Rhino Equipment Corporation
Application Note

Installing Rhino Cards on a New Trixbox Install

1.0 Overview

This process assumes that you have loaded the Trixbox 1.2.3 or 2.0 ISO onto your PC with no issues. Rhino PCI cards are easy to install by following a few steps. The main process is:

1) Install Rhino cards into the PC, turn on PC
2) Download the kernel development headers
3) Download the current Zaptel-1.2 sources (this is an update to Zaptel)
4) Download the Rhino driver code corresponding with your installed hardware
5) Compile Zaptel and the Rhino drivers
6) Configure your system's zaptel.conf and zapata.conf to use your hardware
7) test and run with it!

This is a tested step-by-step document for Trixbox 1.2.3, and we expect that if these instructions are followed you will have an error free installation. Of course we will do our best to help you out if there is trouble!

One special note: Rhino has developed a single analog card driver, called the **rcbfx**, that is to be used on all Rhino analog cards, which include: R4FXO Rev. C and upwards, R8FXX, R24FXX, R24FXO and R24FXS. This also includes all cards with Echo Cancellation on-board. The R4FXO Rev. A and Rev. B boards use the older r4fxo driver.

For those of you who are experts, section 8.0 is a quick reference with just the commands, and not the long explanations.

2.0 Get the Linux Development Headers for CentOS and Fix Two Issues with Distribution

The Linux kernel headers are required any time you are compiling kernel modules (drivers) for the system. Getting the sources is as simple as using the YUM package management utility. This process assumes that you can either login directly with a keyboard and monitor, or use an SSH session to connect remotely. What you will type at the keyboard will be shown in ***bold italics*** text.

You MUST be connected to the internet for this to work!

2.1 Login as ***root***, and enter the password that you set during the install process.

2.2 Type

yum install –y kernel-devel

to pull the development kernel sources onto your machine. This will take some time, since the download is about 3.7 megabytes; watch for the process to complete.

If you are running an SMP kernel, for instance if you are running a multi-processor or multi-core machine, the correct command is:

yum install –y kernel-smp-devel

2.3 There is a typo in the headers we have just downloaded that must be gixed manually. The error is in the spinlock.h file, which can be edited using the following command:

nano +407 /lib/modules/`uname –r'/build/include/linux/spinlock.h

The 'uname –r' is an important parameter, which uses the uname application to find the running kernel version. You can type that at the command prompt and replace that parameter with the output of that application, or simply use the Linux "back tick operator", which is the "`" symbol to the left of the "1" (one) key on the top row of the keyboard. An example for this value is "2.6.9-34.0.2.EL"

You should now see the spinlock.h file open in the nano text editor with the cursor on the erroneous line. Change (edit)

rw_lock_t

to

rwlock_t

by removing the underscore character after the "rw" by moving the cursor and using the backspace key. Now that we are done, we can save and exit. Hold ***Ctrl***, and press "**X**", then answer ***y***, and you will be returned to the terminal.

3.0 Get the Zaptel Source Files from Digium

The next step is to download the current Zaptel sources from Digium. As of this writing, the most tested and stable version is Zaptel 1.2.12. You MUST be connected to the internet for this to work!

3.1 Change the working directory to the /usr/src directory with

cd /usr/src

3.2 Get the source files in tar.gz format from Digium's FTP site.

wget http://ftp.digium.com/pub/telephony/zaptel/releases/zaptel-1.2.12.tar.gz

3.3 Once that is done downloading, uncompress the downloaded file using

tar –xzvf zaptel-1.2.12.tar.gz

This will create a directory under /usr/src called zaptel-1.2.12 (/usr/src/zaptel-1.2.12)

4.0 Get the Rhino Driver Source Files from Rhino

You MUST be connected to the internet for this to work! The Rhino source files are divided up into the following categories:

r4fxo – used on the Rev. A and Rev. B boards only (Rev. C and onward, see rcbfx below) **** Note:If your card is a Rev C board or upwards, or if the board has echo cancellation built into it, then use the rcbfx driver in section 4.2b ****
r1t1 – Rhino R1T1 cards, all revisions including PCI Express (section 4.2a)
rxt1 – Rhino R2T1 and R4T1 cards, all revisions including PCI Express (section 4.2a)
rcbfx – all Rhino analog cards, including the R4FXO Rev. C and onward, including all PCI express cards, and all cards with and without on-board Echo Cancellation. See the special section 4.2b below on how to install the rcbfx driver.

