PCI Card Common Installation Issues

Reference Number:	NVX-CP-TBSH-01
Author:	Richard Spencer
Last Updated:	18 May 2008
Document Status:	Issue

Technical Support: This guide is freely available to anyone in the Open Source Community to help solve configuration and installation issues. However, email based technical support is only available to customers that have purchased Open Source telephony equipment directly from Novavox.

Updates: Due to the number of alternative Open Source IP PBX software packages available and new versions being regularly released, it is difficult to keep Open Source documentation fully up to date. If you find any out of date or inaccurate information within this document then please let us know by emailing us at feedback@novavox.co.uk and we will update the document as soon as possible.

Disclaimer: This document has been compiled based on experience resolving real world customer configuration issues as well as information available within the Open Source community. References to source material have not been included as it would make the document less easy to read. Also, keeping the references up to date would be very time consuming. However, if you discover information contained within this document for which you believe you are the original source and you would like to receive acknowledgement, then please let us know and we will add the appropriate reference.

About Novavox

Novavox helps create reliable, affordable, integrated communication solutions to meet the needs of small businesses and medium sized enterprises. An office network connected to the Internet serves as the basis for the convergence of data, voice, video and mobile communications in a secure, integrated architecture. With an integrated communications solution, companies can save on costly phone bills, integrate customer information, and help make employees more mobile without sacrificing communication functionality.

To realise the benefits that integrated communications solutions can offer, a company needs an affordable, reliable solution tailored for the small/medium office and designed for ease of use. Novavox provides Open Source telephony equipment specifically designed for small businesses and medium size enterprises. Using Open Source solutions allows companies to benefit from protection against vendor lock-in and abandonment, a dramatic reduction in capital expenditure costs and an end to per user/feature licensing costs.

Novavox is dedicated to supplying reliable, affordable, feature rich Open Source computer telephony equipment to support businesses upgrading existing systems and those deploying new systems. Working in partnership with Novavox, small businesses and medium sized enterprises can deploy Open Source communication solutions to benefit from immediate return on investment (ROI) through dramatically reduced call charges and significant savings in equipment and infrastructure costs.

Product Range

Novavox supplies a range of alternative Open Source telephony products including:

- X100P Special Edition – The best of breed single port FXO card available
- OpenVox 4 port modular analogue PCI Express cards
- OpenVox 1-4 port ISDN BRI PCI cards
- OpenVox 1-2 port ISDN PRI/E1/T1/J1 PCI cards
- OctWare SoftEcho carrier grade echo cancellation software licences

For full details of our alternative products please visit our websites product section:

www.novavox.co.uk/products

Contact Details

Contact Sales

If you require additional product information or reseller pricing information, have an invoice/order query, or would like to request a quote for a communications solution to meet your company's specific needs then please contact us at sales@novavox.co.uk.

Contact Support

For assistance with equipment installation or for help with a device/service issue please visit our website support section in the first instance:

www.novavox.co.uk/support

Novavox customers only: If after following the guidance provided on the support page you are unable to resolve the issue then please contact us at support@novavox.co.uk.

Contents

1 Introduction

This document describes how to overcome some of the most common issues experienced when installing PCI telephony cards in Open Source IP PBX systems. Setting up an Open Source IP PBX for the first time can be a daunting and sometimes frustrating task, particularly if you are not familiar with Linux. There is online help available in the form of forums and "wikis", and if you encounter a problem someone else has probably had the same problem and found a solution. However, searching through the forums and wikis for the relevant information and finding a solution can be very time consuming.

The main objective of the document is to help make card installation as easy as possible and to minimise the amount of time required. The most common issues are covered in depth and detailed step-by-step solutions are provided along with background information explaining the purpose of each step. The information provided focuses on Digium® Asterisk® and Trixbox® but can be used to help install an Open Source telephony card in any Zaptel/Asterisk based IP PBX system including AsteriskNOW®, Elastix and PBX in a Flash.

2 Card Not Detected by Motherboard/Bios

2.1 Problem Description

If the card you have installed does not appear to be listed in any show command outputs then the first step is to look at the Kernels boot process message buffer. The boot process message buffer can be viewed using the dmesg (diagnostic message) command and will show if the card has been detected by the motherboard/bios:

```
#dmesg | more
```

If the card has successfully been detected, the dmesg output should show:

```
...
wcfxo: DAA mode is 'FCC'
Found a Wildcard FXO: Wildcard X100P
...
```

The above output is for an X100P SE card but similar output should be received for other cards, e.g. "Found a Wildcard TDM: Wildcard TDM400P REV E/F" for 4-port OpenVox analogue cards.

2.2 Solution

If you cannot find the lines for the relevant card in the dmesg output, then:

1. Check that the PCI slot supports the relevant standard, most cards use PCI 2.2
2. Check that card has been inserted correctly, occasionally it helps to pull the card 1-2mm out of the slot so that it does not go right to the bottom
3. Try moving the card to a different PCI slot
4. Try another card (e.g. Network card) in the PCI slot to find out to check that it is not faulty
5. Ensure the Motherboard has the latest BIOS installed

The card must be detected by the motherboard/bios before you can proceed any further. If the card is shown in the dmesg output but is still not listed in some of the relevant show commands then there may be an IRQ issue as described in the following section.

3   PCI Bus IRQ Sharing Issues

3.1   Problem Description

The Peripheral Component Interconnect (PCI) bus provides communication between input/output (I/O) devices and a computer's processor. In order to communicate with the computer processor an I/O device must signal that it wishes to use the bus by sending an interrupt request (IRQ). I/O devices are allocated IRQ bus lines/channels (commonly referred to as IRQs) to allow them to signal an interrupt, which are identified by an index. In the case of Open Source Telephony cards, the line is an actual contact (finger) on the card, and it is a corresponding pin in the sockets on the bus connected by a trace on the motherboard. A card sends an interrupt to the motherboard's interrupt circuitry by changing the voltage level on the interrupt request line. This voltage change acts via interrupt controller circuitry to interrupt the processor to service the card needing the CPU's attention.

If there are more I/O devices than there are spare IRQs then two or more devices can share the same IRQ. For some I/O devices this does not cause any issues. However, the Zaptel driver generates 1000 interrupts a second for each telephony card that it controls. If the interrupts are not handled in time then this can cause significant issues for audio passing through the telephony cards and can even prevent the cards from working altogether. For this reason, it should be made certain that cards that use the Zaptel driver like the X100P SE have their own dedicated IRQ to minimise interrupt handling latency.

When installing a PCI telephony card in an IP PBX for the first time, or moving it to a different slot, you may receive the following error message:

```
Failed to initailize DAA, giving up...
wcfxo: probe of 0000:00:0c.0 failed with error -5
```

To determine if the card has been detected, use the Linux utility lspci which displays information about devices connected to the PCI bus with the "-v" option to check what IRQ has been allocated by the Kernel:

```
#lspci -v
```

If the X100P SE card has been detected something similar to the following lines should be included in the output:

```
04:06.0 Communication controller: Motorola Wildcard X100P
Subsystem: Unknown device b100:0003
        Flags: bus master, medium devsel, latency 32, IRQ 217
        I/O ports at a800
        Memory at dfafe000 (32-bit, non-prefetchable)
        Capabilities: [40] Power Management version 2
```

The next step is to verify that the X100P SE card is recognised by the Linux Kernel and that it is using its allocated IRQ. This can be achieved using the following command:

```
#cat /proc/interrupts
```

If the card is recognised and is being handled correctly by the Zaptel driver then the output should include:

```
  7:    25680730       XT-PIC  wcfxo
```

The above output is from a system using a standard XT Programmable Interrupt Controller (XT-PIC). If a system is using an Advanced Programmable Interrupt Controller (APIC) then the output will look like this instead:

```
  217:   718680730        IO-APIC-level  wcfxo
```

In both cases, the Zaptel driver for the card must be allocated its own dedicated IRQ.

