Wireless Setup

From ArchWiki

Summary

A complete guide to enabling and configuring wireless networking.

Overview

Arch Linux provides netctl for network management. `netctl` supports wired connections on desktops and servers, as well as **wireless** setups and roaming for mobile users, facilitating easy management of network profiles. NetworkManager and Wicd are popular third-party alternatives.

Configuring wireless is a two-part process; the first part is to identify and ensure the correct driver for your wireless device is installed (they are available on the installation media, so make sure you install them), and to configure the interface. The second is choosing a method of managing wireless connections. This article covers both parts, and provides additional links to wireless management tools.

About new Arch Linux systems: Most wireless drivers and tools are available during Arch set-up under the `base` (https://www.archlinux.org/groups/i686/base/) group. Be sure to install the proper driver for your card. Udev will usually load the appropriate module, thereby creating the wireless interface, in the initial live system of the installer, as well as the newly installed system on your hard drive. If you are configuring your wireless functionality after, and not during, Arch Linux installation, simply ensure the required packages are installed with pacman, (driver, firmware if needed, `wireless_tools` (https://www.archlinux.org/packages/?name=wireless_tools) , `iw` (https://www.archlinux.org/packages/?name=iw) , `wpa_supplicant` (https://www.archlinux.org/packages/?name=wpa_supplicant) , etc.) and follow the guidelines below. Note that `wireless_tools` (https://www.archlinux.org/packages/?name=wireless_tools) may be optional depending on how recent your wireless hardware is.

Contents

Part I: Identify Card/Install Driver

Identify and Discover if Supported

First you will need to check and see if the Linux kernel has support for your card or if a user-space driver is available for it.

Identify your card

You can find your card type by command:

```
# lspci | grep -i net
```

Or, if you have a USB device, run:

```
# lsusb
```

Note: The internal Wi-Fi card in some laptops may actually be a USB device, so make sure you check both commands.

Discover if the card is supported

- The Ubuntu Wiki (https://help.ubuntu.com/community/WifiDocs/WirelessCardsSupported) has a good list of wireless cards and whether or not they are supported either in the Linux kernel or by a user-space driver (includes driver name).
- Linux Wireless Support (http://linux-wless.passys.nl/) and The Linux Questions' Hardware Compatibility List (http://www.linuxquestions.org/hcl/index.php?cat=10) (HCL) also have a good database of kernel-friendly hardware.
- The kernel page (http://wireless.kernel.org/en/users/Devices) additionally has a matrix of supported hardware.

If your card is not listed

If your wireless hardware is not listed above, likely it is supported only under Windows (some Broadcom, 3com, etc). For these, you will need to use ndiswrapper (http://ndiswrapper.sourceforge.net/wiki/index.php/List) .

Ndiswrapper is a wrapper script that allows you to use some Windows drivers in Linux. See the compatibility list here (http://ndiswrapper.sourceforge.net/mediawiki/index.php/List) . You will need the `.inf` and `.sys` files from your Windows install. If you have a newer card, or a more exotic card, you might want to look up your exact model name and 'linux' and search the Internet before doing this step.

Install user space tools

If you have wired Internet access available

If you have wired Ethernet available and are simply adding wireless functionality to an existing system, and you did not include `wireless_tools (https://www.archlinux.org/packages/?name=wireless_tools)` during initial installation, then install the package `wireless_tools (https://www.archlinux.org/packages/?name=wireless_tools)` .

Note: `wireless_tools (https://www.archlinux.org/packages/?name=wireless_tools)` may not be required depending on the age of your hardware and whether your hardware/drivers support `wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)` . If your configuration is supported well enough to work using only `wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)` , then it is recommended to stick with wpa_supplicant only.

The drivers' corresponding package names are either highlighted in **bold** or via `monospaced font` on this page. The packages can be installed during initial package selection on the Arch Linux installation media and can also be installed later.

If you have only wireless internet available

The `wireless_tools (https://www.archlinux.org/packages/?name=wireless_tools)` package is now available as part of the base system and is also on the live installation media (CD/USB stick image) under the base-devel group.

You cannot initialize wireless hardware without these user-space tools, so ensure they are installed from the installer media, especially if you have no means of networking other than wirelessly. Otherwise, you will be stuck in a "catch 22" when you reboot your newly installed Arch Linux system: you will need `wireless_tools (https://www.archlinux.org/packages/?name=wireless_tools)` and drivers, but in order to get them, you will need `wireless_tools (https://www.archlinux.org/packages/?name=wireless_tools)` and drivers.

Drivers and firmware

The default Arch Linux kernel is *modular*, meaning many of the drivers for machine hardware reside on the hard drive and are available as *modules*. At boot, udev takes an inventory of your hardware. Udev will load appropriate modules (drivers) for your corresponding hardware, and the driver, in turn, will allow creation of a kernel *interface*.

The interface name for different drivers and chipsets will vary. Some examples are *wlan0*, *eth1*, and *ath0*.

Note: Udev is not perfect. If the proper module is not loaded by udev on boot, simply `modprobe` it and add the module name in a `.conf` file in `/etc/modules-load.d/` . Note also that udev may occasionally load more than one driver for a device, and the resulting conflict will prevent successful configuration. Be sure to blacklist the unwanted module.

