ACID System Administrators Guide

Version 0.7 12/27/2008

Introduction

The purpose of this guide is to help new administrators of app_rpt/Asterisk previously installed with the Allstar Centos Installation Disk (ACID) ISO downloaded from http://dl.allstarlink.org. It assumes that you know how to use the command prompt on a Linux system and that you are familiar with one of two Linux text editors; vi or nano, so that you can view and modify the configuration files on the system.

This guide will not teach you Linux or Asterisk. There are several resources listed in Appendix A which will do a good job of that. The goal of this guide is to teach you just enough to do basic system administration tasks related to maintaining and operating an Allstar Link node.

Chapter 1: Basic Concepts

This section attempts to instruct the new system administrator about the way things are structured and where key information is located.

SSH Access

SSH Access is enabled in ACID by default once the entire installation is complete. This facilitates remote administration of the ACID installation. SSH is instructed to listen on port 222. This is not the standard SSH port and it was changed to help improve system security. If you would like to change the SSH access port to something completely different, edit the file /etc/ssh/sshd_config and change the port number specified by the Port keyword to a new value.

Helper Scripts

There are several helper scripts located in /root/acid. The table below explains the purpose of each script:

Script Name	**Purpose**
astdn.sh	Shuts down Asterisk
astres.sh	Restarts Asterisk. Use to fore a reload of all configuration files.
astupd.sh	Fetch, compile and install the latest version of Asterisk
astup.sh	Starts up Asterisk
backup.sh	Backs up Asterisk config files, node name sound files, and the IRLP configuration (if installed).
irlpsetup.sh	Installs a new IRLP installation or re-installs a backed up IRLP configuration.
nodesetup.sh	Allows the Allstar node number, node number password, and identification message to be set on a single node system.
restore.sh	Restores the Asterisk config files, node name sound files, and puts the IRLP backup file irlp_backup.tgz in the /tmp directory so it can be used by irlpsetup.sh

Updating Asterisk

As we fix bugs and add additional features, new source code is posted to http://dl.allstarlink.org/installcd/files.tar.gz from time to time. A script with will check to see if your Asterisk source code needs updating, and then automatically fetch the new version, compile, and install Asterisk is now part of the ACID distribution. The script will only download and compile Asterisk if the version you have does not match what is located at the above web site. To run the script, change

directories to /root/acid and type the following:

```
./astupd.sh
```

If the script prints "Asterisk at latest version. No update required.", then no further action is required on your part. If the script downloads and compiles Asterisk, you should reboot your system once the compilation and installation completes. You can reboot your system by typing `reboot` at the root shell prompt.

Adjusting the Time Zone

By default ACID is configured for the US Pacific time zone. The file /etc/localtime controls which time zone the system thinks it is in. All of the time zone files the rest of the world are located in /usr/share/zoneinfo. To change the time zone, find the time zone file you want to use, and make a soft link to it like this:

```
cd /etc
cp localtime localtime.orig
ln -sf /usr/share/zoneinfo/Australia/Victoria localtime
```

You can check to see that the time zone is now correct by using the `date` command from the shell.

Structure of Config Files

All config files in an ACID installation are ASCII text files which can be modified with a text editor. The files are structured as key=value pairs in groups with a stanza heading like this:

```
[stanza1]
key1=value1
key2=value2
.
.
[stanza2]
key1=value1
key2=value2
...
```

Comments can be inserted to the left of stanzas and key=value entries, but depending upon the name of the configuration file, the comment delimiter character could either be a semicolon ';' or a an octothorpe '#' depending upon the name of the configuration file. *iax.conf*, *extensions.conf, rpt.conf*, and *usbradio.conf* all use semicolons only as comment delimiters. *zaptel.conf* uses an octothorpe as comment delimiter.

```
[stanza1] ; A comment
```

```
key=value ; Another comment
```

Important Config Files

These 4 configuration files are used to set up and customize one or more radio nodes

Configuration File	Function
/etc/asterisk/extensions.conf	Asterisk Dialplan (call routing)
/etc/asterisk/iax.conf	IAX (inter-asterisk exchange) Configuration
/etc/asterisk/rpt.conf	Repeater/Node Configuration
/etc/asterisk/usbradio.conf	USB fob Configuration
/etc/zaptel.conf	Zaptel Device Configuration

extensions.conf

extensions.conf is mainly used to route incoming connections from remote nodes to the correct node number in app_rpt. It also is used to route outgoing autopatch connections to various channels and VOIP termination providers, but that is a subject which will be covered in a future version of this document.

Here is an example of how *extensions.conf* is used to handle incoming connections for a single node:

```
[radio-secure]
exten => 1234,1,rpt,1234
```

In the above case, two nodes are defined as asterisk extensions in a “context” called radio-secure defined by a stanza [radio-secure] . An incoming connection directed to extension 1234 will end up calling app_rpt (rpt) with a value of 1234 which should a node number defined in *rpt.conf*.

iax.conf

iax.conf controls defines how to register a node with the Allstar link node assignment authority, and assigns a context incoming connections receive so that statements in *extensions.conf* can direct the connection to the correct local node. *iax.conf* contains two sections relevant to app_rpt. The [general] section, and the [radio] section.

[general]

The [general] section sets up the global configuration for *iax.conf*, and also is used to hold the register statements for each node defined on the system. A register statement looks like this:


```
register=A1999:1234567@register.allstarlink.org
```

A separate register statement is required for every node number hosted on the computer. In other words, if you have 2 nodes hosted on the computer, then 2 register statements will be required.