3.2 Solution

If the card doesn't have its own dedicated IRQ (listed in /proc/interrupts) then:

1. Try moving the card to a different PCI slot
2. Check that card has been inserted correctly, occasionally it helps to pull the card 1-2mm out of the slot so that it does not go right to the bottom
3. Make more IRQs available by disabling all unnecessary I/O devices in the BIOS, e.g. serial ports, parallel ports, USB, etc.
4. See if you can assign an IRQ to a specific PCI slot in the BIOS (most modern bioses don't allow this though)
5. Enable/disable the plug 'n play OS option in the BIOS
6. Enable/disable ACPI (Advanced Configuration and Power Interface) in the bios or by adding "acpi=no" in the grub bootloader
7. Enable/disable APIC (Advanced Programmable Interrupt Controller) in the bios or be adding "noapic" in the grub bootloader
8. Enable/disable hyperthreading by adding "noht" in the grub bootloader
9. Enable/disable SMP (Symmetric Multi-Processing) on dual core/process IP PBXs in the bios or by selecting a non-smp Kernel from the grub bootloader
10. Enable/disable the irqbalance daemon

As stated previously, Open Source telephony cards must have their own dedicated IRQ (confirmed via cat /proc/interrupts) to avoid any PCI interrupt related issues. It can sometimes take a while, but using various combinations of the steps listed above should eventually help provide a dedicated IRQ.

4 PCI BUS IRQ Misses/Latency

4.1 Problem Description

As stated in the previous section, the Zaptel driver generates 1000 interrupts a second for each telephony card that it controls. If the interrupts are not handled in time then this can cause significant issues for audio passing through the telephony cards and can cause clicks, pops, and glitches. The most important function of an IP PBX system is to switch/move digital sound samples from one port/protocol to the other at very constant rate. Further, although lost audio samples may be tolerable for voice applications, they may be detrimental to data/fax applications.

If you experience audio issues and have confirmed that the card has its own dedicated IRQ (via cat /proc/interrupts) then there may be a missing interrupt or interrupt latency issue. As previously stated, the Zaptel driver generates 1000 interrupts a second for each telephony card that it controls. One way to check that the correct number of interrupts are being received is to use the following command:

```
#cat /proc/interrupts; sleep 10; cat /proc/interrupts
```

The output should be similar to the output shown below:

```
          CPU0
217:    1961653    IO-APIC-level   wcfxo
...
217:    1971654    IO-APIC-level   wcfxo
...
```

This command displays the number of interrupts for each device connected to the PCI bus, waits 10 seconds and then displays the information again. If we take 1961653 from 1971654 the result is 10001. As the test is not completely accurate 1001 is extremely close to the expected 1000 (i.e. 1000 per second) so we can see that the correct number of interrupts are being received.

If the output shows the number of interrupts as 0 or a very low number then you need to check that the relevant zaptel modules are loaded and that the /etc/zaptel.conf and /etc/asterisk/zapata.conf files are configured correctly.

Another way of testing if there is an issue with missing interrupts is to use the command zztool which runs a timing test:

```
#zttest  (stop with ctrl+c)
```

The output should be similar to:

```
Opened pseudo zap interface, measuring accuracy...
99.995216% 100.000000% 99.995598% 99.997948% 99.996674% 99.997070%
99.996376% 99.995995% 99.999413% 99.996780% 99.996483% 99.996483%
99.996185% 99.997078% 99.996780% 99.996674% 99.996864% 99.995125%
99.999413% 99.995895% 99.997948% 99.996689% 99.996010% 99.997459%
99.997162% 99.996971% 99.996780% 99.995995% 99.997559%
--- Results after 29 passes ---
Best: 100.000 -- Worst: 99.995 -- Average: 99.996918, Difference:
100.003041
```

The higher the percentage the better the sounds quality will be. If the results are 100% or 99% then there definitely shouldn't be any missing IRQ related issues. If the results are 98%

then the card should work OK, however, if the results under 99.975%, then there will probably be some audio problems and the card may not even work.

4.2 Solution

4.2.1 Upgrade to Linux Kernel 2.6

The Linux kernel is responsible for controlling disk access using kernel I/O scheduling. The I/O scheduler found in the 2.4 Linux kernel was designed to maximize global I/O throughput. In doing so a trade-off was made with regards to fairness which means that IP PBX systems using a 2.4 kernel are more likely to suffer from scheduling delays. In order to decide which task should run next, the scheduler had to look at each task and make a computation to determine each task's relative importance. Because the time required to complete the algorithm varied with the number of tasks, applications could suffer from slow scheduling.

The kernel I/O scheduler embedded in the 2.6 kernel has advanced the I/O capabilities of Linux significantly. The scheduler in Linux 2.6 no longer scans all tasks every time. Instead, when a task becomes ready to run it is sorted into position in a queue. When the scheduler runs it selects the task at the most favourable position within the queue. This means that scheduling is done in a constant amount of time, which is ideal for audio applications like IP PBX software. When a task is running it is given a period of time in which it may use the processor before it has to give way to another thread. When its time period has expired the task is moved to a different queue according to its priority. This new I/O scheduler in kernel 2.6 is considerably faster than the old one, and it works equally as well whether there are a large number of tasks in the queue or just a few.

There are actually 4 I/O schedulers embedded in the 2.6 kernel that can be selected at boot time, each of which offers a different combination of optimisations. The alternative schedulers are the Completely Fair Queuing elevator (cfq), Deadline elevator (deadline), NOOP elevator (noop), and Anticipatory elevator (as). The I/O schedulers are called elevator algorithms because they address a problem similar to that of keeping an elevator moving smoothly in a large building. The default scheduler is usually as or cfq depending on the distribution.

In addition to the I/O scheduler improvements, there are a number of other performance enhancing improvements to the 2.6 kernel including better drivers/feature-sets for IDE/SCSI drives, support for kernel pre-emption, and an improved threading model/library. The Linux 2.6 Kernel also include a new process scheduler that scales better in multiprocessor, multicore and hyperthreaded CPU systems than the 2.4 kernel process scheduler.

4.2.2 Disable X Windows and Framebuffer

X Windows generates a large number of interrupts (especially when using an input device such as a mouse) that can delay the processing of telephony card interrupts. It can be useful to run X when configuring the IP PBX system, but once the system up and running it is advisable to disable X Windows by changing the runlevel.

Almost all Linux distribution use runlevel 2/3 for text mode and runlevel 5 for GUI mode. The runlevel modes are defined in the /etc/inittab configuration file. If the IP PBX system is currently using X Windows then:

`#cat /etc/inittab`

should show the following line in the configuration:

`id:5:initdefault:`

To diable X Windows by setting the default runlevel to 3 (text mode) change the line using a text editor (e.g. vi, nano, etc.) to:

```
id:3:initdefault:
```

Next time the IP PBX system is rebooted it will boot in text only mode.

The Linux framebuffer (fbdev) used to display graphics on a console without relying on system specific libraries or X Windows can also cause interrupt processing issues. The framebuffer can be disabled by including the "vga=normal" option in the grub bootloader.

4.2.3   Change IDE Hard Drive Settings

Hard disk controllers can cause audio issues if they consume too much PCI bus resources as they can cause the Open Source telephony card to suffer from interrupt latency/misses. To determine if your IDE hard disk is causing issues you use:

```
#hdparm -t /dev/[IDE Drive]
```

The hdparm –t command causes a substantial amounts of I/O to be generated on the IP PBX system. If audio issues are experience whilst using hdparm –t then this signifies that there is a hard drive issue.

When using IDE drives, Digium recommends using Direct Memory Access (DMA) mode with an Ultra DMA (UDMA) setting of 2 or 3. UDMA mode 2 is ATA33 and UDMA mode 3 is ATA44. To set the hard drive to UDMA mode 2 for example use:

```
# hdparm -d 1 -X udma2 -c 3 /dev/[IDE Device]
```

In the command above the option "–d 1" enables DMA mode, "-X udma2" sets the IDE transfer mode to UDMA mode 2 and "-c 3" enables 32-bit data transfers. It should be noted that any changes to hard drive settings will only be used until the IP PBX is rebooted. Therefore, to ensure the hard drive settings remain when the IP PBX is rebooted the hdparm command needs to be included in one of the IP PBX startup scripts.

4.2.4   Assign IRQs to Specific Processors

Dual core/processor IP PBX systems running a Symmetric Multi-Processing (SMP) kernel allow IRQs to be bound to single or multiple CPUs. It is advantageous to bind the X100P SE card IRQ to a separate processor used for other IRQs, particularly IRQs allocated to thing like IDE drives. Using a separate CPU for the X100P SE card reduces interrupt latency and also improves cache coherency which reduces cache misses.

