Build Instructions from Scratch for "APRS Appliance" Image on Raspberry OS "Bullseye"

Don Rolph AB1PH/Jeff Marden N1JCM
Rev. 0.4
Mar. 5, 2022

Discussion:

Bullseye is the officially supported version of Raspberry OS. This is a major upgrade from the previous "Buster" version. Unfortunately, from the perspective of the APRS Appliance effort, this has not been a smooth upgrade. There are challenges with "Bullseye" which releases gpsd 3.22 as standard, the gpsd team which does not consider gpsd 3.22 supported but requires gpsd 3.23 or higher and Dire Wolf. After several months of experimentation, this appears to be the "sweet spot" for installs with the present software environment surrounding "Bullseye".

Background:

The "ATGP APRS Appliance '' is an effort to establish an APRS tool to support the Appalachian Trail Golden Packet effort, but which is also suitable for general APRS use. The design goal is to establish an open approach which can develop an "APRS Appliance" which is comparable in performance to the Kenwood D7XX series of APRS enabled transceivers.

This effort is dependent on the Dire Wolf development efforts of John Langner WB2OSZ and should be perhaps considered an application note for the Dire Wolf software. My heartfelt thanks to John for his development efforts.

The approach also leverages the AllStar sound interface design. My thanks to Stephen Smith N8AR for his development of the DINAH implementation of the AllStar interface and for his discussion on how to package the hardware moving forward.

I would like to thank Jeff Marden N1JCM for his testing efforts.

I would like to thank Bob Bruninga, now an SK, for his discussions and direction on testing.

And Jeff Hochberg W4JEW is to be commended for instigating this effort and holding the AT Golden Packet teams feet to the fire to keep the project moving.

Introduction:

At its simplest the "ATGP APRS Appliance" consists of:

- a lightweight application running basic ARPS functions, in this case the Dire Wolf software TNC
- using the AllStar interface sound card
- A connection to the transceiver through the 6 pin mini-din port: this is to standard sound levels across systems as well as provide standardized PTT controls and 1200 and 9600 baud functionality

An auxiliary user interface for monitoring APRS traffic, mapping positions, and messaging is supported.

A default software image of the Raspberry OS is available, but for this to be a truly open build instructions and clarity in where to make personal modifications where desired need to be provided. The amateur operator should have easy access to internals to make changes as desired and needed. This also ensures that evolution of the system to support OS and infrastructure changes can be readily supported and can be upgraded and updated by anyone.

The build instructions are heavily dependent on the build instructions for Dire Wolf by John WB2OSZ:

- https://github.com/wb2osz/direwolf/blob/master/doc/Raspberry-Pi-APRS.pdf

WB2OSZ's tracker documentation:

- https://github.com/wb2osz/direwolf/blob/master/doc/Raspberry-Pi-APRS-Tracker.pdf

Assumptions about the assembler of the image:

You are doing an image build of Raspberry OS for the Raspberry PI. It is assumed you have a reasonable working knowledge of Raspberry OS and its configuration. If you do not have this level of experience, then perhaps the prebuilt image might be more appropriate, see Raspberry OS image at:

- https://drive.google.com/file/d/1ZkZQmRpCHOKm-dbgMWxH5YxCqe-QIvHT/view?usp=sharing

The choice of editor is a religious issue among many LINUX users. It is assumed that you know how to use your editor of choice and that you can insert/replace strings as required.

You will also find it convenient if a set of changed files for the install are copied to your Raspberry PI, There are many choices of tools to use to put these files on the Raspberry PI, but if you use MobaXterm, https://mobaxterm.mobatek.net/, it includes a file browser interface to move files to and from the Raspberry PI. Remember, the file browser interface does not work with the root account, so you will need to copy to and from the pi account.

Build Instructions from Scratch for "APRS Appliance" Image on Raspberry OS "Bullseye"

At this stage in the development all actions are done as root. It is assumed for all steps that you will login as the pi account and then execute:

sudo su –

which will make you root and place you in the /root home directory.

Build Instructions for the mage:

1: Build a Raspberry OS image and boot the Raspberry PI

Standard build instructions for a base Raspberry PI can be used, typically using the Raspberry PI Imager:

- https://www.raspberrypi.com/software/

It is probably easiest to connect the Raspberry PI to a USB keyboard and an HDMI monitor for the build, but those who are courageous can use the advanced features of the Raspberry Imager to configure the image at build time. Make sure that ssh is enabled and uses a simple password authentication.

My default setting for the freely available image ATGP APRS Appliance are:

- Hostname: atgp
- Enable SSH
- Connection settings for your WIFI network
  - Enter WIFI-SSID
  - Enter password
- Set your locale
- Enable ssh

You will need to establish the WIFI-SSID for your established WIFI and put in the appropriate password to connect to this WIFI SSID.

You can set these in the Raspberry PI Imager (locating the IP address after it boots) or using the first time setup routine.

Once these are configured, the rest of the build can be performed over SSH.

This step is perhaps not required, but since there are many build options, we include it to ensure that the full SD is being utilized.

raspi-config --expand-rootfs

reboot

Now we update software components to the latest versions.

apt-get update

reboot

2) put the support files in a directory aprsappliance on the Raspberry PI:

The files can be found as a zip file at:

- https://github.com/APRSFoundation/aprsappliance/blob/main/configs/allstar-based-build/aprsappliancebullseye.zip

Lisintg os the key completely new files are in the appendix. It is perhaps simplest to have these files in the aprsappliance directory of the PI account, but one could also edit the master files directly. If you put the files on the Raspberry PI, the aprsappliance directory of the pi account should look like:

```
pi@atgp:~ $ ls -ld /home/pi/aprsappliance/*
-rw-r--r-- 1 pi pi    931 Mar  3 13:00 /home/pi/aprsappliance/crontab.master
-rw-r--r-- 1 pi pi   1846 Mar  3 13:00 /home/pi/aprsappliance/dhcpcd.conf
-rw-r--r-- 1 pi pi  16790 Mar  3 13:00 /home/pi/aprsappliance/direwolf.conf
-rw-r--r-- 1 pi pi  27707 Mar  3 13:00 /home/pi/aprsappliance/dnsmasq.conf
-rw-r--r-- 1 pi pi   5887 Mar  3 13:00 /home/pi/aprsappliance/dw-start.sh
-rw-r--r-- 1 pi pi    385 Mar  3 13:00 /home/pi/aprsappliance/fstab
-rw-r--r-- 1 pi pi 569532 Mar  3 13:00 /home/pi/aprsappliance/gpsd-3-23-2
-rw-r--r-- 1 pi pi    269 Mar  3 13:00 /home/pi/aprsappliance/gpsd.conf
-rw-r--r-- 1 pi pi    206 Mar  3 13:00 /home/pi/aprsappliance/hostapd.conf
-rw-r--r-- 1 pi pi    233 Mar  3 13:00 /home/pi/aprsappliance/wifi-switch.sh
-rw-r--r-- 1 pi pi    141 Mar  3 13:00 /home/pi/aprsappliance/wpa_supplicant.conf
```

3) Remove pulse audio

This is probably not required but was included in WB2OSZ's instructions, and doesn't seem to hurt unless you are also using other sound applications with the GUI interface. All images tested so far have pulse audio removed and the most recent versions of "Bullseye" seem to require this.

Remember, perform all the following steps as root

From WB2OSZ build instructions (I find the dpkg instructions does not perform as documented so I skip it)

apt-get remove --purge pulseaudio (This is dash dahs purge: Word hates the dash dash)

apt-get autoremove

Now

reboot

4: Install all needed packages for the build:

Again do these steps as root

The next packages are for the basic direwolf executable:

apt-get install cmake

apt-get install libasound2-dev

apt-get install libudev-dev

To allow direwolf to run unattended, we need Install the screen utility

apt-get install screen

These packages are for gps support. They must be installed before we build direwolf or gps support will not be included in the direwolf build.

apt-get install gpsd

apt-get install gpsd-clients

apt-get install libgps-dev

This package supports announcing of the TNC socket for use with the iPhone aprs.fi application, see:

- https://github.com/wb2osz/direwolf/tree/dev

apt-get install libavahi-client-dev

To provide hotspot support we need the following packages:

apt-get install hostapd

apt-get install dnsmasq

Now for consistency:

reboot

5) now clean up gpsd and make the log files virtual to reduce chances of corrupting SD card

Perform the following as root.

gpsd 3.22 is the default gpsd daemon under "Bullseye". However, 3.22 has several bugs when accessed by Dire Wolf AND the gpsd team does not support gpsd 3.22. Experiments suggest that the gpsd 3.23.2 daemon is a sweet spot BUT:

- The build of gpsd 3.23.2 is complex
- The build install differs from the package install causing Dire Wolf to complain

By:

- Installing gpsd 3.22 form the package repository for Bullseye
- In /usr/sbin replacing the gpsd 3.22 executable with the gpsd 3.23.2 executable

We get a working system BUT with occasional warning messages from various gps tools (i.e. cgps) regarding library version. However, nothing seems to break. Hopefully this set of issues under Bullseye will be resolved soon.