The format for the register statement is:

```
register=Anode:password@register.allstarlink.org
```

Where:

node is your assigned Allstar link node number
password is your Allstar link password for the above node number

In addition to register statements, the permitted codecs for outgoing connections are also defined in the [general] stanza of *iax,conf*. The way codec definitions work is we first disallow all of them with a disallow=all statement, then we define the codecs we wish to permit with allow statements. The suggested outgoing codecs are defined in this example:

```
disallow=all
allow=gsm
allow=g726aal2
allow=ulaw
```

The statements above only allow GSM, ADPCM, and ULAW codecs to be used for outgoing connections. This is the suggested contents of the [general] stanza to be included in *iax.conf*

```
[general]
bindaddr=0.0.0.0
disallow=all
allow=gsm
allow=g726aal2
allow=ulaw
jitterbuffer=yes
forcejitterbuffer=yes
dropcount=2
maxjitterbuffer=4000
maxjitterinterps=10
resyncthreshold=1000
maxexcessbuffer=80
minexcessbuffer=10
jittershrinkrate=1
tos=0x1E
autokill=yes
delayreject=yes
iaxthreadcount=30
iaxmaxthreadcount=150
```

[radio]

The [radio] stanza controls the types of codecs which can be used, how they are selected (or negotiated) and the context to call in *extensions.conf* when an incoming connection occurs from a remote node. The suggested [radio] section for *iax.conf* should look like this:

```
; Incoming radio connections

[radio]
type=user
disallow=all
allow=g726aal2
allow=gsm
codecpriority=host
context=radio-secure
transfer=no
```

Let's now pick the above configuration apart. The configuration allows incoming connections only (type=user), and only ADPCM and GSM codecs (disallow=all, allow=g726aal2, and allow=gsm). The order of selection (negotiation) is determined by this machine (codecpriority=host), and is in the order specified by the allow statements (ADPCM first, GSM second). The context to call in extensions.conf is radio-secure (context=radio-secure). And call transfers to external systems are disallowed (transfer=no)

Adding support for another codec is fairly straightforward. For example, if we want to allow the use of the ulaw codec on incoming connections all that would be required is to add an allow=ulaw statement to the [radio] stanza so it looks like this:

```
[radio]
type=user
disallow=all
allow=ulaw ; added
allow=g726aal2
allow=gsm
codecpriority=host
context=radio-secure
transfer=no
```

Now the system will try to negotiate a ulaw connection first, ADPCM second and GSM last.

A final note about codec selection. There are quite a few codecs in Asterisk to choose from, but ulaw, alaw, GSM and ADPCM should only be used, the rest of the standard Asterisk codecs (speex, ilbc, lpc10, etc) should be avoided. The ulaw and alaw codecs have the best audio quality, followed by ADPCM, and lastly GSM, Bandwidth used is in the reverse order to audio quality. GSM uses the least bandwidth, and alaw/ulaw the most. Here is a quality versus bandwidth trade off table:

CODEC	AUDIO QUALITY	BANDWIDTH (including IP and Ethernet headers)
ADPCM	good	55 kbps
GSM	mediocre	36 kbps
ulaw/alaw	best	87 kilobits per second (kbps)

rpt.conf

rpt.conf holds configuration information for app_rpt, the Asterisk repeater application. It is a complex configuration file, with a large number of options. We will not define any configuration options here unless they require further clarification from what is documented in rpt.conf.sample. It will be helpful to have a copy of rpt.conf.sample in front of you while reading this section.

rpt.conf structure

r*pt.conf* relies heavily on the use of stanzas to glue related bits of information together. Stanzas are also referenced by other stanzas by using key=value pairs to reference the other stanza within a given stanza. The node stanza makes use of a lot of references to other stanzas within *rpt.conf*.

There may be several stanzas of the same type in rpt.conf. For example, a system with two nodes defined will have two node stanzas.

Stanza Type	Example in rpt.conf.sample	Description
Control State	[control-states]	Defines groups of control operator commands to be executed all at once. Can be combined with macros to allow changes to a nodes operating mode to be made quickly.
Function	[functions-repeater]	Defines DTMF function digit sequences. A function stanza has a reference to it defined inside a node stanza. Multiple function stanzas may be defined and used to provide different function lists for different sources (radio.phone, and link)
Macro	[macro]	Defines DTMF macro sequences
Memory	[memory]	Holds channel frequencies for remote base nodes
Morse	[morse]	Contains definitions for morse code messages. A morse stanza has a reference defined to it inside a node stanza
Node	[000]	Defines configuration options relevant to a specific node number
Remote Nodes	[nodes]	Hard coded node addresses and node addresses not part of the Allstar link. Local addresses of Allstar Link nodes to support local connections.
Scheduler	[scheduler]	Defines macros to be executed at a specific time and date. Uses a cron-like syntax

Stanza Type	Example in rpt.conf.sample	Description
Telemetry	[telemetry]	Contains definitions for telemetry tones (courtesy tones, roger beeps, etc). A telemetry stanza has a reference to it defined inside a node stanza.
Tx Limits	[tx-limits]	Limits where the remote base transmitter can transmit depending upon login privileges. Used with remote base nodes only.
Wait Times	[wait-times]	Contains time delays used to time audio telemetry events

Important Configuration Options for Node Stanzas in rpt.conf

This section covers the important configuration keys used in an *rpt.conf* node stanza. Not all configuration options are documented here as the sample configuration file does a good job of doing that. Only those options which require further clarification, and those options which are significant are documented here.

duplex=

This configuration option tells app_rpt how to handle audio passed through the radio interface. Please note that it does not perform the same function as the duplex= configuration option in the configuration file *usbradio.conf*. In *rpt.conf*, duplex= can be set to one of 5 different modes:

Mode	Description
0	Half duplex with no telemetry tones or hang time. Special Case: Full duplex if linktolink is set to yes. This mode is preferred when interfacing with an external multiport repeater controller.
1	Half duplex with telemetry tones and hang time. Does not repeat audio. This mode is preferred when interfacing a simplex node.
2	Full Duplex with telemetry tones and hang time. This mode is preferred when interfacing a repeater.
3	Full Duplex with telemetry tones and hang time, but no repeated audio.
4	Full Duplex with telemetry tones and hang time. Repeated audio only when the autopatch is down.

functions=

This key points to the name of a function stanza which describes what to do when DTMF commands are received. Please refer to the section on function stanzas below for more information. An example usage is as follows:

```
functions=functions
```

hangtime=

This controls the length of the repeater hang time. It is specified in milliseconds. An example usage is as follows:

```
hangtime=1000
```

idrecording=

The setup.sh script installs a morse code identifier using a callsign you enter during the setup process. This identifier string is stored in the node stanza using the idrecording key. It can be changed to a different call sign by changing the value to something different. The value can be either a morse code identification string prefixed with |*i*, or the name of a sound file containing a voice identification message. When using a sound file, the default path for the sound file is /var/lib/asterisk/sounds. Example usages are as follows:

```
idrecording=|iwa6zft/r ; Morse Code ID
idrecording=/var/lib/asterisk/sounds/myid ; Voice ID
```

Note: ID recording files must have extension gsm,ulaw,pcm, or wav. The extension is left off when it is defined as the example shows above. File extensions are used by Asterisk to determine how to decode the file. All ID recording files should be sampled at 8KHz.

