Rpt.conf

From "AllStarLink Wiki"

Rpt.conf is where the majority of user-facing features, such as the node's CW and voice ID, DTMF commands and timers are set. There is a lot of capability here which can be difficult to grasp. Fortunately the default DIAL rpt.conf is well commented and will work fine for most node admins.

Contents

DTMF Commands

DTMF commands are placed in any one of three named stanzas. Function stanzas control access to DTMF commands that a user can issue from various control points.

- The fuctions stanza - to decode DTMF from the node's local receiver.
- The link_functions stanza - to decode DTMF from linked nodes.
- The phone_functions stanza - to decode DTMF from telephone connects.

A function stanza key/value pair has the following format:

dtmfcommand=functionclass,[functionmethod],[parameters] Where:

- dtmfcommand is a DTMF digit sequence **minus** the start character (usually *)

- functionclass is a string which defines what class of command; link, status or COP
- functionmethod defines the optional method number to use in the function class.
- functionoptions are one or more optional comma separated parameters which further define a command.

Status Commands

Status commands (functioncalss 'status') provide general information about the node.

Sample:

```
712 = status,12   ; Give Time of Day (local only)
```

Status	Description
1	Force ID (global)
2	Give Time of Day (global)
3	Give software Version (global)
4	Give GPS location info
5	Speak the last (dtmf) user
11	Force ID (local only)
12	Give Time of Day (local only)

Link Commands

Link commands (functioncalss 'link') connect to, disconnect from, monitor (Rx only) other nodes and give link status.

Sample:

```
3 = ilink,3   ; Connect specified link -- transceive
```

ilink	Description
1	Disconnect specified link
2	Connect specified link -- monitor only
3	Connect specified link -- tranceive
4	Enter command mode on specified link
5	System status
6	Disconnect all links
7	Last Node to Key Up
8	Connect specified link -- local monitor only
9	Send Text Message (9,<destnodeno or 0 (for all)>,Message Text, etc.
10	Disconnect all RANGER links (except permalinks)
11	Disconnect a previously permanently connected link
12	Permanently connect specified link -- monitor only
13	Permanently connect specified link -- tranceive
15	Full system status (all nodes)
16	Reconnect links disconnected with "disconnect all links"
17	MDC test (for diag purposes)
18	Permanently Connect specified link -- local monitor only

COP Commands

Control operator (functionclass 'cop') commands are privileged commands. Node admins may provide some of these to their user community based on personal preference.

Sample:

```
99 = cop,7   ; enable timeout timer
```

Some COP commands can take multiple parameters. For example this COP 48 would send #3#607 on command.

```
900 = cop,48,#,3,#,6,0,7
```

COP	Description
1	System warm boot
2	System enable
3	System disable
4	Test Tone On/Off
5	Dump System Variables on Console (debug)
6	PTT (phone mode only)
7	Time out timer enable
8	Time out timer disable
9	Autopatch enable
10	Autopatch disable
11	Link enable
12	Link disable
13	Query System State
14	Change System State
15	Scheduler Enable
16	Scheduler Disable
17	User functions (time, id, etc) enable
18	User functions (time, id, etc) disable
19	Select alternate hang timer
20	Select standard hang timer
21	Enable Parrot Mode
22	Disable Parrot Mode
23	Birdbath (Current Parrot Cleanup/Flush)
24	Flush all telemetry
25	Query last node un-keyed
26	Query all nodes keyed/unkeyed
27	Reset DAQ minimum on a pin
28	Reset DAQ maximum on a pin
30	Recall Memory Setting in Attached Xcvr
31	Channel Selector for Parallel Programmed Xcvr
32	Touchtone pad test: command + Digit string + # to playback all digits pressed
33	Local Telemetry Output Enable
34	Local Telemetry Output Disable
35	Local Telemetry Output on Demand
36	Foreign Link Local Output Path Enable
37	Foreign Link Local Output Path Disable
38	Foreign Link Local Output Path Follows Local Telemetry
39	Foreign Link Local Output Path on Demand
42	Echolink announce node # only
43	Echolink announce node Callsign only