4.1 Change the working directory to the /usr/src/zaptel-1.2.12 directory with

cd /usr/src/zaptel-1.2.12

4.2a FOR R4FXO, R1T1 and RXT1 ONLY - The following is the generic command line that you will use to get any of the Rhino driver source code into your zaptel working directory. Please note that you will replace the driver name (r1t1, r4fxo, rxt1) in the line. **For the rcbfx there is a different method – see 4.2b below for that process.**

(Please note that the R4FXO card has two possible driver types – Rev. A and Rev. B cards use this one (r4fxo), BUT the Rev. C and onward (including EC versions) use the RCBFX (rcbfx) driver, please see that section below)

wget ftp://ftp.rhinoequipment.com/R4FXO/Drivers/latest/*

wget ftp://ftp.rhinoequipment.com/R1T1/Drivers/latest/*

wget ftp://ftp.rhinoequipment.com/RXT1/Drivers/latest/*

Please note that you do not need to get drivers for a card that you do not have, just use the one(s) above for the card that you are trying to install.

You will watch the process of pulling each file to your hard drive, there should be a .h file and a .c file, for example rxt1.h and rxt1.c.

4.2b FOR RCBFX - The following is the exact command line that you will use to get the Rhino rcbfx driver source code into your zaptel working directory. **For the r4fxo (for the Rev. A and Rev. B cards only (see above), the rcbfx is the correct version for the Rev. C and onward card, including EC versions), r1t1 and rxt1 there is a different method – see 4.2a above for that process.**

First, download the source files from the Rhino FTP site.

wget ftp://ftp.rhinoequipment/com/RCBFX/Drivers/latest/rcbfx.tbz2

You will watch the process of pulling a single, compressed file to your hard drive. When it is complete, you will get a command line prompt, where we will now decompress the file using the tar utility.

tar –xjvf rcbfx.tbz2

Now execute the following patch command to make the necessary changes to the zaptel Makefile (it needs to be 1.2.12, which we pulled earlier from the Digium site):

patch –p0 < RhinoMakefile-1.2.12.patch

When that is complete, we need to compile the rcbfx driver, which is covered in the next section under 5.1 below – you do not need to make any Makefile edits now, that was done with the patch step.

5.0 Compile the Rhino Drivers (non RCBFX module)

The generic method is to edit the Makefile to let the "make" process know that you want to add the Rhino drivers to the zaptel build flow.

nano +106 Makefile

You should be looking at the line starting with "MODULES:=zaptel". Place the editing cursor one space after "zaptel," and with one space before "tor2", add the names of the modules(s) that you want to include in the zaptel make process. For example, the R4FXO would look like this:

```
MODULES:=zaptel r4fxo tor2 torisa wcusb wcfxo wctdm wctdm24xxp \
        ztdynamic ztd-eth wct1xxp wcte11xp pciradio \
        ztd-loc # ztdummy
```

You can also add multiple names to the modules list, for example to make the R4FXO, R1T1 and RXT1 modules, the result would look like this:

```
MODULES:=zaptel r4fxo r1t1 rxt1 tor2 torisa wcusb wcfxo wctdm wctdm24xxp \
        ztdynamic ztd-eth wct1xxp wcte11xp pciradio \
        ztd-loc # ztdummy
```

Now that we are done, we can save and exit. Hold ***Ctrl***, and press "**X**", then answer ***y***, and you will be returned to the terminal.

5.1 Compiling the Module (all cards, all drivers)

Now the process gets easy since the computer will do the hard work, the compiling. At your terminal, type the following followed by Enter.

make install

The Makefile will now be read and Zaptel will build itself along with the Rhino drivers. This will result in a kernel object module. Look for error messages in the last few lines of the output stream, if there are none, then all went well. This will take the compiled driver and install it into a place where the Linux system will know how to use it, plus make the Rhino driver "modprobe aware".

If there were no errors, the drivers are now ready to use. The next time that zaptel boots, it will look for the Rhino hardware, and load the module if the hardware is found. You can test this by manually trying to load the driver yourself with the modprobe command. For example, to load the r4fxo driver, use the following command:

modprobe r4fxo

Almost any Linux driver can be loaded in this manner if the driver is modprobe aware. One advantage of using modprobe is that other Linux commands can be executed at the same time. For example, in /etc/modprobe.conf, it can be seen that ztcfg is also executed since the AND operator is used (r4fxo && ztcfg).

Another method to load the driver using the inmod command, which will also allow you to pass parameters to the module, if it has any. Remember that if the module is already loaded using modprobe, this step is not necessary.

insmod r4fxo

The module removal tool, rmmod, is also useful when trying to debug module problems. Modprobe (or insmod) and rmmod can be used to load and unload Linux modules at will.

rmmod r4fxo

is the syntax for the module remove command.