Methods and procedures for installing kernel modules for various chipsets are covered below. In addition, certain chipsets require the installation of corresponding *firmware*

(also covered below). Read Kernel modules for general informations on operations with modules.

rt2860 and rt2870

From Linux kernel 3.0, the staging driver `rt2860sta` is replaced by the mainline driver `rt2800pci`, and `rt2870sta` is replaced by `rt2800usb`. As a result, the staging drivers are deleted. Source: Kernel commit (https://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commitdiff;h=fefecc6989b4b24276797270c0e229c07be02ad3) . The rt2800 driver automatically works with devices using the rt2870 chipset.

It has a wide range of options that can be configured with `iwpriv`. These are documented in the source tarballs (http://web.ralinktech.com/ralink/Home/Support/Linux.html) available from Ralink.

rt3090

For devices which are using the rt3090 chipset it should be possible to use `rt2860sta` driver. The mainline driver `rt2800pci` is not working with this chipset very well (e.g. sometimes it's not possible to use higher rate than 2Mb/s).

The best way is to use the `rt3090 (https://aur.archlinux.org/packages/rt3090/)` driver from AUR. Compile the `rt3090 (https://aur.archlinux.org/packages/rt3090/)` driver from AUR, delete/move the /etc/Wireless/RT2860STA/RT2860STA.dat firmware file to allow installation of the compiled RT3090 package, blacklist the rt2860sta module and setup the rt3090sta module to load at boot.

Note: This driver also works for rt3062 chipsets.

rt2x00

Unified driver for Ralink chipsets (replaces `rt2500`, `rt61`, `rt73`, etc). This driver has been in the Linux kernel since 2.6.24, but some devices may require extra firmware. It can be configured using the standard `wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)` and `iwconfig` tools.

Some chips require a firmware file, which is installed by default in Arch Linux via the package `linux-firmware (https://www.archlinux.org/packages/?name=linux-firmware)` .

See: Using the new rt2x00 beta driver

rt3573

New chipset as of 2012. It may require proprietary drivers from Ralink. Different manufacturers use it, see Belkin N750 example (https://bbs.archlinux.org/viewtopic.php?pid=1164228#p1164228)

rt5572

New chipset as of 2012 with support for 5 Ghz bands. It may require proprietary drivers from Ralink and some effort to compile them. At the time of writing a how-to on compilation is available for a DLINK DWA-160 rev. B2 here (http://bernaerts.dyndns.org/linux/229-ubuntu-precise-dlink-dwa160-revb2) .

w322u

Treat this Tenda card as an `rt2870sta` device. See: rt2870

rtl8180

Realtek rtl8180 PCI/Cardbus 802.11b is now fully supported in the kernel. It can be configured using the standard `wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)` and `iwconfig` tools.

rtl8187

See: rtl8187

rtl8192e

The driver is part of the current kernel package. It can be configured using the standard `wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)` and `iwconfig` tools.

> **Note:** wicd may cause excessive dropped connections with this driver, while NetworkManager appears to work better.

Module initialization fails

The module initialization may fail at boot giving this error message :

```
rtl819xE:ERR in CPUcheck_firmware_ready()
rtl819xE:ERR in init_firmware() step 2
rtl819xE:ERR!!! _rtl8192_up(): initialization is failed!
r8169 0000:03:00.0: eth0: link down
```

A workaround is to simply unload the module:

```
# modprobe -r r8192e_pci
```

and reload the module (after a pause):

```
# modprobe r8192e_pci
```

rtl8192s

The driver is part of the current kernel package. Firmware may need to be added manually if `/usr/lib/firmware/RTL8192SU/rtl8192sfw.bin` does not exist. (`dmesg` will report "*rtl819xU:FirmwareRequest92S(): failed*" if the firmware is missing)

To download and install firmware:

```
$ wget http://launchpadlibrarian.net/33927923/rtl8192se_linux_2.6.0010.1012.2009.tar.gz
# mkdir /lib/firmware/RTL8192SU
# tar -xzOf rtl8192se_linux_2.6.0010.1012.2009.tar.gz \
 rtl8192se_linux_2.6.0010.1012.2009/firmware/RTL8192SE/rtl8192sfw.bin > \
 /lib/firmware/RTL8192SU/rtl8192sfw.bin
```

Note: An alternate version of the firmware may be found here (http://launchpadlibrarian.net/37387612/rtl8192sfw.bin.gz) , but this version may cause dropped connections.

Note: wicd may cause excessive dropped connections with this driver, while NetworkManager appears to work better.

madwifi-ng

There are three modules maintained by the MadWifi team:

- `ath_pci` is the older driver.
- `ath5k` will eventually phase out `ath_pci` . Currently a better choice for some chipsets, but not all chipsets are supported (see below)
- `ath9k` is the new, official, superior driver for newer Atheros hardware (see below)

For old `ath_pci` driver, install package `madwifi (https://aur.archlinux.org/packages/madwifi/)` and optionally `madwifi-utils-svn (https://aur.archlinux.org/packages/madwifi-utils-svn/)` . Then:

```
# modprobe ath_pci
```

If using `ath_pci` , you may need to blacklist `ath5k` . See Kernel_modules#Blacklisting for instructions.