44	Echolink announce node # & Callsign
45	Link Activity timer enable
46	Link Activity timer disable
47	Reset "Link Config Changed" Flag
48	Send Page Tone (Tone specs separated by parenthesis)
49	Disable incoming connections (control state noice)
50	Enable incoming connections (control state noicd)
51	Enable sleep mode
52	Disable sleep mode
53	Wake up from sleep
54	Go to sleep
55	Parrot Once if parrot mode is disabled
56	Rx CTCSS Enable
57	Rx CTCSS Disable
58	Tx CTCSS On Input only Enable
59	Tx CTCSS On Input only Disable
60	Send MDC-1200 Burst (cop,60,type,UnitID[,DestID,SubCode]) Type is 'I' for PttID, 'E' for Emergency, and 'C' for Call (SelCall or Alert), or 'SX' for STS (ststus), where X is 0-F. DestID and subcode are only specified for the 'C' type message. UnitID is the local systems UnitID. DestID is the MDC1200 ID of the radio being called, and the subcodes are as follows: Subcode '8205' is Voice Selective Call for Spectra ('Call') Subcode '8015' is Voice Selective Call for Maxtrac ('SC') or Astro-Saber('Call') Subcode '810D' is Call Alert (like Maxtrac 'CA')
61	Send Message to USB to control GPIO pins (cop,61,GPIO1=0[,GPIO4=1].....)
62	Send Message to USB to control GPIO pins, quietly (cop,62,GPIO1=0[,GPIO4=1].....)
63	Send pre-configred APRSTT notification (cop,63,CALL[,OVERLAYCHR])
64	Send pre-configred APRSTT notification, quietly (cop,64,CALL[,OVERLAYCHR])
65	Send POCSAG page (equipped channel types only)

Remote Base Commands

Remote base commands (functionclass 'remote') provide remote base control. This stanza is named by the remote base Remote Base Functions key/value pair.

remote	Description
1	Retrieve Memory
2	Set freq.
3	Set tx PL tone
4	Set rx PL tone
5	Long status query
99,CALLSIGN,LICENSETAG	Remote Base login
100	Rx PL off
101	Rx PL on
102	Tx PL off
103	Tx PL on
104	Low Power
105	Medium Power
106	High Power
107	Bump -20
108	Bump -100
109	Bump -500
110	Bump +20
111	Bump +100
112	Bump +500
113	Scan - slow
114	Scan - quick
115	Scan - fast
116	Scan + slow
117	Scan + quick
118	Scan + fast
119	Tune
140	Short status query
210	Send a *
211	Send a #

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.

Remote Base Login

When authlevel= is greater than zero, remote 99 is used to define a different dtmf sequence for each user authorized to use the remote base. The remote base will announce the callsign as access is granted.

Sample:

```
9139583 = remote,99,WB6NIL,G   ; grant access to Jim (general)
9148351 = remote,99,WA6ZFT,E   ; grant access to Steve (extra)
```

When the remote base is disconnected from the originating node, the user will be logged out.

The LICENSETAG argument is used to enforce tx frequency limits. Need info [txlimits].

Settings to name other Stanzas

Within the node stanza, some key=value pairs point to other stanzas. This allows nodes on the same Asterisk/app_rpt server to have the same settings (without duplicate entries) or different settings in some cases. For example the phone patch command may be *6 on one node, yet *61 on another.