Note: with the MT3333 gps unit (AdaFruit Ultimate GPS) only, there is an intermittent 1 minute flood of beacons at time of fix which then self-quenches. We are still working on this issue.

cd /usr/sbin

mv gpsd gpsd-3.22

cp /home/pi/aprsappliance/gpsd-3-23-2 .
cp gpsd-3-23-2 gpsd

chmod 755 gpsd

chmod 755 gpsd-3-23-2

Now we can speed up the gps fix by having gpsd start up the GPS device at boot time

cd /etc/default

Edit (using your preferred editor the following line in gpsd (this is the gpsd config file, see /home/pi/aprsappliance/gpsd.conf):

GPSD_OPTIONS=""

To

GPSD_OPTIONS="-n"

Now log files are being written to the physical /var/log directory. If the system is powered off during this write activity, the SD card may become corrupted. So:

cd /etc

Edit fstab to add the following line as the last line (see fstab file in /home/pi/aprsappliance):

tmpfs /var/log tmpfs defaults,noatime,nosuid,mode=0755,size=100m 0 0

And for consistency:

reboot

6) Now build Dire Wolf

Again we will build as root. If we want this to run on both PI 3 B+ and PI Zero, the executable needs to be built on a PI Zero. A PI 3 B+ executable will not execute properly on PI Zero.

git clone https://www.github.com/wb2osz/direwolf

cd direwolf

If you want announcement support for aprs.fi app and are willing to use the dev branch:

git checkout dev

As of the writing of the instructions this installs the 1.7 E version of Dire Wolf which seems to have the UIFLOOD digipeating algorithm incorporated in the build.

Now continue with:

mkdir build

cd build

cmake -DUNITTEST=1 ..

make -j4

make test

make install

make install-conf

cd /root

7) Configuring direwolf and matching direwolf to the soundcard:

We will assume:

- You are using a DINAH soundcard with JP3 cut (to disable COS)

- You have no other sound device on the system (many HDMI monitors will configure a sound device: so do this over ssh)

You need to identify the sound device:

aplay -l

and you should see:

**** List of PLAYBACK Hardware Devices ****
card 1: Device [USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0

If the soundcard is card 1 device 0, then the standard direwolf.conf should work.

If aplay -l shows that the sound is from card 1 device 0:

mv direwolf.conf direwolf.conf.orig
cp /home/pi/aprsappliance/direwolf.conf direwolf.conf

Now edit the line in direwolf.conf from:

MYCALL NOCALL-XXX''

To:

MYCALL <your_callsign-SSID>

e.g.:

MYCALL AB1PH-13

You can mow test Dire Wolf by running:

direwolf -t 0

and look at the log stream for any errors.

The sound chain is important in the proper functioning of the Dire Wolf sound based TNC. Experiments suggest that with:

- The typical 6 pin mini-din interface on the radio
- The DINAH sound card

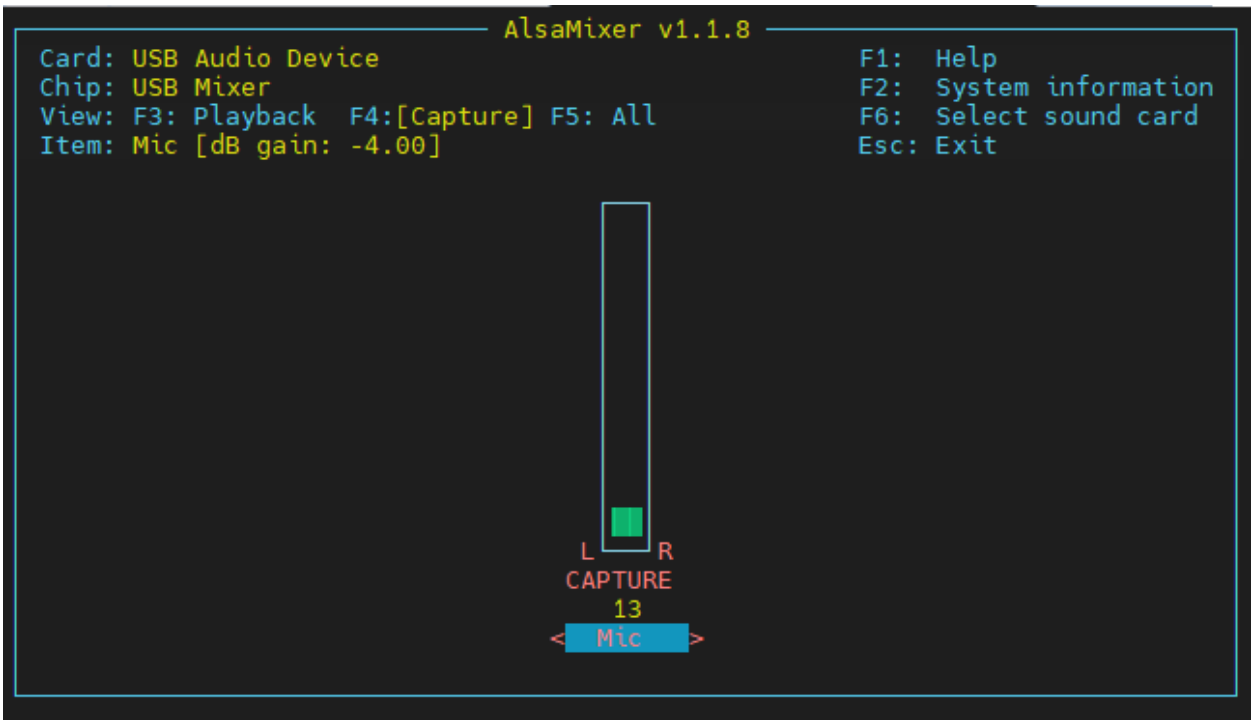
The key issue is the line input receive volume.

So enter:

alsamixer -c 1

tab to mixer/capture screen (line level inputs)

And use the down arrow key to set sound level to about 13:



Exit using <*CTRL*>*c*

Now:

mv dw-start.sh dw-start.sh.orig
cp dw-start.sh /home/pi/aprsappliance/dw-start.sh dw-start.sh

We need to make the script executable so:

chmod 755 dw-start.sh

Start up Dire Wolf:

./dw-start.sh

And after it starts, you can connect to the log data by:

screen -D -r direwolf

To exit screen mode:

<CTRL>a
d

And direwolf will keep running detached from the session.

To kill Dire Wolf from within the direwolf screen session:

<CTRL>c

Once last step: we want direwolf to start automatically when we boot. The simplest approach is perhaps to have a crontab entry, so enter:

crontab -e

if needed, select your editor of choice

and at the end of the file add the line:

@reboot /root/dw-start.sh >/dev/null 2>&1

Which will startup Dire Wolf on reboot.

Look at the provided file crontab.master.

Now let's see if this works as expected.

reboot

When the system restarts, login as pi, *sudo su* – and enter

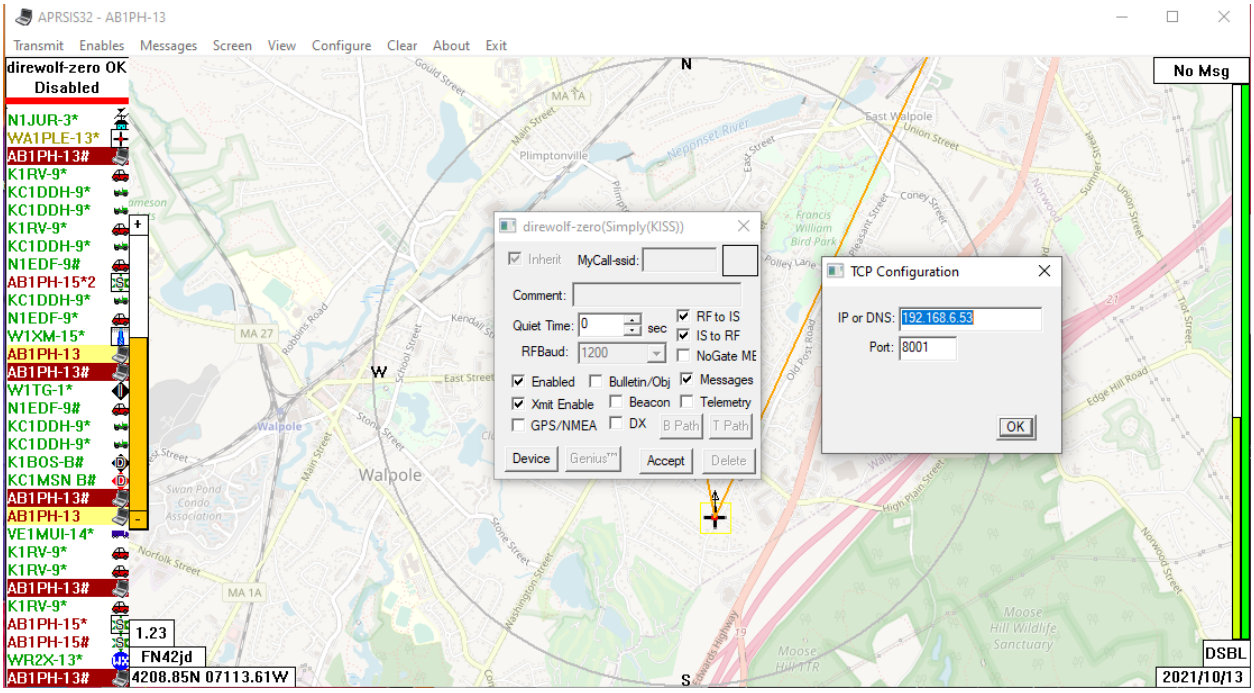
screen -D -r direwolf

If the log file is showing entries, you have a working direwolf based APRS appliance.

If all works as expected you have a running Dire Wolf on the prototype APRS Appliance.

8) Connecting user interface:

One can connect APRSIS32 to Dire Wolf by creating a port using TCP/IP to the direwolf IP address on part 8001.