Before binding the X100P SE card IRQ to a specific processor first of all the irqbalance daemon should be disabled otherwise it will override the binding. This can be achieved using the following commands:

```
#service stop irqbalance
#chkconfig irqbalance --level 345 off
```

To bind the IRQs to a particular processor first of all check which IRQ has been assigned to each device using:

```
#cat /proc/interrupts

           CPU0
    0:   29421608    IO-APIC-edge    timer
    1:         39    IO-APIC-edge    i8042
    8:          3    IO-APIC-edge    rtc
    9:          0    IO-APIC-level   acpi
   14:     287820    IO-APIC-edge    ide0
```

```
205:       12333   IO-APIC-level   eth0
209:        8001   IO-APIC-level   ide1
217:    29406181   IO-APIC-level   wcfxo
```

Make a note of the IRQs for each peripheral device, any disk drives and your telephony card. Next change the assignments using the following commands:

```
#echo 1 > /proc/irq/217/smp_affinity #wcfxo
#echo 2 > /proc/irq/14/smp_affinity #ide0
#echo 2 > /proc/irq/205/smp_affinity #eth0
#echo 2 > /proc/irq/209/smp_affinity #ide1
```

Using the commands above the telephony card IRQ (in this case an X100P SE) has been assigned to processor 1, whilst all other relevant IRQs have been assigned to processor 2.

Once the IRQ binding configuration has been completed, each of the echo commands used should be added to the /etc/rc.d/rc.local file to ensure that the IRQ bindings will remain when the IP BPX server reboots, i.e.

```
#cat /etc/rc.d/rc.local
```

Should include:

```
echo 1 > /proc/irq/217/smp_affinity #wcfxo
echo 2 > /proc/irq/14/smp_affinity #ide0
echo 2 > /proc/irq/205/smp_affinity #eth0
echo 2 > /proc/irq/209/smp_affinity #ide1
```

Please note: If PCI Bus devices are removed or added to the system then the IRQ assigned may change. Therefore the steps above should be repeated after removing/adding PCI Bus devices to ensure the IRQs remain assigned to the most appropriate processor.

4.2.5   Change IRQ Priorities

If the IP PBX CPU gets two interrupts at the same time, it will answer them in order of priority based on the IRQ of the device requesting the interrupt. Therefore one way to reduce interrupt latency/misses is to use an IRQ with a higher priority.

IRQs are prioritized and serviced in priority order by the CPU as determined by the controller. When a standard XT Programmable Interrupt Controller (XT-PIC) is being used IRQs 8 through 15 have a higher priority than IRQs 3 through 7. The IRQ priority order is shown below with priority decreasing from left to right (0 has the highest priority, 7 the lowest):

[0 1 2 8 9 10 11 12 13 14 15 3 4 5 6 7]

It may be possible to assign a card an IRQ with a higher priority by moving it to a different slot or by trying some of the other suggestions for changing IRQ assignments listed in Section 3.2.

If an Advanced Programmable Interrupt Controller (APIC) is being used then interrupt priority isn't related with the pin, but the vector of the pin. How the vector of a pin is allocated is quite random. Usually the device driver that communicates with the PCI controller first will get a lower priority vector. Therefore, which vector gets allocated to a device depends on how many device drivers have already contacted the PCI controller. There is no method available for reserving/setting vectors for a device/IRQ and no way of checking what vector has been assigned to what IRQ. Therefore if IAPIC is being used it is not possible to try and assign a card with a higher priority.

4.2.6 Change IRQ Latency Timer Settings

Even if you manage to assign an IRQ with a higher priority to your card, a lower-priority interrupt can still block higher-priority interrupts if it takes over the processor for several milliseconds and disables all other interrupts during that time. Another way to try and reduce interrupt latency/misses is to change the default IRQ latency timers. Interrupt latency is the time between the generation of an interrupt by a device and the servicing of the device which generated the interrupt. There is usually a tradeoff between interrupt latency, throughput, and processor utilization.

PCI bus latency timers can range from 0 to 248, the lower the figure the quicker the device will give up the bus. If a device has a setting of 0, then it will immediately give up the bus if another device needs to transmit. If a device has a setting of 248, it will continue to use the bus for a longer period of time before stopping whilst the other device waits for its turn. It is the role of the device driver to set the correct PCI bus latency timer value for the PCI device and most of the time the default settings work well.

Using low PCI bus latency timer settings results in devices quickly giving up the bus if another device needs to transmit data. The advantage of this approach is that it results in a much lower data transmit latency, since no device is going to hold on to the bus for an extended period of time causing other devices to wait. However, this also reduces the effective PCI bus because large data bursts become much less frequent and rapid changes in bus control increases overhead. Configuring the majority of devices with high PCI bus latency timer settings increases interrupt latency, but also increase throughput. As each device gets to burst large amounts of data across the bus without interruption, the PCI bus is used more efficiently and the PCI devices can transmit more data.

In general, the best approach to reducing interrupt latency is to assign a higher PCI bus latency timer to the Open Source telephony card and assign a lower PCI bus timer to the other PCI bus devices. This helps to ensure that the other PCI bus devices give up the PCI bus when the Open Source card needs it.

However, it should be noted that if an IP PBX is being used to provide multiple services that require large amount of data to be sent/processed, e.g. file/web server, firewall, etc. then lowering the PCI bus timer for things like the IDE drive may actually decrease performance. In such cases, CPU utilisation may increase significantly due the processor not having enough time to send the volume of data required and due to the frequency of PCI bus control changes. Therefore, in some cases it may be more beneficial to actually increase the PCI latency of some of the other devices in order to allow them to burst relatively large amounts of data across the bus in one go. Each system is different and therefore it is a case of trial and error in finding the optimum PCI bus latency settings.

Before changing the PCI Bus latency timer settings first of all we need to find out the specified bus, slot and function information for each PCI bus device we want to change the latency for. This information is provided in the [[<bus>]:][<slot>][.[<func>]] format next to each PCI device when issuing the lcpci –v command. For example:

```
#lspci –v

00:0f.0 IDE interface: VIA Technologies, Inc. Unknown device 5337
(rev 80) (prog-if 8f [Master SecP SecO PriP PriO])
        Subsystem: Foxconn International, Inc. Unknown device 0c87
        Flags: bus master, medium devsel, latency 32, IRQ 11
        Capabilities: [c0] Power Management version 2

04:04.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL-
8139/8139C/8139C+ (rev 10)
        Subsystem: Foxconn International, Inc. Unknown device 0c87
```

```
        Flags: bus master, medium devsel, latency 32, IRQ 209
        I/O ports at 9c00 [size=256]
        Memory at dfaff000 (32-bit, non-prefetchable) [size=256]
        Capabilities: [50] Power Management version 2

04:06.0 Communication controller: Motorola Wildcard X100P
        Subsystem: Efar Microsystems Unknown device 0001
        Flags: bus master, medium devsel, latency 32, IRQ 217
        I/O ports at 9800 [size=256]
        Memory at dfafe000 (32-bit, non-prefetchable) [size=4K]
        Capabilities: [40] Power Management version 2
```

After making a note of the relevant bus, slot and function details for each device the PCI latency timers values can be changed using the following commands:

```
#setpci -v -s 04:06.0 LATENCY_TIMER=f8 #wcfxo
#setpci -v -s 00:0f.0 LATENCY_TIMER=8 #ide0
#setpci -v -s 04:04.0 LATENCY_TIMER=8 #eth0
#setpci -v -s 00:0f.0 LATENCY_TIMER=8 #ide0
```

Using the commands above the PCI latency timer for the Open Source telephony card (in this case an X100P SE) has been increased to the maximum of 248 (F8 in hex), whilst the PCI latency timers for the other PCI devices have been reduced to 8.

After the PCI latency timers have been configured using the commands above, each of the setpci commands used should be added to the /etc/rc.d/rc.local file to ensure that the PCI latency timers will remain when the IP BPX server reboots, i.e.

```
#cat /etc/rc.d/rc.local
```

Should include:

```
setpci -v -s 04:06.0 LATENCY_TIMER=f8 #wcfxo
setpci -v -s 00:0f.0 LATENCY_TIMER=8 #ide0
setpci -v -s 04:04.0 LATENCY_TIMER=8 #eth0
setpci -v -s 00:0f.0 LATENCY_TIMER=8 #ide0
```

Please note: If PCI Bus devices are removed or added to the system then the relevant bus, slot and function details may change. Therefore the steps above should be repeated after removing/adding PCI Bus devices to ensure the PCI latency timer values remain optimal.