For example:

```
[1000]
functions=functionsVHF

[1001]
functions=functionsVHF

[1002]
functions=functionsUHF

[functionsVHF]
; Two meter Autopatch up is *6
6=autopatchup,noct=1,farenddisconnect=1,dialtime=20000
0=autopatchdn       ; Autopatch down

[functionsUHF]
; 440 Autopatch up is *61
61=autopatchup,noct=1,farenddisconnect=1,dialtime=20000  ; Autopatch up
0=autopatchdn       ; Autopatch down
```

- controlstates=
- functions=
- link_functions=
- macro=
- phone_functions=
- telemetry=
- wait_times=

Node Number Stanza

The node number stanza is the first stanza in rpt.conf.

```
[1999]    ; Replace with your assigned node number
```

The node number stanza contains the following key/value pairs.

althangtime=

This controls the length of the repeater hang time when the alternate hang timer is selected with a control operator function. It is specified in milliseconds.

Sample:

```
althangtime=4000   ; 4 seconds
```

beaconing=

When set to 1 will send the repeater ID at the idtime interval regardless of whether there was repeater activity or not. This feature appears to be required in the UK, but is probably illegal in the US.

Sample:

```
beaconing=1   ;Set to 1 to beacon. Defaults to 0
```

callerid=

This setting allows the autopatch on the node to be identified with a specific caller ID. The default setting is as follows

```
callerid="Repeater" <0000000000>
```

Note: The value in quotes is the name of the autopatch caller, and the numbers in angle brackets are the originating telephone number to use.

context=

This setting directs the autopatch for the node to use a specific context in extensions.conf for outgoing autopatch calls. The default is to specify a context name of radio.

```
context=radio
```

controlstates=

This setting defines the name of the variable named control state stanza. Control states are an optional feature which allows groups of control operator commands to be executed all at once. To use control states, define an entry in your node stanzas to point to a dedicated control state stanza like this:

```
controlstates = controlstates   ; points to control state stanza


[controlstates]
0 = rptena,lnkena,apena,totena,ufena,noicd  ; Normal operation
1 = rptena,lnkena,apdis,totdis,ufena,noice  ; Net and news operation
2 = rptena,lnkdis,apdis,totena,ufdis,noice  ; Repeater only operation
```

The control state stanza describes these mnemonics.

duplex=

The duplex key/value pairs sets the duplex mode for desired radio operation. Duplex mode 2 is the default if none specified.

Duplex 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. Comment out idrecording and idtalkover to suppress IDs.
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. See details
4	Full Duplex with telemetry tones and hang time. Repeated audio only when the autopatch is down.

Sample:

```
duplex = 0     ; 0 = Half duplex with no telemetry tones or hang time.
```

endchar=

This setting allows the end character used by some control functions to be changed. By default this is a #. The endchar value must not be the same as the funcchar default or overridden value.

erxgain=

Echolink receive gain adjustment in +/- db-volts. Used to balance Echolink recieve audio on an app_rpt node.

```
erxgain = -3
```

See Echolink How to

etxgain=

Echolink transmit gain adjustment in +/- db-volts. Used to balance Echolink transmit audio on an app_rpt node.

```
etxgain = 3
```

See Echolink How to

funcchar=

This setting can be used to change the default function starting character of * to something else. Please note that the new value chosen must not be the same as the default or overridden value for endchar=.

functions=

This names (points to) the function stanza.

Sample:

```
functions = functionsUHF   ; pointer to 440 repeater functions stanza
```

GUI

Key	Valid Values	Default
guilinkdefault	0 = telemetry output off, 1 = telemetry output on, 2 = timed telemetry output on command execution and for a short time thereafter, 3 = follow local telemetry mode	1
guilinkdynamic	0 = disallow users to change the gui telemetry setting with a COP command, 1 = Allow users to change the setting with a COP command	1

Function Classes

Function classes are names for categories of functions. There are several function classes defined. They are described in the table below:

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 a function method and possibly one or more additional command parameters. Function methods are discussed next.

