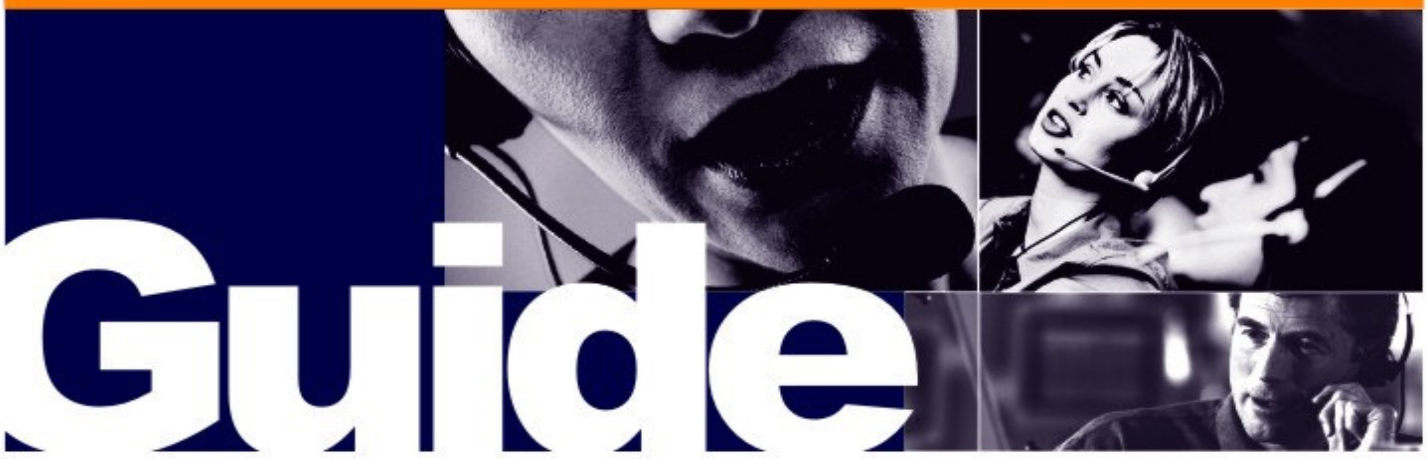
Flávio E. Gonçalves

Ready for the 1.4 Version!

Asterisk PBX

3RD GENERATION



Guide

Configuration Guide

How to build and configure a PBX with open source Software

How to build and configure a PBX with Open Source Software

Featuring release 1.4

Flavio E. Gonçalves

Third Generation

2nd Edition/March/2007

rev. 8.3

By Flavio E. Goncalves

Asterisk PBX Configuration Guide
Flavio E. Gonçalves
Revision: Luis F. Gonçalves

Printing History

First Edition: November 2006,
File Date: Thursday, March 15, 2007
ISBN: 978-85-906904-2-9

Preface

This book is for anyone who wants to learn how to install and configure a PBX (Private Branch eXchange) based on Asterisk PBX. Asterisk is an open source telephony platform capable to use VoIP and TDM channels.

This is the third generation of the e-Book Asterisk Configuration Guide. The e-Book is also available in Spanish and Portuguese. The material that I present in this book helped to prepare for the dCAP certification from Digium last May 2006 and to pass it in the first try. Originally, this e-Book was written for version 1.0. The second generation was updated to version 1.2 and this one is based on version 1.4. However, you may still find examples that were based in the older version. Wherever possible, those examples have been suppressed or changed.

I have always been a fan of e-Books. They are easy to carry around, ecologically correct and simpler to publish. Piracy is the major drawback of this strategy. We will use any possible means to restrict piracy. Unfortunately, it will happen, but I sincerely hope you buy this e-Book legally.

The Asterisk Open Source PBX is revolutionary. Telephony will never be the same after this program. For many years, telephony has been dominated by huge companies with proprietary systems. Finally, users can recover their buying power by having access to an open telephony platform. Thus, things that were not possible before because they were not economically viable are likely to start happening. Examples include resources like CTI (computer telephony integration, IVR (interactive voice response), ACD (automatic call distribution), and voicemail, that are now available to everybody.

This book was not designed to teach every single detail of Asterisk. In fact, you will probably not become a guru simply by reading this e-Book. However, you will be able to build and configure a PBX with advanced features like voicemail, IVR an ACD by the end of reading. I hope you enjoy as much learning about Asterisk as I have enjoyed writing about it.

