Installing Madwifi on Slackware

Wireless support in Slackware has much improved since Slackware 10.0 (the first release to support non-PCMCIA wireless cards out of the box). The madwifi driver for Atheros based chipsets works fine with the current 2.6.x series of kernels and the 2.4.x kernels found in Slackware releases prior to 12.0. The Slackware configuration files support unencrypted, WEP- and WPA-protected connections. For WPA encryption, the wpa_supplicant package (part of Slackware since 12.0) is an additional requirement.
If you need WPA encryption, be sure to read the Support for WPA encryption section of this page.

This Wiki page explains how to install an appropriate Madwifi package on your Slackware computer, and gives directions on how to build your own package if you need to (for instance if you run a non-Slackware kernel).

The madwifi driver is capable of creating so-called *virtual access points* or VAPs. This is being done on a base device called *wifi0* which will show up in your listings of ifconfig and iwconfig, and is not linked to the wireless extensions. You should never have to use this *wifi0* network device. The *user station* (a VAP of type “sta”) will by created by default when the kernel module loads and is called *ath0*. This *ath0* is the real network device, which you will be configuring and using.
People who want to create something other than a user station, for instance a real Access Point (master mode) will want to read this MadWiki documentation.

Obtaining slackware packages for madwifi

Binary Slackware packages for the madwifi driver can be found at slackware.com. They can be installed onto your computer using `installpkg` or `upgradepkg`. They are packages for specific kernels. The package naming convention is *madwifi-${VERSION}_${KERNELVERSION}-i486-${NUMBER}.tgz*. Here, *${VERSION}* is the version of the madwifi source package, and *${KERNELVERSION}* is the version of the kernel that the package is meant for. The *${NUMBER}* is the build number. In case there are multiple packages where only the build number differs, then the package with the highest build number is the most recent and should be used.

Building a slackware package for madwifi from source

You can build your own package for any kernel version that you are running, too. If you want to know if any of the downloadable packages matches your running kernel, you can check the version of your running kernel with the command

```
uname -r
```

- Start with downloading all the source files here to a directory on your local hard disk. This command will do exactly that:

  ```
  lftp -c "open http://www.slackware.com/~alien/slackbuilds/madwifi/;
  mirror build"
  ```

This will create a subdirectory `build` in your current directory. Change to that directory, `su -` to root if you haven't done so already, and start the `madwifi.SlackBuild` script after making it executable:

```
chmod +x madwifi.SlackBuild
./madwifi.SlackBuild
```

A binary Slackware package will be created in directory /tmp and have a name of `madwifi-${VERSION}_${KERNELVERSION}-i486-1.tgz`. Here, `${VERSION}` is the version of the madwifi source package, and `${KERNELVERSION}` is the version of your running kernel.

- If you want to build a package for another kernel that you've already installed on your computer, and you don't want to reboot into that kernel for building the madwifi package, you can run the script like this (the example builds a package for Linux kernel `2.6.23.16-smp`):

```
KVER=2.6.23.16-smp ./madwifi.SlackBuild
```

If the kernel source tree cannot be found (i.e. if the kernel source tree is not found where the link /lib/modules/${KVER}/build points) you can specify your kernel source directory as well (again, this is only an example):

```
KVER=2.6.23.16-smp KSRC=~/src/linux-2.6.23.16 ./madwifi.SlackBuild
```

If you downloaded a newer/other version of the madwifi sources and you want to build a package out of that, you will have to edit the madwifi.SlackBuild script, and substitute the correct value for the **VERSION** parameter in the line

```
VERSION=<value>
```

before executing the script.
Be warned that the madwifi sources are in constant development, so the possibility exists that the madwifi.SlackBuild script will not produce correct results for newer (or older) madwifi sources.

- Install the madwifi package with

```
installpkg madwifi-VERSION_KERNELVERSION-i486-BUILDNR.tgz
```

or if you already had a previous package installed, use

```
upgradepkg madwifi-VERSION_KERNELVERSION-i486-BUILDNR.tgz
```

Loading the kernel modules automatically

You can use hotplug/udev to load the madwifi driver automatically. There is nothing to configure; hotplug/udev takes care of everything. If you do not use hotplug or udev, you can add a line like this to `/etc/rc.d/rc.modules`:

```
/sbin/modprobe ath_pci
```

which will then load the kernel module when the computer boots.