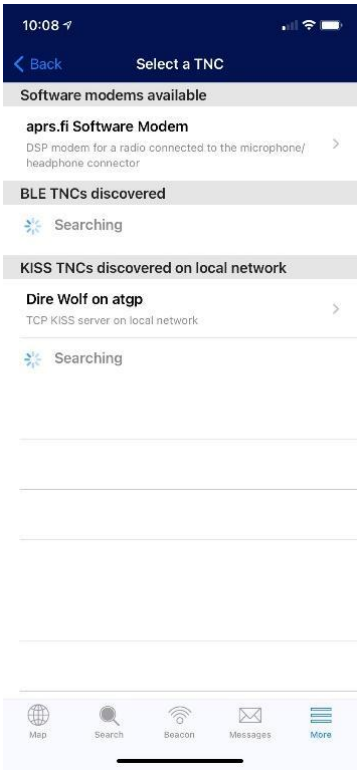


If:

- You install the aprs.fi applications on your iPhone

- You compiled Dire Wolf off the present development branch

You can connect to Dire Wolf from you cell phone by connecting to the advertised TNC:



Other applications which use the KISS interface via TCP/IP should also work.

These interfaces will typically allow you to:

- See flow of packets and stations
- See locations of stations on a map
- Messaging from the application (for aprs.fi app, this requires a yearly subscription)

8) Building the hotspot:

The instructions so far support building the image for the APRS Appliance prototype, but it will only connect to an established network. This will work if the appliance is used standalone (no client interface) or it is used at home with the configured WIFI.

In the field, however, we usually do not have an established network. So we need the appliance to be able to support a network as a hotspot. There are several approaches to this configuration:

- A script to autodetect the presence of an established network and start the hotspot if there is no established network

- A script to allow one to turn off the hotspot and enable normal WIFI

  - A sample script of this is provided as wifi-switch.sh

Experiments suggest some temperamental behavior, so a more direct approach has been pursued.

- We establsih two WIFI controllers
- The first controller, wlan0 (the onboard WIFI controller), is configured as a hotspot
- The second controller, wlan1 (a USB WIFI controller), is configured to the default established WIFI network if it is available

This design supports the needed hotspot by default but enables an established network which can be used for maintenance and upgrades if this network is present.

And all one needs to do is add a USB WIFI controller, except …, not all USB WIFI adapters will be automatically recognized by Raspberry OS. Unless you enjoy the additional amusement of installing USB WIFI drivers (and perhaps having to patch the kernel), one should select a USB WIFI adapter which is plug and play with Raspberry OS. Example suitable USB WIFI adapters are:

- https://www.amazon.com/dp/B00GFAN498

- https://www.amazon.com/dp/B06Y2HKT75

You can tell if the USB WIFI adapter is automatically recognized if without the USB adapter plugged in you execute:

*ls -ld /sys/class/net/wlan**

and you should see:

lrwxrwxrwx 1 root root 0 Oct 13 08:09 /sys/class/net/wlan0 ->
../../devices/platform/soc/20300000.mmcnr/mmc_host/mmc1/mmc1:0001/mmc1:0001:1/net/wlan0

or only the wlan0 adapter should be present.

Now plug in the USB WIFI adapter and execute:

*ls -ld /sys/class/net/wlan**

and you should see something like:

lrwxrwxrwx 1 root root 0 Oct 13 08:09 /sys/class/net/wlan0 ->
../../devices/platform/soc/20300000.mmcnr/mmc_host/mmc1/mmc1:0001/mmc1:0001:1/net/wlan0
lrwxrwxrwx 1 root root 0 Oct 13 08:09 /sys/class/net/wlan1 ->
../../devices/platform/soc/20980000.usb/usb1/1-1/1-1.3/1-1.3:1.0/net/wlan1

showing that the USB WIFI adapter has been identified.

If your card is identified and configured, you can build the hotspot as follows:

systemctl unmask hostapd
systemctl enable hostapd
systemctl enable dnsmasq

Now it is simplest to just use the exemplar files. So:

cd /etc
mv dhcpcd.conf dhcpcd.conf.orig
cp /home/pi/aprsappliance/dhcpcd.conf dhcpcd.conf
mv dnsmasq,conf dnsmasq,conf.orig
cp /home/pi/aprsappliance/dnsmasq.conf dnsmasq.conf
cd /etc/hostapd
mv hostapd.conf hostapd.conf.orig (this should fail because there should be no hostapd.conf)
cp /home/pi/aprsappliance/hostapd.conf hostapd.conf
reboot

If everything works as expected:

- There will be a WIFI with SSID ATGP: connect to this network, password: ChangeOnInstall
- Use ssh to connect to the system at ip address 10.0.0.5
- Login as pi and *sudo su –*
- Check for ip addresses with *hostname -I* you should see 10.0.0.5 and the local WIF ip address

Example output:

10.0.0.5 192.168.6.53

You can now mount your clients to the established WIFI network and connect to the local ip address over port 8001 for the client to access the ATGP APRS Appliance.

Final Comments:

Once the system is configured, you can wire the PI with sound card to the transceiver and start testing.

We can state:

- There are errors in the instructions: give feedback on the errors
- There are better ways to do this: let us know your thoughts
- We have limited service experience with this system: let us know how it works out