linktolink=

When operating in duplex mode 0, this forces the radio interface to operate in full duplex mode, but keeps all the other duplex mode 0 semantics. This is used when a radio interface is connected to a multiport analog repeater controller. The linktolink= key can take two values: yes or no.

rxchannel=

This defines the name of the radio interface to use. Radio interface names are defined in *usbradio.conf*. A node stanza requires that one (and only one) rxchannel be defined. Channels are defined as *tech/id* where *tech* is the type of channel and *id* is the the radio interface name or number. Examples:

```
rxchannel=radio/usb
rxchannel=radio/usb1
rxchannel=zap/1
```

totime=

This defines the time out timer interval. The value is in milliseconds. An example usage would be:

```
totime=180000
```

Function Stanzas

Function stanzas control access to DTMF commands that a user can issue from various control points in app_rpt. There can be separate function stanzas defined for radio interfaces, and dial-in access. A function stanza key-value pair has the following format:

dtmfcommand=commandclass,parameters

Where:

dtmfcommand is a DTMF digit sequence minus the start character (usually *)
commandclass is a string which defines what class of command.
parameters are one or more comma separated parameters which further define a command.

Command Classes

Class	Description
cop	Control operator commands
ilink	Internet linking commands
status	User Status Commands
autopatchup	Autopatch up commands
autopatchdn	Autopatch down commands
remote	Remote base commands
macro	Command Macros

Most of the above command classes require one or more additional command parameters.
Below is an example of a function stanza:

```
[functions]
1=ilink,1           ; Specific link disconnect
2=ilink,2           ; Specific Link connect -  monitor only
3=ilink,3           ; Specific Link connect - transceive
4=ilink,4           ; Enter command mode on a specific link
7=ilink,5           ; Local Link status
```

A complete definition of all DTMF commands can be found in rpt.conf.sample.

usbradio.conf

The *usbradio.conf* file holds the configuration settings for one or more USB radio interfaces. There are two types of stanzas in this file. A general stanza, and one or more radio interface stanzas. Typically, the general stanza is defined with no key=value pairs in it, and the radio interface stanzas contain the individual settings for each radio interface.

Structure of *usbradio.conf*

```
[general]
[usb] ; Name of first radio interface.
.
.
.
[usb1] ; Name of second radio interface.
.
.
.
```

Contents of the Radio Interface Stanza

The table below describes all of the valid keys in a radio interface stanza:

Key	Value
hwtype=	Defines the type of hardware interface. A 0 specified a modified USB fob or a URI a 1 specifies a Dingotel interface.
rxboost=	Defines whether 20dB of gain is added to the receiver audio input or not. A value of 0 indicates no gain is to be added. A value of 1 indicates that 20dB of gain is to be added.
rxctcssrelax=	This should always be set to 1. Reduces talkoff from radios without a transmit high pass audio filter. Almost all amateur-grade FM radios do not have a high pass filter to attenuate the CTCSS frequencies in the TX audio path.
txctcssdefault=	This is the default TX CTCSS tone when no signal is present on the input. This tone is transmitted during the hang time, telemetry, ID messages etc.
rxctcssfreqs=	List of CTCSS tones for receive with one decimal point of precision. Must be a valid EIA tone. Specify one or more tone frequencies separated by commas. When CTCSS decode enabled, If a match to a tone in this list is detected, the squelch will be opened. If operating in full duplex mode, and the CTCSS tone is also in the transmit frequency list txctcssfreqs, then that tone will be transmitted whenever there is a matching tone on the received signal. If operating in full duplex mode, and only 1 tone is specified, then it should match the txctcssdefault setting.
txctcssfreqs=	List of CTCSS tones for transmit. Specify one or more tone frequencies separated by commas. These tones will be transmitted whenever there is a match with a received tone in the rxctcssfreqs list. If operating in full duplex. and only one tone is specified, it should match that specified by txctcssdefault.
carrierfrom=	Carrier detection source. Options (*no, dsp, usb, usbinvert*). *no* is for no carrier detection, *dsp* is for carrier derived from discriminator noise, *usb* is for carrier derived from the USB fob COR input, and *usbinvert* is for an inversion of the USB fob COR input.
ctcssfrom=	CTCSS detection source. Options (*no, dsp*). *no* disables the CTCSS requirement and relies on carrier squelch alone. *dsp* uses the software CTCSS decoder to decode CTCSS tones.
rxdemod=	Selects the type of receive audio the software is to expect. Options: (*no, flat, speaker*) *no* turns off the receive audio completely, *flat* selects discriminator audio (before de-emphasis), and *speaker* selects audio after de-emphasis.

Key	Value
txprelim=	TX audio processing. Options: (*no,yes*). Choosing yes turns on transmitter pre-emphasis and peak limiting, along with a lowpass filter. The *yes* option is used when driving an FM transmitter directly by injecting audio right into the modulator. The *no* option provides audio which should be injected into the microphone input of the transmitter or at some point in the transmitter' speech amplifier before pre-emphasis, and limiting.
txtoctype=	Transmit CTCSS handling. Options: (*no,phase,notone*). The *no* option provides no special CTCSS tone handling; the transmit CTCSS tone will remain on for as long as the transmitter is keyed. The *phase* option is "reverse burst" and it inverts the phase of the tone generator just before the transmitter unkeys. A small amount of hang time is added to allow the reverse phase CTCSS tone to be sent. The *notone* option is "chicken burst" and it adds a small amount of hang time to the transmitter during which no CTCSS tone is transmitted.
txmixa= txmixb=	TX mixer A/B output control. There are two outputs from the URI or modified USB fobs as the CM108 is a stereo chip. txmixa is output mixer the left channel and txmixb is the output mixer for the right channel. These mixers can be independently configured with different output options. The options are: (no,voice,tone,composite,auxvoice). The *no* option disables the specific mixer output. The *voice* option configures the output to be voice-only without any CTCSS tone summed into it. The *tone* option configures the output to be CTCSS tone only. The *composite* option configures the output to be a combination of voice and CTCSS tone. The *auxvoice* option configures the output to be a headphone level voice-only output.
invertptt=	TX PTT polarity. Options: (0,1). The 0 option configures the PTT output to be a low when there is PTT and open when there is no PTT. The 1 option forces the output to be low when there is no PTT and open when there is PTT.
duplex=	Duplex processing. Options (1,0). A value of 1 enables additional DSP processing to support full duplex audio. A value of 0 turns off the additional DSP processing, and allows the administrator to conserve CPU bandwidth in applications where only half-duplex audio is required. Note that this duplex setting does not do the same thing as the one found in the config file *rpt.conf*. If you are directly interfacing to a repeater, or full duplex controller port, this option should be set to 1.