If you need to pass parameters to the madwifi driver, like a different country code, you can create a file in the `/etc/modprobe.d/` directory which contains these parameters. The filename is yours to choose - let's use 'madwifi' as the name. For example, suppose you need to tell the driver to use country code 25. You create the file "`/etc/modprobe.d/madwifi`" and add the following single line to it:

```
options ath_pci countrycode=25
```

(by the way, a list of country codes is available on the madwifi Wiki).

If your (older) version of Slackware does not have a `/etc/modprobe.d` directory, you can add the line

```
options ath_pci countrycode=25
```

to the file `/etc/modprobe.conf` (for 2.6 kernels) or `/etc/modules.conf` (for 2.4 kernels) instead.

Network configuration

rc.wireless.conf

If you have a Wireless Access Point that is broadcasting its station ID (the *ESSID*), and is not configured for encrypted traffic, then you're ready to go with the default configuration as it comes with Slackware. This kind of open wireless network is typical when

1. you just took your Wireless Access Point out of the box you bought it in, and didn't have time yet to configure it;
1. you are at an airport/hotel/pub where they offer free wireless access.

If you need to configure specific parameters to make the wireless card talk to your Access Point - for instance, the ESSID (in case the Access Point is hiding its station ID), or the channel, or a WEP key, etc) then you will need to edit either the file

```
/etc/rc.d/rc.wireless.conf
```

or the file

```
/etc/rc.d/rc.inet1.conf
```

(one of the two will do) and add a specific configuration that matches your wireless card and Access Point. Let's use `/etc/rc.d/rc.wireless.conf` as an example of how to configure your wireless.

You will notice that the content of `/etc/rc.d/rc.wireless.conf` is basically a number of sections that apply to certain (ranges of) wireless network cards. The distinguishing factor is the hardware address (the *MAC address*) of a card. A section for a specific card or range of cards looks like this:

```
MAC_Address)
    INFO="a string that decribes your card type"
    PARAMETER1="value1"
    PARAMETER2="value2"
    [more parameters] .......
    ;;
```

The MAC_Address in this example can be a full MAC address (six *HEX* bytes separated by colons, like `00:12:8E:A0:32:DC`) that matches a single network card, or a wildcard address that matches a whole range of cards, typically all cards from a specific vendor (like `00:12:8E:A0:*`).
You are going to add such a section for your card, and this is how to do it:

- After the module is loaded and your ath0 interface is available, run `ifconfig ath0` and get the MAC address of your wireless interface (called `ath0` for a madwifi supported card).
- Edit /etc/rc.d/rc.wireless.conf and comment out this section right in the beginning of the file:

  ```
  *)
      INFO="Any ESSID"
      ESSID="any"
      ;;
  ```

  so that it will look like this:

  ```
  # *)
  #     INFO="Any ESSID"
  #     ESSID="any"
  #     ;;
  ```

- Somewhere further below the lines you just commented out, add a few lines (easiest is to add it to the bottom of the file for instance, **right above** the `esac` line) that will apply to your card, like these:

  ```
  00:06:25:13:2B:D4)
      INFO="D-LINK DWL-G510 revB1"
      ESSID="your_ap_essid"
      KEY="0100030203"
      ;;
  ```

  The first line is your card's MAC address followed by a ')', and the last line must be two semicolons all by themselves (copy and paste one of the available examples if you're unsure). Your MAC address, ESSID, KEY and info comment are obviously going to be different from the values in the above example.

rc.inet1.conf

Slackware since release 10.2 understands network interfaces whose names do not start with `eth`. A network card that shows up as **`ath0`** can not be setup with an IP address using the configuration files that are part of pre-Slackware 10.2 releases. We are going to assume here that you are running Slackware 10.2 or newer. For older releases, read Updating the network scripts and Configuration the manual way.