6.0 Make Sure That Devices Start

Sometimes with Trixbox devices that are added to the build after the main install have issues starting up. It is best to force udevstart to load and then to force zaptel to start using ztcfg. We will do this in the file named rc.local, which is run after all of the other Linux startup processes are complete.

6.1 Edit the file rc.local using nano:

nano /etc/rc.local

Add the following lines just after the fxotune line.

udevstart
ztcfg –vvv

Now that we are done, we can save and exit. Hold ***Ctrl***, and press "**X**", then answer ***y***, and you will be returned to the terminal.

6.2 Shutdown the machine and install Rhino hardware.

6.3 When CentOS boots again, the Rhino card may now be seen by the hardware manager, it is very important at this time to answer either one of two ways:

Ignore

or

Remove Configuration

Telephony hardware is recognized by CentOS as Network cards, so it is important to not let the OS manage the cards for you as network cards, but rather as zaptel cards. Use the keyboard to navigate to the correct reply and press Enter.

6.4 Reboot the machine to be certain that all hardware loading has been resolved.

7.0 Getting zaptel.conf and zapata.conf Ready

The last step is to get zaptel.conf (/etc/zaptel.conf) and zapata.conf (/etc/asterisk/zapata.conf), which are the two hardware configuration mechanisms that zaptel uses to load channels into zaptel and Asterisk. Use the following command to show how the cards(s) have been loaded, this is important to zaptel in terms of channel identification.

cat /proc/zaptel/*

Important: zaptel.conf uses fxsks and fxols for FXO and FXS line types, respectively. Note that this seems backwards, since a FXO channel needs to be assigned as a FXS channel, and vice versa. Also, zaptel.conf uses an underscore in the definition (fxs_ks) and zapata.conf does not use an underscore (fxsks). AND…zaptel.conf uses the # as a comment, and zapata.conf uses the semicolon for comments.

Open the zaptel.conf file for editing using nano with the following command:

nano /etc/zaptel.conf

Note that this file is simply for letting zaptel know about the very low level operation of the installed Rhino hardware. If you are installing a Rhino digital card, then you will either have one, two or four spans to identify, plus the signaling type.

6.1 Defining Zaptel Digital Card Interfaces (www.voip-info.org is also a great resource)

The first lines of the zaptel configuration file (comments excluded) will be the span definitions for T1/E1 interfaces.

A span definition is in this format:
span=(spannum),(timing),(LBO),(framing),(coding)
(see next page for more examples)
Spannum:
Spannum defines the number assigned to the span these definitions apply to. If you only have one R1T1 card installed you will only have one span (span 1), if there are two single port R1T1 cards installed you will have two spans (span 1 corresponds to the first port on the first module loaded).

Timing:
Use '1' if you want to use the circuit as your primary timing source. If '0' is used Asterisk will try to provide timing to the span (if you were connecting to a Rhino CB24 this would be correct.). If Asterisk is connected directly to the telco you will want to use '1' to accept timing from them. If you have multiple spans; set the timing accordingly such as 2, 3, 4, etc. Note that 4 is the maximum numbers

supported.

0: to not use this span as sync source
1: to use as primary sync source
2: to set as secondary and so forth

LBO:

Line Build Out (LBO) is taken from the table below.
0: 0 db (CSU) / 0-133 feet (DSX-1) (**normal setting**)
1: 133-266 feet (DSX-1)
2: 266-399 feet (DSX-1)
3: 399-533 feet (DSX-1)
4: 533-655 feet (DSX-1)
5: -7.5db (CSU)
6: -15db (CSU)
7: -22.5db (CSU)

Framing:

For T1 - Framing is either d4 or esf. Coding is either ami or b8zs.
For E1 – Framing is either cas or ccs. Coding is either ami or hdb3. E1's spans may also need to enable crc checking.

If you have a voice T1 connecting to the digital port, one possible example would be:

span=1,0,0,esf,b8zs
fxsks = 1-24

If you have an E1 connecting to the digital port, one possible example would be:

span=1,0,0,cas,ami,crc4
(Leave off the ',crc4' if crc checking should not be enabled.)

If you have a T1 PRI connecting to the digital port, one possible example would be:

span=1,0,0,esf,b8zs
bchan = 1-13
dchan = 24

Now that we are done, we can save and exit. Hold ***Ctrl***, and press "**X**", then answer ***y***, and you will be returned to the terminal.

6.2 Defining Zaptel Analog Card Interfaces (www.voip-info.org is also a great resource)

It is easier to define just the analog channels in zaptel since there are no spans, so just the definition of the channel type, fxs or fxo, is needed. This is done by defining the channels (remember to use the opposite type for analog channels, fxs and fxo are oppositely defined).

