# EAS SAME to APRS Message Converter

The U.S. National Weather Service (NWS) operates more than 1,000 VHF FM radio stations that continuously transmit weather information. The frequencies, in MHz, are:

| | | | | | | |
|---|---|---|---|---|---|---|
| 162.400 | 162.425 | 162.450 | 162.475 | 162.500 | 162.525 | 162.550 |

These stations also transmit special warnings about severe weather, disasters (natural & manmade), and public safety.

Alerts are sent in an obnoxious (or maybe beautiful, depending on your viewpoint) sounding digital form known as Emergency Alert System (EAS) Specific Area Message Encoding (SAME). You can hear a sample here.

It is possible to buy radios that decode these messages but what fun is that? We are ham radio operators so we want to build our own from stuff that we already have sitting around.

It has been suggested that retransmitting these alerts over the APRS network could aid in local situational awareness. This is not difficult. There are already various open source packages to demodulate these beautiful (or was it obnoxious?) sounds and decipher the strange resulting text. We just need to gather some of those components and add a little software "glue" to hold them together.

Here is one possible approach.

The "**direwolf**" software TNC has the ability to receive EAS SAME signals while sending and receiving APRS on other radios.

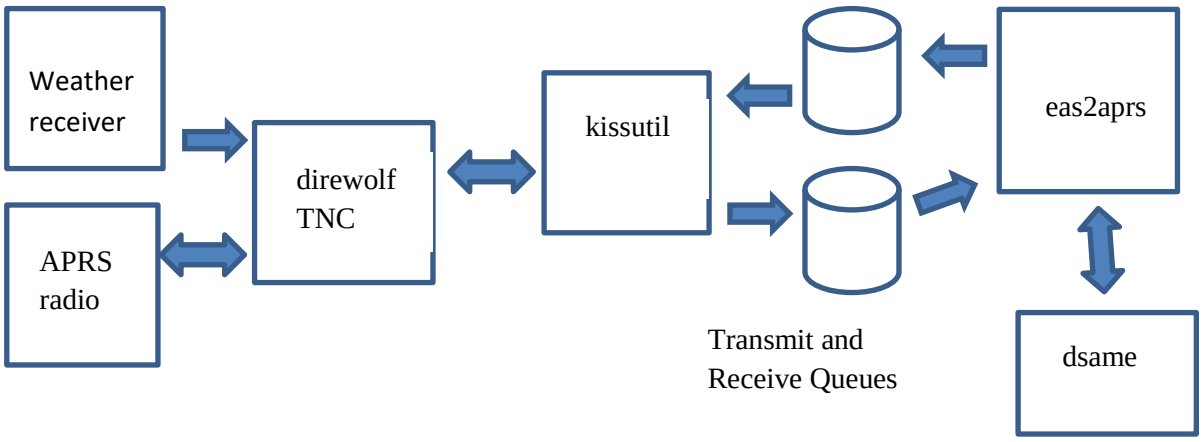
There is an existing python application, called **dsame**, that will convert the encoded character format to plain English. For example,

```
ZCZC-WXR-RWT-020103-020209-020091-020121-029047-029165-029095-
029037+0030-1051700-KEAX/NWS
```

Would be turned into:

```
The National Weather Service in Kansas City/Pleasant Hill, MO has issued a
Required Weekly Test valid until 01:30 PM for the following counties in
Kansas: Johnson, Leavenworth, Miami, Wyandotte, and for the following counties
in Missouri: Cass, Clay, Jackson, Platte. (KEAX/NWS)
```

We just need to package that into APRS "messages" and connect it all together somehow.




“kissutil” is a general purpose utility for talking to a KISS TNC. One of its capabilities is to communicate with another application through a file system based queue. For each received packet, a file is created in the receive queue directory. Any files found in the transmit queue directory is sent to the KISS TNC for transmission.

The examples here are for Linux, sometimes specifically the Raspberry Pi, but it should all work on Windows, Mac OSX, and other operating systems with some minor adjustments.