GonçalvesFlavio E. Gonçalves
CEO
V.Office Networks
flavio@asteriskguide.com

Audience

This book is intended for those who are new to Asterisk. We assume your are familiar with Linux, Linux shell commands and Linux text editors. You could test Asterisk using a Linux system with a graphical interface which may be easier for Linux newbies. Some users will try to execute Asterisk using VMWare and this is really not a problem, except for poorer voice quality. For production systems we do not encourage VMware or Linux with a graphical user interface.

It is also desirable that the reader has some knowledge of IP networks, voice over IP (VoIP) and telephony concepts.

Acknowledgments

I have to thank my family for the patience to see me work at late hours and during the weekends for several months. A special thanks to my sister Ana Cristina Gama for her help with publishing and my brother Luis F. Gonçalves who made the English revision of this e-Book.

Mistakes and errors in the e-Book

We always try to find and eliminate errors and mistakes. Please, if you find something wrong, give us feedback and we will act on it immediately. You will receive a revised copy of the book in case your feedback results in a change (the beauty of e-publishing!).

E-mail address for feedback: oops@voffice.com.br

Summary

Chapter 1

Asterisk Introduction

In the current chapter, we will learn what Asterisk is, its architecture, and how it can be used.

1.1 OBJECTIVES

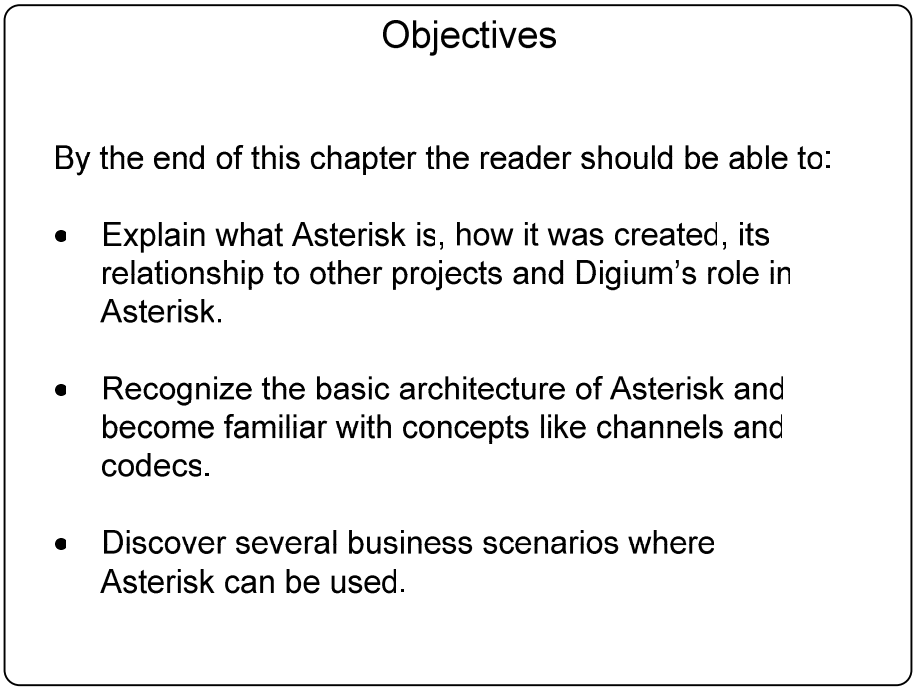


Figure 1.1 Objectives

1.2 WHAT IS ASTERISK?

Asterisk is an "Open Source PBX software" that once installed in a PC hardware along with the correct interfaces, can be used as a full featured PBX for home users, enterprises, VoIP service providers and telecoms. Asterisk is also both an Open Source Community and a commercial product from Digium. You are free to use and modify Asterisk to suit your needs.

Asterisk allows real time connectivity between PSTN and VoIP networks. Since Asterisk is much more than a PBX, you have not only an exceptional upgrade to your existing PBX, but you can do new things in telephony, such as:

- Connect employees working from home to an Office PBX over broadband Internet.
- Connect several offices in different places over an IP network, private network, or even through the Internet itself.
- Give your employees web and e-mail integrated voicemail.
- Build applications like IVRs that allow connections to your ordering system or other applications.
- Give traveling users access to the company PBX from anywhere with a simple broadband or VPN connection
- And much more....

Asterisk includes several advanced resources, only found before in high-end systems, for example:

- Music on hold for costumers waiting in call queues, supporting media streaming and MP3 files.
- Call queues, where a team of agents can answer calls and monitor queues.
- Integration with text-to-speech and voice recognition.
- Detailed records transferred to both text files and SQL databases.
- PSTN connectivity through both digital and analog lines.

1.2.1 Digium's role in Asterisk

Digium, a company located in Huntsville, Alabama, is the creator and primary developer of Asterisk. Besides being the primary sponsor of Asterisk development, Digium also produces telephony interface cards and other hardware for the Asterisk's PBX.