Here is an example of a four channel FXO and a 16 channel FXS system, we use Kewl (fxsks) and Loop (fxols) to define the signaling types, which is most common for FXO and FXS channels, respectively.

```
; First 4 channels are FXO
fxsks=1-4
; Next 16 channels are FXS
fxols=5-20
```

Now that we are done, we can save and exit. Hold ***Ctrl***, and press "**X**", then answer ***y***, and you will be returned to the terminal.

6.3 NOW TEST: Checking out your typing skills are necessary with zaptel, use the zaptel configuration application ztcfg to see if the entries are correct.

ztcfg –vvv

The –vvv allows for a higher verbosity level to be output to the screen, allowing you to see better any issues that may have arisen, potentially from editing the zaptel.conf file. Should there be an issue, the final output line will print the error message, rather than "X channels configured." Please double check the zaptel.conf file, and then repeat this section.

6.4 Defining Zapata Digital Card Interfaces (www.voip-info.org is also a great resource)

Open the zapata.conf file for editing using nano with the following command:

nano /etc/asterisk/zapata.conf

Editing this file is a bit trickier, since there are rules to the order in which the entries are understood by zaptel. The general rule is, everything that is entered until the channel => command is seen is remembered in memory, and when the channel => is encountered, all parameters prior to this statement are stored in that channel's definition table.

In other words, once you define a parameter in zapata.conf, you do not have to redefine them for each channel if they are not changed. Zaptel remembers the last entries, and only the ones that you want changed will need to be entered.

The best place to add your entries is to the bottom of the file, BUT before the two #include statements, in which FreePBX manages the two files later during your configuration of the Asterisk environment.

To define the channel, there are four minimum items that are needed:

context = allows for equating extensions.conf sections
group = sets the channel grouping, normally 0 for inbound, and 1 for extensions
signalling = sets the technical method to "talk" to the channel (note the TWO l's)
channel => x (or x-y) (note the use of the > symbol here!)

Typically, in Trixbox, there are two types of channels – telco inputs, and extensions, when dealing with digital or analog hardware. We will define the telco inputs as follows:

```
; define a T1 coming in from the telco that uses Loop (Kewl in this case) signaling
context = from-pstn
group = 0
signaling = fxs_ks
channel => 1-24
```

We will define the extensions, assuming a FXS channel connected to an extension, as follows:

```
; define a R24FXS card that uses Loop signaling
context = from-zaptel
group = 1
signaling = fxo_ls
channel => 25-48
```

Now that we are done, we can save and exit. Hold ***Ctrl***, and press "**X**", then answer ***y***, and you will be returned to the terminal.

7.0 Testing and Validating

These are some useful testing ideas.

7.1 Test Zaptel Configuration

Test to be sure that zaptel.conf is correct:

```
ztcfg -vvv
```

and look for errors. Most messages are cryptic at best, and are most likely because of typos. Trial and error helps here.

7.2 Test FXO Telco Channels

You can test the FXO telco channels before you have added them to Asterisk in FreePBX by editing extensions.conf, and adding a little script to the default ([default]) context.

nano /etc/asterisk/extensions.conf

Go to the end of this file, and there will be a few lines that need to be changed to look like the following:

```
[default]
exten => s,1,Answer
exten => s,n,Wait(1)
exten => s,n,Play(Goodbye)
exten => s,n,Hangup
```

When you call into the FXO channel, it will answer and say "Goodbye".

Another good test is to bridge a FXO to FXS, this is an example that will answer the FXO line, and dial (bridge) to a ZAP channel, in this example Zap1.

```
[default]
exten => s,1,Answer
exten => s,n,Wait(1)
exten => s,n,Dial(Zap/1)
```

7.3 Restarting the Asterisk PBX Server

Restart Asterisk and see if zapata.conf and the hardware are operating correctly;

amportal restart

This will restart the Asterisk server and a lot of data will be passed to the screen. If at the end of this data you see "Asterisk Started", then so far, so good.

7.4 Test FXS Extensions

If you have an analog FXS extension, connect a telephone and pick it up, you should hear dial tone. If not, something in zaptel.conf and zapata.conf are not matching, or zapata.conf is not correct. Unfortunately the results of the zapata.conf file are not readily found like that with zaptel.conf and ztcfg.

7.5 See What Asterisk Thinks is Connected

Another good test is to see what Asterisk has in memory for the zaptel channels. Start the Asterisk CLI with

asterisk –Rvvvvvvvvvvvvvvv

to connect to the CLI, and type

zap show channels

this should show a listing of the channels that Asterisk was told about in zaptel.conf and zapata.conf. Use this to find any issues.