Some users **may need** to use the `countrycode` option when loading the MadWifi driver in order to use channels and transmit power settings that are legal in their country/region. In the Netherlands, for example, you would load the module like this:

```
# modprobe ath_pci countrycode=528
```

You can verify the settings with the `iwlist` command. See `man iwlist` and the CountryCode page on the MadWifi wiki (http://madwifi-project.org/wiki/UserDocs/CountryCode) . To have this setting automatically applied during boot, refer to Kernel_modules#Configuration, and note the following module option setting:

```
options ath_pci countrycode=528
```

ath5k

`ath5k` is the preferred driver for AR5xxx chipsets including those which are already working with `madwifi-ng` and for some chipsets older than AR5xxx.

If `ath5k` is conflicting with `ath_pci` on your system, blacklist (and unload using `rmmod` or reboot) the following drivers:

```
ath_hal
ath_pci
ath_rate_amrr
ath_rate_onoe
ath_rate_sample
wlan
wlan_acl
wlan_ccmp
wlan_scan_ap
wlan_scan_sta
wlan_tkip
wlan_wep
wlan_xauth
```

then `modprobe ath5k` manually or reboot. *wlan0* (or *wlanX*) in sta mode should spawn and become ready to use.

If the device is unable to lease an IP after being loaded, try `modprobe ath5k nohwcrypt=1`. See below for details about the nohwcrypt option.

Info:

- http://wireless.kernel.org/en/users/Drivers/ath5k
- http://wiki.debian.org/ath5k

Note: Some laptop have problems with their wireless LED indicator flickering red and blue. To solve this problem, do:

```
echo none > "/sys/class/leds/ath5k-phy0::tx/trigger"
echo none > "/sys/class/leds/ath5k-phy0::rx/trigger"
```

For alternatives, look here (https://bugzilla.redhat.com/show_bug.cgi?id=618232) .

Note: If you find web pages randomly loading very slow in Firefox/Opera/Chromium, or if the adapter has problems leasing an IP, try to switch from hardware to software encryption:

```
rmmod ath5k
modprobe ath5k nohwcrypt
```

And restart your connection. If it helps, make the change permanent by adding into `/etc/modprobe.d/010-ath5k.conf` :

```
options ath5k nohwcrypt
```

More about modprobe options: Modprobe#Options

ath9k

`ath9k` is Atheros' officially supported driver for the newer 802.11n chipsets. All of the chips with 802.11n capabilities are supported, with a maximum throughput around 180 Mbps. To see a complete list of supported hardware, check this page (http://wireless.kernel.org/en/users/Drivers/ath9k) .

Working modes: Station, AP and Adhoc.

`ath9k` has been part of the Linux kernel as of v2.6.27. (In the unlikely event that you have stability issues that trouble you, you could try using the compat-wireless (http://wireless.kernel.org/en/users/Download) package. An ath9k mailing list (https://lists.ath9k.org/mailman/listinfo/ath9k-devel) exists for support and development related discussions.)

Info:

- http://wireless.kernel.org/en/users/Drivers/ath9k
- http://wiki.debian.org/ath9k

ath9k_htc

`ath9k_htc` is Atheros' officially supported driver for 802.11n USB devices. Station and Ad-Hoc modes are supported. The driver is included in the kernel. For more information, see http://wireless.kernel.org/en/users/Drivers/ath9k_htc .

ipw2100 and ipw2200

These modules are fully supported in the kernel, but they require additional firmware. It can be configured using the standard `wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)` and `iwconfig` tools.

Depending on which of the chipsets you have, install either `ipw2100-fw (https://www.archlinux.org/packages/?name=ipw2100-fw)` or `ipw2200-fw (https://www.archlinux.org`

`/packages/?name=ipw2200-fw)` .

If installing after initial Arch Linux installation, the module may need to be reloaded for the firmware to be loaded; run the following as root:

```
rmmod ipw2200
modprobe ipw2200
```

Enabling the radiotap interface

Launch the following as root:

```
rmmod ipw2200
modprobe ipw2200 rtap_iface=1
```

Enabling the LED

Most laptops will have a front LED to indicate when the wireless is connected (or not). Add the following to `/etc/modprobe.d/ipw2200.conf` :

```
options ipw2200 led=1
```

iwl3945, iwl4965 and iwl5000-series

Intel's open source **W**i-Fi drivers for **L**inux (See iwlwifi (http://intellinuxwireless.org)) will work for both the 3945 and 4965 chipsets since kernel 2.6.24. And iwl5000-series chipsets (including 5100BG, 5100ABG, 5100AGN, 5300AGN and 5350AGN) have been supported since **kernel 2.6.27**, by the in-tree driver **iwlagn**.

Since the 2.6.34 kernel update, the firmware files were moved to the `linux-firmware` package. Manually installing firmware packages is not required.

Loading the Driver

udev should load the driver automatically. To manually load the driver at start-up, read Kernel modules#Loading, and add `iwl3945` or `iwl4965` respectively to the new file. For example:

```
# Load Intel Wi-Fi modules
iwl3945
```

The drivers should now load after a reboot, and running `ip addr` from a terminal should report *wlan0* as a new network interface.