## Step 1 - Obtain dsame

Download the application:

```
cd
git clone https://github.com/nivex/dsame
cd dsame
chmod +x dsame.py
```

Test it.

```
./dsame.py --msg "ZCZC-WXR-RWT-020103-020209-020091-020121-029047-
029165-029095-029037+0030-1051700-KEAX/NWS"
```

This should appear

```
The National Weather Service in Kansas City/Pleasant Hill, MO has issued a
Required Weekly Test valid until 01:30 PM for the following counties in
Kansas: Johnson, Leavenworth, Miami, Wyandotte, and for the following counties
in Missouri: Cass, Clay, Jackson, Platte. (KEAX/NWS)
```

## Step 2 - Obtain EAS to APRS converter

Download the converter from github:

```
cd
git https://github.com/wb2osz/eas2aprs
cd eas2aprs
```

Copy in the dsame files.

```
cp  ../dsame/*.py .
chmod +x *.py
```

Start it up:

```
./eas2aprs
```

In another window, we will simulate reception of a EAS SAME message.  This is all one line.

```
echo "X>X:{DEZCZC-WXR-RWT-020103-020209-020091-020121-029047-029165-
029095-029037+0030-1051700-KEAX/NWS" > /dev/shm/RQ/x
```

eas2aprs should spring into action and print something to show what is going on.

```
Processing /dev/shm/RQ/x ...

X>X:{DEZCZC-WXR-RWT-020103-020209-020091-020121-029047-029165-029095-
029037+0030-1051700-KEAX/NWS

Transmitting...

HAM123>APZEAS::NWS      :[1/4] The National Weather Service in Kansas
City/Pleasant Hill, MO has issued a

HAM123>APZEAS::NWS      :[2/4] Required Weekly Test valid until 01:30
PM for the following counties in

HAM123>APZEAS::NWS      :[3/4] Kansas: Johnson, Leavenworth, Miami,
Wyandotte, and for the following counties
```

```
HAM123>APZEAS::NWS      :[4/4] in Missouri: Cass, Clay, Jackson,
Platte. (KEAX/NWS)HAM123>APZEAS::NWS      :[4/4] Cass, Clay, Jackson,
Platte. (KEAX/NWS)
```

Notice that several files are now in /dev/shm/TQ

```
cat /dev/shm/TQ/*

HAM123>APZEAS::NWS      :[1/4] The National Weather Service in Kansas
City/Pleasant Hill, MO has issued a

HAM123>APZEAS::NWS      :[2/4] Required Weekly Test valid until 01:30
PM for the following counties in

HAM123>APZEAS::NWS      :[3/4] Kansas: Johnson, Leavenworth, Miami,
Wyandotte, and for the following counties

HAM123>APZEAS::NWS      :[4/4] in Missouri: Cass, Clay, Jackson,
Platte. (KEAX/NWS)
```

The APRS “messages” are in the transmit queue. Now we just need a way to get them to the transmitter.

## Step 3 - Install direwolf

This requires a new feature in version 1.6. At the time this is being written, version 1.6 is not released yet so you will need to use the development branch as shown here.

```
sudo apt-get install cmake
sudo apt-get install libasound2-dev
sudo apt-get install libudev-dev

cd ~
git clone https://www.github.com/wb2osz/direwolf
cd direwolf
git checkout dev

mkdir build
cd build
cmake ..
make -j4

sudo make install
make install-conf
```

This should work for Raspbian on the Raspberry Pi. Consult the direwolf documentation for other operating systems.

# Step 4 - Connect to radios and configure audio devices

## Weather Receiver

For the weather receiver, I'm using an RTL-SDR dongle. If you're planning to buy one, you will probably regret getting the cheapest one. They are not all the same. I have an early cheap one and it has a frequency error of about 63 parts per million (ppm). On the 2 meter band that is an error of 9 kHz!!!