5 Echo

5.1 Problem Description

Echo is the term used to describe the experience of hearing your own voice back during a telephone call. There are two main sources of echo in IP PBX systems:

- **Acoustic echo** – Generally caused by feedback from the telephone handset
- **Line echo** - Dependant on the impedance characteristics of the line to the exchange

The two main factors that determine how serious echo is are the time period after which you hear the echo and the loudness of the echo. If the echo is fairly quite and there are only a few milliseconds of delay then the echo may be quite bearable. However, if the delay is a few hundred milliseconds and the echo is quite loud then it can be extremely annoying and even render the voice service unusable.

5.1.1 Acoustic Echo

Acoustic echo is usually caused by voice travelling out of the earpiece or speaker back into the mouthpiece or microphone, although it can also be caused by sound bouncing off the walls of a room. Acoustic echo is only usually an issue when using low quality or incompatible headsets, or when using a speakerphone. However, it can also occur when using a telephone handset if the microphone is sensitive and the volume of the handset speaker is quite loud. When suffering from acoustic echo both the local and remote parties hear echo when speaking and echo is present when calling another extension connected to the same IP PBX. Acoustic echo varies in strength and delay depending on the changing acoustic environment of the echo source, e.g. moving a microphone further away from a headset.

5.1.2 Line Echo

Line echo is common in PSTN networks and is caused by impedance mismatches. Line echo is a property of the line connection and remains mostly constant throughout the call. When making a call over the PSTN, the signal path between two telephones requires amplification using a four-wire circuit. Four wire circuits are not extended to customer premises due to cost, instead the four-wire circuits are coupled to two-wire circuits using a device called a hybrid. The hybrid in the telephone providers exchange couples the analogue signal from the four-wire circuit (where incoming and outgoing signals are separated) to the two-wire circuit where the incoming and outgoing signals are combined.

As signals pass from the four-wire to the two-wire portion of the network, the energy in the four-wire section is reflected back, creating an echo of the signal. The severity of the echo depends on how well the impedance is matched between both sides of the hybrid. The impedance of the two-wire circuit can vary significantly depending on factors including the line set-up in the exchange equipment, the distance between the exchange and the IP PBX, the electrical characteristics of the wire, etc. The actual amount of signal that is reflected back depends on how well the balance circuit of the hybrid matches the two-wire line.

Line echo isn't an issue when using a standard analogue POTS telephone as the reflected voice is heard back at the same time as you are speaking. If the delay is under 5ms it merges with the side tone, which is the sound of your own voice in your earpiece and becomes unnoticeable. However, when using a telephone connected to an IP PBX the voice is delayed slightly due to the speech encoding and digital processing, which results in echo. The hybrid device in the providers exchange is supposed to subtract the correct proportion of the signal transmitted in order to eliminate the echo from the local loop-back portion of the circuit. Unfortunately though poorly designed hybrid devices are often used and it's very common for impedance settings to be improperly balanced.

It should be noted that it is possible for some calls made using a PSTN line to suffer from echo problems whilst other calls made over the same line remain echo free. For example, long-distance calls over 600km are routinely echo cancelled at each end and mobile phone calls to the PSTN are also echo cancelled so these types of call will not suffer from echo issues.

In addition to echo caused by hybrid balancing issues in the exchange, poor quality in building cabling and splitters can also generate echo due to bounce back caused by impedance variations.

5.2   Solution

5.2.1   Check for Acoustic Echo

When solving echo issues the first step is to make sure that the source of the echo is not acoustic echo being caused by a telephone handset/softphone. Methods to determine whether or not the issue is due to acoustic echo are:

- Does the remote party hear echo or just the local party? If its both parties the echo must be acoustic echo.
- Try making an internal call between extensions connected to the same IP PBX or a call over a SIP trunk, if the echo is still present it must be acoustic echo.
- Hold your hand over the mouthpiece/microphone during a call to see if the echo is still present.

If it is acoustic echo there are a number of relatively simple solutions available:

- Try turning down the volume on the handset/speakerphone, most acoustic echo problems can be greatly reduced by simply turning the volume down.
- Try moving the speakerphone to a room that is less likely to create echo, e.g. a carpeted room with soft furniture that can absorb the sound rather than reflect it back.
- Try using a different handset/headset/speakerphone.

5.2.2   Check the Zaptel Module Operating Mode

If the echo can only be heard when making calls using a PSTN line then the echo must be line echo. The primary object in dealing with line echo is to try and prevent the echo from occurring in the first place. All Open Source telephony cards sold by Novavox have tuneable parameters that affect the operating characteristics and have an impact on voice quality and echo. Some of the parameters are country dependant, and therefore the zaptel driver module for the card needs to know which country parameter to use.

To set the correct operating mode the appropriate option needs to be added to the /etc/modprobe.conf file.

Note: Debian based Linux distributions like Ubuntu do not have a /etc/modprobe.conf file. Instead modules can be loaded/unloaded by creating a configuration file with the relevant options and storing it in the /etc/modprobe.d/ directory. For example, you could create a file called modprobe.conf in /etc/modprobe.d/ and add the lines described above. After creating the file run `update-modules` to merge the changes with the system modules configuration.

The options are Open Source telephony card specific, two examples are provided below:

X100P SE cards

The default operating mode for the Zaptel wcfxo driver is FCC/US line standards. To change the operating mode to CTR21 to support European line standards add the following lines to modprobe.conf:

```
options wcfxo opermode=1
install wcfxo /sbin/modprobe --ignore-install wcfxo && /sbin/ztcfg
```

Then shutdown and restart the IP PBX:

```
#shutdown -r now
```

To check that the correct mode is in operation use the following:

```
#dmesg | more
```

The output should include:

```
    ...
    wcfxo: DAA mode is 'CTR21'
    Found a Wildcard FXO: Wildcard X100P
    ...
```

This confirms that the card is running in European CTR21 mode.

Please note: The Zaptel wcfxo driver only supports CTR21 mode with 600 Ω AC termination, which may or may not be the correct setting depending on the country and the phone system in use. For example, in the UK BT PSTN lines use complex impedance, whereas PSTN lines from some cable companies may use 600 Ω impedance. Also, the Silicon labs Si3014/Si3034 DAA chip used on the X100P Special Edition (SE) sold by Novavox supports global line standards. However, the Si3012/Si3035 DAA chip used in the original Digium X100P card and low-cost X100P clone cards only supports FCC/JATE line standards. Therefore, unless you have an authentic X100P SE card it is unlikely that your X100P card will support European CTR21 mode.

To support global line standards we have put together a setup guide that uses a patch to provide additional operational modes to support line standard for all countries. If your country is not supported by the default FCC mode or CTR21 600 Ω impedance mode then use the instructions in the following guide to configure the appropriate settings for your country:

Novavox X100P SE Global Line Standards Setup Guide

OpenVox A400P cards

The default operating mode for the Zaptel wctdm driver is FCC US line standards. To change the operating mode to support the line standards for a different country add the following lines to modprobe.conf:

```
...
options wctdm opermode=<country_code>
install wctdm /sbin/modprobe --ignore-install wctdm && /sbin/ztcfg
...
```

For example, the country code for the UK is "UK". A list of country codes for other countries is provided in Appendix B. Please be aware that not all country codes are supported.

After configuring the correct operating mode for your country shutdown and restart the IP PBX:

```
#shutdown -r now
```

To check that the correct mode is in operation use the following:

```
#dmesg | more
```

The output (using the UK operating mode as an example) should include:

```
Module 0: Installed -- AUTO FXO (UK mode)
Module 1: Installed -- AUTO FXO (UK mode)
Module 2: Installed -- AUTO FXS/DPO
Module 3: Installed -- AUTO FXS/DPO
```

This output is for an OpenVox A400P22 with 2 FXS and FXO modules, the output will vary depending on the modules installed.