Disabling LED blink

The default settings on the module are to have the LED blink on activity. Some people find this extremely annoying. To have the LED on solid when Wi-Fi is active:

```
# echo 'w /sys/class/leds/phy0-led/trigger - - - - phy0radio' > /etc/tmpfiles.d/phy0-led.conf
# systemd-tmpfiles --create phy0-led.conf
```

To see all the possible trigger values for this LED:

```
# cat /sys/class/leds/phy0-led/trigger
```

Here is an example for the old way, if you do not have `/sys/class/leds/phy0-led` :

```
# echo "options iwlcore led_mode=1" >> /etc/modprobe.d/modprobe.conf
# rmmod iwlagn
# rmmod iwlcore
# modprobe iwlcore
# modprobe iwlagn
```

On Linux kernels 2.6.39.1-1 and up, the `iwlcore` module was deprecated. Use `options iwlagn led_mode=1` or `options iwl_legacy led_mode=1` instead (find out what module is loaded with `lsmod`).

Note: iwl_legacy was renamed iwlegacy in Linux kernel 3.3.1. For this version, use `options iwlegacy led_mode=1` .

Other Notes

- The MS Windows NETw4x32 driver can be used with ndiswrapper as an alternative to the `iwl3945` and `ipw3945` drivers.
- In some cases (specifically a Dell Latitude D620 with Arch 2008.06, though it could happen elsewhere), after installation you may have both `iwl3945` and `ipw3945` modules loaded. The card will not work with both modules loaded, so you will have to blacklist the `ipw3945` module.
- By default, `iwl3945` is configured to only work with networks on channels 1-11. Higher frequency bands are not allowed in some parts of the world (e.g. the US). In the EU however, channels 12 and 13 are used quite commonly (and Japan allows for channel 14). To make `iwl3945` scan for all channels, add `options cfg80211 ieee80211_regdom=EU` to `/etc/modprobe.d/modprobe.conf` . With `iwlist f` you can check which channels are allowed.
- If you want to enable more channels on Intel Wifi 5100 (and quite possible other cards too), you can do that with the `crda (https://www.archlinux.org/packages/?name=crda)` package. After installing the package, edit `/etc/conf.d/wireless-regdom` and uncomment the line where your country code is found. When executing `sudo iwlist wlan0 channel` , you should now have access to more channels (depending on your location).

orinoco

This should be a part of the kernel package and be installed already.

> **Note:** Some Orinoco chipsets are Hermes I/II. You can use the AUR package `wl_lkm (https://aur.archlinux.org/packages/wl_lkm/)` to replace the `orinoco` driver and gain WPA support. See this post (http://ubuntuforums.org /showthread.php?p=2154534#post2154534) for more information.

To use the driver, blacklist `orinoco_cs` , and then add `wlags49_h1_cs` .

ndiswrapper

Ndiswrapper is not a real driver, but you can use it when there are no native Linux kernel drivers for your wireless chipset, so it is very useful in some situations. To use it, you need the `*.inf` file from your Windows driver (the `*.sys` file must also be present in the same directory). Be sure to use drivers appropriate to your architecture (e.g. 32/64bit). If you need to extract these files from an `*.exe` file, you can use `cabextract (https://www.archlinux.org/packages/?name=cabextract)` .

Follow these steps to configure ndiswrapper.

1. Install the driver to `/etc/ndiswrapper/*`

```
ndiswrapper -i filename.inf
```

2. List all installed drivers for ndiswrapper

```
ndiswrapper -l
```

3. Write configuration file in `/etc/modprobe.d/ndiswrapper.conf`

```
ndiswrapper -m
depmod -a
```

Now the ndiswrapper install is almost finished; follow the instructions on Kernel modules#Loading to automatically load the module at boot.

The important part is making sure that ndiswrapper exists on this line, so just add it alongside the other modules. It would be best to test that ndiswrapper will load now, so:

```
modprobe ndiswrapper
iwconfig
```

and *wlan0* should now exist. Check this page if you are having problems: Ndiswrapper installation wiki (http://ndiswrapper.sourceforge.net/joomla/index.php?/component /option,com_openwiki/Itemid,33/id,installation/) .

prism54

Download the firmware driver for your appropriate card from this site (http://linuxwireless.org/en/users/Drivers/p54) . Rename the firmware file to `isl3890` . If non-existent, create the directory `/usr/lib/firmware` and move the file `isl3890` inside it. This should do the trick. [1] (https://bbs.archlinux.org/viewtopic.php?t=16569&start=0&postdays=0&postorder=asc&highlight=siocsifflags+such+file++directory)

If that did not work, try this:

- Reload the prism module (`modprobe p54usb` or `modprobe p54pci` , depending on your hardware)

Alternatively, remove your Wi-Fi card and then reconnect it.

- Use the `dmesg` command, and look at the end of the output it prints out.

Look for a section similar to this:

```
firmware: requesting isl3887usb_bare
p54: LM86 firmware
p54: FW rev 2.5.8.0 - Softmac protocol 3.0
```

and try renaming the firmware file to the name corresponding to the part bolded here.