I'm currently using the RTL-SDR BLOG V3, which has 1 ppm accuracy and other improvements.

We need to install the **rtl_fm** application if using this type of SDR for the weather receiver.

Instructions are summarized from the rtl-sdr wiki. Refer to the original for more details.

```
sudo apt-get update
sudo apt-get install cmake build-essential libusb-1.0-0-dev
cd ~
git clone git://git.osmocom.org/rtl-sdr.git
cd rtl-sdr
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON -DDETACH_KERNEL_DRIVER=ON
make
sudo make install
sudo ldconfig
```

## APRS Transceiver

For the APRS radio interface, I normally use a cheap USB-to-audio adapter but there are many other choices available. More details can be found in ***Raspberry-Pi-APRS.pdf*** and ***User-Guide.pdf*** in the direwolf documentation directory. No point in repeating all of that here.

## Configuration File

This next part is important. A proper configuration file (or command line options in some cases) is needed to select the desired audio devices and modems. The distribution contains a file called **eas.conf** which contains something like:

```
# The first audio device will be the USB (or other type of)
# “sound card.”  It normally shows up as card 1 but,
# under some circumstances, it could be different.
# The modem defaults to 1200 bps AFSK so that does not need
# to be specified.
# You do need to specify a method to activate the transmitter.
# Most popular methods are a Raspberry Pi GPIO pin or a GPIO
# pin of the USB audio adapter.

ADEVICE  plughw:1,0
CHANNEL 0
#PTT GPIO 25
PTT CM108

# The second audio device (1 because numbering starts at 0)
# is the RTL SDR.  There is no audio output for this channel.
# The rtl_fm application writes to stdout with a sample rate
# of 24000/sec.  We need to be listening to stdin with the
# same sample rate.

ADEVICE1  stdin  null
ARATE 24000
CHANNEL 2
MODEM EAS /1
```

Start up direwolf, specifying this configuration file.

```
rtl_fm -f 162.475M  |  direwolf -c eas.conf -a 60
```

Naturally, you need to change the frequency for a station in your area.

In cases like this, you usually see “-“, at the end of the line, meaning read audio from stdin.  We can’t use that in this case.  Most options on the command line apply to the first audio device and first channel.  In this example the SDR is the second audio device so this needs to be specified in the configuration file.

The “-a 60” option is just for troubleshooting.  You should see something like this every 60 seconds:

```
ADEVICE0: Sample rate approx. 44.1 k, 0 errors, receive audio level CH0 0

ADEVICE1: Sample rate approx. 24.0 k, 0 errors, receive audio level CH2 96
```

Notice the ADEVICE1 lines.  The sample rate should be 24.0 k and the audio level for channel 2 should be somewhere in the vicinity of 90 or 100.  This means you are getting audio from **rtl_fm**.

## Step 5 - Run kissutil

This is the final missing link. Run in a separate command window:

```
kissutil -f /dev/shm/TQ -o /dev/shm/RQ
```

**kissutil** will attempt to connect to a network TNC. It does not need to be **direwolf**; it could be any other network KISS TNC. In the absence of a specific different host on the command line, it will look on the same host.

**kissutil** will then look for files in the transmit queue (-f option), convert from text to KISS format, and send them to the TNC for transmission.

Any received packets are placed in the receive queue (-o option).

Received EAS SAME messages are encapsulated as APRS user defined data so attached applications can read them over the KISS interface, just like the APRS packets. The fake source and destination addresses are not important. The information part begins with:

| | |
|---|---|
| { | means user defined data |
| D | indicates direwolf |
| E | is for EAS SAME |

The rest of the information part should being with ZCZC or NNNN.

## Step 6 - Wait for it…. Or simulate reception

Emergency alerts are not that frequent, so how can we test it? Test messages are sent between 10:00 AM and noon, local time, on Wednesdays. We can still do some testing while waiting for that time.