Starting and Stopping Asterisk

Stopping Asterisk:

```
/root/acid/astdn.sh
```

Starting Asterisk:

```
/root/acid/astup.sh
```

Restarting Asterisk:

```
/root/acid/astres.sh
```

Asterisk is automatically started when the system completes the booting process.

The Asterisk CLI

Access

From a root shell type:

```
asterisk -r
```

You should get the command prompt:

```
test*CLI>
```

CLI Commands

There are a number of useful Asterisk CLI commands listed in the table below.

Asterisk CLI Command	**Function**
`core set verbose` *`level`*	Sets the verbosity level for console messages in Asterisk. The *level* parameter is a number from 0 to 7 indicating the level of detail to report. Level 0 turns off all all verbosity.
`dialplan reload`	Forces a reload of *extensions.conf*.
`exit`	Forces an exit from asterisk and returns control to a system shell prompt
`iax2 reload`	Forces a `r`eload of *iax.conf*.
`radio active interface`	Selects the active USB radio interface. For example if you have two USB fobs named usb and usb1 in usbradio.conf, you can select which one is active for commands by typing `radio active usb` or `radio active usb1`
`radio key`	Keys the radio (asserts PTT) attached to the active USB radio interface.
`radio unkey`	Unkeys the radio (de-asserts PTT) attached to the active USB radio interface.
`reload`	Forces a reload of all configuration files
`rpt fun` *`nodenum function`*	Executes a DTMF function on a specified node as if it was done from the radio. This command takes two arguments *nodenum* and *function*. The *nodenum* parameter is the node you want to send the command to, and function parameter is the actual DTMF function to perform.
`rpt lstats` *`nodenum`*	Returns linking statistics for a specified node number *nodenum*
`rpt reload`	Forces a reload of *rpt.conf*
`rpt set debug level` *`level`*	Sets the debug level for app_rpt debug messages. The *level* parameter is a number from 0 to 7 indicating the level of detail to report. Level 0 turns off all debugging messages.
`rpt stats` *`nodenum`*	Returns statistics for specified *nodenum*

Locations of Important Files

You should know whee to find important files in the system. This section of the guide documents where these important files are located.

Type of File	**Location**
Asterisk Configuration Files	/etc/asterisk
Zaptel Configuration File	/etc/zaptel.conf
Reference Copies of Configuration Files	/usr/src/configs
Example Configuration Files	/usr/src/configs/examples
System Scripts	/etc/rc.d/rc.updatenodelist
Master Node List	/var/lib/asterisk/rpt_extnodes
Asterisk Sound Files	/var/lib/asterisk/sounds, /var/lib/asterisk/sounds/rpt, /var/lib/asterisk/sounds/rpt/nodenames
Asterisk Source Code	/usr/src/asterisk

Chapter 2: HOWTO's

This chapter contains popular howto's and is built around the concepts discussed in Chapter 1.

Adjusting Audio Levels

This howto describes how to properly set audio levels on an ACID system interfaced to a radio.

Required Equipment

Communications service monitor, or deviation meter and FM signal generator.

Preliminary Checks

Modify the config files for the type of radios you have (repeater or half-duplex) and the type of carrier detection you want to do.

Make sure your radio is connected to the USB interface and turned on before proceeding.

Procedure

From the Asterisk CLI, (to get into the Asterisk CLI, type `asterisk -r` from a root shell) set the receiver noise baseline (with no signal on the receiver frequency) as follows:

```
radio tune rxnoise
```

Set the receive audio input level by suppling an FM signal at 1KHz with 3KHz of deviation (with no CTCSS) on the receiver's frequency. then once the signal is present, type:

```
radio tune rxvoice
```

Set the CTCSS tone level by supplying a CTCSS tone at 600Hz deviation your desired frequency (with no other modulation), and type:

```
radio tune rxtone
```

Save the receiver levels by typing:

```
radio tune save
```

Set the CTCSS tone level to zero by typing:

```
radio tune txtone 0
```

To set the transmit audio level, monitor the transmitter with a deviation meter and start with the setting of 500 (which is the midpoint) by typing:

```
radio tune txvoice 500
```

This will cause the transmitter to be keyed, and a brief tone be sent so that you can measure the transmit audio level with your deviation meter or service monitor. The last parameter is an audio level setting with a range of 000 to 999. Set the number to that which gives you 3KHz of deviation. Repeat the `radio tune txvoice` command using different numbers (successive approximation) until you get the desired 3 KHz of deviation.

Set the TX CTCSS tone level by typing the following:

```
radio tune txtone 500
```

This will cause the transmitter to be keyed, and a brief CTCSS tone be sent so that you can measure the TX CTCSS level with your deviation meter or service monitor. The last parameter is a TXCTCSS level setting with a range of 000 to 999. Set the number to that which gives you 0.6 KHz of deviation. Repeat the radio tune command using different numbers until you get the desired 0.6 KHz of deviation.

Save the settings by typing

```
radio tune save
```

You may now test the setup to see if the audio levels are acceptable.

Enabling Node Status Reporting to stats.allstarlink.org

To enable statistics reporting to stats.allstarlink.org, uncomment or add the following lines to your node stanza(s) in rpt.conf:

```
statpost_program=/usr/bin/wget,-q,--output-document=/dev/null
statpost_url=http://stats.allstarlink.org/uhandler.php
```

Not all nodes will want to send statistics information to stats.allstarlink.org. The choice is up to the system owner. This feature is disabled by default.

Configuring a Two Node System

Changing a system from one node to two nodes requires that 4 text files be modified: *extensions.conf* *iax.conf*, *rpt.conf*, and *usbradio.conf*.

USB Device Enumeration

Because of the way USB bus device identification works, new USB fobs must be added one at a time. In other words, you must have a working single node system before you can make it a two node system. This is because the USB channel driver remembers where all the previously defined nodes are located, and when it finds new hardware, it will find the next unassigned stanza in *usbradio.conf* and assign the new hardware to that node. It is very important the order that the USB fobs connected to the computer are not disturbed or plugged in to different USB sockets, as this will confuse the USB channel driver and require that *usbradio.conf* have all the interfaces removed or commented out and added back in one at a time.