If you get the message

```
SIOCSIFFLAGS: Operation not permitted
```

when performing `ip link set wlan0 up` OR

```
prism54: Your card/socket may be faulty, or IRQ line too busy :(
```

appears in `dmesg` 's output this may be because you have both the deprecated kernel module `prism54` and one of the newer kernel modules (`p54pci` or `p54usb`) loaded at the same time and they are fighting over ownership of the IRQ. Use the command `lsmod | grep prism54` to see if the deprecated module is being loaded. If so, you need to stop `prism54` from loading by blacklisting it (there are several ways to do this which are described elsewhere). Once blacklisted, you may find you have to rename the firmware as `prism54` and `p54pci` / `p54usb` look for different firmware filenames (i.e. recheck the `dmesg` output after performing `ip link set eth0 up`).

ACX100/111

Packages: `tiacx` `tiacx-firmware`

The driver should tell you which firmware it needs; check `/var/log/messages.log` or use the `dmesg` command.

Link the appropriate firmware to `/usr/lib/firmware` :

```
ln -s /usr/share/tiacx/acx111_2.3.1.31/tiacx111c16 /usr/lib/firmware
```

For another way to determine which firmware revision number to use, see the "Which firmware" section (http://acx100.sourceforge.net/wiki/Firmware) of the acx100.sourceforge wiki. For ACX100, you can follow the links provided there to a table of card model numbers vs. "firmware files known to work"; you can figure out the rev. number you need, by looking at the suffix there. For example, a dlink_dwl650+ uses "1.9.8.b", in which case you would do this:

```
ln -s /usr/share/tiacx/acx100_1.9.8.b/* /usr/lib/firmware
```

If you find that the driver is spamming your kernel log, for example because you are running Kismet with channel-hopping, you could put this in `/etc/modprobe.d/modprobe.conf` :

```
options acx debug=0
```

Note: The open-source `acx` driver does not support WPA/RSN encryption. Ndiswrapper will have to be used with the Windows driver to enable the enhanced encryption. See ndiswrapper, this page, for more details.

b43, broadcom-wl and brcmsmac (previously brcm80211)

See the Broadcom wireless page.

zd1211rw

`zd1211rw` (http://zd1211.wiki.sourceforge.net/) is a driver for the ZyDAS ZD1211 802.11b/g USB WLAN chipset, and it is included in recent versions of the Linux kernel. See [2] (http://www.linuxwireless.org/en/users/Drivers/zd1211rw/devices) for a list of supported devices. You only need to install the firmware for the device, provided by the `zd1211-firmware (https://www.archlinux.org/packages/?name=zd1211-firmware)` package.

carl9170

`carl9170` (http://wireless.kernel.org/en/users/Drivers/carl9170/) is the 802.11n USB driver with GPLv2 firmware for Atheros USB AR9170 devices. It supports these devices (http://wireless.kernel.org/en/users/Drivers/carl9170#available_devices) . The **firmware** is not yet part of the `linux-firmware (https://www.archlinux.org/packages/?name=linux-firmware)` package; it is available in the AUR (`carl9170-fw (https://aur.archlinux.org/packages/carl9170-fw/)`). The **driver** is a part of the Linux kernel v2.6.37 and higher.

In order to use this driver, the following older driver modules must be blacklisted:

- `arusb_lnx`
- `ar9170usb`

hostap_cs

Host AP is the Linux driver for Prism2/2.5/3 like WCP11. `hostap_cs` should be a part of the `linux` package and should be installed already.

`orinico_cs` can cause problems, so it must be blacklisted. After blacklisting, the driver should work.

More information : Home page (http://hostap.epitest.fi/)

compat-drivers-patched

Patched compat wireless drivers correct the "fixed-channel -1" issue, whilst providing better injection. Please install the `compat-drivers-patched (https://aur.archlinux.org/packages/compat-drivers-patched/)` package from the AUR.

`compat-drivers-patched (https://aur.archlinux.org/packages/compat-drivers-patched/)` does not conflict with any other package and the modules built reside in `/usr/lib/modules/`*`your_kernel_version`*`/updates`.

These patched drivers come from the Linux Wireless project (http://wireless.kernel.org/) and support many of the above mentioned chips such as:

```
ath5k ath9k_htc carl9170 b43 zd1211rw rt2x00 wl1251 wl12xx ath6kl brcm80211
```

Supported groups:

```
atheros ath iwlagn rtl818x rtlwifi wl12xx atlxx bt
```

It is also possible to build a specific module/driver or a group of drivers by editing the PKGBUILD, particularly uncommenting the **line #46**. Here is an example of building the atheros group:

```
scripts/driver-select atheros
```

Please read the package's PKGBUILD for any other possible modifications prior to compilation and installation.

Test installation

After loading your driver, run `ip link` to ensure a wireless interface (e.g. *wlanX*, *ethX*, *athX*) is created.

If no such interface is visible, modprobing it might work. To start your driver, use the `rmmod` and `modprobe` commands. If `rmmod` fails, continue with `modprobe`. See Kernel modules for more info.

