/doc)** - for further detailed documentation