Digium offers Asterisk under three different types of license agreement:

- **General Public License (GPL) Asterisk**. This is the most used version. It includes all features and is free to be used and modified according to the terms of the GPL license.
- **Asterisk Business Edition** is a recent version of Asterisk. It doesn't have some of the extra features found in the GPL version, such as ??? The business edition is used by some companies that don't want or can't use the GPL license, mostly because they don't want to release their source code together with Asterisk. The GPL license requires that any further code development to a GPL licensed code must be released to the source code.

- **Asterisk OEM**. Mostly used by PBX manufacturers who do not want to reveal to the public that their software is based on Asterisk.

1.2.2 The Zapata project and its relationship with Asterisk

The Zapata project was developed by Jim Dixon, who was also responsible for the development of a revolutionary hardware that is used with Asterisk. Note that the hardware is Open Source too and, therefore, it can be used by any company. Digium, Sangoma and Varion are some of the main telephony card manufacturers of the Asterisk PBX. The Zapata project can be seen at:

http://www.asteriskdocs.org/modules/tinycontent/index.php?id=10)

The main feature of the Asterisk hardware is the use of the PC CPU to process media streaming, echo cancellation and transcoding. In contrast, most of the existing cards use DSP (Digital Signal Processors) to perform these tasks. The decision to use the PC CPU reduced the board's price dramatically. Thus, Digium boards are several times cheaper then previously available boards from, for example, Dialogic, Aculab and others, since they don't require expensive DSPs. The drawback is that these boards need a lot of CPU and a misuse of the PC CPU can have a major impact in voice quality.

1.3 WHY ASTERISK?

I remember my first contact with Asterisk. The first reaction to something new, moreover one that competes with something you already know, is to reject it!

It happened in 2003. Asterisk was competing with a solution that I was selling to a costumer (4 E1 VoIP Gateway) and it costed ten times less than the price I was charging for the solution I already knew. Due to the disproportionate price, I started studying Asterisk in order to identify potential pitfalls or drawbacks. I found, for example, that the PC CPU at that time would not support 120 g.729 simultaneous sections and, at the end of the day, I won the proposal with my gateway solution. However, this exercise led me to the discovery that Asterisk could solve a variety of very expensive problems for my costumer base. We were in trouble with expensive quotes for IVR, unified messaging, call recording, and dialers. I found that with appropriate dimensioning, the CPU problems could be worked around and, indeed, Asterisk became the flagship product of my company in just three years (I actually decided to open another company just for the Asterisk business). In my opinion, Asterisk is a revolution in

telecommunication and it represents to IP telephony what Apache represents to web services.

1.3.1 Extreme cost reduction

If you compare a traditional PBX with Asterisk with digital interfaces and phones, Asterisk is a little cheaper than those PBXs. However, Asterisk really pays off when you add advanced features like voicemail, ACD, IVR and CTI. With these advanced features, Asterisk is several times less expensive than traditional PBXs. Indeed, comparing Asterisk PBXs with low-end analog PBXs is unfair because it has a lot of features that are not available in low-end analog systems.

1.3.2 Telephony system control and independence

One of the most often quoted benefits from the costumer's point of view is the independence that Asterisk provides. Some of today manufacturers do not even give the costumer the system's password or the configuration documentation. With Asterisk's "do it yourself" approach, the user achieves total freedom and, as a bonus, has access to a standard interface.

1.3.3 Easy and rapid development environment

Asterisk can be extended using scripting languages like PHP and Perl with AMI and AGI interfaces. Asterisk is Open Source and its source code can be modified by the user. The source code is written mostly in ANSI C programming language.

1.3.4 Feature rich

Asterisk has several features that are either not found or optional in traditional PBXs (e.g. voicemail, CTI, ACD, IVR, built in music on hold, and recording). The costs of these features in some platforms supersede the price of the platform itself.

1.3.5 Dynamic content on the phone

Asterisk is programmed using C language and other languages common in today's development environment. The possibility to provide dynamic content is almost limitless.

1.3.6 Flexible and powerful dial plan

One more Asterisk breakthrough. If you look at traditional PBXs, even simple things like LCR (Least Cost Routing) are either not feasible or optional. With Asterisk, choosing the best route is easy and clean.

1.3.7 Open source running on top of Linux

One of the greatest features of Asterisk is it's community. When I access Wiki (www.voip-info.org), e-mail distribution lists, and forums, I see that the adoption of Asterisk has been fast with patches quickly provided to any bugs eventually found. Asterisk is probably the most tested PBX software in the world. From versions 1.0 to 1.2, more than 3000 changes and bugs in the source code were corrected. This process ensures a code that is both stable and almost error free.