5.2.3   Balancing the Hybrid

Some Open Source telephony cards such as the OpenVox analogue card range sold by Novavox have a programmable digital hybrid for near-end echo reduction. Fxotune is a tool used to auto-configure the PSTN line impedance settings for cards that have a programmable digital hybrid. Before using the fxotune tool, first of all ensure that the FXO ports are connected to the PSTN and then stop Asterisk (otherwise fxotune considers the lines as busy):

```
#amportal stop / CLI>stop now
```

Next start the fxotune tool:

```
#fxotune -i 5
```

The number "5" is a dialled digit used to break dialtone on the line, any number that will break dial tone on the line may be used. The "-i" option instructs fxotune to test the PSTN line impedance for each connected PSTN line (may take up to 10mins per line) and set the corresponding parameters in the fxotune.conf file (normally located in /etc/). Once complete, apply the calibration settings defined in the fxotune.conf file using:

```
#fxotune –s
```

Before finally restarting Asterisk:

```
#amportal start / #asterisk
```

Every time a new PSTN trunk line is added or changed "fxotune -i 5" should be run.

The latest Zaptel driver versions include the command "fxotune -s" in the Zaptel init script (usually located in /etc/init.d/) to automatically load the fxotune calibration settings when the system is reloaded. If you are using an older Trixbox version like 2.2 then the "fxotune -s" command is included in /etc/rc.d/rc.local script.

To check whether or not the "fxotune -i" command is already included in one of your IP PBX scripts open up the relevant scripts using a text editor like nano and search the text for "fxotune". If the command is not included then add "/usr/sbin/fxotune –s" to the /etc/rc.d/rc.local script to ensure that the fxotune calibration settings are automatically reloaded following a reboot.

5.2.4   Adjusting the receive/transmit gain

One of the factors that can have an impact on echo is the setting of correct RX/TX audio gain values for your telephone line in the /etc/asterisk/zapata.conf file. Unfortunately there is no such thing as a standard RX/TX gain setting that will work for all installations as every installation is different due to differences in telephone line quality, impendence, line length,

etc. However, Asterisk provides a useful tool called ztmonitor to help set the appropriate RX/TX gain values. Using the command:

```
#ztmonitor -v
```

Will provide you with the following output:

```
Visual Audio Levels.
-------------------
 Use zapata.conf file to adjust the gains if needed.

( # = Audio Level  * = Max Audio Hit )
<---------------(RX <---------------(TX
 ################# *                   ################# *
```

To use the ztmonitor tool make a telephone call and observe the RX/TX levels whilst you are speaking. The RX/TX bars should both ideally be about 50%, the peaks should not be too high or too low.

If the bars are too high/low then you can adjust the RX/TX gain settings in the zapata.conf file:

```
rxgain=0.0
txgain=0.0
```

The TX/RX values that can be configured range from -100 to 100 (-100% to 100% of capacity). However, it is generally recommended not to put the values to more than -11 to +11 and in some case anything outside of the -5 to +5 range may cause audio issues.

During a call, if you need to adjust the RX/TX settings, make the changes to the zapata.conf file and save it, then switch to the Asterisk CLI and type the following:

```
CLI> module reload chan_zap.so
```

This will reload the zap module without hanging up the call. After configuring the correct RX/TX values and hanging up the call stop and restart Asterisk to ensure the new settings take effect.

```
#amportal stop / CLI>stop now
#amportal start / #asterisk
```

The RX/TX gains will vary depending on the location of the called/calling party. Therefore, it is advisable when using ztmonitor to call different numbers in different locations to achieve the optimal settings.

5.2.5   In building Cabling/Splitters

Amongst other things, poor quality in building cabling, long cable spans, splitters can create echo due to bounce back caused by impedance variations. If you think the quality of the in building cabling/splitters could be a cause of echo, then try connecting the port on the IP PBX PSTN line card directly to the exchange NTE box using a good quality modem cable. If that fixes the issue then follow the cabling route and carry out section tests to try and locate the source of the echo.

5.2.6   IP PBX Voice Switching Delay

As stated in section 4, the most important function of an IP PBX system is to switch/move digital sound samples from one port/protocol to the other at very constant rate. If the sound samples are delayed as they traverse the IP PBX then this can cause echo. The delay introduced by an IP PBX may be due to speech encoding, digital processing, or both.

If it is suspected that the IP PBX itself could be the cause of the echo the first step should be to check for and resolve any IRQ issues using the methods described in Section 4.

If there does not appear to be an IRQ issue, then the general performance of the IP PBX should be assessed. System information provided by an IP PBX GUI (e.g. The System Status screen in the Trixbox GUI) can provide helpful information on system performance, i.e. things like memory usage, network usage and file system capacity. Command line tools that are useful include:

`#vmstat` - Provides information on CPU/memory usage as well system I/O information
`#uptime` - CPU load information based on 5, 10, 15 minute averages
`#ps aux` – Static snapshot overview of system processes
`#top` – Dynamic overview of system processes updated every 10 seconds

Solving IP PBX server performance issues is outside the scope of this document. However, the commands listed above should act as a good starting point to help determine if there is a performance issue and what the cause may be to allow a solution to be found, e.g. install extra memory or remove unnecessary applications/processes.

5.2.7 Echo Cancellation

Background

The suggestions above are all intended to help stop the echo being created in the first place. If these do not work then the solution is to use echo cancellation to remove echo from the voice communication. Echo cancellation first of all requires recognizing the originally transmitted signal that after a delay re-appears in the transmitted or received signal. To recognise the echo return level (ERL) needs to be lower than the output signal level by a certain amount in order for it to interpret the signal as echo. Once the echo canceller recognises the echo it can be removed by subtracting it from the transmitted or received signal.

Echo cancellation can be implemented in hardware using a dedicated digital signal processor (DSP), or in software. It is often assumed that hardware based echo cancellation is superior to software based echo cancellation. However, a hardware based echo canceller is just an echo cancellation algorithm implemented in firmware on a DSP chip. There is no inherent reason why the same algorithm couldn't work equally as well in software on a CPU. Saying that, software based echo cancellation does have an impact on CPU usage and its use is not feasible for large numbers of channels, e.g. multiple E1/T1s.

One of the factors that determines how much processing resources a software based echo canceller requires is the tail circuit delay. The tail circuit encompasses everything from the Open Source telephony card PSTN port all the way down to the terminating phone. Tail circuit delay is the time period between the point at which the original audio signal exists the echo canceller and the time at which it returns as echo. The echo canceller needs to keep a record of the audio it has sent for a time period equal to the tail circuit delay to determine if the same audio has come back. To keep a record of the audio it takes digital sound samples at set intervals called taps.

The Zaptel/Asterisk echo cancellers take digital samples at a rate of 8,000 per second, which means the interval between each tap is 0.125ms. The default number of taps in Asterisk is 128, which will handle echo paths of up to 16ms and is usually adequate for most echo issues. The number of taps is always configured as a power of two, i.e. 32, 64, 128, or 256. The lowest number of taps as possible should be used while still adequately cancelling the echo as it reduces the computing load and memory requirement on the IP PBX. Also, the training time is shorter and the canceller will adapt faster.

Zapata.conf Echo Configuration

An example Zapata.conf configuration (usually located in /etc/asterisk/) for enabling echo cancellation is provided below:

```
echocancel=yes
echocancelwhenbridged=no
echotraining=400
```

echocancel

This parameter disables [no] or enables [yes] echo cancellation. It is recommended to leave it enabled even of there doesn't appear to be an echo issue unless there is a reason not to, e.g. lack of CPU resource. The [yes] setting sets the number of taps to 128, but an integer can be used instead to change the number of taps to 16, 32, 64, 128, or 256.

echocancelwhenbridged

This parameters enables [yes] or disables [no] echo cancellation when TDM calls are bridged. Generally its not necessary or advisable to enable echo cancellations for bridged TDM calls but it can be enabled if required.

echotraining

The echo training option in Asterisk is a mechanism for improving the convergence time of the echo canceller. Transmitted voice is disabled for a short time period during ringing and a spike of sound is transmitted to measure the difference in the received echo and transmitted signal directly instead of learning it iteratively over many samples. The echo training option eliminates most of the echo at the beginning of the call in the majority of cases. Valid settings for echotraining are 10 – 2000 millseconds which specifies the delay before training.

Please note: echotraining is not supported by the Open Source Line Echo Canceller (OSLEC) and must be commented out otherwise it will cause the channel to be silent. OSLEC is installed by default in Trixbox versions 2.4 and upwards and offers better echo cancellation performance than the zaptel echo cancellers.

Removing Echo

The echocancel and echotraining options may need to be fine-tuned in order to get rid of echo completely. Different parameters should be used on an iterative trial and error basis until the optimum echocancel and echotraining settings are determined for an IP PBX system.