Example: If your driver is called "driverXXX", you would run the following commands:

```
# rmmod driverXXX
# modprobe driverXXX
```

Bring the interface up with `ip link set <interface> up`. For example, assuming the interface is *wlan0*:

```
# ip link set wlan0 up
```

If you get this error message: `SIOCSIFFLAGS: No such file or directory`, it most certainly means your wireless chipset requires a firmware to function, which you need to install as explained above.

Part II: Wireless management

Assuming that your drivers are installed and working properly, you will need to choose a method for managing your wireless connections. The following subsections will help you decide the best way to do just that.

Procedure and tools required will depend on several factors:

- The desired nature of configuration management; from a completely manual command line setup procedure to a software-managed, automated solution.
- The encryption type (or lack thereof) which protects the wireless network.
- The need for network profiles, if the computer will frequently change networks (such as a laptop).

The manual method requires more work from you, but gives you much more control over your configuration. Usually you will have to enter a set of commands which have no persistant effect, i.e. they won't apply after a reboot. Either you enter those commands on every boot which may be quite cumbersome, or you put all these commands in a shell script to automate the process. This script can even be executed automatically at boot time. See Arch Boot Process.

Management methods

The following table shows the different methods that can be used to activate and manage a wireless network connection, depending on the encryption and management types, and the various tools that are required. Although there may be other possibilities, these are the most frequently used:

Management	No encryption/WEP	WPA/WPA2 PSK
Manual	iproute2 (https://www.archlinux.org/packages/?name=iproute2) + iwconfig + dhcpcd / iproute2	iproute2 + iwconfig + wpa_supplicant + dhcpcd / iproute2

Automatically managed, with network profiles support	netctl, netcfg, Wicd, NetworkManager, etc.

More choice guide:

Management	Auto connect at boot	Auto connect if dropped or changed location	support 3G Modem	GUI	Console tools
Netctl	Yes	Yes	-	Yes	`netctl`
Netcfg	Yes	Yes	-	Yes	`wifi-select`
Wicd	Yes	Yes	-	Yes	`wicd-curses`
NetworkManager + `network-manager-applet (https://www.archlinux.org/packages/?name=network-manager-applet)`	Yes	Yes	Yes	Yes	`nmcli`

Please note that the Linux wireless extensions and corresponding commands like `iwconfig` or `iwlist` have become deprecated and are replaced by `iw`, which has to be installed seperately from the core-repository. This is not fully reflected in this wiki yet and both still work. A comparison of common commands is found on Linuxwireless (http://linuxwireless.org/en/users/Documentation/iw/replace-iwconfig) .

Tip:

- Whatever your choice, **you should try to connect using the manual method first**. This will help you understand the different steps that are required and debug them in case a problem arose.
- If possible (e.g. if you manage your Wi-Fi access point), try connecting with no encryption, to check everything works. Then try using encryption, either WEP (simpler to configure -- but crackable in a matter of seconds, so it is hardly more secure than an unencrypted connection), WPA, or WPA2.

When it comes to ease of use, NetworkManager (with GNOME's `network-manager-applet (https://www.archlinux.org/packages/?name=network-manager-applet)`) and `wicd` have good GUI's and can provide a list of available networks to connect, and they prompt for passwords, which is straightforward and highly recommended. WPA Supplicant has also a GUI configuration tool, `wpa_supplicant_gui (https://www.archlinux.org/packages/?name=wpa_supplicant_gui)` .

Note: GNOME's `network-manager-applet (https://www.archlinux.org/packages/?name=network-manager-applet)` also works under Xfce if you install `xfce4-xfapplet-plugin (https://aur.archlinux.org/packages/xfce4-xfapplet-plugin/)` first. `xfce4-xfapplet-plugin (https://aur.archlinux.org/packages/xfce4-xfapplet-plugin/)` is in the AUR, but it is orphaned

and may not work. Additionally, there are applets available for KDE.

Manual setup

The programs provided by the package `wireless_tools (https://www.archlinux.org/packages/?name=wireless_tools)` are the basic set of tools to set up a wireless network. Additionally the `iw (https://www.archlinux.org/packages/?name=iw)` package provides the new tool. Moreover, if you use WPA/WPA2 encryption, you will need the package `wpa_supplicant (https://www.archlinux.org/packages/?name=wpa_supplicant)` . These powerful user-space console tools work extremely well and allow complete, manual control from the shell.

These examples assume your wireless device is *wlan0*. Replace *wlan0* with the appropriate device name.

Note: Depending on your hardware and encryption type, some of these steps may not be necessary. Some cards are known to require interface activation and/or access point scanning before being associated to an access point and being given an IP address. Some experimentation may be required. For instance, WPA/WPA2 users may directly try to activate their wireless network from step 3.

Operating mode

(Optional, may be required) At this step you may need to set the proper operating mode of the wireless card. More specifically, if you are going to connect an ad-hoc network, you might need to set the operating mode to *ad-hoc:*

```
# iw wlan0 set type ibss
```

Note: Ideally, you should already know which type of network you are going to connect to. If you do not, scan the network as described in step 2 below, then, if necessary, return back to this step and change the mode. Also, please keep in mind that changing the operating mode might require the wireless interface to be *down* (`ip link set wlan0 down`).