1.3.8 Asterisk architecture limitations

Some limitations in Asterisk come from the use of the Zapata telephony design. In this design, Asterisk uses the PC CPU to process voice channels instead of dedicated DSPs (Digital Signal Processors) which are common in other platforms. Although this allows for a huge cost reduction in hardware interface, the system becomes dependent on the PC CPU. My recommendation is to run Asterisk in a dedicated machine and to be conservative about hardware dimensioning. It is also interesting to use Asterisk in a separate VLAN to avoid excessive broadcasts that consume the CPU (broadcast storms caused by loops or viruses). Some newer interface cards from several vendors are now including DSPs to process echo cancellation, codecs and other features. This will make Asterisk even better.

1.4 ASTERISK ARCHITECTURE

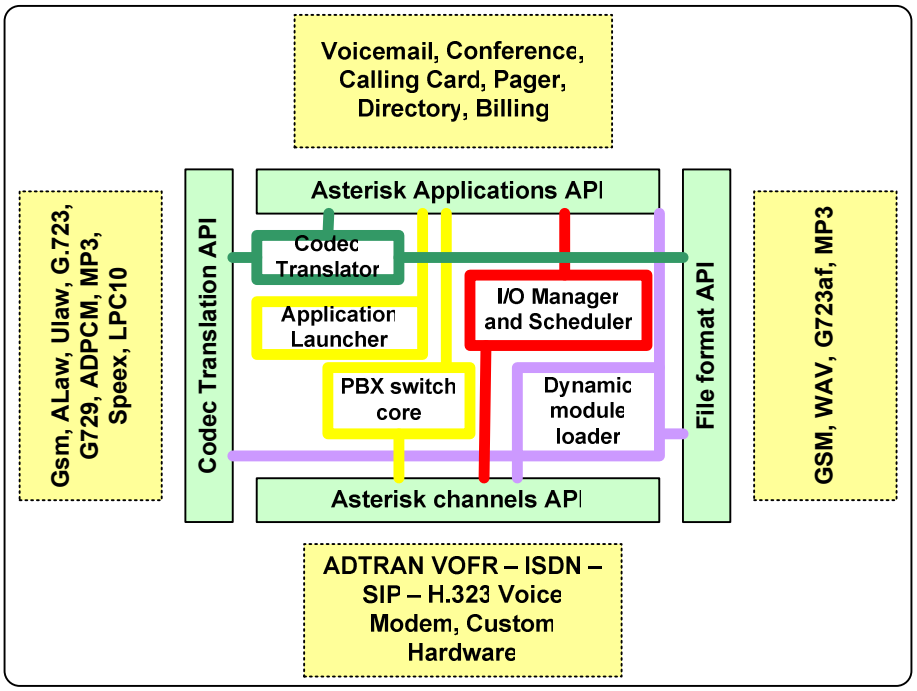


Figure 1.1 - Asterisk Architecture.

The figure above shows the basic Asterisk architecture. Next, we will explain concepts related to the architecture like channels, codecs and applications.

1.4.1 Channels

A channel is the equivalent of a telephone line, but in digital format. It usually consists of an analogic or digital (TDM) signaling system or a combination of codec and signaling protocol (e.g. SIP-GSM, IAX-uLaw). In the beginning, all telephony connections were analog and susceptible to echo and noise. Later, most systems were converted to digital systems, with the analogic sound converted into digital format by PCM (Pulse Code Modulation) in most cases. This format allows voice transmission in 64 kilobits/second without compression.

TDM hardware supported:

Zaptel Cards (usually Digium produced)

- Wildcard T410P –Four E1/T1 interfaces (PCI 3.3 volts only)
- Wildcard T405P – Four E1/T1 interfaces (PCI 5.0 volts only)

- TE110P – One port E1/T1 interface
- TDM400P – Four analog interfaces FXO or FXS
- TDM2400 – Twenty-four ports FXS or FXO

These boards use chan_**zap** channel drivers

- Linux Cards
- Quicknet Phonejack and linejack can be used
- ISDN Cards and drivers
- ISDN4Linux – Old driver, don't even try to use it. I have tried with a ISDN BRI Eicon Diva Card. The result is bad voice quality and instability. It was removed from compilation in version 1.2 and deleted in version 1.4. These boards use chan_modem channel drivers
- ISDN CAPI – It is a third party Linux ISDN driver. It is used with junghanns boards (www.junghanns.net) and integrates with hylafax. These boards use chan_**capi** channel drivers
- There are other channel drivers for BRI stuff like chan_misdn, which are considered experimental. These drivers were created by Beronet, Europe. vISDN can be also used with HFC ISDN chips.
- Voicetronix (www.voicetronix.com.au): Produces high density analog boards, 4, 8 and 16 ports. Now they produce E1/T1 cards too. These boards use chan_vbp channel drivers
- Other manufacturers include Dialogic and Aculab. Most of them do not release the source code.