It is advisable to start with an echocancel setting of 32 and a echotraining setting of 100. Increment the echotraining in steps of 100 but do not exceed an echotraining setting of 1200. Asterisk must be restarted each time one of the parameters is modified in order for the changes to take effect. If echo is still present after reaching an echotraining setting of 1200, increment the echocancel option to the next setting (64) and repeat the process until the echo is completely gone.

To speed up the procedure you can increase the settings using larger intervals (e.g. increase the echotraining delay by 400 each time) and can even start at the highest settings and work back if the echo is extremely bad (e.g. 256 taps, 1200millsiceonds training).

Changing the Echo Canceller

If there is still echo present after following the procedure described above then it is possible to try using a different echo canceller. One option is to try the OSLEC echo canceller, instructions on installing the OSLEC echo canceller can be found here:

http://www.rowetel.com/ucasterisk/oslec.html#install

Another option is to try a different Zaptel echo canceller. In older Zaptel versions the default echo canceller is ECHO_CAN_KB1. A newer echo canceller is ECHO_CAN_MG2, which is the default for newer Zaptel versions. It also possible to use the AGGRESSIVE_SUPPRESSOR in conjunction with ECHO_CAN_MG2 for aggressive residual echo suppression.

To change the default echo canceller you need to open up the zconfig.h file (located in /user/src/zaptel or /user/src/zaptel/kernel) using a text editor like nano and comment in/out the required echo canceller. For example, if your default echo canceller was ECHO_CAN_KB1 and you wanted to change to ECHO_CAN_MG2 with AGGRESSIVE_SUPPRESSOR you would need to edit your zconfig.h file so it shows:

```
/*
 * Pick your echo canceller: MARK2, MARK3, STEVE, or STEVE2 :)
 *
 */
/* #define ECHO_CAN_STEVE */
/* #define ECHO_CAN_STEVE2 */
/* #define ECHO_CAN_KB1 */
/* This is the new latest and greatest */
#define ECHO_CAN_MG2

….
 * Uncomment for aggressive residual echo suppression under
 * MARK2, KB1, and MG2 echo canceler
 */
#define AGGRESSIVE_SUPPRESSOR
```

After changing the file save it and then recompile Asterisk. Instructions on downloading and compiling the Zaptel source code it can be found in Appendix D.

OctWare SoftEcho

An alternative to spending time tweaking zapata.conf settings changing rx/tx gains is to simply install OctWare SoftEcho. The SoftEcho algorithm is auto-tuning meaning that no adjustments are necessary making integration quick and simple. SoftEcho provides carrier grade voice quality for Asterisk IP PBX voice channels, key features include:

- Excellent double-talk handling
- Fast convergence
- High quality background noise handling
- Long echo tail up to 128 ms
- Performance and voice quality statistics
- Auto-tuning transparent algorithm
- Dial tone and digit transparency
- G.168-2004 compliant

Leveraging Octasic's sound processing expertise based on years of experience in trunk side as well as local side network echo cancellation, OctWare have developed a software based echo cancelling algorithm called SoftEcho. With support for up to 16 voice channels, SoftEcho is a cost-effective way for Asterisk® IP PBX integrators to offer carrier-grade voice quality. SoftEcho effectively eliminates customers' support calls about unsatisfactory voice quality. SoftEcho can be used with any Open Source telephony cards including analogue cards like the X100P SE and digital cards like the OpenVox B100P.

OctWare SoftEcho licences are available directly from Novavox:

http://www.novavox.co.uk/products/software/softecho.html

6 Far end Disconnect

6.1 Problem Description

Far end disconnect is the term used to describe the situation when the far end hangs up during a telephone call, i.e. the remote end from the perspective of the IP PBX. If a local user is on the phone when the far end disconnects the user will know the call has ended and will hang up the phone at local end. However, even though the local user hangs up the IP PBX may not release the PSTN line if it thinks that the far end is still connected. Also, if someone is leaving a voicemail message then the IP PBX needs to know when the far end has hung up otherwise it will record silence or dial/busy tone.

The method by which the IP PBX learns that the far end has disconnected is known as disconnect supervision signalling. The most reliable form of disconnect supervision is where the local exchange drops battery/voltage momentarily on the phone line to indicate to the phone/PBX that the other end of the party has disconnected the call, i.e. to provide hang-up notification. This voltage drop is known as CPC (Calling Party Control) or Disconnect Clear Time (DCT) in the UK. This type of disconnect has various names in different countries including forward disconnect, battery removal, open loop disconnect and clear forward answer reversal.

Another type of disconnect supervision signal used in some countries is reverse polarity, also called reverse battery. When reverse polarity disconnect supervision is being used the PSTN provider reverses the polarity of the battery current for both answer supervision and disconnect supervision.

The least reliable method of disconnect supervision is the use of disconnect/busy tone. When the far end disconnects the IP PBX receives a disconnect/busy tone to indicate that the call is disconnected.

Far end disconnect issues arise when the far end disconnects but the IP PBX does not receive or misinterprets the disconnection supervision signal.

6.2 Solution

The first thing to check when experiencing far end disconnect issues that the correct signalling type is configured in the zaptel.conf (e.g. **fxsks=1**) and zapata.conf (e.g. **signalling=fxs_ks**) configuration files. Kewlstart is the preferred signalling protocol for analogue circuits in Asterisk and is the signalling protocol that should be used in the UK and the majority of European countries. A summary of the different signalling methods is provided in Appendix A.

If your PSTN provider uses voltage/battery drop for disconnect supervision but far end disconnect is not working then the CPC/DCT provided by the PSTN provider may be set too low. For example in the UK the ideal DCT is 800ms, however, it may be set as low as 100ms and may not be detected by your IP PBX. To solve this issue contact your PSTN provider and ask them to increase the CPC/DCT for you.

If you PSTN provider provides reverse polarity/battery disconnect supervision then the following two lines should be included in the zapata.conf file:

```
hanguponpolarityswitch=yes
answeruponpolarityswitch=yes
.
```

For lines that rely on disconnect/busy tone disconnect supervision the following lines should be included in the zapat.conf file:

```
busydetect=yes
busycount=6
```

With busydetect enabled Asterisk will listen for busy signals on the line, if your PSTN provider sends you a busy tone following far end hang-up Asterisk should detect it. It should be noted that this option is less reliable than other hang-up detection methods and as such should only be used if it's the only option available. Busycount is the number of busy tones to detect before recognising that the far end has disconnected. The lower the value the quicker the call will hang-up, but there is a tradeoff t consider as this increases the possibility of false hangups.

You can also try enabling `callprogress`, and set your `progzone` to your country code (see appendix B). This option is similar to the busydetect option, but will also listen for a ringing tone, congestion tone and will try to detect if the line got answered. However, callprogress is experimental, so if you experience frequent disconnects, you should disable it.

7 Kernel Panic on Reboot

7.1 Problem Description

Sometimes after installing a new Open Source telephony card you may get a Kernel panic error similar to the one shown below when you reboot the IP PBX:

```
Code: Bad EIP value
<0>Fatal exception: panic in 5 seconds
Kernel Panic - not syncing: Fatal exception
```

The issue effects some IP PBX builds but not others and is due to the Zaptel modules being unloaded before Asterisk has been stopped when rebooting.

7.2 Solution

To solve the issue we need to stop the zaptel modules from being unloaded before Asterisk is stopped when rebooting. This can be achieved by commenting out the section that unloads the drivers in the zaptel stop script located in the /etc/rc6.d directory, i.e. the directory that contains the scripts that are executed when the system is rebooted.

The name of the relevant script is Kxxzaptel, where xx is a number between 00-99 (the number corresponds to the order in which the script is executed). In our Trixbox 2.6.07 test system the relevant script is /etc/rc6.d/K92zaptel and the relevant lines that need to commented out (by placing a '#' at the start of each line) are shown below (lines 228 – 233 and line 238) as shown below:

```
 stop)
     #Unload drivers
     #shutdown_dynamic # FIXME: needs test from someone with dynamic
spans
     #echo -n "Unloading zaptel hardware drivers:"
     #unload_module zaptel
     #RETVAL=$?
     #echo "."

     #[ $RETVAL -eq 0 ] && rm -f $LOCKFILE
     ;;
unload)
     # We don't have zaptel helper, so let's not replicate too much
code:
     # allow others to use the unload command.
     #unload_module zaptel
```

In older Zaptel versions the relevant lines are 115-123 as shown below:

```
     # Unload drivers
     # echo "Unloading zaptel hardware drivers:"
     # for x in $RMODULES; do
     # action "Unloading ${x}:" rmmod ${x}
     # done
     # action "Removing zaptel module:" rmmod zaptel
     # RETVAL=$?

     # [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/zaptel
```

The relevant lines in your zaptel stop script may be slightly different; the section to look out for is the one that says includes "Unload drivers".