Type this all on a single line.

```
echo "X>X:{DEZCZC-WXR-RWT-020103-020209-020091-020121-029047-029165-
029095-029037+0030-1051700-KEAX/NWS" > /dev/shm/RQ/x
```

aes2aprs.py should print:

```
Processing /dev/shm/RQ/x ...

X>X:{DEZCZC-WXR-RWT-020103-020209-020091-020121-029047-029165-029095-
029037+0030-1051700-KEAX/NWS

Transmitting...

HAM123>APZEAS::NWS      :[1/4] The National Weather Service in Kansas
City/Pleasant Hill, MO has issued a

HAM123>APZEAS::NWS      :[2/4] Required Weekly Test valid until 01:30
PM for the following counties in

HAM123>APZEAS::NWS      :[3/4] Kansas: Johnson, Leavenworth, Miami,
Wyandotte, and for the following counties

HAM123>APZEAS::NWS      :[4/4] in Missouri: Cass, Clay, Jackson,
Platte. (KEAX/NWS)
```

Kissutil should print:

```
Processing 200613.224157.910 for transmit...

HAM123>APZEAS::NWS      :[4/4] in Missouri: Cass, Clay, Jackson,
Platte. (KEAX/NWS)

Processing 200613.224157.904 for transmit...

HAM123>APZEAS::NWS      :[3/4] Kansas: Johnson, Leavenworth, Miami,
Wyandotte, and for the following counties

Processing 200613.224157.899 for transmit...

HAM123>APZEAS::NWS      :[2/4] Required Weekly Test valid until 01:30
PM for the following counties in
```

```
Processing 200613.224157.893 for transmit...

HAM123>APZEAS::NWS      :[1/4] The National Weather Service in Kansas
City/Pleasant Hill, MO has issued a
```

Do you notice anything odd? They are in the wrong order. Currently kissutil doesn't sort the file names found in a directory. That needs to be fixed.

And finally direwolf should show transmissions:

```
[0L] HAM123>APZEAS::NWS      :[4/4] in Missouri: Cass, Clay, Jackson,
Platte. (KEAX/NWS)

[0L] HAM123>APZEAS::NWS      :[3/4] Kansas: Johnson, Leavenworth,
Miami, Wyandotte, and for the following counties

[0L] HAM123>APZEAS::NWS      :[2/4] Required Weekly Test valid until
01:30 PM for the following counties in

[0L] HAM123>APZEAS::NWS      :[1/4] The National Weather Service in
Kansas City/Pleasant Hill, MO has issued a
```

This has tested everything other than the weather receiver and **direwolf** providing the modem function.

Now we just have to wait until Wednesday morning, for the weekly required test, to see if it actually receives over the radio.

TO BE CONTINUED….

## Conclusion

This example illustrates how different building blocks can be combined to construct a new APRS application.

- **direwolf** is a high performance APRS / Packet Radio network TNC with the ability to receive EAS SAME messages and communicate over multiple radio channels at the same time.
- **kissutil** provides the glue between the KISS protocol and a file based queuing system.
- **dsame** converts the coded EAS SAME messages to normal text.
- A new python script provides the interface between **dsame** and a file based queuing system.

Currently, this is just a proof of concept to show how it can all work together. More work is needed to make it a practical system. Suggested improvements:

(1) Add command line options so the script does not need to be edited for things like source callsign.

(2) Remember the messages and retransmit periodically until the expiration time.

(3) The local weather radio station should send a test transmission each week. If nothing is received for 8 days, provide some indication of system failure.

(4) Use the rest of the APRS message addressee field to convey some useful information.

(5) Add an APRS region specifier so the affected region can be highlighted on a map.

(6) The EAS SAME messages have no error checking so errors from radio interference could go unnoticed. If all 3 messages were received, within the expected time frame, see if they are all identical. If any mismatch, compare them bit by bit and take the more popular value for each bit position.

## Share Your Experience

https://groups.io/g/direwolf is the best place to ask questions and share information with other users. You might find your questions have already been answered here. Others would like to hear about your interesting activities.