Function Methods

Function methods are numbers which identify a specific function to execute within a function class. Function Methods are optional and in some cases should be omitted (Such as with the autopatchup method) A complete and up-to-date description of all functon methods can be found in the app_rpt.c source file. Some function methods are shown below as an example below:

```
1  - Force ID (global)
2  - Give Time of Day (global)
3  - Give software Version (global)
11 - Force ID (local only)
12 - Give Time of Day (local only)
```

Function Options

Some Function Methods can take function options. These are specified after the Function Method separated with commas. Not all commands require or take Function options. An example of a method which can accept Function Options is the autopatchup method.

Putting it all Together

A small excerpt from the function stanza of rpt.conf is hown below.

```
[functions]
1=ilink,1                                                ; Specific link disconnect
6=autopatchup,noct=1,farenddisconnect=1,dialtime=2000    ; Autopatch up
```

The above example contains DTMF functions with all of the parameters discussed on previous pages.

In the above example:

- *1 followed by a node number disconnects a specific link. The function class is *ilink* and the function method is *1*
- *6 followed by a phone number brings up the autopatch with the function options specified. Note that there is no function method defined, but there are function options present.

hangtime=

This controls the length of the repeater hang time. It is specified in milliseconds.

Sample:

```
hangtime = 1000   ;Set hang time for 1 second
```

holdofftelem=

This node stanza configuration key forces all telemetry to be held off until a local user on the receiver or a remote user over a link unkeys. There is one exception to this behavior: When an ID needs to be sent and there is activity coming from a linked node.

Sample:

```
holdofftelem = 1   ;Set to 1 to hold off. Default is 0
```

telemdefault=

```
telemdefault =  0 = telemetry output off, 1 = telemetry output on, 2 = timed telemetry output on command executi
```

telemdynamic=

```
telemdynamic = 0 = disallow users to change the local telemetry setting with a COP command, 1 = Allow users to c
```

phonelinkdefault=

```
phonelinkdefault = 0 = telemetry output off, 1 = telemetry output on, 2 = timed telemetry output on command exec
```

phonelinkdynamic=

```
phonelinkdynamic = 0 = disallow users to change phone telemetry setting with a COP command, 1 = Allow users to c
```

idrecording=

The identifier message 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.

Sample:

```
idrecording = |iwa6zft/r   ; Morse Code ID
```

or

```
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. See Recording audio files

idtalkover=

The ID talkover message is stored in the node stanza using the idtalkover setting. The purpose of idtalkover is to specify an alternate ID to use when the ID must be sent over the top of a user transmission, This can be a shorter voice ID or an ID in morse code. 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:

Sample:

```
idtalkover = |iwa6zft/r   ; Morse Code ID
```

or

```
idtalkover = /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. See Recording audio files

inxlat=

The inxlat setting allows complete remapping of the funcchar and endchar digits to different digits or digit sequences. inxlat acts on the digits received by the radio receiver on the node.

Format: inxlat = funchars,endchars,passchars,dialtone

where:

- funcchars is the digit or digit sequence to replace funcchar
- endchars is the digit or digit sequence to replace endchar
- passchars are the digits to pass through (can be used to block certain digits)
- dialtone is optional. Set to Y for dial tone on successful funcchars.

Sample:

```
inxlat = #56,#57,0123456789ABCD ; string xlat from radio port to sys
```

link_functions=

This names (points to) the link_functions stanza.

The link_functions setting directs the node to use a particular function stanza for functions dialed by users accessing the node via a link from another node. The traditional default is to point it to a function stanza named "functions".

Sample:

```
link_functions = functions ; pointer to the Link Function stanza
```

linkmongain=

Link Monitor Gain adjusts the audio level of monitored nodes when a signal from another node or the local receiver is received. If linkmongain is set to a negative number the monitored audio will decrease by the set amount in db. If linkmongain set to a positive number monitored audio will increase by the set amount in db. The value of linkmongain is in db. The default value is 0 db.

Sample:

```
linkmongain = -20   ; reduce link volume 20dB
```

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.