If amending the zaptel stop script doesn't fix the issue then you will need to ensure that Asterisk is shut down before rebooting, i.e.

```
#amportal stop / CLI>stop now
#shutdown -r now
```

It is advisable to add the command to stop Asterisk to one of the reboots scripts to ensure that Asterisk is always stopped before the Zaptel modules are unloaded when the IP PBX server is rebooted.

8 Quick Reference

Useful Commands

Command	Description
#amportal start	Start Asterisk
#amportal stop	Stop Asterisk
#asterisk –r	Connect to the Astersik CLI
#cat /proc/interrupts	Lists PCI bus drivers/devices recognised by the Linux Kernel along with the allocated IRQ
#chkconfig	Command for adding/removingg startup scripts and managing which run level they operate at
#dmesg | more	Diagnostic message command used to look at the Kernels boot process message buffer
#fxotune	Tool used to auto-configure the PSTN line impedance settings for cards that have a programmable digital hybrid.
#hdparm	Program that interacts with Linux IDE driver to obtain and alter disk parameters
#lsmod	Lists information about all loaded modules
#lspci –v	Linux utility that displays information about devices connected to the PCI bus, "-v" option displays IRQ allocation details
#modprobe	Adds a loadable module to the kernel, e.g. zaptel or wcfxo
#ps aux	Static snapshot overview of system processes
#setpci	Utility for querying and configuring PCI devices
#top	Dynamic overview of system processes updated every 10 seconds
#uptime	CPU load information based on 5, 10, 15 minute averages
#vmstat	Provides information on CPU/memory usage as well system I/O information
#ztcfg -vv	Reloads the zaptel.conf configuration, "-vv" option provides information on configured zap channels
#ztmonitor	Tool to help set the appropriate RX/TX gain values
#zttool	The Zaptel Tool (zttool) command shows the current status of any Zaptel supported interface cards
#zttest	Zaptel Test (zttest) is a timing test to determine the accuracy of the Zaptel timing
CLI>exit	Exit from Asterisk CLI
CLI>reload	Reload Asterisk
CLI>zap show channels	Shows Zap channels recognised by Asterisk
CLI>zap show status	Shows Zaptel devices recognised by Asterisk
CLI>stop now	Stops Asterisk

Configuration File Descriptions

File	Description
/etc/inittab	File used to configure runlevel modes
/etc/zaptel.conf	Used by the Zaptel driver to define the relevant parameters for zaptel supported hardware devices
/etc/asterisk/zapata.conf	Used by Asterisk to store information about Zaptel devices and the features supported
/etc/asterisk/zapata-auto.conf	Automatically generated by genzaptelconf, contains Zaptel hardware configuration information used by Asterisk
/etc/asterisk/extensions.conf	Contains dial plan information in the form of instructions called extensions, which are grouped by contexts
/etc/asterisk/indications.conf	Contains information about the sounds that a phone system in a particular country makes for various indications
/etc/rc6.d/KxxZaptel	Zaptel reboot level stop script

9 Further Assistance

The following are useful links for IP PBX server configuration:

Trixbox Without Tears - http://dumbme.voipeye.com.au/trixbox2/trixbox2_without_tears.pdf
Elastix Without Tears - http://www.elastixconnection.com/downloads/elastix_without_tears.pdf
Asterisk Installation Guide - http://www.voip-info.org/wiki-Asterisk+installation+tips
Trixbox Installation Guide - http://www.trixbox.org/wiki/trixbox-quick-install-guide
PBX in a Flash Installation Guide - http://pbxinaflash.net/docs/InstallPBXInAFlash-1.2.pdf

Links to the online forums and wikis for assistance installing Asterisk/Trixbox are provided below:

Asterisk Wiki - http://www.voip-info.org/wiki-Asterisk
Asterisk Forum - http://forums.digium.com
Trixbox Forum - http://www.trixbox.org/forum
Trixbox Wiki - http://help.trixbox.com

For product installation guides and support contact information please visit our support page:

www.novavox.co.uk/support

Novavox customers only: If after following the guidance provided on the support page you are unable to resolve the issue then please contact us at support@novavox.co.uk.

Appendix A - Signalling methods

Groundstart (gs)

Groundstart signalling also known as Earth Start and is not very common. The PBX/phone signals a call by connecting an earth to one of the wires and current flows from the PBX earth to the local exchange battery. The local exchange responds by sending dial tone to the PBX/phone. The local exchange signals a call to the PBX/phone by putting an earth on one of the wires. The PBX/phone detects the earth and looks for ringing voltage which the local exchange sends.

Loopstart (ls)

Loopstart signalling is supported by the majority of analogue phone lines. In Loopstart signalling the local exchange signals a call by sending ringing voltage to the phone/PBX. It allows a phone to indicate on hook/offhook (and a PBX to seize the line) by putting a loop across the wires. The local exchange responds by sending dial tone to the phone/PBX.

Kewlstart (ks)

Kewlstart signalling works in the same way as loopstart, but it allows the local exchange to drop battery/voltage momentarily on the phone line to indicate to the phone/PBX that the other end of the party has disconnected the call, i.e. to provide hang-up notification. This voltage drop is known as CPC (Calling Party Control) or Disconnect Clear Time (DCT) in the UK.

Appendix B - Country codes / zones

Code	Country / Zone
at	Austria
au	Australia
be	Belgium
br	Brazil
ch	Switzerland
cl	Chile
cn	China
cz	Czech Republic
de	Germany
dk	Denmark
ee	Estonia
es	Spain
fi	Finland
fr	France
gr	Greece
hu	Hungary
il	Israel
in	India
it	Italy
lt	Lithuania
mx	Mexico
nl	Netherlands
no	Norway
nz	New Zealand
pl	Poland
pt	Portugal
ru	Russia
se	Sweden
sg	Singapore
tw	Taiwan
uk	United Kingdom / Ireland
us	United States
us-o	United States - Old
ve	Venezuela
za	South Africa

Appendix C – Zaptel.conf configuration parameters

busycount - Busycount is the number of busy tones to detect, when **busydetect** is enabled. The lower the value the quicker the call will hang-up, but this increases the possibility of false hangups. [positive integer]

busydetect - With this option enabled, asterisk will listen for busy signals on the line, if your carrier sends you a busy tone on a hang-up, Asterisk should detect it. It should be noted that this option is less reliable than other hang-up detection methods such as polarity reversal and CPC (Calling Party Control) so should only be used to detect hang-ups if the other methods are not supported. [yes / no]

callerid - CallerID can be set to 'asreceived' or a specific number if you want to override it. Please note that asreceived only applies to trunk interfaces. [asreceived or a specific number]

callgroup, pickupgroup - Supports ring groups (a.k.a. call groups) and pickup groups. If a phone is ringing and it is a member of a group which is one of your pickup groups, then you can answer it by picking up using the pickup code. For simple offices, just make these both the same positive integer.

callprogress, progzone - This option is similar to the busydetect option, but will also listen for a ringing tone, congestion tone and will try to detect if the line got answered. However, this feature is experimental and can easily detect false answers and errors. Few zones are supported, but may be selected with progzone [yes / no, and two letter country code for progzone].

callreturn - Support call return. [yes / no]

callwaiting - Enable call waiting on FXO lines. [yes / no]

callwaitingcallerid - Sets whether to receive caller ID during call waiting indication. [yes / no]

canpark - Allow call parking. [yes / no]

cidsignalling - Type of caller ID signalling in use.

bell =bell202 (US)
v23 = v23 (UK)
dtmf = DTMF (Denmark, Sweden, & Netherlands)

cidstart - Identifies what signals the start of caller ID.

ring = a ring signals the start
polarity = polarity reversal signals the start

context - Defines the initial context for the channel. This will be the context available to the channel upon the initiation of a call. Note that contexts are an important part of maintaining site security. The initial context will govern the availability of extensions to a given channel. If an extension is placed in a different context from the initial context, that extension is unavailable to the caller.