USB Hubs

The use of USB hubs should be avoided if at all possible as we have found that some hubs corrupt the audio streams coming form and going to the USB fobs. If you must use a USB hub, use only multi-TT types, but be forewarned that you may have to test several models until you find one which works acceptably, if at all.

The procedure to configure a Multi-node system consisting of fictional nodes 1234 and 5678 is as follows:

1. Make backup copies of rpt.conf, extensions.conf, iax.conf, and usbradio.conf.
2. Edit the usbradio.conf file. Make a copy of the [usb] stanza in usbradio.conf.
3. Paste the copy just below the original stanza.
4. Rename the stanza name of the copy to [usb1]
5. Change the configuration settings of [usb1] to suit the new radio.
6. Save the new usbradio.conf file.
7. Edit the rpt.conf file. Make a copy of your Node stanza in rpt.conf
8. Paste the copy just below the original node stanza 1234

9. Change the node stanza of the copy to the new node number 5678. and change rxchannel to point to the name of the device in usbradio.conf:

```
[1234]
rxchannel=usb
.
.
.
[5678]
rxchannel=usb1
.
.
.
```

10. Change any configuration settings in the new node stanza to suit the new radio
11. Add a second local entry for node 5678 to the nodes stanza to allow local connections to be made:

```
[nodes]
1234 = radio@127.0.0.1/1234,NONE
5678 = radio@127.0.0.1/5678,NONE
```

12. Save the new rpt.conf file
13. Edit the iax.conf file
14. Add an additional register statement for the second node, 5678 just below the first statement.

```
register=A1234:12345678@register.allstarlink.org ; First Node
register=A5678:12345678@register.allstarlink.org ; Second Node
```

15. Save the new iax.conf file.
16. Edit the extensions.conf file
17. Add the 5678 extension in the radio-secure context to call the rpt application using the new node number:

```
[radio-secure]
exten => 1234,1,rpt,1234
exten => 5678,1,rpt.5678
```

18. Save extensions.conf and exit the text editor.
19. Stop and restart asterisk
20. Adjust the signal levels on the new interface using the usbradio audio setup procedure documented in the previous HOWTO. Issue the command `radio active usb1` from the Asterisk CLI to switch to the new USB interface (usb1) before starting the audio adjustment process.

Example configuration files for a two node system are located in /usr/src/configs/examples/twonodes

Setting up a Remote Base

Remote base nodes are configured differently than a standard node. A completely different set of internal functions in app_rpt is used when operating a node as a remote base. Usually, the only reason to set up a node as a remote base is when you wish to change the operating parameters of the the attached radio remotely, or if you only want the radio to be used by a single user at a time. The operating parameters would be frequency, emission mode, power level, and receive and transmit CTCSS tones.
Remote bases which are configured to allow a remote user to command a compatible radio are known as frequency/mode agile remote bases.

Behavior of Standard Nodes Versus Remote Base Nodes

Behavior	**Standard Node**	**Remote Base Node**
Command Decoding	Remote or Local. DTMF can be optionally decoded on the receive audio input.	Remote only. No DTMF will be decoded on the receive audio input.
Duplexing	Configurable: duplex or half-duplex.	Half-duplex only
Frequency and Mode Agility	Fixed frequency operation, and channelized operation only using arguments passed in to app_rpt from *extensions.conf*	Frequency and Mode agile. Support for several radio types using asynchronous serial, CAT, and synchronous serial.
Multiple connections	Multiple nodes can connect. Operates as a conference bridge.	Only one node can connect at a time.
User Authorization	No	Optional

Security Issues

Unfettered access to remote bases can be a security issue. If the remote base has no login protection, it could be used by unscrupulous individuals to violate amateur radio rules and regulations. We strongly advise that all remote bases be protected by requiring a login code.

Requesting a remote base node from http://allstarlink.org

Remote base nodes must be specially requested from http://allstarlink.org. When requesting a remote

base node, check the box Rem. Base, and if the remote base is frequency agile, check the box Freq. Agile.

Specifying a Standard Node and a Remote Base Node in rpt.conf

In the nodes stanza, a remote base node is denoted when it is suffixed with a y option:

```
[nodes]
1234 = radio@127.0.0.1/1234,NONE,y ; Remote base node
5678 = radio@127.0.0.1/5678,NONE ; Standard node
```

In the above example node 1234 is a remote base node, and node 5678 is a standard node.

Compatible Radios and Interfaces Which Support Frequency/Mode Agility

Brand	Model	Modes	Notes
Doug Hall Electronics	RBI-1	FM	Is an interface to Kenwood FM mobile radios including the TMx21 series, TMx31 series, and TMx41 series. Not recommended for new installations.
ICOM	IC-706	FM,SSB,AM	160M-70CM all mode radio.
Kenwood	TM-271	FM	2 meter FM only. Requires modification to being the RXD and TXD lines out to support frequency and CTCSS control.
Kenwood	TMG-707	FM	2 meter and 70 centimeters. Requires a custom interface to multiplex COR and PTT with RXD and TXD. The TMG-707 was discontinued by Kenwood in 2006, but if you can find one, this is one of the better radios to use.
YAESU	FT-897	FM,SSB,AM	160M-70CM all mode radio.

Remote Base Node Stanza

There are only a few key-value pairs required to get a remote base node configured properly. This section covers the most common key-value pairs.

authlevel=

The authlevel= key is used to enable or disable login requirements for a remote base.

authlevel=	**Description**
0	No login required
1	Log in required. Wait for transmit before prompting for a log in.
2	Log in required. Asks for login following connection by a remote node

Note: log in is handled by putting special remote base commands in the function table stanza used by the remote base. If no valid login is received within 20 seconds, the calling node will be disconnected.