Interface activation

(Also optional, may be required) Some cards require that the kernel interface be activated before you can use the `wireless_tools` :

```
# ip link set wlan0 up
```

Access point discovery

See what access points are available:

```
# iw dev wlan0 scan | less
```

or

```
$ iwlist wlan0 scanning | less
```

Note: If it displays "*Interface doesn't support scanning*" then you probably forgot to install the firmware. You can also try bringing up the interface first as shown in point 1. In some cases this message is also displayed when not running iw as root. Also, your wireless network card may be soft-blocked. Try getting `rfkill` (https://www.archlinux.org/packages/?name=rfkill) and running `rfkill list all` to check.

The important points to check:

- ESSID: the "name" of the access point.
- Quality: in general try something above 40/70.
- Encryption key: if it is "on", check if you can see any line regarding
  - WEP, WPA, or RSN. Note that RSN and WPA2 are different names for the protocol.
  - Group cipher: value in TKIP, CCMP, both, others.
  - Pairwise ciphers: value in TKIP, CCMP, both, others. Not necessarily the same value than Group cipher.
  - Authentication Suites: value in PSK, 802.1x, others. For home router, you'll usually find PSK (*i.e.* passphrase). In universities, you are more likely to find 802.1x suite which requires login and password. Then you will need to know which key management is in use (e.g. EAP), and what encapsulation it uses (e.g. PEAP). Find more details at Wikipedia:List_of_authentication_protocols and the sub-articles.

Association

Depending on the encryption, you need to associate your wireless device with the access point to use and pass the encryption key.

Assuming you want to use the ESSID `MyEssid` :

Note: The essid is usually just the name of the network you want to connect to.

- **No encryption**

```
# iwconfig wlan0 essid "MyEssid"
```

Or, alternatively, for the new netlink interface

```
# iw wlan0 connect MyEssid
```

- **WEP**

using a hexadecimal key:

```
# iwconfig wlan0 essid "MyEssid" key 1234567890
```

using an ASCII key:

```
# iwconfig wlan0 essid "MyEssid" key s:asciikey
```

- **WPA/WPA2**

You need to edit the `/etc/wpa_supplicant.conf` file as described in WPA_Supplicant and according to what you got from #Access point discovery. Then, issue this command:

```
# wpa_supplicant -i wlan0 -c /etc/wpa_supplicant.conf
```

This is assuming your device uses the `wext` driver. If this does not work, you may need to adjust these options. If connected successfully, continue in a new terminal (or quit `wpa_supplicant` with **`Ctrl+c`** and add the `-B` switch to the above command to run it in the background). WPA_Supplicant contains more information and troubleshooting.

Regardless of the method used, you can check if you have associated successfully as follows:

```
# iwconfig wlan0
```

Or, alternatively, for the new netlink interface

```
# iw dev wlan0 link
```

Note: In some setups it may still display "Access Point: Not-Associated", continue on to the next step.

Getting an IP address

Finally, provide an IP address to the network interface. Simple examples are:

```
# dhcpcd wlan0
```

for DHCP, or

```
# ip addr add 192.168.0.2/24 dev wlan0
# ip route add default via 192.168.0.1
```

for static IP addressing.

Note: If you get a timeout error due to a *waiting for carrier* problem, then you might have to set the channel mode to `auto` for the specific device.

```
# iwconfig wlan0 channel auto
```

Before changing the channel to auto, make sure your wireless interface (in this case, 'wlan0') is **down**. After it has successfully changed it, you can again bring the interface up and continue from there.

Note: Although the manual configuration method will help troubleshoot wireless problems, you will have to re-type every command each time you reboot. You can also quickly write a shell script to automate the whole process, which is still a quite convenient way of managing networks while keeping full control over your configuration.

Manual wireless connection at boot using systemd and dhcpcd

To have systemd connect to a manually configured wireless network at boot:

Create `/etc/conf.d/network` to store your interface or static IP settings in:

/etc/conf.d/network

```
interface=wlan0
address=192.168.0.10
netmask=24
broadcast=192.168.0.255
gateway=192.168.0.1
```

Create a systemctl unit e.g: `/etc/systemd/system/network.service` . This example uses dhcpcd and WPA supplicant.

/etc/systemd/system/network.service

```
[Unit]
Description=Network Connectivity
Wants=network.target
Before=network.target
BindsTo=sys-subsystem-net-devices-${interface}.device
After=sys-subsystem-net-devices-${interface}.device

[Service]
Type=oneshot
RemainAfterExit=yes
EnvironmentFile=/etc/conf.d/network
ExecStart=/sbin/ip link set dev ${interface} up
ExecStart=/usr/sbin/wpa_supplicant -B -i ${interface} -c /etc/wpa_supplicant.conf
ExecStart=/sbin/dhcpcd ${interface}

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

Or without `/etc/conf.d/network` :

```
/etc/systemd/system/network.service
```

```
[Unit]
Description=Network Connectivity
Wants=network.target
Before=network.target
BindsTo=sys-subsystem-net-devices-wlan0.device
After=sys-subsystem-net-devices-wlan0.device

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/sbin/ip link set dev wlan0 up
ExecStart=/usr/sbin/wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant.conf
ExecStart=/sbin/dhcpcd wlan0

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

Do not forget to enable it!

```
# systemctl enable network
```

To test, reboot or make sure all other network daemons are stopped and then issue as superuser

```
# systemctl start network
```

Systemd with wpa_supplicant and static IP

This example configuration uses the new systemd-197 interface naming scheme.

