Blog and Rants

A place to share and reference rants, thoughts, musing and emails to help our customers. Actual customer names, IP addresses, etc may be edited to protect the innocent and/or guilty.

Raspberry Pi Speaker/PA Phone

First, this little project is not as capable as the serious hardware you get from https://www.cyberdata.net , https://www.algosolutions.com/ or other vendors. But if you don't mind experimenting with a Raspberry Pi or similar hardware, this is a fun useful project that will create a VoIP/SIP endpoint that auto answers as a phone extension that allows you to feed into a PA system, loud speaker, etc. You should have some basic skills with an rPi, a text editor and linux. There is no security, implicit or implied, but if you practice sane PBX/SIP Server configs and put a real password on your rPi user and root accounts, you should be ready to go. Depending on your PBX, it might be possible that outside callers find this extension and say rude things loudly.

Build your device. Lots of options here, we are using basic Raspian aka Raspberry Pi OS without a GUI. Command Line Only. My reference build is an rPi 3, but this should work on any variations. rPi's have sound out, but no microphone in via the 3.5mm jack. Because of this, we need to install the **snd-dummy** driver via modprobe. Just plug something loud into the 3.5mm plug.

Step 1:

Install the needed things. I like Joe as an editor, nano and pico and vi work as well. mpg321 is mostly so you can test audio out without a call. We are using the console/command line version of Twinkle, a powerful softphone application. Expect and screen are optional. I use them because I do.

```
apt install twinkle-console joe mpg321 openssh-server expect screen
```

Step 2:

Run Twinkle and configure it how you would like. **alsa:plughw:1,0** is the snd-dummy driver microphone. Important parts of my reference configs:

/root/.twinkle/twinkle.sys

```
# AUDIO
dev_ringtone=alsa:plughw:0,0
dev_speaker=alsa:plughw:0,0
dev_mic=alsa:plughw:1,0
validate_audio_dev=no
```

/root/.twinkle/twinkle.svc

```
dnd=no
auto_answer=yes
```

/root/.twinkle/twinkle.cfg example values, not real. Use the IP address or FQDN of your SIP server

```
user_name=142
user_domain=192.168.1.22
user_display=142
user_organization=
auth_realm=
auth_name=142
auth_pass=42-put-realpasswdhere-42
auth_aka_op=00000000000000000000000000000000
auth_aka_amf=0000
# SIP SERVER
outbound_proxy=192.168.1.22
all_requests_to_proxy=no
non_resolvable_to_proxy=no
registrar=
register_at_startup=yes
registration_time=1200
reg_add_qvalue=no
reg_qvalue=1
```

Step 3:

Next, run **twinkle-console** and test. You can put the snd-dummy module in /etc/modules as well.

***/root/runtwinkle.sh**

```
/usr/sbin/modprobe snd-dummy
/usr/bin/twinkle-console
```

Step 4:

One tested, configure so this runs automatically at boot. Twinkle needs a console, and there are lots of ways to do this. This is one way and is easy to troubleshoot.

I like to add this to **/etc/rc.local** so it runs on bootup.

```
/usr/sbin/modprobe snd-dummy
/usr/bin/amixer set Headphone 90%
```

```
/usr/bin/mpg123 /root/iamalive.mp3
```

I'm not sure if twinkle-console will run forever or not, this is a hack method. I created a script and run it from crontab every 10 minutes.

/etc/crontab

- /10 * * * * root /root/keepitallrunning.sh

/root/keepitallrunning.sh

```
#!/usr/bin/bash
ps -axf | grep twinkle | grep -v grep
if [ "$?" == "0" ]; then
echo "twinkle found"
else
echo restart attempt
/root/star.expect &
sleep 2
fi
ps -axf | grep twinkle | grep -v grep
if [ "$?" == "0" ]; then
echo "TWINKLE IS ALIVE"
else
echo "TWINKLE NOT ALIVE"
fi
```

Expect runs star.expect (twinkle twinkle little star)

```
#!/usr/bin/expect
#not proud of this, but sometimes you gotta make things work the hard way.
set timeout 720
spawn /usr/bin/twinkle-console
expect "# " { send "help" }
interact
```

So, in the end, you have a little Raspberry Pi plugged into an amplifier of some kind, that connects to your PBX (Asterisk/FreeSwitch/3CX/Hello Hub) that on boot up plays an MP3 file (a speaker test) then connects to the PBX as a phone extension/endpoint with auto answer turned on. If you dial the extension you have configured, it will auto answer and you can holler whatever you want over the PA system. This project has worked for me, at least twice. I've given variations of them to grateful **ring-u** customers. What I wrote up here is a starting point. Your methods and results may vary. –Mike–

2022/06/10 17:13 · mike

Older entries >>