civaddr=

The civaddr= key is used with ICOM band radios to set the CIV address. The value is a 2 digit hexadecimal number. If this key is not specified, then the CIV address will be set to the default of 88. An example usage of this key-vale would be:

```
civaddr=98
```

functions=

The functions= key is used as a pointer to a remote base function table stanza. The function table stanza contains a list of key=value pairs whch describe valid DTMF commands for the remote base. For details about the function table, please refer to the section **Remote Base Function Table Stanza**. An example usage of this key-value would be:

```
functions=functions-remote
```

ioaddr=

The ioaddr= key refers to a parallel port I/O address. It is specified as a hexadecimal number with a 0x prefix. The parallel port is used when the Doug Hall RBI-1 interface is employed. An example usage of this key-value would be:

```
ioaddr=0x378
```

ioport=

The ioport= stanza is used to select a serial port on the PC where radio commands will be sent. On Linux Systems, these are typically path names to special files in the /dev directory. An example

usage of this key-value would be:

```
ioport=/dev/ttyS1
```

phone_functions=

The phone_functions= key is used as a pointer to a remote base function table stanza. The phone_functions= stanza allows for a different function table to be used when a connection is made from a phone. The function table stanza contains a list of key=value pairs which describe valid DTMF commands for the remote base. For details about the function table, please refer to the section **Remote Base Function Table Stanza**. An example usage of this key-value would be:

```
phone_functions=functions-remote
```

remote=

The remote= key allows the type of radio to be defined. Specifying remote= also ensures that the node will be defined as a remote base node and not a standard node.

Radio	Value	Comments
Dumb	y	Use for single channel remote base radios
FT-897	ft897	Must specify serial port using ioport=
TMG-707	kenwood	Must specify serial port using ioport=
IC-706	ic706	Must specify serial port using ioport=. Must specify civaddr using civaddr=
TM-271	tm271	Must specify serial port using ioport=
Doug Hall RBI	rbi	Must specify parallel port address using ioaddr=

rxchannel=

This should contain the name of a usb radio interface which has been defined in *usbradio.conf*

A Sample Remote Base Node Stanza

Here is a sample remote base stanza assembled using the information in the previous section:

```
[1234]

rxchannel = Radio/usb              ; Rx audio/signaling channel
ioport = /dev/ttyS1                ; Serial port for control
remote = ft897                     ; Radio Type
functions = remote-functions       ; Function list from link
phone_functions = remote-functions ; Function list from phone
authlevel = 0                      ; Authorization level
```

Remote Base Function Table Stanza

This is a sample remote base function table stanza which contains all of the valid DTMF commands currently implemented in app_rpt:

```
[functions-remote]

0=remote,1                              ; Retrieve Memory
1=remote,2                              ; Set freq.
2=remote,3                              ; Set tx PL tone
3=remote,4                              ; Set rx PL tone
40=remote,100                           ; Rx PL off
41=remote,101                           ; Rx PL on
42=remote,102                           ; Tx PL off
43=remote,103                           ; Tx PL on
44=remote,104                           ; Low Power
45=remote,105                           ; Medium Power
46=remote,106                           ; High Power
711=remote,107                          ; Bump -20
714=remote,108                          ; Bump -100
717=remote,109                          ; Bump -500
713=remote,110                          ; Bump +20
716=remote,111                          ; Bump +100
719=remote,112                          ; Bump +500
721=remote,113                          ; Scan - slow
724=remote,114                          ; Scan - quick
727=remote,115                          ; Scan - fast
723=remote,116                          ; Scan + slow
726=remote,117                          ; Scan + quick
729=remote,118                          ; Scan + fast
79=remote,119                           ; Tune (brief AM transmission for automatic
antenna tuner)
51=remote,5                             ; Long status query
52=remote,140                           ; Short status query
67=remote,210                           ; Send a *
69=remote,211                           ; Send a #
91=remote,99,CALLSIGN,LICENSETAG        ; Remote base login.
; Define a different dtmf sequence for each user which is
; authorized to use the remote base to control access to it.
; For examble 9139583=remote,99,WB6NIL,G would grant access to
; the remote base and announce WB6NIL as being logged in.
; Another entry, 9148351=remote,99,WA6ZFT,E would grant access to
; the remote base and announce WA6ZFT as being logged in.
; When the remote base is disconnected from the originating node, the
; user will be logged out. The LICENSETAG parameter can be optionally specified
; to enforce TX band limits.
98=cop,6                                ; Remote base telephone key
```

Not all commands above are supported by all radios. For example radios which don't support SSB, would not be able to be placed in LSB or USB mode.

Setting up a VOIP Autopatch

The autopatch feature in app_rpt allows users on the radio to interconnect with the public switched telephone network.

Regulatory Issues

Some countries which disallow third-party traffic do not allow autopatching or telephone interconnection on amateur radio frequencies. In addition, if a system is linked between one country which allows autopatching and one which doesn't, just the passing of the traffic itself across the link could be considered a violation of the rules in the prohibiting country. In countries which permit autopatching, users need be made aware of this and should take the node off-link before using the autopatch.

Security Issues

The examples provided here do not secure the autopatch against toll fraud. A lot can be done to prevent or reduce toll fraud, but the prevention measures vary by where you are located geographically. Most of the prevention measures will be in *extensions.conf,* as you can write contexts to reject certain number sequences. The asterisk book in the references section should be consulted to see how to do this, Before placing any autopatch in service, please make sure you have secured your system so that toll fraud does not become an issue.

Selecting an ITSP

Call termination into the public switched telephone network is a service offered by an Internet Telephone Service Provider (ITSP). This is a highly competitive business, and there are lots of ITSP's offering termination for Asterisk users. In the reference section is a link to a site which lists several ITSP's.
Termination is usually provided using the SIP protocol, and IAX is also offered by a few providers.

When selecting an ITSP, make sure they provide setup instructions with example configurations including usernames, and passwords. Quite a few ITSP's will automatically generate a custom SIP or IAX stanza for insertion into sip.conf, or iax.conf respectively.