See http://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames

https://mailman.archlinux.org/pipermail/arch-dev-public/2013-January/024231.html

Run this script as non-root to find your interface names:

```
for i in /sys/class/net/*; do
  echo "==$i"
  udevadm test-builtin net_id "$i";
  echo
done 2>/dev/null
```

Create `/etc/conf.d/network`

```
/etc/conf.d/network
```

```
address=192.168.0.10
netmask=24
broadcast=192.168.0.255
gateway=192.168.0.1
```

Install `wpa_supplicant` and create `/etc/wpa_supplicant.conf`. See WPA supplicant

```
/etc/wpa_supplicant.conf
```

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=network
update_config=1
network={
        ssid="My-Wireless"
        psk=b705a6bfcd5639d5c40cd972cd4048cfb94572987f30d324c82036317b91a138
}
```

Create a systemd unit file containing the name of the interface:
`/etc/systemd/system/network@wlp0s26f7u3.service`

```
/etc/systemd/system/network@wlp0s26f7u3.service
```

```
[Unit]
Description=Network Connectivity (%i)
Wants=network.target
Before=network.target
BindsTo=sys-subsystem-net-devices-%i.device
After=sys-subsystem-net-devices-%i.device

[Service]
Type=oneshot
RemainAfterExit=yes
EnvironmentFile=/etc/conf.d/network
ExecStart=/usr/sbin/ip link set dev %i up
ExecStart=/usr/sbin/wpa_supplicant -B -i %i -c /etc/wpa_supplicant.conf
ExecStart=/usr/sbin/ip addr add ${address}/${netmask} broadcast ${broadcast} dev %i
ExecStart=/usr/sbin/ip route add default via ${gateway}
ExecStop=/usr/sbin/ip addr flush dev %i
ExecStop=/usr/sbin/ip link set dev %i down

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

Enable the unit and start it.

```
# systemctl enable network@wlp0s26f7u3.service
# systemctl start network@wlp0s26f7u3.service
```

Automatic setup

There are many solutions to choose from, but remember that all of them are mutually exclusive; you should not run two daemons simultaneously.

Netctl

`netctl` is a replacement for netcfg designed to work with systemd.

See: Netctl

Netcfg

`netcfg` provides a *versatile, robust and fast* solution to networking on Arch Linux.

netcfg uses a profile based setup and is capable of detection and connection to a wide range of network types. This is no harder than using graphical tools.

See: Netcfg

Wicd

Wicd is a network manager that can handle both wireless and wired connections. It is written in Python and Gtk with fewer dependencies than NetworkManager, making it an ideal solution for lightweight desktop users. Wicd is available in the official repositories.

See: Wicd

NetworkManager

NetworkManager is an advanced network management tool that is enabled by default in most popular GNU/Linux distributions. In addition to managing wired connections, NetworkManager provides worry-free wireless roaming with an easy-to-use GUI program for selecting your desired network.

If you do not use GNOME but use a window manager like Openbox or xmonad, do not forget to install `polkit-gnome (https://www.archlinux.org/packages/?name=polkit-gnome)` , `gnome-keyring (https://www.archlinux.org/packages/?name=gnome-keyring)` , `libgnome-keyring (https://www.archlinux.org/packages/?name=libgnome-keyring)` , and `pyxdg (https://www.archlinux.org/packages/?name=pyxdg)` to manage WEP, WPA, and WPA2 connections.

See: NetworkManager

WiFi Radar

WiFi Radar is a Python/PyGTK2 utility for managing wireless profiles (and *only* wireless). It enables you to scan for available networks and create profiles for your preferred networks.

See: Wifi Radar

wlassistant

wlassistant is a very intuitive and straight-forward GUI application for managing your wireless connections.

Install the `wlassistant (https://aur.archlinux.org/packages/wlassistant/)` package from the AUR.

wlassistant must be run with root privileges:

```
# wlassistant
```

One method of using wlassistant is to configure your wireless card within `/etc/rc.conf`, specifying the access point you use most often. On start-up, your card will automatically be configured for this ESSID, but if other wireless networks are needed/available, `wlassistant` can then be invoked to access them. Background the `network` daemon in `/etc/rc.conf`, by prefixing it with a `@` to avoid boot-up delays.

Power saving

See Power_saving#Wireless_power_saving.

See also

- Sharing PPP Connection
- Ad-hoc networking

External links

- NetworkManager (http://www.gnome.org/projects/NetworkManager/) -- The official website for NetworkManager
- WICD (http://wicd.sourceforge.net/) -- The official website for WICD
- WiFi Radar (http://wifi-radar.berlios.de/) -- WiFi Radar information page
- The MadWifi project's method of installing (http://madwifi-project.org/wiki/UserDocs/FirstTimeHowTo) -- Recommended if you are having trouble after reading this article

Retrieved from "https://wiki.archlinux.org/index.php?title=Wireless_Setup&oldid=258966"
Category: Wireless Networking