To exit the Asterisk CLI, use the ***exit*** command.

7.6 Reboot the System

Finally, don't forget reboot the system, and when it comes back up, test again!

reboot

8.0 *All the Commands in One Place*

This section is provided to just allow you to go through the commands, sort of an expert section. All of the commands here are discussed in more detail in the above sections.

8.1 Login as root!
8.2 Get the latest CentOS kernel headers
yum update –y kernel-devel
8.3 Fix spinlock.h! Replace "rw_lock_t" with "rwlock_t"
nano +407 /lib/modules/`uname –r'/build/include/linux/spinlock.h
8.4 Fix the CentOS swap file issue, make last line 'LABEL=SWAP'
nano /etc/fstab
8.5 Go to the user source directory
cd /usr/src
8.6 Get the latest Digium zaptel 1.2.12 source
wget http://ftp.digium/com/pub/telephony/zaptel/releases/zaptel-1.2.12.tar.gz
tar –xzvf zaptel-1.2.12.tar.gz
8.7 Move to the working zaptel 1.2.12 directory
cd /usr/src/zaptel-1.2.12

8.8a Get the Rhino drivers for the r4fxo (Rev. A and Rev B boards only, Rev. C boards use the rcbfx driver, see below!), r1t1 or rxt1 drivers.

wget ftp://ftp.rhinoequipment.com/R4FXO/Drivers/latest/*
wget ftp://ftp.rhinoequipment.com/R1T1/Drivers/latest/*
wget ftp://ftp.rhinoequipment.com/RXT1/Drivers/latest/*

8.8b for the r1t1, r4fxo (Rev. A and Rev. B boards only!), and rxt1, edit the line containing MODULES:=

nano +106 Makefile

8.9 Get the Rhino RCBFX driver (including R4FXO Rev. C and onward)

wget ftp://ftp.rhinoequipment.com/RCBFX/Drivers/latest/rcbfx.tzb2
tar –xjvf rcbfx.tzb2
patch –p0 <RhinoMakefile-1.2.12.patch

8.10 Compile the Rhino drivers, for all modules

make install

8.11 Edit the file rc.local using nano:

nano /etc/rc.local

Add the following lines just after the fxotune line.

udevstart
ztcfg –vvv

8.12 Power off the machine, install Rhino cards, and turn system back on.
8.13 Answer "Ignore" or "Remove Configuration" to any questions about finding new hardware or configuration changes seen.
8.14 Reboot machine to assure that all hardware installation issues are resolved.

reboot

8.15 Identify the card and channel load order for zaptel and zapata configuration.

cat /proc/zaptel/*

All that is left is to edit the zaptel.conf, and zapata.conf to provide zaptel and Asterisk with the necessary lines to make the Rhino hardware come alive. This is the same process as used with Digium cards, since Rhino cards are Zaptel compliant.

9.0 *Some Useful Linux and Asterisk Commands*

Be sure to login as **root**!

Commands from the Linux command line

Tab key – use this to "auto complete" command line entries, very useful!

uname –r – displays the kernel version

cat /proc/zaptel/* - shows the installed zaptel compatible cards and channel assignments

modinfo rcbfx – shows information about the associated linux driver (rcbfx)

modprobe rcbfx – load a module, in this case the rcbfx module, using the rules found in /etc/modprobe.conf (used to load the module AND perform a ztcfg, for example)

insmod rcbfx.ko – load a module, all by itself. This is particularly useful when a module has command line options, use modinfo to see those

rmmod rcbfx – unload a module, in this case the rcbfx module

lsmod – shows the installed linux modules, sometimes if the list is long using grep to isolate the zaptel associated entries with
lsmod | grep zaptel helps to lessen the screen output

lspci – shows the PCI cards installed

cat /proc/interrupts = shows how the interrupts in a linux system are routed

ztcfg –vvv – forces zaptel to read zaptel.conf, and to associate entries there with the module interface

zttool – show the zaptel compatible cards, their overall status, and if selecting a card, its individual status and signaling states

amportal restart – special method to get Trixbox installations to restart Asterisk

udevstart – restarts the module handling utility, normally this starts on boot

<u>Commands to use from the Asterisk CLI</u>

zap show channels – shows the installed zap channels, this should be the same as what zaptel.conf has

extensions reload – used to load changes made to any dial plan file without having to restart Asterisk

*10.0 **Help Resources***

www.rhinoequipment.com
www.voip-info.org
www.trixbox.org

Email support@rhinoequipment.com

*** END OF DOCUMENT ***