Configuration Files

Three configuration files will be affected: *rpt.conf,* extensions.conf and *iax.conf* or *sip.conf*. The ITSP should provide a stanza for you to insert in *iax.conf* or *sip.conf.* The flow diagram below shows how each of these files are used when an outgoing autopatch call is made:

In the *rpt.conf* configuration file, a context in extensions.conf is defined in a node stanza using the context key:

```
[1234]
.
.
.
context=autopatch
.
.
.
```

Also in the *rpt.conf* file, entries are added to the function table so that users can bring up or take down the autopatch:

```
[functions]
.
.
.
6=autopatchup,noct=1,farenddisconnect=1,dialtime=20000
0=autopatchdn
.
.
.
```

In *extensions.conf*, a stanza for the autopatch context is added which refers to the peer stanza in *iax.conf* or *sip.conf*. For outbound IAX connections extensions.conf

```
[autopatch]

exten => _1NXXNXXXXXX,1,Dial,IAX2/peername/${EXTEN}
exten => _1NXXNXXXXXX,2,Congestion
```

For outbound SIP connections it should look like this:

```
[autopatch]
```

```
exten => _1NXXNXXXXXX,1,Dial,SIP/peername/${EXTEN}
exten => _1NXXNXXXXXX,2,Congestion
```

The peer stanza you insert in *iax.conf* or *sip.conf* should preferably be provided by your ITSP. You'll want to use the stanza they identify for performing outgoing connections. A nonoperational example of an iax2 peer stanza looks like this:

```
[peername]
type=peer
host=127.0.0.1
secret=nunya
auth=md5
disallow=all
allow=gsm
allow=ulaw
```

Autopatch Options

You may have noticed several options being passed in to the autopatchup command class.
This table summarizes what they do:

Autopatch Option	Description
context	Override the context specified for the autopatch in rpt.conf
dialtime	The maximum time to wait between DTMF digits when a telephone number is being dialed. The patch will automatically disconnect if this time is exceeded. The value is specified in milliseconds.
farenddisconnect	When set to 1, the patch will automatically disconnect when the called party hangs up. The default is send a circuit busy tone until the radio user brings the patch down.
noct	When this is set to 1 the courtesy tone during an autopatch call will be disabled. The default is to send the courtesy tone whenever the radio user unkeys.
quiet	When set to 1, Don't send dial tone, voice responses, just try to connect the call.

Setting Up Dial-in Access

The Asterisk app_rpt application supports dial in access which include facilities for dial-in control, and reverse autopatches. In this section we will cover only those features used for dial-in control, because there is not enough space in this guide to completely cover reverse autopatch configuration.

Security Issues

Dial-in access needs to be password protected if it can be accessed from the public switched telephone network. The example shown here does not provide password protection because it is not intended to be accessed outside of the Asterisk PBX. The authenticate application in asterisk should be used to protect any access to the radio system from dial in numbers on the public switched telephone network.

Phone Control Modes

The Asterisk app_rpt application supports several phone control modes. Phone control modes are accessed by appending an option to the Rpt application call in *extensions.conf*.

Dumb Phone Control Mode

In this mode, a dial-in user has partial control and audio access to the radio system. PTT will be activated when a user dials in and will remain activated for the entire duration of the call. For the user to have DTMF control, the 'dphone_functions' key must be defined and it must point to a valid function stanza, however, DTMF functions typically are not used in conjunction with this phone control mode. This phone control mode is useful when you have a full duplex radio, and you want to allow users to connect who have little knowledge about radio systems.

Normal Phone Control Mode

In this mode, a dial-in user has full control and audio access to the radio system. When a user dials in, PTT will remain de-activated until the user dials a specialized DTMF command to activate it. For the user to have DTMF control, the 'phone_functions' key must be defined and it must point to a valid function stanza. This phone control mode is useful when you want to monitor a radio system, or you have a half-duplex radio interfaced. This phone control mode is intended to be used by more technically inclined users.

Simple Phone Control Mode

Simple Phone Control Mode is a compromise between Dumb Phone Control Mode and Phone Control Mode. When a user dials in, they will only have audio access to the radio system, and the only DTMF characters which will be recognized are * and #. The * key will cause the transmitter PTT to be activated,
and the # key will cause the transmitter PTT to be de-activated. No other DTMF commands will be recognized. This mode is a good alternate for Dumb Phone Control Mode when a half-duplex radio is interfaced.

VOX Phone Control Mode

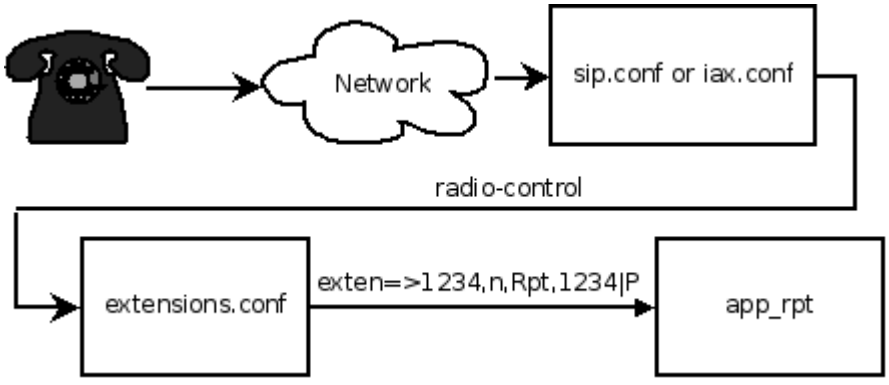
Vox Phone Control Mode is like phone mode, except the transmitter will automatically be keyed when there is sound is detected from the calling party.

Example usages for Extensions.conf

Phone Control Mode	**Command Example in *extensions.conf***
Dumb	exten=>1234,nRpt,1234|D
Normal	exten=>1234,n,Rpt,1234|P
Phone	exten=>1234,n,Rpt,1234|P
Simple	exten=>1234,n,Rpt,1234|S
VOX	exten=>1234,n,Rpt,1234|Pv

Setup

To set up dial in access, two files need to be modified: *extensions.conf,* and either *iax.conf* or *sip.conf*. A user stanza needs to be added to *iax.conf* or *sip.conf*, and a context needs to be added to *extensions.conf* to allow the user to call the rpt application with special arguments**.** Calls originating from a sip phone or softphone are first sent to the user stanza in sip.conf or iax.conf, then transferred to the incoming call context in extensions.conf. A flow diagram is shown below



When asterisk receives a connection request from the IP phone, it looks for a stanza in *iax,conf* or *sip.conf* which contains information which matches an ip address or shared secret. If it finds a match, then the context specified in that stanza (radio-control) will be called in extensions.conf. The context in extensions.conf will try to match the extension dialed (1234) and if there is a match, it will then call the app_rpt application (Rpt) with the node number and a |P flag.