Appendix 1 – direwolf.conf file

```
############################################################
#                                          #
#            Configuration file for Dire Wolf          #
#                                          #
#               Linux version                    #
#                                          #
############################################################
#
# Extensive documentation can be found here:
# Stable release -       https://github.com/wb2osz/direwolf/tree/master/doc
# Latest development -   https://github.com/wb2osz/direwolf/tree/dev/doc
#
# The complete documentation set can also be found in
# /usr/local/share/doc/direwolf/ or /usr/share/doc/direwolf/
# Concise "man" pages are also available for Linux.
#
# This sample file does not have examples for all of the possibilities.
# Consult the User Guide for more details on configuration options.%C%#
#
# These are the most likely settings you might change:
#
#        (1)       MYCALL        -  call sign and SSID for your station.
#
#                                 Look for lines starting with MYCALL and
#                                 change NOCALL to your own.
#
#        (2)       PBEACON       -  enable position beaconing.
```

```
#
#                       Look for lines starting with PBEACON and
#                       modify for your call, location, etc.
#
#       (3)     DIGIPEATER  -  configure digipeating rules.
#
#                       Look for lines starting with DIGIPEATER.
#                       Most people will probably use the given example.
#                       Just remove the "#" from the start of the line
#                       to enable it.
#
#       (4)     IGSERVER, IGLOGIN  - IGate server and login
#
#                       Configure an IGate client to relay messages between
#                       radio and internet servers.
#
#
# The default location is "direwolf.conf" in the current working directory.
# On Linux, the user's home directory will also be searched.
# An alternate configuration file location can be specified with the "-c" command line option.
#
# As you probably guessed by now, # indicates a comment line.
#
# Remove the # at the beginning of a line if you want to use a sample
# configuration that is currently commented out.
#
# Commands are a keyword followed by parameters.
#
# Command key words are case insensitive.  i.e. upper and lower case are equivalent.
#
# Command parameters are generally case sensitive.  i.e. upper and lower case are different.
#


############################################################
#                                          #
#           FIRST AUDIO DEVICE PROPERTIES            #
#           (Channel 0 + 1 if in stereo)             #
#                                          #
############################################################

#
# Many people will simply use the default sound device.
# Some might want to use an alternative device by choosing it here.
#
# Linux ALSA is complicated.  See User Guide for discussion.
# To use something other than the default, generally use plughw
# and a card number reported by "arecord -l" command.  Example:

# ADEVICE  plughw:1,0
ADEVICE plughw:1,0
```

```
# You can also use "-" or "stdin" to pipe stdout from
# some other application such as a software defined radio.
# "stdin" is not an audio device.  Don't use this unless you
# understand what this means.  Read the User Guide.
# You can also specify "UDP:" and an optional port for input.
# Something different must be specified for output.

# ADEVICE stdin plughw:1,0
# ADEVICE UDP:7355 default


#
# Number of audio channels for this souncard:  1 (mono) or 2 (stereo).
# 1 is the default so there is no need to specify it.
#

#ACHANNELS 2


##############################################################
#                                    #
#            SECOND AUDIO DEVICE PROPERTIES            #
#            (Channel 2 + 3 if in stereo)            #
#                                    #
##############################################################

#ADEVICE1  ...


##############################################################
#                                    #
#            THIRD AUDIO DEVICE PROPERTIES            #
#            (Channel 4 + 5 if in stereo)            #
#                                    #
##############################################################

#ADEVICE2  ...


##############################################################
#                                    #
#            CHANNEL 0 PROPERTIES                    #
#                                    #
##############################################################

CHANNEL 0

#
# The following MYCALL, MODEM, PTT, etc. configuration items
# apply to the most recent CHANNEL.
```

```
#

#
# Station identifier for this channel.
# Multiple channels can have the same or different names.
#
# It can be up to 6 letters and digits with an optional ssid.
# The APRS specification requires that it be upper case.
#
# Example (don't use this unless you are me):  MYCALL     WB2OSZ-5
#

MYCALL NOCALL-XXX

#
# Pick a suitable modem speed based on your situation.
#         1200    Most common for VHF/UHF.  Default if not specified.
#         2400    QPSK compatible with MFJ-2400, and probably PK232-2400 & KPC-2400.
#         300     Low speed for HF SSB.  Default tones 1600 & 1800.
#         EAS     Emergency Alert System (EAS) Specific Area Message Encoding (SAME).
#         9600    G3RUH style - Can't use Microphone and Speaker connections.
#         AIS     International system for tracking ships on VHF.
#                         Also uses 9600 bps so Speaker connection won't work.
#
# In most cases you can just specify the speed.  Examples:
#

MODEM 1200
#MODEM 9600

#
# Many options are available for great flexibility.
# See User Guide for details.
#

#
# Uncomment line below to enable the DTMF decoder for this channel.
#

#DTMF

#
# If not using a VOX circuit, the transmitter Push to Talk (PTT)
# control is usually wired to a serial port with a suitable interface circuit.
# DON'T connect it directly!
#
# For the PTT command, specify the device and either RTS or DTR.
# RTS or DTR may be preceded by "-" to invert the signal.
# Both can be used for interfaces that want them driven with opposite polarity.
#
# COM1 can be used instead of /dev/ttyS0, COM2 for /dev/ttyS1, and so on.
```

```
#

#PTT COM1 RTS
#PTT COM1 RTS -DTR
#PTT /dev/ttyUSB0 RTS

#
# On Linux, you can also use general purpose I/O pins if
# your system is configured for user access to them.
# This would apply mostly to microprocessor boards, not a regular PC.
# See separate Raspberry Pi document for more details.
# The number may be preceded by "-" to invert the signal.
#

#PTT GPIO 25
PTT CM108

# The Data Carrier Detect (DCD) signal can be sent to the same places
# as the PTT signal.  This could be used to light up an LED like a normal TNC.

#DCD COM1 -DTR
#DCD GPIO 24


############################################################
#                                          #
#           CHANNEL 1 PROPERTIES                  #
#                                          #
############################################################

#CHANNEL 1

#
# Specify MYCALL, MODEM, PTT, etc. configuration items for
# CHANNEL 1.   Repeat for any other channels.


############################################################
#                                          #
#           TEXT TO SPEECH COMMAND FILE              #
#                                          #
############################################################

#SPEECH dwespeak.sh


############################################################
#                                          #
#           VIRTUAL TNC SERVER PROPERTIES            #
#                                          #
############################################################
```

```
#
# Dire Wolf acts as a virtual TNC and can communicate with
# client applications by different protocols:
#
#         - the "AGW TCPIP Socket Interface" - default port 8000
#         - KISS protocol over TCP socket - default port 8001
#         - KISS TNC via pseudo terminal   (-p command line option)
#

AGWPORT 8000
KISSPORT 8001

#
# It is sometimes possible to recover frames with a bad FCS.
# This applies to all channels.
#
#         0  [NONE] - Don't try to repair.
#         1  [SINGLE] - Attempt to fix single bit error.  (default)
#         ... see User Guide for more values and in-depth discussion.
#

#FIX_BITS 0

#
############################################################
#                                          #
#          FIXED POSIION BEACONING PROPERTIES            #
#                                          #
############################################################


#
# Beaconing is configured with these two commands:
#
#         PBEACON               - for a position report (usually yourself)
#         OBEACON               - for an object report (usually some other entity)
#
# Each has a series of keywords and values for options.
# See User Guide for details.
#
# Example:
#
# This results in a broadcast once every 10 minutes.
# Every half hour, it can travel via two digipeater hops.
# The others are kept local.
#

#PBEACON delay=1  every=30 overlay=S symbol="digi" lat=42^37.14N long=071^20.83W
power=50 height=20 gain=4 comment="Chelmsford MA" via=WIDE1-1,WIDE2-1
```

```
#PBEACON delay=11 every=30 overlay=S symbol="digi" lat=42^37.14N long=071^20.83W
power=50 height=20 gain=4 comment="Chelmsford MA"
#PBEACON delay=21 every=30 overlay=S symbol="digi" lat=42^37.14N long=071^20.83W
power=50 height=20 gain=4 comment="Chelmsford MA"
#PBEACON delay=1  every=3 overlay=S symbol="digi" lat=42^08.85N long=071^13.61W
power=1 height=20 gain=1 comment="East Walpole MA" via=WIDE1-1,WIDE2-1
#PBEACON delay=11 every=4 overlay=S symbol="digi" lat=42^08.85N long=071^13.61W
power=1 height=20 gain=1 comment="East Walpole MA"
#PBEACON delay=21 every=4 overlay=S symbol="digi" lat=42^08.85N long=071^13.61W
power=1 height=20 gain=1 comment="East Walpole MA"



# With UTM coordinates instead of latitude and longitude.

#PBEACON delay=1 every=10 overlay=S symbol="digi" zone=19T easting=307477
northing=4720178


#
# When the destination field is set to "SPEECH" the information part is
# converted to speech rather than transmitted as a data frame.
#

#CBEACON dest="SPEECH" info="Club meeting tonight at 7 pm."

# Similar for Morse code.  If SSID is specified, it is multiplied
# by 2 to get speed in words per minute (WPM).

#CBEACON dest="MORSE-6" info="de MYCALL"


#
# Modify for your particular situation before removing
# the # comment character from the beginning of appropriate lines above.
#

############################################################
#                                          #
#          GPSD properties                         #
#                                          #
############################################################

GPSD

TBEACON every=2 symbol=horse alt=1 power=1 height=1 gain=1 via=WIDE1-1,WIDE2-2

SMARTBEACONING 40 2:00 2 5:00 0:05 20 10


############################################################
```

```
#                                                     #
#          APRS DIGIPEATER PROPERTIES                 #
#                                                     #
#######################################################

#
# For most common situations, use something like this by removing
# the "#" from the beginning of the line below.
#

DIGIPEAT 0 0 ^WIDE[3-7]-[1-7]$|^TEST$ ^WIDE[12]-[12]$ TRACE

# See User Guide for more explanation of what this means and how
# it can be customized for your particular needs.

# Filtering can be used to limit was is digipeated.
# For example, only weather weather reports, received on channel 0,
# will be retransmitted on channel 1.
#

#FILTER 0 1 t/wn

# Traditional connected mode packet radio uses a different
# type of digipeating.  See User Guide for details.

#######################################################
#                                                     #
#            INTERNET GATEWAY                         #
#                                                     #
#######################################################

# First you need to specify the name of a Tier 2 server.
# The current preferred way is to use one of these regional rotate addresses:

#       noam.aprs2.net                  - for North America
#       soam.aprs2.net                  - for South America
#       euro.aprs2.net          - for Europe and Africa
#       asia.aprs2.net                  - for Asia
#       aunz.aprs2.net                  - for Oceania

#IGSERVER noam.aprs2.net
#IGSERVER noam.aprs2.net

# You also need to specify your login name and passcode.
# Contact the author if you can't figure out how to generate the passcode.


# That's all you need for a receive only IGate which relays
# messages from the local radio channel to the global servers.

# Some might want to send an IGate client position directly to a server
```

```
# without sending it over the air and relying on someone else to
# forward it to an IGate server.  This is done by using sendto=IG rather
# than a radio channel number. Overlay R for receive only, T for two way.

#PBEACON sendto=IG delay=0:30 every=60:00 symbol="igate" overlay=R lat=42^37.14N
long=071^20.83W
#PBEACON sendto=IG delay=0:30 every=60:00 symbol="igate" overlay=T lat=42^37.14N
long=071^20.83W


# To relay messages from the Internet to radio, you need to add
# one more option with the transmit channel number and a VIA path.

#IGTXVIA 0 WIDE1-1


# Finally, we don't want to flood the radio channel.
# The IGate function will limit the number of packets transmitted
# during 1 minute and 5 minute intervals.   If a limit would
# be exceeded, the packet is dropped and message is displayed in red.

IGTXLIMIT 6 10


############################################################
#                                          #
#            APRStt GATEWAY                        #
#                                          #
############################################################

#
# Dire Wolf can receive DTMF (commonly known as Touch Tone)
# messages and convert them to packet objects.
#
# See separate "APRStt-Implementation-Notes" document for details.
#

#
# Sample gateway configuration based on:
#
#         http://www.aprs.org/aprstt/aprstt-coding24.txt
#         http://www.aprs.org/aprs-jamboree-2013.html
#

# Define specific points.

TTPOINT  B01  37^55.37N  81^7.86W
TTPOINT  B7495088  42.605237  -71.34456
TTPOINT  B934  42.605237  -71.34456

TTPOINT B901  42.661279  -71.364452
```

```
TTPOINT B902  42.660411  -71.364419
TTPOINT B903  42.659046  -71.364452
TTPOINT B904  42.657578  -71.364602


# For location at given bearing and distance from starting point.

TTVECTOR  B5bbbddd  37^55.37N  81^7.86W  0.01  mi

# For location specified by x, y coordinates.

TTGRID   Byyyxxx   37^50.00N  81^00.00W  37^59.99N  81^09.99W

# UTM location for Lowell-Dracut-Tyngsborough State Forest.

TTUTM  B6xxxyyy  19T  10  300000  4720000



# Location for the corral.

TTCORRAL   37^55.50N  81^7.00W  0^0.02N

# Compact messages - Fixed locations xx and object yyy where
#         Object numbers 100 - 199     = bicycle
#         Object numbers 200 - 299     = fire truck
#         Others                       = dog

TTMACRO  xx1yy  B9xx*AB166*AA2B4C5B3B0A1yy
TTMACRO  xx2yy  B9xx*AB170*AA3C4C7C3B0A2yy
TTMACRO  xxyyy  B9xx*AB180*AA3A6C4A0Ayyy

TTMACRO  z  Cz

# Receive on channel 0, Transmit object reports on channel 1 with optional via path.
# You probably want to put in a transmit delay on the APRStt channel so it
# it doesn't start sending a response before the user releases PTT.
# This is in 10 ms units so 100 means 1000 ms = 1 second.

#TTOBJ 0 1 WIDE1-1
#CHANNEL 0
#DWAIT 100

# Advertise gateway position with beacon.

# OBEACON DELAY=0:15 EVERY=10:00 VIA=WIDE1-1 OBJNAME=WB2OSZ-tt
SYMBOL=APRStt LAT=42^37.14N LONG=71^20.83W COMMENT="APRStt Gateway"


# Sample speech responses.
# Default is Morse code "R" for received OK and "?" for all errors.
```

```
#TTERR  OK              SPEECH  Message Received.
#TTERR  D_MSG           SPEECH  D not implemented.
#TTERR  INTERNAL        SPEECH  Internal error.
#TTERR  MACRO_NOMATCH   SPEECH  No definition for digit sequence.
#TTERR  BAD_CHECKSUM    SPEECH  Bad checksum on call.
#TTERR  INVALID_CALL    SPEECH  Invalid callsign.
#TTERR  INVALID_OBJNAME SPEECH  Invalid object name.
#TTERR  INVALID_SYMBOL  SPEECH  Invalid symbol.
#TTERR  INVALID_LOC     SPEECH  Invalid location.
#TTERR  NO_CALL         SPEECH  No call or object name.
#TTERR  SATSQ           SPEECH  Satellite square must be 4 digits.
#TTERR  SUFFIX_NO_CALL  SPEECH  Send full call before using suffix.
```