Asterisk channels supported:

1. chan_console: supports a sound card (OSS or ALSA) - dial string: console/dsp)

2. chan_sip: supports voice over IP using SIP protocol. – dial string: sip/channel

3. chan_iax: supports voice over IP using IAX2 protocol, - dial string: iax2/channel

4. chan_h323: H.323 is one of the oldest and most implemented voice over IP protocols. It's useful when one attempts to connect to existing H.323 networks. There are different flavors of H.323 in Asterisk: chan_h323 and chan_oh323. A third implementation is being developed by Digium: the first version in asterisk-add-ons subversion. Chan_h323 can be used in Asterisk as a gateway. Asterisk can point to

a gatekeeper, but can't work as one. Dial string h323/hostname if using a gatekeeper, h323/extension@hostname if going directly to the gateway.

5. chan_mgcp: supports the voice over IP protocol using MGCP. Currently Asterisk supports MGCP phones, but it cannot connect to a VoIP provider using MGCP. Dial string: MGCP/aaln/1@hostname

6. chan_sccp: Supports Cisco voice over IP skinny protocol. There are two versions: chan_skinny and chan_sccp2 (http://chan-sccp.berlios.de). Supports most Cisco phones. More information can be found in http://www.voip-info.org/wiki/view/SCCP-HOWTO2. Dial String: SCCP/channel.

7. chan_unicall: Implements MFC/R2 as a signaling protocol for E1s used in China, Latin America and several other countries. It is supported by a third party channel driver called Unicall. In chapter 4, we will detail how to implement it. It uses the chan_unicall channel driver. Dial string: Unicall/channel
   •
8. chan_agent: Used for ACD (Automatic Call Distribution). It is not related to a specific hardware or protocol. It can also be used for mobility, allowing any person to use any phone just by logging in to the agent.

9. chan_local: Is a pseudo channel, it simply loops back into the dialplan in a different context. Useful for recursive routing. Dial string: Local/extension@context

1.4.2 Codecs and codec translation

We usually try to put as many voice connections as possible in a data network. Codecs enable new features in digital voice. Compression is one of the most important ones, since it allows compression rates larger than 8 to 1. Other features include voice activity detection, packet loss concealment and comfort noise generation. There are several codecs available for Asterisk and these codecs can be transparently translated from one to another. Internally, Asterisk uses slinear as the stream format when it needs to convert from one codec to another. Some codecs in Asterisk are supported only in pass-through mode, and these codecs can't be translated.

The following codecs are supported:

- G.711 ulaw (USA) – (64 Kbps).

- G.711 alaw (Europe) – (64 Kbps).
- G.723.1 – Only pass-through mode
- G.726 – (16/24/32/40kbps)
- G.729 – Needs licensing (8Kbps)
- GSM – (12-13 Kbps)
- iLBC – (15 Kbps)
- LPC10 - (2.5 Kbps)
- Speex - (2.15-44.2 Kbps)

1.4.3 Protocols

Sending data from one phone to another should be easy provided that the data find a path to the other phone by themselves. Unfortunately, it doesn't happen this way, and a signaling protocol is necessary in order to establish connections between phones, discover end devices and implement telephony signaling. Recently, it has become very common to use of SIP as a signaling protocol. H.323 is largely implemented in voice over IP networks and most legacy implementations use this protocol. IAX is another option that is becoming popular because it works well with NAT traversal and some bandwidth can be saved as well.

Asterisk supports:

- SIP
- H323
- IAXv1 e v2
- MGCP
- SCCP (Cisco Skinny)
- Nortel unistim

1.4.4 Applications

To bridge calls from one phone to another an application named Dial() is used. Most Asterisk features like voicemail and conferencing are implemented as applications. You can see available Asterisk applications by using the "core show applications" console command. You can add applications from asterisk-addons, from third-party providers or even develop some applications yourself.