A user stanza in sip.conf or iax.conf should look like this:

```
[my-ip-phone]
type=user
context=radio-control
auth=md5
secret=nunya   ; Important! Change this!!!
disallow=all
allow=gsm
allow=ulaw
transfer=no
```

The radio-control context in extensions.conf should look like this:

```
[radio-control] ; Change all instances of 1234 to your node num.
exten=1234,1,Answer
exten=1234,n,Playback,rpt/node
exten=1234,n,Playback,digits/1
exten=1234,n,Playback,digits/2
exten=1234,n,Playback,digits/3
exten=1234,n,Playback,digits/4
exten=1234,n,Rpt,1234|P
```

Note that the extension logic for the above stanza answers the call, and announces the node number which the caller is about to be connected to. Control is transferred to app_rpt using the last statement in the stanza.

Setting up IAXRPT Access

Security Issues

Incoming IAXRPT connections need to be authorized by making sure the incoming IAX user stanza is protected with a secret. This is the case because incoming IAX calls are not checked against a nodes stanza.

IAXRPT is a specialized soft phone program which allows users to connect from their PC's to an Allstar node. Setting up IAXRPT access is similar to seting up dial-in access, but a different command line switch is used in *extensions.conf*, and user autheorization is done in *iax.conf.*

First, a user stanza needs to added to iax.conf to allow incoming IAXRPT calls to be routed to a context in *extensions.conf*. A modified on the following stanza should be placed in iax.conf:

```
[gui]
type=user
context=radio-gui
auth=md5
secret=nunya  ; *** Important! Change this!!!
host=dynamic
disallow=all
allow=ulaw
allow=gsm
transfer=no
```

Second, a context stanza is added to extensions.conf to transfer control to the Rpt application:

```
[radio-gui]
exten = 1234,1,Rpt,1234|X    ; Change 1234 to your node number
```

The X option passed to the Rpt application disables the normal security checks. Because incoming connections are validated in iax.conf, and we don't know where the user will be coming from in advance, the X option is required.

Setting up Echolink Connectivity

With the addition of the chan_echolink Asterisk channel driver direct connections to and from echolink to Acid are now supported. No other programs (e.g. rtpDir, or the link box are required. With chan_echolink, Echolink nodes become part of the Allstar link number space.

From an app_rpt/Allstar node, Echolink connections look just like Allstar/App_rpt connections except the Echolink node numbers have been prefixed with a 3 and **padded out to 7 digits with leading zeroes.** For instance, if you want to connect to Echolink node 1234 on your app_rpt system you would dial *3 followed by 3001234. If you have a 6 digit Echolink node number 123456, you would dial *3 followed by 3123456. As you can see we have reserved Allstar node numbers with a leading 3 for the Echolink number space.

For users originating from an Echolink node using Echolink supplied software, nothing changes for them, they just dial the 4 or 6 digit Echolink node number assigned to your app_rpt system and they get connected!

To activate the Echolink channel driver, all that's required is a properly formatted configuration file. A base configuration file has been included with ACID and is located in /usr/src/configs/examples/echolink. Copy this file to /etc/asterisk, and edit it to match your callsign, password and node number assignment from Echolink.org. The sample echolink.conf file is as follows:

```
[el0]
confmode=no
call=W6ABC-R   ; Change this!
pwd=XXXXX ; Change this!
name=Asterisk-EL-channel-driver ; Change this to your real name!
qth=Asterisk-EL-channel-driver ; Change this to your actual QTH!
email=foo@bar.com ; Change this!
maxstns=20
rtcptimeout=10
node=123456 ; Change this!
recfile=/tmp/recorded.gsm
astnode=2345 ; Change this to your active Allstar node number!
context=radio-secure
server1=server1.echolink.org
server2=server2.echolink.org
server3=server3.echolink.org
```

Once you have made the changes, restart Asterisk, and within a few minutes time, the node should show up on the http://echolink.org website.

Denying incoming connections from selected Echolink nodes

> Echolink connections may be denied on a per-callsign basis. This is done by using the deny and permit key values in the above [el0] stanza. The default is to allow all connections if the permit and deny keywords are not present. If a permit is specified, then only the callsigns specified in

the permit statement will be allowed to connect. If deny is specified, then the callsign(s) specified will be denied access and the connection will be terminated. Commas are used to delimit multiple callsigns for permit and deny keys. Wildcards are supported so that whole classes of connections can be rejected. Examples:

1. To deny w6xxx you would add the statement: `deny=w6xxx`
2. To prohibit computer-based connections you would write: `permit=*-*`
3. To allow access to only a select group of callsigns: `permit=w6abc,w6def,...`

Setting up IRLP Connectivity

In order to install IRLP connectivity on your ACID installation, it helps to be familiar with how IRLP nodes are installed in general. A lot of what happens with IRLP in an ACID installation is similar to how IRLP preforms node installations and re-installations using their hardware.

ACID comes with a setup script to install the IRLP binaries and scripts. The setup script can either perform a new installation, or an installation from a backup. The script is located in the /root directory. To run the script type:

```
./irlpsetup.sh
```

Follow the instructions and select NEW or REINSTALL You will be asked for some information about your node or where to read in the backup file.

If you are installing a new node. Run the script first and complete the installation before contacting IRLP for a new node number. Make a note of the stnXXXX number in the script and reference it when contacting IRLP. Additionally, after installation of a new node is complete, you must provide additional data by accessing a special “status database” web page from the public IP address of the system. See the IRLP owner's FAQ for details.

If you are re-installing an IRLP node, make a backup using the backup script in the scripts directory before running the IRLP install script on the ACID installation. Place the backup file in the /tmp directory and refer to it when the script asks for its location.

Use

From an app_rpt/Allstar node, IRLP connections look just like Allstar/App_rpt connections except the IRLP node numbers have been prefixed with a 4. For instance, if you want to connect to IRLP node 1234 on your app_rpt system you would dial *3 followed by 41234. As you can see we have reserved Allstar node numbers with a leading 4 for the IRLP number space.

For users originating from an IRLP node using IRLP supplied software, nothing changes for them, they just dial the 4 IRLP node number assigned to your app_rpt system and they get connected!

Appendix A: References

1. Linux Resources:
   aa. Command line: http://www.physics.ubc.ca/mbelab/computer/linux-intro/html/
   bb. Reference Guide for Newbies: http://blog.lxpages.com/ultimate_linux.html
2. Asterisk Resources:
   aa, http://voip-info.org
   bb. http://asterisk.org
   cc. O'Reilly publishing: Asterisk the Future of Telephony Second Edition ISBN 978-0596510480 Free online version available at:
   http://cachefly.oreilly.com/books/9780596510480.pdf

3. ITSP list: http://voip-info.org/wiki/view/Cheapest+ATAs+and+Service