Appendix 2 – dwconfig.sh

```
#!/usr/bin/env bash

# Why not simply "#!/bin/bash" ?

# For OpenBSD, the bash location is /usr/local/bin/bash.
# By using env here, bash is found based on the user's $PATH.
# I hope this does not break some other operating system.


# Run this from crontab periodically to start up
# Dire Wolf automatically.

# See User Guide for more discussion.
# For release 1.4 it is section 5.7 "Automatic Start Up After Reboot"
# but it could change in the future as more information is added.


# Versioning (this file, not direwolf version)
#-----------
# v1.3 - KI6ZHD - added variable support for direwolf binary location
# v1.2 - KI6ZHD - support different versions of VNC
# v1.1 - KI6ZHD - expanded version to support running on text-only displays with
#        auto support; log placement change
# v1.0 - WB2OSZ - original version for Xwindow displays only



#How are you running Direwolf : within a GUI (Xwindows / VNC) or CLI mode
#
#  AUTO mode is design to try starting direwolf with GUI support and then
#    if no GUI environment is available, it reverts to CLI support with screen
#
#  GUI mode is suited for users with the machine running LXDE/Gnome/KDE or VNC
#    which auto-logs on (sitting at a login prompt won't work)
#
#  CLI mode is suited for say a Raspberry Pi running the Jessie LITE version
#      where it will run from the CLI w/o requiring Xwindows - uses screen

RUNMODE=CLI

# Location of the direwolf binary.  Depends on $PATH as shown.
# change this if you want to use some other specific location.
# e.g.  DIREWOLF="/usr/local/bin/direwolf"

DIREWOLF="direwolf"


#Direwolf start up command :: Uncomment only one of the examples.
#
```

```
# 1. For normal operation as TNC, digipeater, IGate, etc.
#    Print audio statistics each 100 seconds for troubleshooting.
#    Change this command to however you wish to start Direwolf

DWCMD="$DIREWOLF -a 100"

# 2. FX.25 Forward Error Correction (FEC) will allow your signal to
#    go farther under poor radio conditions.  Add "-X 1" to the command line.

#DWCMD="$DIREWOLF -a 100 -X 1"

#------------------------------------------------------------
#
# 3. Alternative for running with SDR receiver.
#    Piping one application into another makes it a little more complicated.
#    We need to use bash for the | to be recognized.

#DWCMD="bash -c 'rtl_fm -f 144.39M - | direwolf -c sdr.conf -r 24000 -D 1 -'"


#Where will logs go - needs to be writable by non-root users
LOGFILE=/var/tmp/dw-start.log


#-----------------------------------
# Main functions of the script
#-----------------------------------

#Status variables
SUCCESS=0

function CLI {
  SCREEN=`which screen`
  if [ $? -ne 0 ]; then
     echo -e "Error: screen is not installed but is required for CLI mode.  Aborting"
     exit 1
  fi

  echo "Direwolf in CLI mode start up"
  echo "Direwolf in CLI mode start up" >> $LOGFILE

  # Screen commands
  #  -d m :: starts the command in detached mode
  #  -S   :: name the session
  $SCREEN -d -m -S direwolf $DWCMD >> $LOGFILE
  SUCCESS=1

  $SCREEN -list direwolf
  $SCREEN -list direwolf >> $LOGFILE

  echo "----------------------"
```

```
  echo "----------------------" >> $LOGFILE
}

function GUI {
  # In this case
  # In my case, the Raspberry Pi is not connected to a monitor.
  # I access it remotely using VNC as described here:
  # http://learn.adafruit.com/adafruit-raspberry-pi-lesson-7-remote-control-with-vnc
  #
  # If VNC server is running, use its display number.
  # Otherwise default to :0 (the Xwindows on the HDMI display)
  #
  export DISPLAY=":0"

  #Reviewing for RealVNC sessions (stock in Raspbian Pixel)
  if [ -n "`ps -ef | grep vncserver-x11-serviced | grep -v grep`" ]; then
    sleep 0.1
    echo -e "\nRealVNC found - defaults to connecting to the :0 root window"
   elif [ -n "`ps -ef | grep Xtightvnc | grep -v grep`" ]; then
    #Reviewing for TightVNC sessions
    echo -e "\nTightVNC found - defaults to connecting to the :1 root window"
    v=`ps -ef | grep Xtightvnc | grep -v grep`
    d=`echo "$v" | sed 's/.*tightvnc *\(:[0-9]\).*/\1/'`
    export DISPLAY="$d"
  fi

  echo "Direwolf in GUI mode start up"
  echo "Direwolf in GUI mode start up" >> $LOGFILE
  echo "DISPLAY=$DISPLAY"
  echo "DISPLAY=$DISPLAY" >> $LOGFILE

  #
  # Auto adjust the startup for your particular environment:  gnome-terminal, xterm, etc.
  #

  if [ -x /usr/bin/lxterminal ]; then
    /usr/bin/lxterminal -t "Dire Wolf" -e "$DWCMD" &
    SUCCESS=1
   elif [ -x /usr/bin/xterm ]; then
    /usr/bin/xterm -bg white -fg black -e "$DWCMD" &
    SUCCESS=1
   elif [ -x /usr/bin/x-terminal-emulator ]; then
    /usr/bin/x-terminal-emulator -e "$DWCMD" &
    SUCCESS=1
   else
    echo "Did not find an X terminal emulator.  Reverting to CLI mode"
    SUCCESS=0
  fi
  echo "----------------------"
  echo "----------------------" >> $LOGFILE
}
```

```
# ------------------------------------------------------------
# Main Script start
# ------------------------------------------------------------

# When running from cron, we have a very minimal environment
# including PATH=/usr/bin:/bin.
#
export PATH=/usr/local/bin:$PATH

#Log the start of the script run and re-run
date >> $LOGFILE

# First wait a little while in case we just rebooted
# and the desktop hasn't started up yet.
#
sleep 30


#
# Nothing to do if Direwolf is already running.
#

a=`ps ax | grep direwolf | grep -vi -e bash -e screen -e grep | awk '{print $1}'`
if [ -n "$a" ]
then
  #date >> /tmp/dw-start.log
  #echo "Direwolf already running." >> $LOGFILE
  exit
fi

# Main execution of the script

if [ $RUNMODE == "AUTO" ];then
   GUI
   if [ $SUCCESS -eq 0 ]; then
      CLI
   fi
  elif [ $RUNMODE == "GUI" ];then
   GUI
  elif [ $RUNMODE == "CLI" ];then
   CLI
  else
   echo -e "ERROR: illegal run mode given.  Giving up"
   exit 1
fi
```

Appendix 3 – wpa_supplicant.conf

```
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
ap_scan=1

update_config=1
network={
        ssid="WIFI-SSID"
        psk="password"
}
```

Appendix 4 – crontab:

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
@reboot /root/dw-start.sh >/dev/null 2>&1
```

Appendix 5 – dhcpcd.conf

```
# A sample configuration for dhcpcd.
# See dhcpcd.conf(5) for details.

# Allow users of this group to interact with dhcpcd via the control socket.
#controlgroup wheel

# Inform the DHCP server of our hostname for DDNS.
hostname

# Use the hardware address of the interface for the Client ID.
clientid
# or
# Use the same DUID + IAID as set in DHCPv6 for DHCPv4 ClientID as per RFC4361.
# Some non-RFC compliant DHCP servers do not reply with this set.
# In this case, comment out duid and enable clientid above.
#duid

# Persist interface configuration when dhcpcd exits.
persistent

# Rapid commit support.
# Safe to enable by default because it requires the equivalent option set
# on the server to actually work.
option rapid_commit

# A list of options to request from the DHCP server.
option domain_name_servers, domain_name, domain_search, host_name
option classless_static_routes
# Respect the network MTU. This is applied to DHCP routes.
option interface_mtu

# Most distributions have NTP support.
#option ntp_servers

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate SLAAC address using the Hardware Address of the interface
#slaac hwaddr
# OR generate Stable Private IPv6 Addresses based from the DUID
slaac private

# Example static IP configuration:
#interface eth0
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1
```

```
# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0
interface wlan0
static ip_address=10.0.0.5/24
nohook wpa_supplicant
```

Appendix 6 – dnsmasq.conf:

```
# Configuration file for dnsmasq.
#
# Format is one option per line, legal options are the same
# as the long options legal on the command line. See
# "/usr/sbin/dnsmasq --help" or "man 8 dnsmasq" for details.