Sample:

```
linktolink = yes   ; set to yes to enable. Default is no.
```

linkunkeyct=

The linkunkeyct setting selects the courtesy tone to send when a connected remote node unkeys. The default is no courtesy tone when a remote node unkeys.

Sample:

```
linkunkeyct = ct8  ; use courtesy tone 8
```

macro=

The marco key/value points to by the Macro Stanza key/value pair. Macros are DTMF shortcuts.

Sample:

```
macro=macro   ; use stanza names 'macros'

[macro]
1 = *32000*32001     ; connect to nodes 2000 and 2001
```

nounkeyct=

The nounkeyct node stanza key completely disables the courtesy tone. This is useful for eliminating TX time in applications using simplex uplinks to repeaters on the repeater pair itself. This practice is **strongly** discouraged.

Sample:

```
nounkeyct = yes  ; :(
```

politeid=

The politeid setting specified the number of milliseconds prior to the end of the id cycle where the controller will attempt to play the ID in the tail when a user unkeys. If the controller does not get a chance to send the ID in the tail, the ID will be played over the top of the user transmission. Optional, default 30000.

Sample:

```
politeid = 30000   ; 30 seconds
```

propagate_dtmf=

Takes either yes or no. When set to yes, DTMF will be regenerated from out-of-band signalling or from from the receiver dtmf decoder whenever a function start character is NOT detected and command decoding has not begun. When set to no, no tones will be regenerated. The default for this setting is no. This setting is meant to be used in conjunction with linktolink, inxlat, and outxlat when interfacing a radio port to a multiport analog repeater controller on an RF-linked system.

Sample:

```
propagate_dtmf = no
```

Note: There is a separate setting propagate_phonedtmf for use by dial-in (phone and dphone) users.

remotect=

The remotect setting allows the selection of the remote linked courtesy tone so that the users can tell there is a remote base connected locally.

Sample:

```
remotect = ct3   ; use courtesy tone 3
```

rxburstfreq=

If rx tone burst operation is desired, specify the frequency in hertz of the desired tone burst. Setting to 0 (or not specifying) indicates no tone burst operation.

Sample:

```
rxburstfreq = 1000
```

rxburstthreshold=

In rx toneburst mode, specifies the minimum signal to noise ratio in db that qualifies a valid tone. Defaults to 16 (db).

rxbursttime=

For rx toneburst operation, specifies minimum amount of time that tone needs to be valid for recognition (in milliseconds). Defaults to 250.

rxchannel=

The rxchannel key/value pair selects the radio interface. There must be one (and only one) rxchannel per node. The selections are:

Value	Description
dahdi/pseudo	No radio, used for hubs
SimpleUSB/usb_1999	SimpleUSB (limited DSP)
Radio/usb_1999	USBRadio (full DSP)
voter/1990	RTCM
Pi/1	Raspberry Pi PiTA
Dahdi/1	PCI Quad card
Beagle/1	BeagleBoard
USRP/127.0.0.1:34001:32001	GNU Radio interface USRP

Sample:

```
rxchannel = dahdi/pseudo     ; No radio (hub)
```

Note: This is selecting what is known as (in Asterisk terminology) the channel driver.

scheduler=

This setting defines the scheduler stanza. The scheduler is used to execute commands at a particular time.

Sample:

```
scheduler = schedule   ; name scheduler to 'schedule'

[schedule]
...
```

sleeptime=

This sets the inactivity period in seconds for no signal on the repeater receiver before the system goes to sleep.

Sample:

```
sleeptime = 300   ; go to sleep after 5 mins of no activity
```

startup_macro=

The startup_macro is executed once on system startup. Each node can have one startup macro defined in its node stanza.

Sample:

```
startup_macro = *31000*31001*31002   ; Connect to 1000, 1001 and 1002 at startup
```

tailmessagelist=

The tailmessagelist setting allows a comma separated list of audio files to be specified for the tail message function. The tail messages will rotate from one to the next until the end of the list is reached at which point the first message in the list will be selected. If no absolute path name is specified, the directory var/lib/asterisk/sounds will be searched for the sound file. The file extension should be omitted.

