Using the Allstar “Autopatch” Function to Perform Tasks

The Allstar autopatch function is primarily associated with dialing an outbound phone line to make a landline call. This was a very desirable function before the proliferation of cell phones. Today few repeaters offer this capability because the need has been largely eliminated. The autopatch function remains in Allstar and if configured with an Asterisk dialplan and voip provider it is still fully functional. In this article I will briefly describe using the autopatch function for outgoing calls but more importantly as a fully functional call passing the DTMF digits to any program or script

<u>Here is an example of an outgoing phone patch in Allstar.</u>

In the rpt.conf file [functions] stanza the default line for autopatch is -

```
61=autopatchup,context=pbx_server,noct=1,farenddisconnect=1,dialtime=20000,quiet=1 ; Autopatch up
0=autopatchdn    ; Autopatch down
```

So a user would dial *61<number> to dial a specific number and *0 to hangup the line.

This autopatch line says go to the pbx_server context in /etc/asterisk/extensions.conf. This name can be any legal Asterisk context name. The dialplan context name and the autopatch “context=” names must match.

It is defined in extensions.conf in what is called a dial plan context. In this example shown below any passed number which has 10 digits where the first digit must be a 1, the second and fifth digit must be 2-9 and all other digits 0-9 would be legal. If the passed number satisfies these parameters it is passed to either a sip or iax stanza within /etc/asterisk/sip.conf or iax.conf which actually completes the call to the provider.

```
[pbx_server]
exten => _1NXXNXXXXXX,1,Dial(IAX2/<iax-stanza>/${EXTEN})
```

OR

```
exten => _1NXXNXXXXXX,1,Dial(SIP/<sip-stanza>/${EXTEN})
```

The Asterisk dialplan can be used to format or limit the called number. Here an example shows dialing a four digit number that always has the same three digit prefix (456) and a fixed area code like an extension in a large company.

```
exten => _xxxx,1,Dial(SIP/<sip-stanza>/202456${EXTEN})
```

so the number called would be 202-456-xxxx   where xxxx is user supplied - *62xxxx.

The number could be fixed so a specific autopatch function always called a single number.

```
exten => _x,1,Dial(SIP/<sip-stanza>/12024561212)
```

would dial only 1-202-456-1212   when *62x was entered. x = any digit.

Using the autopatch feature for extended DTMF capability

There are many times in Allstar when it would be useful to pass the entered DTMF digits directly to a script or program. This could be done in a very convoluted way by defining each DTMF sequence and its intended operation in the function section in rpt.conf but there is a much better way to do it using the autopatch function and an asterisk dialplan. This also gives you an almost unlimited number of function code possibilities.

The autopatch function in rpt.conf could be defined to any number of actual function codes. You would leave the current *61 code intact and create another unused code to call another dialplan context in extensions.conf. Here is an example.

The new /etc/asterisk/rpt.conf call -

```
62=autopatchup,context=command_process,noct=1,farenddisconnect=1,dialtime=7000,quiet=1 ; Dial Plan Processing
```

In this case I have arbitrarily selected 62 (*62) as my prefix. It could be any available unambiguous code. The line reads the same as the above autopatch (61) line except for the context it calls which is now called "command_process" and the dialtime which is 7000 (7 seconds) instead of 20000 (20 seconds). The context can be any legal Asterisk name which would match the name in extensions.conf. The dialtime is set shorter because the total digits in this example are 7 - *62xxxx. The number of digits is defined in the dialplan as shown below.

```
 [command_process]
exten => _xxxx,1,System(/etc/asterisk/local/test_script ${EXTEN})
exten => _xxxx,n,Hangup()
```

In this case the xxxx requires 4 digits each (0-9). These digits are passed to the script located at /etc/asterisk/local/test_script in the EXTEN variable. Pay attention to syntax, it is important in Asterisk.

Here is a quick table of the dialplan codes -

X matches any digit from 0-9
Z matches any digit from 1-9
N matches any digit from 2-9
[1237-9] matches any digit or letter in the brackets
(in this example, 1,2,3,7,8,9)
[a-z] matches any lower case letter
[A-Z] matches any UPPER case letter
. wildcard, matches one or more characters
! wildcard, matches zero or more characters immediately

So a dialplan of -

_12XX   would require the first two digits to be 12 followed by any 2 digits 0-9.