echocancel - Enable echo cancellation. [yes / no, or a power of two from 32 to 256 to set the number of cancellation taps]

echocancelwhenbridged - Generally, it is not necessary (and in fact undesirable) to echo cancel when the circuit path is entirely TDM. You may, however, reverse this behaviour by enabling the echo cancel during pure TDM bridging. [yes / no]

echotraining - In some cases, the echo canceller doesn't train quickly enough and there is echo at the beginning of the call. Enabling echo training will cause Asterisk to briefly mute the channel, send an impulse, and use the impulse response to pre-train the echo canceller so it can start out with a much closer idea of the actual echo. [yes, no, or a number of milliseconds (10 – 2000) to delay before training]

faxdetect - Upon fax detection, routes fax to a fax extension. [Both, incoming, outgoing, or no.]

group - Allows a number of channels to be treated as one for the purpose of dialing. For dialing out, the channels will be called on a first available basis. For the purpose of ringing stations, all channels in the group will ring at once. Takes an integer from 0 to 63 and multiple groups can be specified.

hanguponpolarityswitch - In some countries, a polarity reversal is used to signal the disconnect of a phone line. If this option is enabled, the call will be considered disconnected on a polarity reversal. [yes / no]

hidecallerid - Sets whether to hide outgoing caller ID, defaults to no. [yes / no]

immediate - Specify whether the channel should be answered immediately or if the simple switch should provide dialtone, read digits, etc. [yes / no]

musiconhold - Select which class of music to use for music on hold. If not specified then the default will be used. The music class is defined in musiconhold.conf file. [default, loud, random]

pulsedial - Use pulse dial instead of DTMF for FXO (FXS signalled) devices. [yes / no]

relaxdtmf - If you have trouble with DTMF detection, you can relax the DTMF detection parameters. Relaxing them may make the DTMF detector more likely to have talk off where DTMF is detected. [yes / no]

rxgain - Adjusts receive gain. This can be used to raise or lower the incoming volume to compensate for hardware/line differences. [Positive or negative double, measured in dB]

sendcalleridafter - Some countries have ring tones with a set of cadences which differ from the default. This requires the callerid to be set with a delay, and not right after the first ring. [Positive integer]

threewaycalling - Support three-way calling. [yes / no]

transfer - Support flash-hook call transfer (requires three way calling). Also enables call parking (overrides the **canpark** parameter). [yes / no]

txgain - Adjusts transmit. This can be used to raise or lower the outgoing volume to compensate for hardware/line differences. [Positive or negative double, measured in dB]

usecallerid - Whether or not to use caller ID. [yes / no]

usedistinctiveringdetection - Indicates whether or not to allow distinctive ring detection on FXO lines. [yes / no]

Appendix D - Recompiling Zaptel

Guidelines

Zaptel Software Version

Detailed instructions are provided below for recompiling Zaptel version 1.4.10.1, which is the version used in Trixbox 2.6.07. If you wish to recompile a different Zaptel version, then you will need to ensure you check what Zaptel Version is currently running and modify the instructions to download/install the correct version.

Configuration File Backups

Unless something goes wrong your current configuration files should not be modified when recompiling the Zaptel software. However, it is recommended to backup current configuration files in just case something does goes wrong, particularly if the IP PBX is a working system.

Software Package Dependencies

If you your IP PBX has software packages installed that are dependant on Zaptel, then there is a possibility that they may not work properly after recompiling the Zaptel software and they may need to be reinstalled. Also, if you have installed any other patches or modified any of the original Zaptel source code (e.g. to support vendor specific hardware drivers) then you will need to repeat the same code changes before recompiling the Zaptel code.

Instructions

The instructions below are based on using yum for package updates/installation and wget to download source code tarballs. However, if preferred alternative methods can be used such as using apt or manually downloading rpm's to update/install packages and using svn to download source code.

1. Update Packages

It is recommended that you ensure that all installed software packages are up to date using:

```
#yum update –y
```

Warning: If you have patched any installed packages then these will need to be patched and recompiled again after using yum. As an alternative, you can just update the Zaptel package as this is being recompiled anyway.

2. Install kernel Source Tree

First of all check to see if you need to install a kernel source tree by running:

```
#ls –l /lib/modules/`uname -r`/build/.config || echo "Install kernel"
```

If the output shows "Install kernel" then check what kernel type you need as shown below, otherwise proceed to step 3.

Check if you are using a smp kernel or not:

```
#uname -r | grep -q smp && echo "Install SMP kernel."
```

If the output shows "Install SMP kernel.", then run:

```
#yum install kernel-smp-devel kernel -y
```

Otherwise run:

```
#yum install kernel-devel kernel –y
```

3. Install C/C++ Compiler Packages

Install C/C++ Compilers to Compile Source Code if you don't have them installed on your system:

```
#yum install gcc -y
#yum install gcc-c++ -y
```

4. Shutdown and Restart the IP PBX

Reboot and check that everything is still working correctly after installing/updating relevant packages:

```
#shutdown -r now
```

Download Zaptel Source Code

Change to user source code directory and download/unzip the required software packages:

```
#cd /usr/src
```

5. Download/Unzip Zaptel

```
#wget http://ftp.digium.com/pub/zaptel/releases/zaptel-
1.4.10.1.tar.gz
#tar -xzvf zaptel-1.4.10.1.tar.gz
#ln -s zaptel-1.4.10.1 zaptel
```

6. Make Source Code Modifications

At this point you should make the required changes to the relevant Zaptel source code files. If you are changing the default echo cancellation software then you should modify the zconfig.h file as described in Section 5.2.7.

7. Stop Asterisk and Zaptel

First of all stop Asterisk:

```
#amportal stop / CLI>stop now
```

Then stop Zaptel:

```
#service zaptel stop
```

8. Remove Existing Zaptel modules

The commands below are correct for removing the Zaptel module directories in Trixbox 2.6.07 (the Zaptel modules may be in one or possibly both directories).

```
#rm –r /lib/modules/`uname -r`/extra/zaptel
#rm –r /lib/modules/`uname -r`/misc
```

If you are using a different Trixbox version or an alternative IP PBX package then you will need to find where the Zaptel modules are currently installed. To find the Zaptel modules directory find the directory that the wcfxo.ko module is stored.

> **Note:** As an alternative to deleting the Zaptel module directory you could make a copy of the directories and then delete the old ones to provide a contingency plan if anything goes wrong.

9. Compile Zaptel Source Code

Compile the Zaptel Source Code:

```
#cd /usr/src/zaptel
#make clean
#make
```

You should receive the following message:

```
****
**** The configure script was just executed, so 'make' needs to be
**** restarted.
****
make: *** [config.status] Error 1
```

Continue by running:

```
#make
#make install
#make config
```

The following should be included in the output:
```
...
I think that the zaptel hardware you have on your system is:
pci:0000:04:06.0    wcfxo-       1057:5608 Wildcard X100P
```

10. Shutdown and Restart the IP PBX

```
#shutdown -r now
```

Configuration Complete

The Zaptel source code has been recompiled and your IP PBX is now ready to be used.

Appendix E – Acronyms

Acronym	Description
ACPI	Advanced Configuration and Power Interface
APIC	Advanced Programmable Interrupt Controller
BIOS	Basic Input/Output System
BT	British Telecom
CLI	Command Line Interface
CPC	Calling Party Control
CPU	Central Processing Unit
CTR21	Common Technical Requirements directive 21
DAA	Direct Access Arrangement
DCT	Disconnect Clear Time
DMA	Direct Memory Access
DSP	Digital Signal Processor
ERL	Echo Return Level
FCC	Federal Communications Commission
FXO	Foreign Exchange Office
FXS	Foreign Exchange Station/Subscriber
GUI	Graphical User Interface
I/O	Input / Output
IDE	Intelligent/Integrated Drive Electronics
IP	Internet Protocol
IP PBX	IP Private Branch Exchange
IRQ	Interrupt Request
JATE	Japan Approvals Institute for Telecommunications Equipment
OSLEC	Source Line Echo Canceller
PCI	Peripheral Component Interconnect
POTS	Plain Old Telephony Service
PSTN	Public Switched Telephony Network
SCSI	Small Computer System Interface
SMP	Symmetric Multi-Processing
UDMA	Ultra DMA
UK	United Kingdom
USB	Universal Serial Bus
VoIP	Voice over IP