Sample:

```
tailmessagelist = welcome,clubmeeting,wx   ; rotate 3 tail messages
```

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. See Recording audio files

tailmessagetime=

This sets the amount of time in milliseconds between tail messages. Tail Messages are played when a user unkeys on the repeater input at the point where the hang timer expires after the courtesy tone is sent.

Sample:

```
tailmessagetime = 900000   ; 15 minutes between tail messages
```

tailsquashedtime=

If a tail message is "squashed" by a user keying up over the top of it, a separate time value can be loaded to make the tail message be retried at a shorter time interval than the standard tailmessagetime= setting. The tailsquashedtime= setting takes a value in milliseconds.

Sample:

```
tailsquashedtime = 300000   ; 5 minutes
```

totime=

This defines the time out timer interval. The value is in milliseconds.

Sample:

```
totime = 180000   ;  3 minutes
```

unlinkedct=

The unlinkedct setting selects the courtesy tone to be used when the system has no remote nodes connected and is operating as a standalone repeater.

Sample:

```
unlinkedct = ct2   ; use courtesy tone 2
```

wait-times=

Wait-times points to the Wait Times Stanza.

Sample:

```
wait-times = wait-times   ; name wait-times to 'wait-times'
```

Functions Stanza

The functions stanza is a named stanza pointed to by the function= key/value pair. Functions within this stanza are used to decode DTMF commands when accessing the node from its receiver. This stanza is typically named 'functions'.

Sample:

```
functions = functions   ; name the functions stanza 'functions'

[functions]
...
```

See DTMF Commands for the list of functions available.

Link Functions Stanza

The link functions stanza is a named stanza pointed to by the link_function= key/value pair. Functions within this stanza are used decode DTMF commands when accessing the node via a link from another node. The traditional usage is to point link_functions= to the same stanza as named by functions= thereby having functions from a linked node and from the local node the same.

Sample:

```
functions = functions        ; name the functions stanza 'functions'
link_functions = functions   ; use the same stanza

[functions]
...
```

If a different set of either limited or more capable functions is desired:

```
functions = functions               ; name the functions stanza 'functions'
link_functions = my_link_functions   ; use a different stanza

[functions]
...

[my_link_functions]
...
```

See DTMF Commands for the list of functions available.

Phone Functions Stanza

The phone functions stanza is a named stanza pointed to by the phone_function= key/value pair. Functions within this stanza are used decode DTMF commands when accessing the node from a telephone. The traditional usage is to point phone_functions= to the same stanza as named by functions= thereby having functions from a phone and from the local node the same.

Sample:

```
functions = functions         ; name the functions stanza 'functions'
phone_functions = functions   ; use the same stanza

[functions]
...
```

If a different set of either limited or more capable functions is desired:

```
functions = functions                  ; name the functions stanza 'functions'
phone_functions = my_phone_functions   ; use a different stanza

[functions]
...

[my_phone_functions]
...
```

See DTMF Commands for the list of functions available.

Macro Stanza

The marco stanza is named stanza pointed to by the macro= key/value pair. Macros are DTMF shortcuts.

Sample:

```
macro=macro   ; use stanza names 'macros'

[macro]
1 = *32000*32001     ; connect to nodes 2000 and 2001
```

Remote Base Stanza

Here are key/value pairs required to configure a remote base.

authlevel=

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

- authlevel=0 Disables all access control (not recommended, unsecured)
- authlevel=1 Enables access control, and waits for key up before prompting for the access code
- authlevel=2 Enables access control, and prompts for the access code at the time of connection

Sample:

```
autllevel = 0   ; allow all comers
```

civaddr=

civaddr= 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.