_NXX5 would allow any first digit 2-9, any two digits 0-9, and the digit 5
In general if you were using this method to call a script you would determine the maximum number of digits

you require or desire. One digit would give you 10 possibilities, two would give 100, three 1000, etc. Using the fewest digits would required less time to enter but it would also be a little less secure.

Here is the script that is called by the [command_process] dialplan context above. This script shows some simple command examples that can be executed using both calls to other programs and calls back to Asterisk.

```
#!/bin/bash
#
#  /etc/asterisk/test_script
#
# Test script to show passing of parameters from the Asterisk
# dialplan. Based on the dialplan the digit length could be
# 1 to as many digits as needed. Digits are decoded here to
# perform tasks. This example show repeater power control
# simulated by a GPIO statement. It also says system uptime
# in hours. This script can be executed standalone for testing.
# NODE1 is the defined first node on the server. It could be
# any local node. Audio is localplay.
#
# These are simple examples but any (reasonable) number of
# digits can be passed and and valid Linux command can be
# executed.
#
# D. Crompton 1/2016

 . /usr/local/etc/allstar.env

sleep 2

# Say the passed digits
/usr/local/sbin/speaktext.sh $1 $NODE1

sleep 2

# *624444 would set the repeater to low power

if [ "$1" = "4444" ]
 then
    # gpio write x 0
    # simulated set bit x to 0 - see gpio howto
    /usr/bin/asterisk -rx "rpt localplay $NODE1 /var/lib/asterisk/sounds/repeater"
    /usr/bin/asterisk -rx "rpt localplay $NODE1 /var/lib/asterisk/sounds/rpt/lopwr"

# *625555 would set the repeater to high power

elif [ "$1" = "5555" ]
 then
    # gpio write x 1
    # Simulated set bit x to 1 - see gpio howto
    /usr/bin/asterisk -rx "rpt localplay $NODE1 /var/lib/asterisk/sounds/repeater"
    /usr/bin/asterisk -rx "rpt localplay $NODE1 /var/lib/asterisk/sounds/rpt/hipwr"
fi
sleep 2

# Say the server uptime in hours. Note delays are necesssary as localplay
# does not block

UP=`echo $(awk '{print $1}' /proc/uptime) / 3600 | bc`
/usr/bin/asterisk -rx "rpt localplay $NODE1 /var/lib/asterisk/sounds/system"
sleep 1
/usr/bin/asterisk -rx "rpt localplay $NODE1 /var/lib/asterisk/sounds/uptime"
sleep 1
/usr/local/sbin/speaktext.sh $UP $NODE1
sleep 3
/usr/bin/asterisk -rx "rpt localplay $NODE1 /var/lib/asterisk/sounds/hours"

exit 0

#end script
```

You can use your imagination with the called program. It could be a binary program, Perl, python, bash, etc. script. Anything that executes in Linux. The program or script must be executable.

The script above shows how you can use the gpio command to set bits. You could also use the gpio read command to read the status of bits and report them back or perform an operation based on the status.

You also have the option of using just the dialplan or the dialplan and external program combinations. Dialplans can manipulate text and perform many other functions but it probably makes more sense to do this in a script in most cases.

Multiple dialplan contexts in extensions.conf could be defined along with multiple autopatch functions in rpt.conf but a better idea would be to define one script that read the digits and called other scripts to process the codes. So you could define a three digit code which would be disseminated to ten different scripts by the master script. Each one could be a different sub script with 100 functions each. So 0xx, 1xx, 2xx, 3xx, etc. could control different things from different scripts. Adding more digits gives you more possibilities. Even three digits is probably way more then most would ever need but the capability is there if you need it.

One caution is to not use the dialplan SayDigits, SayNumber commands to send voice out as these would send a global voice to ALL connected nodes. Unless this is what you intend to do use a direct call to Asterisk or a script to a direct call to Asterisk to send via rpt localplay as shown in the above script example. Also as noted the Asterisk calls to do this are not blocked so you need to provide your own timing as shown in the sleep commands above. Without them things can get out of order or not play at all.

Asterisk has quite a few commands available to use in the dialplan. I refer you to the many Internet references and books available on the subject.

As usual have fun!   73, Doug, WA3DSP