- You will need to add or modify a few lines in /etc/rc.d/rc.inet1.conf in order to get a configuration like this:

```
# Config information for ath0 (using dhcp):
IFNAME[1]="ath0"
IPADDR[1]=""
NETMASK[1]=""
USE_DHCP[1]="yes"
DHCP_HOSTNAME[1]="mywirelessbox"
```

  Or like this:

```
# Config information for ath0 (using static IP address):
IFNAME[1]="ath0"
IPADDR[1]="192.168.3.11"
NETMASK[1]="255.255.255.0"
USE_DHCP[1]=""
DHCP_HOSTNAME[1]=""
GATEWAY="192.168.3.1"
```

  These are example values of course and you will have to substitute your own.

- NOTE
  In the above example, where I used the index 1, like in: VARIABLENAME[**1**], you may use whatever index is not used. If you do not have an eth0 interface for instance, you might as well want to use the unused 0 array index. The last configuration example would then look like this:

```
# Config information for ath0 (using static IP address):
IFNAME[0]="ath0"
IPADDR[0]="192.168.3.11"
NETMASK[0]="255.255.255.0"
USE_DHCP[0]=""
DHCP_HOSTNAME[0]=""
GATEWAY="192.168.3.1"
```

  Obviously, any array index value ([0],[1],[2],) in /etc/rc.d/rc.inet1.conf should be used for exactly one card's configuration. If you copy a set of lines, be sure to change the array index to an unused value. If you forget this, and create a double entry, then Slackware will happily forget about the first, and will use only the last value for any parameter found in the file.

- NOTE
  Any configuration setting that is available in *rc.wireless.conf* can also be configured in

rc.inet1.conf by prefixing the parameter with "*WLAN_*". For instance, suppose you want to configure WPA. You could add the following to your card's section in *rc.wireless.conf*

```
WPA="wpa_supplicant"
WPADRIVER="wext"
```

but this is how you could do it in *rc.inet1.conf* as well:

```
IFNAME[1]="ath0"
   ...
WLAN_WPA[1]="wpa_supplicant"
WLAN_WPADRIVER[1]="wext"
```

You can configure your card in either one, or both of the configuration files, but the settings in *rc.inet1.conf* will always have priority. It depends on your own taste which of the two configuration files you want to put your configuration in, but if I may make a suggestion: don't use `rc.wireless.conf` and group all of the network configuration (wireless as well as non-wireless) for an interface nicely together in `rc.inet1.conf`.

(Re) starting the network interface

If you're using Slackware 10.2 or newer, or in case you run an older release but updated your network scripts, you can start your ath0 interface like this:

```
/etc/rc.d/rc.inet1 ath0_start
```

or restart it like this (after making changes to the above configuration files for instance):

```
/etc/rc.d/rc.inet1 ath0_restart
```

Earlier versions of Slackware will (re-)start all configured interfaces at once, because all you can run is

```
/etc/rc.d/rc.inet1
```

Updating the network scripts (Slackware 10.1 and older)

The network scripts of Slackware 10.2 or later can be used in older releases of Slackware as well. You will need /etc/rc.d/rc.wireless* and /etc/rc.d/rc.inet1*
If you have difficulties extracting these files from a Slackware CD or Internet server, you can find them here too.

Network configuration the manual way (Slackware 10.1 and older)

These are the relatively easy ways to get slackware to bring up your interface:

1. Run `/etc/rc.d/rc.wireless` to configure the wireless parameters for your card.
2. Run `dhcpcd ath0` which should be enough to get you up and running;

3. It depends on your local network and your settings if you want `ifconfig` (instead of `dhcpcd`) and possibly additional `iwconfig` commands.
4. Put a `dhcpcd ath0` or `ifconfig ath0` statement at the bottom of `/etc/rc.d/rc.local`
5. If you're using hotplug, you can have it automatically bring up the interface by editing `/etc/hotplug/net.agent` and inserting the following lines in the case add|register) branch:

```
     ath*)
       dhcpcd -n $INTERFACE % If you use dhcp for the interface
       % or if you prefer to just hardcode your interface, put your
ifconfig statement here.
       % ifconfig $INTERFACE 192.168.1.1 netmask 255.255.255.0 ...
       ;;
```

   In the *remove*|*unregister* branch it will automatically bring down dhcpcd if running.