1.5 OVERVIEW

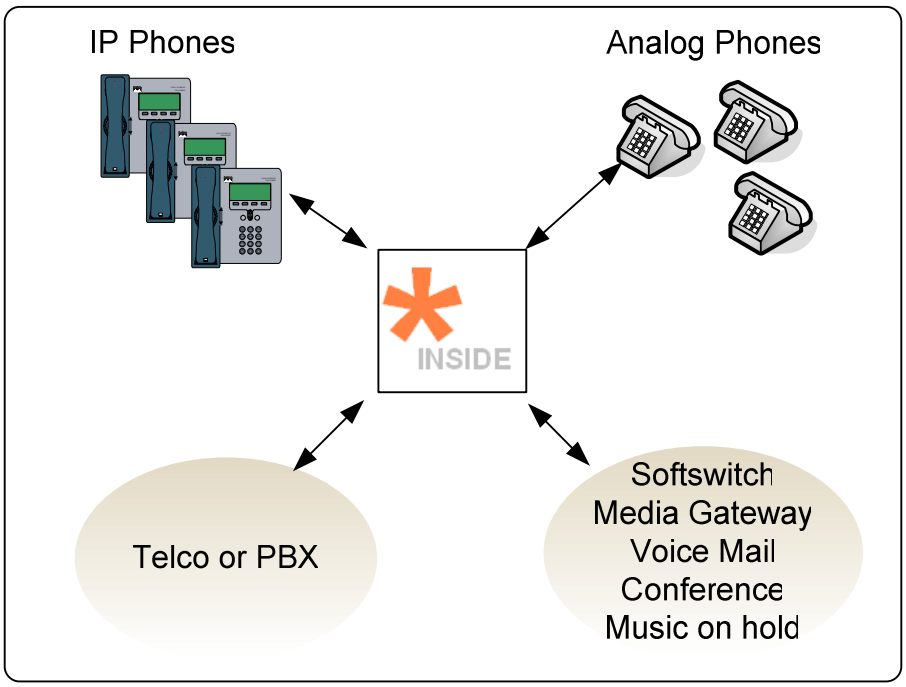


Figure 1.2-Asterisk overview

Asterisk is an open source PBX that acts like a hybrid PBX, integrating technologies such as TDM[1] and IP telephony. Asterisk is ready for IVR (Interactive Voice Response) functionality and ACD (Automatic call distribution) and, as mentioned in previous sections of this book, is open to the development of new applications. In the above figure, you can see that Asterisk connects to telcos and existing PBXs using analog and digital interfaces and also supports analog and IP phones. It can act as a softswitch, media gateway, voicemail, audio conference, and has built in music on hold.

[1] TDM – Time division multiplexing. Today most of digital telephony is based on this concept. With TDM you can use 23 to 30 voice circuits in a single T1 or E1 connection respectively. T1 is mostly used in the USA while E1 is common in Europe.

1.6 DIFFERENCES BETWEEN THE OLD AND THE NEW WORLD.

1.6.1 Telephony using the old PBX/Softswitch model

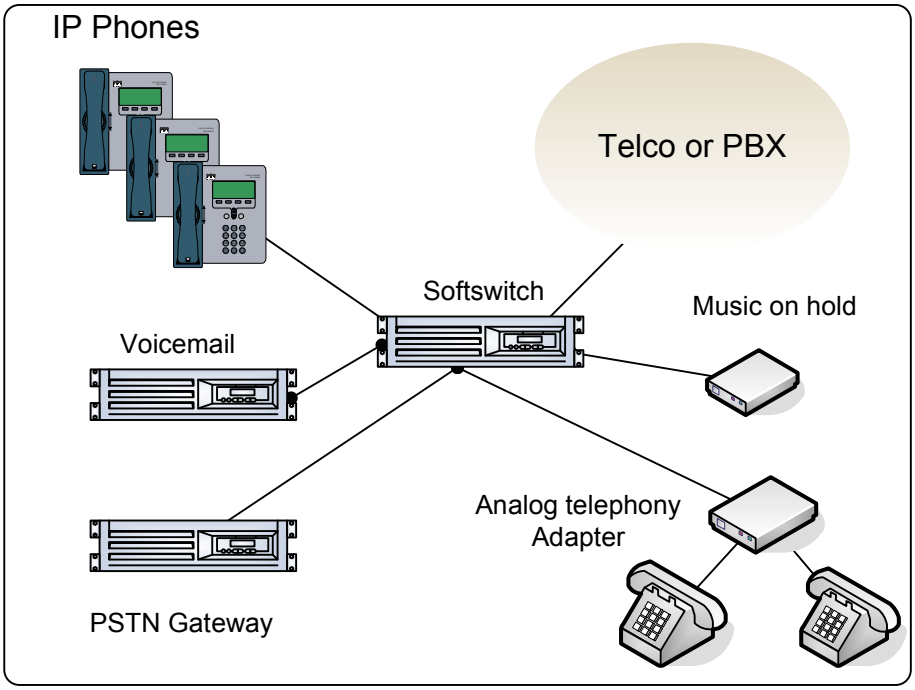


Figure 1.4- Conventional softswitch or legacy PBX model

In the old softswitch model, all components were sold separately. Therefore, you had to purchase each component and then integrate to the PBX or softwitch environment. The costs and risks were high and most of the equipment were proprietary.