# Listen on this specific port instead of the standard DNS port
# (53). Setting this to zero completely disables DNS function,
# leaving only DHCP and/or TFTP.
#port=5353

# The following two options make you a better netizen, since they
# tell dnsmasq to filter out queries which the public DNS cannot
# answer, and which load the servers (especially the root servers)
# unnecessarily. If you have a dial-on-demand link they also stop
# these requests from bringing up the link unnecessarily.

# Never forward plain names (without a dot or domain part)
#domain-needed
# Never forward addresses in the non-routed address spaces.
#bogus-priv

# Uncomment these to enable DNSSEC validation and caching:
# (Requires dnsmasq to be built with DNSSEC option.)
#conf-file=%%PREFIX%%/share/dnsmasq/trust-anchors.conf
#dnssec

# Replies which are not DNSSEC signed may be legitimate, because the domain
# is unsigned, or may be forgeries. Setting this option tells dnsmasq to
# check that an unsigned reply is OK, by finding a secure proof that a DS
# record somewhere between the root and the domain does not exist.
# The cost of setting this is that even queries in unsigned domains will need
# one or more extra DNS queries to verify.
#dnssec-check-unsigned

# Uncomment this to filter useless windows-originated DNS requests
# which can trigger dial-on-demand links needlessly.
# Note that (amongst other things) this blocks all SRV requests,
# so don't use it if you use eg Kerberos, SIP, XMMP or Google-talk.
# This option only affects forwarding, SRV records originating for
# dnsmasq (via srv-host= lines) are not suppressed by it.
#filterwin2k

# Change this line if you want dns to get its upstream servers from
# somewhere other that /etc/resolv.conf
#resolv-file=

# By  default,  dnsmasq  will  send queries to any of the upstream
# servers it knows about and tries to favour servers to are  known
```

```
# to be up. Uncommenting this forces dnsmasq to try each query
# with each server strictly in the order they appear in
# /etc/resolv.conf
#strict-order

# If you don't want dnsmasq to read /etc/resolv.conf or any other
# file, getting its servers from this file instead (see below), then
# uncomment this.
#no-resolv

# If you don't want dnsmasq to poll /etc/resolv.conf or other resolv
# files for changes and re-read them then uncomment this.
#no-poll

# Add other name servers here, with domain specs if they are for
# non-public domains.
#server=/localnet/192.168.0.1

# Example of routing PTR queries to nameservers: this will send all
# address->name queries for 192.168.3/24 to nameserver 10.1.2.3
#server=/3.168.192.in-addr.arpa/10.1.2.3

# Add local-only domains here, queries in these domains are answered
# from /etc/hosts or DHCP only.
#local=/localnet/

# Add domains which you want to force to an IP address here.
# The example below send any host in double-click.net to a local
# web-server.
#address=/double-click.net/127.0.0.1

# --address (and --server) work with IPv6 addresses too.
#address=/www.thekelleys.org.uk/fe80::20d:60ff:fe36:f83

# Add the IPs of all queries to yahoo.com, google.com, and their
# subdomains to the vpn and search ipsets:
#ipset=/yahoo.com/google.com/vpn,search

# You can control how dnsmasq talks to a server: this forces
# queries to 10.1.2.3 to be routed via eth1
# server=10.1.2.3@eth1

# and this sets the source (ie local) address used to talk to
# 10.1.2.3 to 192.168.1.1 port 55 (there must be an interface with that
# IP on the machine, obviously).
# server=10.1.2.3@192.168.1.1#55

# If you want dnsmasq to change uid and gid to something other
# than the default, edit the following lines.
#user=
#group=
```

```
# If you want dnsmasq to listen for DHCP and DNS requests only on
# specified interfaces (and the loopback) give the name of the
# interface (eg eth0) here.
# Repeat the line for more than one interface.
#interface=
# Or you can specify which interface _not_ to listen on
#except-interface=
# Or which to listen on by address (remember to include 127.0.0.1 if
# you use this.)
#listen-address=
# If you want dnsmasq to provide only DNS service on an interface,
# configure it as shown above, and then use the following line to
# disable DHCP and TFTP on it.
#no-dhcp-interface=

# On systems which support it, dnsmasq binds the wildcard address,
# even when it is listening on only some interfaces. It then discards
# requests that it shouldn't reply to. This has the advantage of
# working even when interfaces come and go and change address. If you
# want dnsmasq to really bind only the interfaces it is listening on,
# uncomment this option. About the only time you may need this is when
# running another nameserver on the same machine.
#bind-interfaces

# If you don't want dnsmasq to read /etc/hosts, uncomment the
# following line.
#no-hosts
# or if you want it to read another file, as well as /etc/hosts, use
# this.
#addn-hosts=/etc/banner_add_hosts

# Set this (and domain: see below) if you want to have a domain
# automatically added to simple names in a hosts-file.
#expand-hosts

# Set the domain for dnsmasq. this is optional, but if it is set, it
# does the following things.
# 1) Allows DHCP hosts to have fully qualified domain names, as long
#     as the domain part matches this setting.
# 2) Sets the "domain" DHCP option thereby potentially setting the
#    domain of all systems configured by DHCP
# 3) Provides the domain part for "expand-hosts"
#domain=thekelleys.org.uk

# Set a different domain for a particular subnet
#domain=wireless.thekelleys.org.uk,192.168.2.0/24

# Same idea, but range rather then subnet
#domain=reserved.thekelleys.org.uk,192.68.3.100,192.168.3.200
```

```
# Uncomment this to enable the integrated DHCP server, you need
# to supply the range of addresses available for lease and optionally
# a lease time. If you have more than one network, you will need to
# repeat this for each network on which you want to supply DHCP
# service.
#dhcp-range=192.168.0.50,192.168.0.150,12h

# This is an example of a DHCP range where the netmask is given. This
# is needed for networks we reach the dnsmasq DHCP server via a relay
# agent. If you don't know what a DHCP relay agent is, you probably
# don't need to worry about this.
#dhcp-range=192.168.0.50,192.168.0.150,255.255.255.0,12h

# This is an example of a DHCP range which sets a tag, so that
# some DHCP options may be set only for this network.
#dhcp-range=set:red,192.168.0.50,192.168.0.150

# Use this DHCP range only when the tag "green" is set.
#dhcp-range=tag:green,192.168.0.50,192.168.0.150,12h

# Specify a subnet which can't be used for dynamic address allocation,
# is available for hosts with matching --dhcp-host lines. Note that
# dhcp-host declarations will be ignored unless there is a dhcp-range
# of some type for the subnet in question.
# In this case the netmask is implied (it comes from the network
# configuration on the machine running dnsmasq) it is possible to give
# an explicit netmask instead.
#dhcp-range=192.168.0.0,static

# Enable DHCPv6. Note that the prefix-length does not need to be specified
# and defaults to 64 if missing/
#dhcp-range=1234::2, 1234::500, 64, 12h

# Do Router Advertisements, BUT NOT DHCP for this subnet.
#dhcp-range=1234::, ra-only

# Do Router Advertisements, BUT NOT DHCP for this subnet, also try and
# add names to the DNS for the IPv6 address of SLAAC-configured dual-stack
# hosts. Use the DHCPv4 lease to derive the name, network segment and
# MAC address and assume that the host will also have an
# IPv6 address calculated using the SLAAC algorithm.
#dhcp-range=1234::, ra-names

# Do Router Advertisements, BUT NOT DHCP for this subnet.
# Set the lifetime to 46 hours. (Note: minimum lifetime is 2 hours.)
#dhcp-range=1234::, ra-only, 48h

# Do DHCP and Router Advertisements for this subnet. Set the A bit in the RA
# so that clients can use SLAAC addresses as well as DHCP ones.
#dhcp-range=1234::2, 1234::500, slaac
```

```
# Do Router Advertisements and stateless DHCP for this subnet. Clients will
# not get addresses from DHCP, but they will get other configuration information.
# They will use SLAAC for addresses.
#dhcp-range=1234::, ra-stateless

# Do stateless DHCP, SLAAC, and generate DNS names for SLAAC addresses
# from DHCPv4 leases.
#dhcp-range=1234::, ra-stateless, ra-names

# Do router advertisements for all subnets where we're doing DHCPv6
# Unless overridden by ra-stateless, ra-names, et al, the router
# advertisements will have the M and O bits set, so that the clients
# get addresses and configuration from DHCPv6, and the A bit reset, so the
# clients don't use SLAAC addresses.
#enable-ra

# Supply parameters for specified hosts using DHCP. There are lots
# of valid alternatives, so we will give examples of each. Note that
# IP addresses DO NOT have to be in the range given above, they just
# need to be on the same network. The order of the parameters in these
# do not matter, it's permissible to give name, address and MAC in any
# order.