Support for WPA encryption

I am assuming that you already have your madwifi-powered card up and running. Do not try to add WPA support if you do not yet have a functional wireless network connection! Also, if you run Slackware older than 10.2 you will need the updated network scripts that are mentioned in the previous section Updating the network scripts (Slackware 10.1 and older).

The madwifi package obtained here works well with the wpa_supplicant package found in Slackware 12.0. Starting with the linux kernel 2.6.14, madwifi and wpa_supplicant can communicate using wpa_supplicant's "`wext`" driver using the kernel's "wireless extensions". For older kernel versions, you will need wpa_supplicant's "`madwifi`" driver for which you need the wpa_supplicant package found here.

If you want to compile your own wpa_supplicant package, you might find some useful information in the Setting up a Client Using WPA-PSK Wiki page.

> Note:
> Recent wpa_supplicant should have support for roaming open networks as well
> as for wpa-protected networks. This makes the waproamd
> program obsolete for instance.

> Note:
> When you want to re-build wpa_supplicant and run a Linux kernel older than 2.6.14,
> make sure you have installed the madwifi package on your system as well.
> The wpa_supplicant build needs the include files which the madwifi package installs.
> If you want to use the SlackBuild scripts for madwifi and wpa_supplicant
> which you can find here to rebuild the packages for your system,
> build them in that specific order (first build and install madwifi,
> then build and install wpa_supplicant).
> The madwifi include files will be detected by the wpa_supplicant build script and madwifi support enabled.

- To enable WPA support for your madwifi driver, install the wpa_supplicant package (it is not depending on any kernel version) and then open the file `/etc/wpa_supplicant.conf` in an

editor. It should look something like this:

```
  ctrl_interface=/var/run/wpa_supplicant
  ctrl_interface_group=0
  eapol_version=1
  ap_scan=1
  fast_reauth=1
  network={
        scan_ssid=0
        ssid="your_essid"
        proto=WPA
        key_mgmt=WPA-PSK
        pairwise=CCMP TKIP
        group=CCMP TKIP WEP104 WEP40
        psk=your_64_hex_characters_long_key
  }
```

but you'll need to supply your own values for the *ssid* and the *psk*.

- There is a way to generate the hexadecimal value for the PSK if you have an access point which uses a passphrase. As root, run:

```
wpa_passphrase YOURSSID passphrase
```

with the SSID of your AP and the passphrase youve entered in its WPA-PSK configuration. You'll receive an output, which looks like this:

```
network={
    ssid="YOURSSID"
    #psk="passphrase"
psk=04dffae0172e3a255e5bab6f28ab78cc23d845f3dd8d4a63ba64a37555e2a33b
}
```

Next, you should copy the three lines that you find inside the `network={}` section of the command's output and paste them inside the `network={}` section of the file `/etc/wpa_supplicant.conf`. Do not forget to check the permissions of the configuration file! The key that it contains should be protected from prying eyes.

```
chmod 600 /etc/wpa_supplicant.conf
```

- You will also need to apply the following patch to `/etc/hotplug/net.agent` if you're not yet running Slackware 10.2 or newer. The patch makes the network initialization cleaner if you have more than one network interface, and wpa_supplicant needs that.

```
--- net.agent.org       2005-04-17 00:48:48.000000000 +0200
+++ net.agent   2005-04-17 00:41:56.000000000 +0200
@@ -67,7 +67,7 @@
                 # Interface already up?  If so, skip.
                 if ! /sbin/ifconfig | grep "^${INTERFACE} " 1>
/dev/null ; then
```

```
                    debug_mesg run rc.inet1
-                   exec /etc/rc.d/rc.inet1
+                   exec /etc/rc.d/rc.inet1 ${INTERFACE}_start
                fi
            # RedHat and similar
```

- Finally, you'll need to upgrade your wireless-tools to at least wireless-tools-27 (if you run anything older than Slackware 10.2). There is a package for wireless-tools here in case you need a more advanced version.
- Now, if your network configuration in `rc.inet1.conf` looked like this at first:

```
# Config information for ath0 (using dhcp):
IFNAME[1]="ath0"
IPADDR[1]=""
NETMASK[1]=""
USE_DHCP[1]="yes"
DHCP_HOSTNAME[1]="mywirelessbox"
```

then you'd have to add two lines so that it will read:

```
# Config information for ath0 (using dhcp):
IFNAME[1]="ath0"
IPADDR[1]=""
NETMASK[1]=""
USE_DHCP[1]="yes"
DHCP_HOSTNAME[1]="mywirelessbox"
WLAN_WPA[1]="wpa_supplicant"
WLAN_WPADRIVER[1]="wext"
```

Adapt this to your own configuration of course. With kernels older than 2.6.14, the last line should become

```
WLAN_WPADRIVER[1]="madwifi"
```

and you'll need the version of wpa_supplicant that has support for the madwifi driver (see above).