Sample:

```
civaddr = 98  ; set CIV to 98
```

Remote Base functions=

functions= is a pointer to a remote base function stanza.

Sample:

```
functions = functions-remote   ; name the functions stanza 'functions-remote'

[functions-remote]
...
```

See Remote Base Functions Stanza.

ioaddr=

ioaddr= 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.

Sample:

```
ioaddr = 0x378   ; set RB-1 address
```

ioport=

ioport= sets the serial port. On Linux Systems, these are typically path names to special files in the /dev directory.

Sample:

```
ioport = /dev/ttyS1   ; Linux com1
```

Remote Base phone_functions=

phone_functions= is a pointer to a remote base phone function stanza. It is equivalent to phone_functions in the Phone Functions Stanza but provides a separate set of commands.

Sample:

```
phone_functions = functions-remote
```

remote=

remote= sets the type of radio. It also ensures that the node will be defined as a remote base node and not a standard node.

Sample:

```
remote = xcat   ; set xcat interface
```

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=
Syntor Xcat	xcat	Must specify serial port using ioport=. Must specify civaddr using civaddr=

remote_inact_timeout=

Specifies the amount of time without keying from the link. Set to 0 to disable timeout.

Sample:

```
remote_inact_timeout = 0   ; do not time out
```

remote_timeout=

Session time out for remote base. Set to 0 to disable.

Sample:

```
remote_timeout = 0   ; do not timeout
```

remote_timeout_warning=

remote_timeout_freq=

Remote Base rxchannel=

This should contain the name of a usb radio interface which has been defined in usbradio.conf or a zaptel interface number if using a Quad Radio PCI Card.

Sample:

```
rxchannel = usbRadio/usb1
```

Sample Remote Configuration

```
[1234]
; Rx audio/signaling channel
rxchannel = Radio/usb

; Serial port for control
ioport = /dev/ttyS1

; Radio Type
remote = ft897

; Function list from link
functions = functions-remote

; Function list from phone
phone_functions = functions-remote

; Authorization level
authlevel = 0
```

Nodes Stanza

The [nodes] stanza is a list of nodes, their IP addresses, port and "NONE" for non-remote base nodes. The nodes stanza is used to identify which node is mapped to which Internet call and to determine the destination to send the call to. If you are using automatic update for Allstar link (public) nodes, no Allstar link nodes should be defined here. Only place a definition for your local nodes, and private (off of allstar link) nodes or nodes behind the same NAT router here.

Sample:

```
[nodes]
1000 = radio@127.0.0.1/1000,NONE              ; Private hub on this server
1001 = radio@host.domain.com/1001,NONE        ; Private node on another server
2501 = radio@127.0.0.1/2501,NONE              ; Public node on this server
2502 = radio@127.0.0.1/2502,NONE              ; Another public node on this server
2503 = radio@192.168.1.20:4570/2503,NONE      ; Public node behind the same NAT router
```

The [nodes] stanza performs a function similar to an OS hosts file. When looking up node information, app_rpt looks in the [nodes] stanza first then searches (what could be called the Allstar DNS) the /var/lib/asterisk/rpt_extnodes file.

Telemetry Stanza

This stanza is named by the telemetry= key/value pair. Telemetry entries can be shared across all nodes on the Asterisk/app_rpt server, or defined for each node. Can be a tone sequence, morse string, or a file as follows:

- |t - Tone escape sequence:
  - Tone sequences consist of 1 or more 4-tuple entries (freq1, freq2, duration, amplitude). Single frequencies are created by setting freq1 or freq2 to zero.
- |m - Morse escape sequence:
  - Sends Morse code at the **telemetry amplitude and telemetry frequency** as defined in the [morse] section. Follow with an alphanumeric string.
- |i - Morse ID escape sequence:
  - Sends Morse code at the **ID amplitude and ID frequency** as defined in the [morse] section. Follow with an alphanumeric string.
- Path to sound file:
  - Specify the path to a sound file on the server. Do not include file extension.