# Always allocate the host with Ethernet address 11:22:33:44:55:66
# The IP address 192.168.0.60
#dhcp-host=11:22:33:44:55:66,192.168.0.60

# Always set the name of the host with hardware address
# 11:22:33:44:55:66 to be "fred"
#dhcp-host=11:22:33:44:55:66,fred

# Always give the host with Ethernet address 11:22:33:44:55:66
# the name fred and IP address 192.168.0.60 and lease time 45 minutes
#dhcp-host=11:22:33:44:55:66,fred,192.168.0.60,45m

# Give a host with Ethernet address 11:22:33:44:55:66 or
# 12:34:56:78:90:12 the IP address 192.168.0.60. Dnsmasq will assume
# that these two Ethernet interfaces will never be in use at the same
# time, and give the IP address to the second, even if it is already
# in use by the first. Useful for laptops with wired and wireless
# addresses.
#dhcp-host=11:22:33:44:55:66,12:34:56:78:90:12,192.168.0.60

# Give the machine which says its name is "bert" IP address
# 192.168.0.70 and an infinite lease
#dhcp-host=bert,192.168.0.70,infinite

# Always give the host with client identifier 01:02:02:04
# the IP address 192.168.0.60
#dhcp-host=id:01:02:02:04,192.168.0.60
```

```
# Always give the InfiniBand interface with hardware address
# 80:00:00:48:fe:80:00:00:00:00:00:00:f4:52:14:03:00:28:05:81 the
# ip address 192.168.0.61. The client id is derived from the prefix
# ff:00:00:00:00:00:02:00:00:02:c9:00 and the last 8 pairs of
# hex digits of the hardware address.
#dhcp-host=id:ff:00:00:00:00:00:02:00:00:02:c9:00:f4:52:14:03:00:28:05:81,192.168.0.61

# Always give the host with client identifier "marjorie"
# the IP address 192.168.0.60
#dhcp-host=id:marjorie,192.168.0.60

# Enable the address given for "judge" in /etc/hosts
# to be given to a machine presenting the name "judge" when
# it asks for a DHCP lease.
#dhcp-host=judge

# Never offer DHCP service to a machine whose Ethernet
# address is 11:22:33:44:55:66
#dhcp-host=11:22:33:44:55:66,ignore

# Ignore any client-id presented by the machine with Ethernet
# address 11:22:33:44:55:66. This is useful to prevent a machine
# being treated differently when running under different OS's or
# between PXE boot and OS boot.
#dhcp-host=11:22:33:44:55:66,id:*

# Send extra options which are tagged as "red" to
# the machine with Ethernet address 11:22:33:44:55:66
#dhcp-host=11:22:33:44:55:66,set:red

# Send extra options which are tagged as "red" to
# any machine with Ethernet address starting 11:22:33:
#dhcp-host=11:22:33:*:*:*,set:red

# Give a fixed IPv6 address and name to client with
# DUID 00:01:00:01:16:d2:83:fc:92:d4:19:e2:d8:b2
# Note the MAC addresses CANNOT be used to identify DHCPv6 clients.
# Note also that the [] around the IPv6 address are obligatory.
#dhcp-host=id:00:01:00:01:16:d2:83:fc:92:d4:19:e2:d8:b2, fred, [1234::5]

# Ignore any clients which are not specified in dhcp-host lines
# or /etc/ethers. Equivalent to ISC "deny unknown-clients".
# This relies on the special "known" tag which is set when
# a host is matched.
#dhcp-ignore=tag:!known

# Send extra options which are tagged as "red" to any machine whose
# DHCP vendorclass string includes the substring "Linux"
#dhcp-vendorclass=set:red,Linux

# Send extra options which are tagged as "red" to any machine one
```

```
# of whose DHCP userclass strings includes the substring "accounts"
#dhcp-userclass=set:red,accounts

# Send extra options which are tagged as "red" to any machine whose
# MAC address matches the pattern.
#dhcp-mac=set:red,00:60:8C:*:*:*

# If this line is uncommented, dnsmasq will read /etc/ethers and act
# on the ethernet-address/IP pairs found there just as if they had
# been given as --dhcp-host options. Useful if you keep
# MAC-address/host mappings there for other purposes.
#read-ethers

# Send options to hosts which ask for a DHCP lease.
# See RFC 2132 for details of available options.
# Common options can be given to dnsmasq by name:
# run "dnsmasq --help dhcp" to get a list.
# Note that all the common settings, such as netmask and
# broadcast address, DNS server and default route, are given
# sane defaults by dnsmasq. You very likely will not need
# any dhcp-options. If you use Windows clients and Samba, there
# are some options which are recommended, they are detailed at the
# end of this section.

# Override the default route supplied by dnsmasq, which assumes the
# router is the same machine as the one running dnsmasq.
#dhcp-option=3,1.2.3.4

# Do the same thing, but using the option name
#dhcp-option=option:router,1.2.3.4

# Override the default route supplied by dnsmasq and send no default
# route at all. Note that this only works for the options sent by
# default (1, 3, 6, 12, 28) the same line will send a zero-length option
# for all other option numbers.
#dhcp-option=3

# Set the NTP time server addresses to 192.168.0.4 and 10.10.0.5
#dhcp-option=option:ntp-server,192.168.0.4,10.10.0.5

# Send DHCPv6 option. Note [] around IPv6 addresses.
#dhcp-option=option6:dns-server,[1234::77],[1234::88]

# Send DHCPv6 option for namservers as the machine running
# dnsmasq and another.
#dhcp-option=option6:dns-server,[::],[1234::88]

# Ask client to poll for option changes every six hours. (RFC4242)
#dhcp-option=option6:information-refresh-time,6h

# Set option 58 client renewal time (T1). Defaults to half of the
```

```
# lease time if not specified. (RFC2132)
#dhcp-option=option:T1,1m

# Set option 59 rebinding time (T2). Defaults to 7/8 of the
# lease time if not specified. (RFC2132)
#dhcp-option=option:T2,2m

# Set the NTP time server address to be the same machine as
# is running dnsmasq
#dhcp-option=42,0.0.0.0

# Set the NIS domain name to "welly"
#dhcp-option=40,welly

# Set the default time-to-live to 50
#dhcp-option=23,50

# Set the "all subnets are local" flag
#dhcp-option=27,1

# Send the etherboot magic flag and then etherboot options (a string).
#dhcp-option=128,e4:45:74:68:00:00
#dhcp-option=129,NIC=eepro100

# Specify an option which will only be sent to the "red" network
# (see dhcp-range for the declaration of the "red" network)
# Note that the tag: part must precede the option: part.
#dhcp-option = tag:red, option:ntp-server, 192.168.1.1

# The following DHCP options set up dnsmasq in the same way as is specified
# for the ISC dhcpcd in
# http://www.samba.org/samba/ftp/docs/textdocs/DHCP-Server-Configuration.txt
# adapted for a typical dnsmasq installation where the host running
# dnsmasq is also the host running samba.
# you may want to uncomment some or all of them if you use
# Windows clients and Samba.
#dhcp-option=19,0           # option ip-forwarding off
#dhcp-option=44,0.0.0.0     # set netbios-over-TCP/IP nameserver(s) aka WINS server(s)
#dhcp-option=45,0.0.0.0     # netbios datagram distribution server
#dhcp-option=46,8           # netbios node type

# Send an empty WPAD option. This may be REQUIRED to get windows 7 to behave.
#dhcp-option=252,"\n"

# Send RFC-3397 DNS domain search DHCP option. WARNING: Your DHCP client
# probably doesn't support this......
#dhcp-option=option:domain-search,eng.apple.com,marketing.apple.com

# Send RFC-3442 classless static routes (note the netmask encoding)
#dhcp-option=121,192.168.1.0/24,1.2.3.4,10.0.0.0/8,5.6.7.8
```

```
# Send vendor-class specific options encapsulated in DHCP option 43.
# The meaning of the options is defined by the vendor-class so
# options are sent only when the client supplied vendor class
# matches the class given here. (A substring match is OK, so "MSFT"
# matches "MSFT" and "MSFT 5.0"). This example sets the
# mtftp address to 0.0.0.0 for PXEClients.
#dhcp-option=vendor:PXEClient,1,0.0.0.0

# Send microsoft-specific option to tell windows to release the DHCP lease
# when it shuts down. Note the "i" flag, to tell dnsmasq to send the
# value as a four-byte integer - that's what microsoft wants. See
# http://technet2.microsoft.com/WindowsServer/en/library/a70f1bb7-d2d4-49f0-96d6-
4b7414ecfaae1033.mspx?mfr=true
#dhcp-option=vendor:MSFT,2,1i

# Send the Encapsulated-vendor-class ID needed by some configurations of
# Etherboot to allow is to recognise the DHCP server.
#dhcp-option=vendor:Etherboot,60,"Etherboot"

# Send options to PXELinux. Note that we need to send the options even
# though they don't appear in the parameter request list, so we need
# to use dhcp-option-force here.
# See http://syslinux.zytor.com/pxe.php#special for details.
# Magic number - needed before anything else is recognised
#dhcp-option-force=208,f1:00:74:7e
# Configuration file name
#dhcp-option-force=209,configs/common
# Path prefix
#dhcp-option-force=210,/tftpboot/pxelinux/files/
# Reboot time. (Note 'i' to send 32-bit value)
#dhcp-option-force=211,30i

# Set the boot filename for netboot/PXE. You will only need
# this if you want to boot machines over the network and you will need
# a TFTP server; either dnsmasq's built-in TFTP server or an
# external one. (See below for how to enable the TFTP server.)
#dhcp-boot=pxelinux.0

# The same as above, but use custom tftp-server instead machine running dnsmasq
#dhcp-boot=pxelinux,server.name,192.168.1.100

# Boot for iPXE. The idea is to send two different
# filenames, the first loads iPXE, and the second tells iPXE what to
# load. The dhcp-match sets the ipxe tag for requests from iPXE.
#dhcp-boot=undionly.kpxe
#dhcp-match=set:ipxe,175 # iPXE sends a 175 option.
#dhcp-boot=tag:ipxe,http://boot.ipxe.org/demo/boot.php

# Encapsulated options for iPXE. All the options are
# encapsulated within option 175
#dhcp-option=encap:175, 1, 5b        # priority code
```

```
#dhcp-option=encap:175, 176, 1b       # no-proxydhcp
#dhcp-option=encap:175, 177, string   # bus-id
#dhcp-option=encap:175, 189, 1b       # BIOS drive code
#dhcp-option=encap:175, 190, user     # iSCSI username
#dhcp-option=encap:175, 191, pass     # iSCSI password

