fs_rpt.pl

About

by Bret McDanel

fs_rpt.pl will operate in a similar fashion to Asterisk's app_rpt originally written by Jim Dixon. This will let you build a repeater, a simple radio endpoint, or other operations as the situation dictates.

This was written in Perl specifically to encourage more people to modify the script to suit their needs, without requiring a compiler to be installed nor knowledge of how to compile programs. It is released under the MPL license to encourage adoption into other projects.

FreeSWITCH Has A Radio Endpoint!

by trixter on Sep.03, 2009, under Radio, Telephony

I finished the initial version of my perl script that will link FreeSWITCH to a radio device. While I am primarily doing this with amateur radio in mind, you could use this to connect an FRS, GMRS, or other similar type of radio.

Using the FT-897 PTT control circuit, you can control PTT by connecting the various connectors to virtually any radio. Some will not be compatible, some will require hardware modification (removal of the mic/speaker and replacing them with headphone wires and connecting the PTT button to the serial port). I will not be discussing how to do that here. If you can't figure that out you probably shouldn't be doing it.

In order to use the script, you must install FreeSWITCH. Once you have done that you have to configure a few things for this to work.

Requirements

- Radio with speaker/mic connected to sound card, and a PTT connection to a serial port.
- Serial port for PTT (push to talk) control.
- Perl (Windows, Linux or OSX should all be usable)
    - Device::SerialPins
    - Getopt::Std
    - FreeSWITCH::Client - in {src}/scripts/perl
    - POSIX ':signal_h'
    - Switch
    - File::stat
- mod_event_socket
- mod_conference
- mod_portaudio
- FreeSWITCH-contrib/trixter/radio/fs_rpt.pl

Optional

- morse.js - to play your callsign in morse code; required in some jurisdictions
- TTS engine - to make announcements periodically

Installation and Configuration

Radio interconnect

You will need to make or buy a cable that is suitable to connect to a radio. Some ideas can be found at AllStar USB fob interface.

Conference Configuration

In autoload_configs/conference.conf.xml you must add to the "caller-controls" section: In order for DTMF to be acted upon by fs_rpt.pl we need to map all of the DTMF keys to the "event" option, so they are sent to Event Socket instead of being acted upon by mod_conference itself. To accomplish this we need to add a group to the <caller-controls> section of conference.conf.xml

conf/autoload_configs/conference.conf.xml

```
<group name=”radio”>
    <control action=”event” digits=”1/>
    <control action=”event” digits=”2/>
    <control action=”event” digits=”3/>
    <control action=”event” digits=”4/>
    <control action=”event” digits=”5/>
    <control action=”event” digits=”6/>
    <control action=”event” digits=”7/>
    <control action=”event” digits=”8/>
    <control action=”event” digits=”9/>
    <control action=”event” digits=”0/>
    <control action=”event” digits=”*”/>
    <control action=”event” digits=”#”/>
</group>
```

You must also make a profile for the conference. I am only listing the important sections since most of this is specific to your configuration and preferences.

The important thing is to set the caller-controls to the group you made in the previous step, in this example that is "radio".

conf/autoload_configs/conference.conf.xml

```
<profile name="radio">
    <param name"caller-controls" value="radio"/>
</profile>
```

Dialplan Configuration

dialplan/default/radio.xml

```
<extension name=”radio_conference”>
    <condition field=”destination_number” expression=”^1337$”/>
    <condition field=”source” expression=”mod_portaudio” break=”never”>
        <action application=”answer”/>
        <action application=”sleep” data=”1000/>
        <action application=”start_dtmf”/>
    </condition>
    <condition>
        <action application=”conference” data=”radio@radio”/>
    </condition>
</extension>
```

This will run start_dtmf on the portaudio port so that when the radio receives DTMF tones, they can be decoded and acted upon by fs_rpt.pl. It will put all callers into a conference where they can communicate.

Script Configuration

There are a few different variables that you need to configure in the script itself. These variables control some of the functionality of the script as it interfaces with the radio. When you edit the script you will see the following:

Configure the event socket parameters. If you set the timeout too low, it will constantly try to reconnect, if you set it too high it won't notice that it lost its connection. 30 seconds is a good value and it should only be changed if you are experiencing some problem.

ESL parameters

```
my $password = "ClueCon";    # event socket password
my $host     = "localhost";  # event socket host
my $port     = 8021;         # event socket port
my $timeout  = 30;           # seconds to expect a heartbeat or reconnect
```

Configure the serial port for PTT functionality.

Radio interface parameters

```
my $device   = undef;        # radio control device (/dev/ttyS0, COM1, etc)
my $baud     = 9600;         # radio control device baud rate
```

Configure a courtesy tone, this is a tone played just after the signal into the radio receiver stops, so that people listening to your radio transmission know that you are finished transmitting. If its value is 'undef' then the tone is disabled.

Beep parameters

```
my $courtesy_tone = "tone_stream://%(150,150,500);%(150,0,400)"; # tone played before releasing PTT
#my $courtesy_tone = "/sounds/beep.wav"; # play a sound file
#my $courtesy_tone = undef; # disable courtesy tone
```

Configure the conference information, use the extension you created in your dialplan and the conference name you created in the profile in the previous steps.

Conference parameters

```
my $confname = "radio";      # the name of the conference
my $extension = "1337";      # this is the extension that portaudio will call to join the radio conference
```

Configure your callsign if desired or required. This will use the morse.js script, which is required to be in the 'scripts' directory under your main FreeSWITCH install directory. If it is 'undef' then no callsign will be transmitted, which may violate some laws in some jurisdictions.

Ham callsign parameters

```
my $callsign = undef;     # callsign for morse autoID
my $callsign_interval = 600; # 10 minute intervals
```

Configure the path to Cepstral (or some other TTS engine). If you use another TTS engine you may need to edit the options that are passed to Cepstral. I do not use mod_cepstral since it is unlikely that this information is going to change frequently, so a sound file is generated once, and only updated if the file modification timestamp changes. You just make a file named "announcement.txt" and place the content in there that you want read. It will be read *after* your CWID.

TTS parameters

```
my $voice = "Allison";
my $swift = "/opt/swift/bin/swift";
my $filetime = 0;
```

Running

Start FreeSWITCH as normal, or reloadxml if it was already running to load the new settings. Launch fs_rpt.pl which will place a portaudio call into the conference.

Create an announcement.txt for any announcements that you want audibly spoken after each morse code identification, delete this file to disable announcements when they are no longer relevant.

Have callers go into the conference to talk to people on the radio. **It may be illegal to allow non-licensed operators into the conference with talk ability**. If you want to allow non-licensed people listen it may be better to use the recording capabilities of mod_conference to relay the audio to a Shoutcast or Icecast server so anyone can listen via that mechanism. At the very least they should remain muted for the entire duration they are in the conference which can be controlled with a flag to the conference.