Sample:

```
[telemetry]
ct1=|t(350,0,100,2048)(500,0,100,2048)(660,0,100,2048)
ct2=|t(660,880,150,2048)
ct3=|t(440,0,150,4096)
ct4=|t(550,0,150,2048)
ct4=|t(2475,0,250,768)
ct5=|t(660,0,150,2048)
ct6=|t(880,0,150,2048)
ct7=|t(660,440,150,2048)
ct8=|t(700,1100,150,2048)
ct9=|t(1633,0,50,1000)(0,0,30,0)(1209,0,50,1000);
;remotetx=|t(1633,0,50,3000)(0,0,80,0)(1209,0,50,3000);
remotetx=|t(880,0,150,2048)
remotemon=|t(1209,0,50,2048)
cmdmode=|t(900,903,200,2048)
functcomplete=|t(1000,0,100,2048)(0,0,100,0)(1000,0,100,2048)
patchup=rpt/callproceeding
patchdown=rpt/callterminated

What the numbers mean,
 (000,000,010,000)
   |   |   |   |-------amplitude
   |   |   |-------------duration
   |   |-------------------Tone 2
   |-------------------------Tone 1
```

Morse Stanza

Morse code parameters, these are common to all nodes on a given Asterisk/app_rpt server.

Sample:

```
[morse]
speed = 20            ; Approximate speed in WPM
frequency = 900       ; Morse Telemetry Frequency
amplitude = 4096      ; Morse Telemetry Amplitude
idfrequency = 746     ; Morse ID Frequency
idamplitude = 768     ; Morse ID Amplitude
```

Wait Times Stanza

This stanza is named by the wait-times= key/value pair. The wait time stanza is used to set delay time between various node actions and their response. Values are in milliseconds.

Sample:

```
wait-times = wait-times   ; name the stanza 'wait-times'

[wait-times]
telemwait = 2000
idwait = 500
unkeywait = 2000
calltermwait = 2000
```

Remote Base Memory Stanza

Remote base memories are in the format of:

memory = rxfreq,plfreq,txpower,offset,tone

Sample:

```
[memory]
00 = 146.580,100.0,m
01 = 147.030,103.5,m+t
02 = 147.240,103.5,m+t
03 = 147.765,79.7,m-t
04 = 146.460,100.0,m
05 = 146.550,100.0,m
```

Control States Stanza

There are several predefined nmemonics (keywords) used in the control state stanza to enable and disable the various features of the controller. These nmemonics correspond to the control operator command to be executed and most of these are the same groups of letters sent back when a single control operator command is executed on the controller.

Nmemonic	Description	COP Method
rptena	Repeater Enable	2
rptdis	Repeater Disable	3
totena	Timeout Timer Enable	7
totdis	Timeout Timer Disable	8
apena	Autopatch Enable	9
apdis	Autopatch Disable	10
lnkena	Link Enable	11
lnkdis	Link Disable	12
skena	Scheduler Enable	15
skdis	Scheduler Disable	16
ufena	User Functions Enable	17
ufdis	User Functions Disable	18
atena	Alternate Hangtime Enable	19
atdis	Alternate Hangtime Disable	20
noice	No Incoming Connections Enable	49
noicd	No Incoming Connections Disable	50
slpen	Sleep Mode Enable	51
slpds	Sleep Mode Disable	52

Schedule Stanza

The scheduler can execute commands at certain times. For example for a net on Tuesday nights at 8 PM.

Sample:

```
scheduler=schedule   ; name the stanza 'schedule'

[schedule]
;dtmf_function =  m h dom mon dow  ; ala cron, star is implied
2 = 00 00 * * *   ; at midnight every day, execute macro 2.
```

DAQ List Stanza

Meter Faces Stanza

Alarms Stanza

Categories: Node Configuration | Config Files

- This page was last modified on 16 June 2019, at 18:14.