# Test for the architecture of a netboot client. PXE clients are
# supposed to send their architecture as option 93. (See RFC 4578)
#dhcp-match=peecees, option:client-arch, 0 #x86-32
#dhcp-match=itanics, option:client-arch, 2 #IA64
#dhcp-match=hammers, option:client-arch, 6 #x86-64
#dhcp-match=mactels, option:client-arch, 7 #EFI x86-64

# Do real PXE, rather than just booting a single file, this is an
# alternative to dhcp-boot.
#pxe-prompt="What system shall I netboot?"
# or with timeout before first available action is taken:
#pxe-prompt="Press F8 for menu.", 60

# Available boot services. for PXE.
#pxe-service=x86PC, "Boot from local disk"

# Loads <tftp-root>/pxelinux.0 from dnsmasq TFTP server.
#pxe-service=x86PC, "Install Linux", pxelinux

# Loads <tftp-root>/pxelinux.0 from TFTP server at 1.2.3.4.
# Beware this fails on old PXE ROMS.
#pxe-service=x86PC, "Install Linux", pxelinux, 1.2.3.4

# Use bootserver on network, found my multicast or broadcast.
#pxe-service=x86PC, "Install windows from RIS server", 1

# Use bootserver at a known IP address.
#pxe-service=x86PC, "Install windows from RIS server", 1, 1.2.3.4

# If you have multicast-FTP available,
# information for that can be passed in a similar way using options 1
# to 5. See page 19 of
# http://download.intel.com/design/archives/wfm/downloads/pxespec.pdf


# Enable dnsmasq's built-in TFTP server
#enable-tftp

# Set the root directory for files available via FTP.
#tftp-root=/var/ftpd

# Do not abort if the tftp-root is unavailable
#tftp-no-fail

# Make the TFTP server more secure: with this set, only files owned by
```

```
# the user dnsmasq is running as will be send over the net.
#tftp-secure

# This option stops dnsmasq from negotiating a larger blocksize for TFTP
# transfers. It will slow things down, but may rescue some broken TFTP
# clients.
#tftp-no-blocksize

# Set the boot file name only when the "red" tag is set.
#dhcp-boot=tag:red,pxelinux.red-net

# An example of dhcp-boot with an external TFTP server: the name and IP
# address of the server are given after the filename.
# Can fail with old PXE ROMS. Overridden by --pxe-service.
#dhcp-boot=/var/ftpd/pxelinux.0,boothost,192.168.0.3

# If there are multiple external tftp servers having a same name
# (using /etc/hosts) then that name can be specified as the
# tftp_servername (the third option to dhcp-boot) and in that
# case dnsmasq resolves this name and returns the resultant IP
# addresses in round robin fashion. This facility can be used to
# load balance the tftp load among a set of servers.
#dhcp-boot=/var/ftpd/pxelinux.0,boothost,tftp_server_name

# Set the limit on DHCP leases, the default is 150
#dhcp-lease-max=150

# The DHCP server needs somewhere on disk to keep its lease database.
# This defaults to a sane location, but if you want to change it, use
# the line below.
#dhcp-leasefile=/var/lib/misc/dnsmasq.leases

# Set the DHCP server to authoritative mode. In this mode it will barge in
# and take over the lease for any client which broadcasts on the network,
# whether it has a record of the lease or not. This avoids long timeouts
# when a machine wakes up on a new network. DO NOT enable this if there's
# the slightest chance that you might end up accidentally configuring a DHCP
# server for your campus/company accidentally. The ISC server uses
# the same option, and this URL provides more information:
# http://www.isc.org/files/auth.html
#dhcp-authoritative

# Set the DHCP server to enable DHCPv4 Rapid Commit Option per RFC 4039.
# In this mode it will respond to a DHCPDISCOVER message including a Rapid Commit
# option with a DHCPACK including a Rapid Commit option and fully committed address
# and configuration information. This must only be enabled if either the server is
# the only server for the subnet, or multiple servers are present and they each
# commit a binding for all clients.
#dhcp-rapid-commit

# Run an executable when a DHCP lease is created or destroyed.
```

```
# The arguments sent to the script are "add" or "del",
# then the MAC address, the IP address and finally the hostname
# if there is one.
#dhcp-script=/bin/echo

# Set the cachesize here.
#cache-size=150

# If you want to disable negative caching, uncomment this.
#no-negcache

# Normally responses which come from /etc/hosts and the DHCP lease
# file have Time-To-Live set as zero, which conventionally means
# do not cache further. If you are happy to trade lower load on the
# server for potentially stale date, you can set a time-to-live (in
# seconds) here.
#local-ttl=

# If you want dnsmasq to detect attempts by Verisign to send queries
# to unregistered .com and .net hosts to its sitefinder service and
# have dnsmasq instead return the correct NXDOMAIN response, uncomment
# this line. You can add similar lines to do the same for other
# registries which have implemented wildcard A records.
#bogus-nxdomain=64.94.110.11

# If you want to fix up DNS results from upstream servers, use the
# alias option. This only works for IPv4.
# This alias makes a result of 1.2.3.4 appear as 5.6.7.8
#alias=1.2.3.4,5.6.7.8
# and this maps 1.2.3.x to 5.6.7.x
#alias=1.2.3.0,5.6.7.0,255.255.255.0
# and this maps 192.168.0.10->192.168.0.40 to 10.0.0.10->10.0.0.40
#alias=192.168.0.10-192.168.0.40,10.0.0.0,255.255.255.0

# Change these lines if you want dnsmasq to serve MX records.

# Return an MX record named "maildomain.com" with target
# servermachine.com and preference 50
#mx-host=maildomain.com,servermachine.com,50

# Set the default target for MX records created using the localmx option.
#mx-target=servermachine.com

# Return an MX record pointing to the mx-target for all local
# machines.
#localmx

# Return an MX record pointing to itself for all local machines.
#selfmx

# Change the following lines if you want dnsmasq to serve SRV
```

```
# records.  These are useful if you want to serve ldap requests for
# Active Directory and other windows-originated DNS requests.
# See RFC 2782.
# You may add multiple srv-host lines.
# The fields are <name>,<target>,<port>,<priority>,<weight>
# If the domain part if missing from the name (so that is just has the
# service and protocol sections) then the domain given by the domain=
# config option is used. (Note that expand-hosts does not need to be
# set for this to work.)

# A SRV record sending LDAP for the example.com domain to
# ldapserver.example.com port 389
#srv-host=_ldap._tcp.example.com,ldapserver.example.com,389

# A SRV record sending LDAP for the example.com domain to
# ldapserver.example.com port 389 (using domain=)
#domain=example.com
#srv-host=_ldap._tcp,ldapserver.example.com,389

# Two SRV records for LDAP, each with different priorities
#srv-host=_ldap._tcp.example.com,ldapserver.example.com,389,1
#srv-host=_ldap._tcp.example.com,ldapserver.example.com,389,2

# A SRV record indicating that there is no LDAP server for the domain
# example.com
#srv-host=_ldap._tcp.example.com

# The following line shows how to make dnsmasq serve an arbitrary PTR
# record. This is useful for DNS-SD. (Note that the
# domain-name expansion done for SRV records _does_not
# occur for PTR records.)
#ptr-record=_http._tcp.dns-sd-services,"New Employee Page._http._tcp.dns-sd-services"

# Change the following lines to enable dnsmasq to serve TXT records.
# These are used for things like SPF and zeroconf. (Note that the
# domain-name expansion done for SRV records _does_not
# occur for TXT records.)

#Example SPF.
#txt-record=example.com,"v=spf1 a -all"

#Example zeroconf
#txt-record=_http._tcp.example.com,name=value,paper=A4

# Provide an alias for a "local" DNS name. Note that this _only_ works
# for targets which are names from DHCP or /etc/hosts. Give host
# "bert" another name, bertrand
#cname=bertand,bert

# For debugging purposes, log each DNS query as it passes through
# dnsmasq.
```

```
#log-queries

# Log lots of extra information about DHCP transactions.
#log-dhcp

# Include another lot of configuration options.
#conf-file=/etc/dnsmasq.more.conf
#conf-dir=/etc/dnsmasq.d

# Include all the files in a directory except those ending in .bak
#conf-dir=/etc/dnsmasq.d,.bak

# Include all files in a directory which end in .conf
#conf-dir=/etc/dnsmasq.d/,*.conf

# If a DHCP client claims that its name is "wpad", ignore that.
# This fixes a security hole. see CERT Vulnerability VU#598349
#dhcp-name-match=set:wpad-ignore,wpad
#dhcp-ignore-names=tag:wpad-ignore

# Delays sending DHCPOFFER and proxydhcp replies for at least the specified number of
seconds.
dhcp-mac=set:client_is_a_pi,B8:27:EB:*:*:*
dhcp-reply-delay=tag:client_is_a_pi,2

interface=wlan0  # listening interface
dhcp-range=10.0.0.50,10.0.0.250,255.0.0.0,24h
domain=atgp
address=/gw.wlan/10.0.0.5  # Alias for this router
```

Appendix 7 – hostapd.conf:

```
country_code=US
interface=wlan0
ssid=ATGP
hw_mode=g
channel=7
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=ChangeOnInstall
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Appendix 8 – exemplar script to stop hotspot and enable WIFI:

This works from a console window, but is failing over an ssh connection over WIFI. I include it if people would like to experiment further, but with two WIFI adapters, this function is not necessary.

```
ip link set dev wlan0 down
systemctl stop hostapd
systemctl stop dnsmasq
ip addr flush dev wlan0
ip link set dev wlan0 up
dhcpcd -n wlan0 > /dev/null
wpa_supplicant -B -i wlan0 -c /etc/wpa_supplicant/wpa_supplicant.conf
hostname -I
```