- You can restart your wireless network card (`ath0`) now, by running

```
/etc/rc.d/rc.inet1 ath0_restart
```

If your WPA connection does not activate, try some debugging:

WPA debugging

- [*If you run a kernel older than 2.6.14 :*] Was wpa_supplicant compiled with support for madwifi? Run the command `wpa_supplicant` on the commandline, and verify that the output mentions `madwifi = MADWIFI 802.11 support (Atheros, etc.)` under `drivers:`. If not, you

will have to find another package which has the support for madwifi compiled in, or build a wpa_supplicant package yourself, and make sure that the build script finds the madwifi source code on your box.

- Debug the WPA authentication process.
  Make sure the network interface is down (run

```
/etc/rc.d/rc.inet1 ath0_stop
```

  to make sure). Start the wpa_supplicant daemon as a foreground process with additional debugging enabled:

```
wpa_supplicant -dw -c/etc/wpa_supplicant.conf -Dwext -iath0
```

  Then activate the network interface in another terminal (run

```
/etc/rc.d/rc.inet1 ath0_start)
```

  Look at the output of wpa_supplicant in the first terminal, it might give you pointers to look for a solution.

- Get a run-time status overview of the supplicant:
  As root, run

```
wpa_cli status
```

  to see the current status of wpa_supplicant's authentication process.

- Debug Slackwares network intitialization.
  Change

```
DEBUG_ETH_UP="no"
```

  to

```
DEBUG_ETH_UP="yes"
```

  in `/etc/rc.d/rc.inet1.conf` and look for *logger* messages that are written to `/var/log/messages`. Maybe those messages will help you trace your problem.
  NOTE: with debugging enabled, Slackware will write your WEP/WPA keys to the message log as well, in clear text!

- The WPA association might take a long time.
  Start the interface again after a little time, this may help if it takes wpa_supplicant a long time to associate (no *restart*, just a *start*):

```
/etc/rc.d/rc.inet1 ath0_start
```

  If this makes your wirelesss work, but the problem occurs often, you can change the 'wait' time for the WPA authentication process by editing the file `/etc/rc.d/rc.inet1.conf` and adding

the line

```
WLAN_WPAWAIT[?]=30
```

or any other larger value that helps your particular setup.
NOTE: in the last line make sure that you replace the questionmark in **[?]** with the array value that matches your wireless card configuration In the above example this array index would be [**1**].

- The Access Point is not broadcasting the SSID.
  I have tried and failed in getting WPA to work when the Access Point has a *hidden SSID*. Check if your AP is broadcasting the SSID and if not, enable it. There is little point in hiding the SSID anyway, with WPA as a protection layer you should not fear break-ins (as long as you do not use easy-to-guess passphrases!!! WPA can be cracked with dictionary attacks and no I will not supply a link here).

Author(s) and further pointers

- The original information on this page was written by unknown author for the original DokuWiki based http://madwifi.org/wiki/MadWiki
- This page was updated with Slackware 10.2 information and WPA configuration by Eric Hameleers <alien at slackware dot com>
  For questions about this page, you can contact me at my email address, or visit the #madwifi IRC channel on Freenode.
- Updated with information about using madwifi-ng on Slackware 10.2 by Joe Feise <jfeise at feise dot com>.
- Updated with information about using madwifi-ng (builds later than 1407) and updated WPA information by Eric Hameleers <alien at slackware dot com>.
- Exported this page from the MadWiki and translated back to DokuWiki format, for my own Slackware Wiki.
- Added some configuration info for Slackware 12.0 and kernels >= 2.6.14, removed sections about the early madwifi-ng drivers.