1.6.2 Telephony using Asterisk

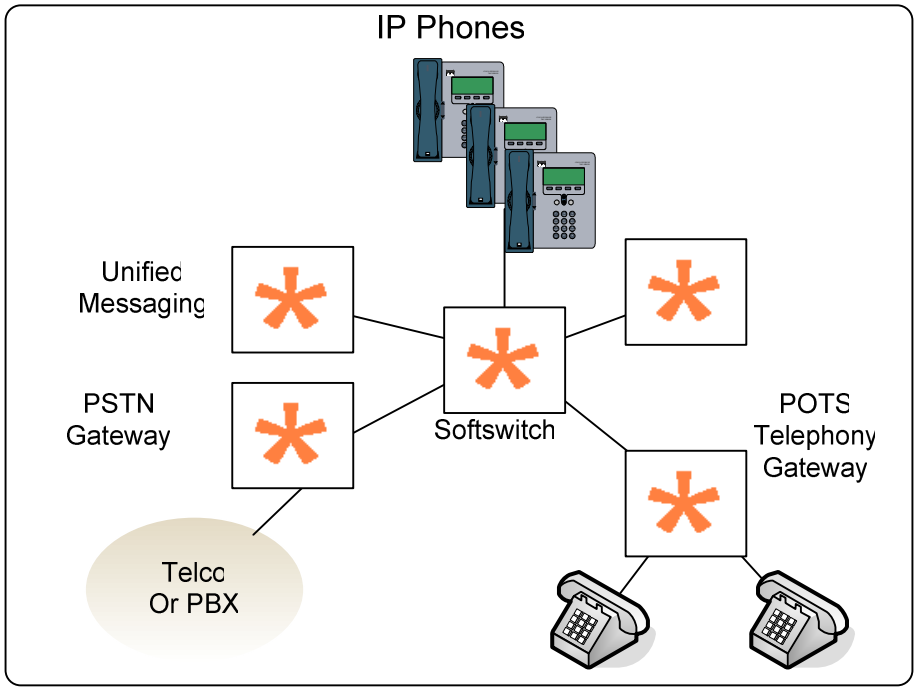


Figure 1.5 –Telephony in the Asterisk way.

All functions are integrated in the Asterisk platform in the same or in different boxes according to dimensioning, and are licensed according to the GPL agreement.

1.7 BUILDING A TEST SYSTEM

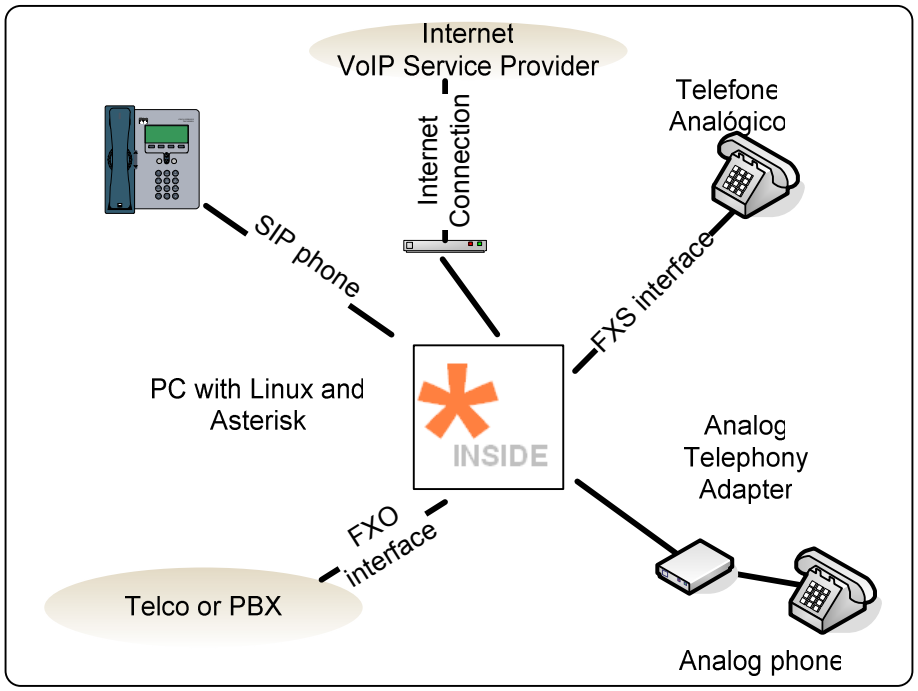


Figure 1.6 – The 1x1 PBX

When implementing an Asterisk solution, our first step is generally to build a test machine. The easiest test machine is the 1x1 PBX with at least one phone and one line. There are several ways to do it. Let's examine some of them.

