Broadcast your Node for Public Listening

The Broadcastify arm of Radio Reference allows the public to broadcast radio transmissions via the Internet. There are many restrictions on what can and can not be broadcast but amateur radio is generally allowed.

I have setup a Broadcastify channel for my Philadelphia area Allstar hub using an Odroid C1+ which also runs my hub connections. While this could and has been done with a Raspberry Pi 2 the Odroid has a higher clock speed and on most operations is noticeably faster than the RPi2. I brought the Odroid up in Archlinux and ported most of the RPi2 Allstar features over. The Odroid loafs along as a hub with 20-30 nodes connected and data to Broadcastify which by itself takes very little resources.

Allstar has a mostly undocumented command called 'outstreamcmd' that can be placed in the node definition section of rpt.conf which can be used to send the audio data stream to external sources.

But first you need to setup a broadcast channel starting at the radioreference.com web page.

Setup your Account

At the site select 'Live Audio' and on the drop down 'Become a Feed Provider' You will be presented with a lot of information. You can ignore any hardware requirements as Allstar with the help of a couple of other programs will feed your audio directly to Broadcastify. No other hardware is required. If you want to continue you need to register with Radio Reference and then you can proceed to register your Allstar broadcast.

Before I continue just a word on who should do this. Generally this is done on hubs or connections that see a lot of activity. It would make little sense to do this for a lone simplex node. Some amount of Linux capability and understanding is required and once established you are generally expected to keep your broadcast active. Dormant accounts are deleted.

Software Requirements

You will need the following packages installed to proceed.

libshout
libxml (libxml2)
taglib
lame – which should already be installed

To install a package in Archlinux use 'pacman -S <packagename>'

(Note that you may need to use 'pacman -Sy <packagename>')

Once these packages are installed you will need to download the 'ezstream' package. This can be found at icecast.org/ezstream/

Scroll down to the downloads. The current version (10/2015) is 0.6.0. If you are using a windows computer to browse you can either download the tar.gz file and transfer it to your Asterisk server or put your cursor over the link, right click, copy link location. Then if you have a putty or winscp session opened to your Allstar server just go to a directory on the server where you want to process the file, at the linux prompt type 'wget' then right click the mouse. The link should be inserted on the line and hitting enter will download the file, then untar.

tar xvzf ezstream-0.6.0.tar.gz

Then cd to the directory created and follow the install instructions.

Software Setup

At this point you have all the software required but you do need to create and configure the ezstream xml file and add one command to your node stanza in rpt.conf.

Create and edit the ezstream.xml with your information. The url and password are supplied by broadcastify when your broadcast is setup at the site. You can copy, paste, and edit the xml script shown here editing out the comments to the right.

```
<ezstream>
   <url>http://audio9.radioreference.com:80/#######</url>   < Entire URL given by Broadcastify
   <sourcepassword>######</sourcepassword>                   <  Also password
   <format>MP3</format>
   <filename>stdin</filename>
   <svrinfoname>#########</svrinfoname>                      < Name you gave your broadcast
   <svrinfourl>http://www.radioreference.com/</svrinfourl>
   <svrinfogenre>Amateur Radio</svrinfogenre>
   <svrinfodescription>##############</svrinfodescription> < Short description
   <svrinfobitrate>16</svrinfobitrate>
   <svrinfochannels>1</svrinfochannels>
   <svrinfosamplerate>22050</svrinfosamplerate>
   <svrinfopublic>1</svrinfopublic>
</ezstream>
```

Save this file in /etc/ezstream.xml

Now edit your /etc/asterisk/rpt.conf file and add the following line in the node stanza of the node you want to broadcast. This pipes the data through lame to convert to mp3 with the correct data format and finally to ezstream to connect and send the data to Broadcastify.

outstreamcmd=/bin/sh,-c,/usr/bin/lame --preset cbr 16 -r -m m -s 8 --bitwidth 16 - - 2> /tmp/stream.status | /usr/local/bin/ezstream -qvc /etc/ezstream.xml

This must be **ALL** on one line. The '2> /tmp/stream.status' is optional. It displays any error status from lame and is helpful for testing. If all is OK nothing is output to the file. Be sure to type this in exactly

as shown and in particular the - - (dash,dash) with and without a space. You could cut and paste from here but make sure it all ends up on one line before you save it.

At this point you can restart Asterisk or reboot your system and it should start broadcasting. This is no feedback that it is working other than checking at Radio Reference. You also will get an email telling you your broadcast is up and also down if it goes down. You can do a 'ps ax' at the Linux prompt and see the processes. They should look similar to this of course with different job numbers and times. This shows a system that has been running for several weeks -

```
 1557 pts/0    Sl   2079:48 /usr/sbin/asterisk -f -vvvg -c
 1596 pts/0    S     0:00 /bin/sh -c /usr/bin/lame --preset cbr 16 -r -m m -s 8 --bitwidth 16 - - 2
 1597 pts/0    S    63:28 /usr/bin/lame --preset cbr 16 -r -m m -s 8 --bitwidth 16 - -
 1598 pts/0    S     0:42 /usr/local/bin/ezstream -qvc /etc/ezstream.xml
```

ezstream is prone to occasional failure. I find it usually runs for weeks between failure but others have reported lesser times. When it stops broadcasting your first indication will probably be an email message from Radio Reference. You have several options to get it going again. The first one to try that usually works is to kill -9 the ezstream process. The process is immediately recreated. You could do this on a cron job daily but on one hand it might be unnecessary and on the other it could have failed 23 hours before and be down all that time. I have found that when it fails the ezstream cpu time skyrockets. So I created a script that looks at the process time for the ezstream and it it goes above 1 it kills the process. I am still fine tuning this but it seems to work. To just use the manual kill do -

```
kill -9 `pidof ezstream`
```

Here is the script which looks at ezstream cpu time and kills the process as a background job -

```
# !/bin/bash
# Check CPU usage of ezstream - normally should be 0
# If it fails kill the process
#
# WA3DSP - 10/2015

ezstreampid=`pidof ezstream`
if [ -z "$ezstreampid" ]
  then
    echo "EZstream not running"
    exit
fi
#ezstreamcpu=`ps -p $ezstreampid -o %cpu | sed -n '2p'`
ezstreamcpu=`ps -p $ezstreampid -o time= | awk -F: '{ print ($1 * 3600) + ($2 * 60) + $3 }'`
ezstreamstat=`echo "60>$ezstreamcpu" | bc`

if [ "$ezstreamstat" -eq "1" ]
  then
    echo "Ezstream OK"
  else
```

```
    echo "Ezstream fail"
    kill -9 $ezstreampid
    echo "`date` - Ezstream restarted" >> /tmp/ezstream.status
fi

# End of script
```

Put this script in any convenient location and call it from a cron job. Here is an example cron showing the file stored in /root which runs once a minute. You could run it less often but the overhead is minimal.

* * * * * /root/chk_ezstream.sh

The other options although less desirable, but maybe necessary in some cases, are to restart asterisk or reboot the system.

Final Tweaking

Once you are broadcasting you can fine tune your settings such as the description on your Broadcastify account. Note that the broadcasts are delayed about 20 seconds. This is NOT a delay through Allstar but rather a forced delay at Broadcastify. You also have the ability in a player to pause the broadcast and start back up where you left off and also back up in time to listen to prior broadcasts.

Remember this is not something everyone needs to do but if you have a higher profile node it might be an asset to local listeners. Once it is put on Broadcastify the whole world can listen.

Have fun!

WA3DSP