1.7.1 One FXO, one FXS

The first and simplest way to build a test machine is to purchase a Digium TDM400 board with one FXO and one FXS interface. Connect the FXO port to an existing line and one FXS to an analog phone. There you have: a 1x1 PBX.

1.7.2 VoIP Service Provider, ATA

This is the VoIP option. In this case you would sign up with a voice service provider to have the SIP trunks and will have to purchase a SIP analog telephony adapter. You will probably spend less than US$100.00 if you already have the PC.

1.7.3 Inexpensive FXO board, ATA

This is the way I started. There are some V.90 fax/modems that work with Asterisk as an FXO board. Some of the first Digium boards were created using those (X100P and X101P). These boards are old V.90 fax/modems based on Motorola and Intel chipsets (Motorola 68202-51, Intel 537PU, Intel 537PG, Intel Ambient MD3200 are known to work). They are not easy to find since they are not produced anymore; however, some are sold as X100P clones. I found ten of these boards in a PC recycling company. For the FXS you can use an Analog Telephony Adapter. Once again you could spend less than US$100.00 to start if you already have the PC.

1.8 ASTERISK SCENARIOS

There are several different scenarios where Asterisk can be used. We will list some of them and explain the advantages and possible limitations of each.

1.8.1 IP PBX

The most common scenario is the installation of a new or the replacement of an existing PBX. If you compare Asterisk with some other alternatives, you will find it to be cheaper and richer in features then most available PBXs in the market. Several companies are now changing their specifications to Asterisk instead of other brand named PBXs.

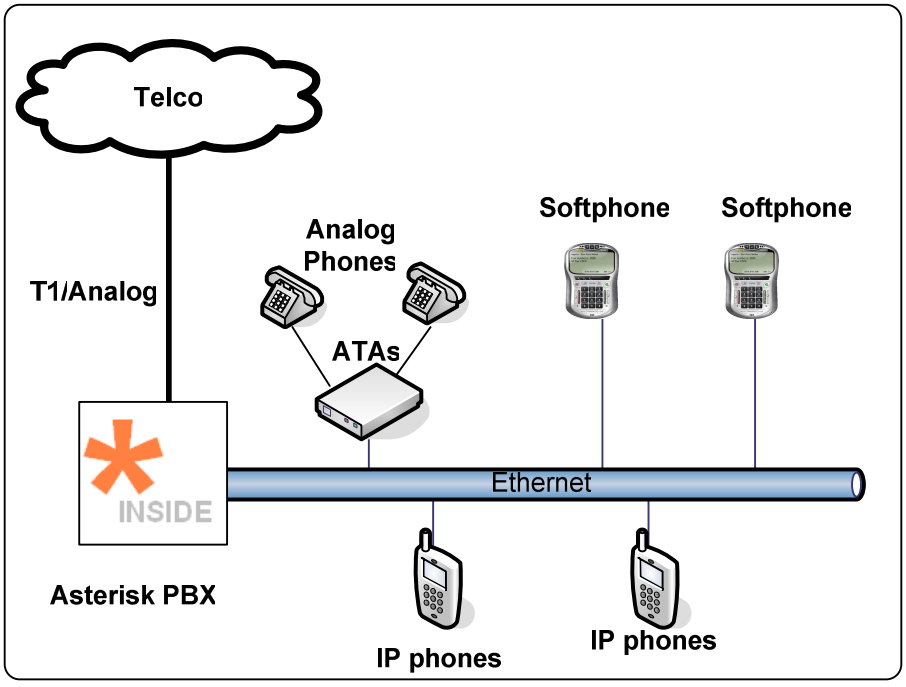


Figure 1.7 IP PBX

1.8.2 IP enabling legacy PBXs

The image below illustrates one of the most commonly used setups. Large companies usually don't want to take a high risk when investing in new technologies and, at the same time, wish to preserve their investments in legacy equipment. IP enabling a legacy PBX can be very expensive and, thus, connecting an Asterisk PBX using T1/E1 lines can be a good alternative to cost conscious costumers. Another benefit is to the possibility of connecting to a VoIP service provider with better telephony rates.

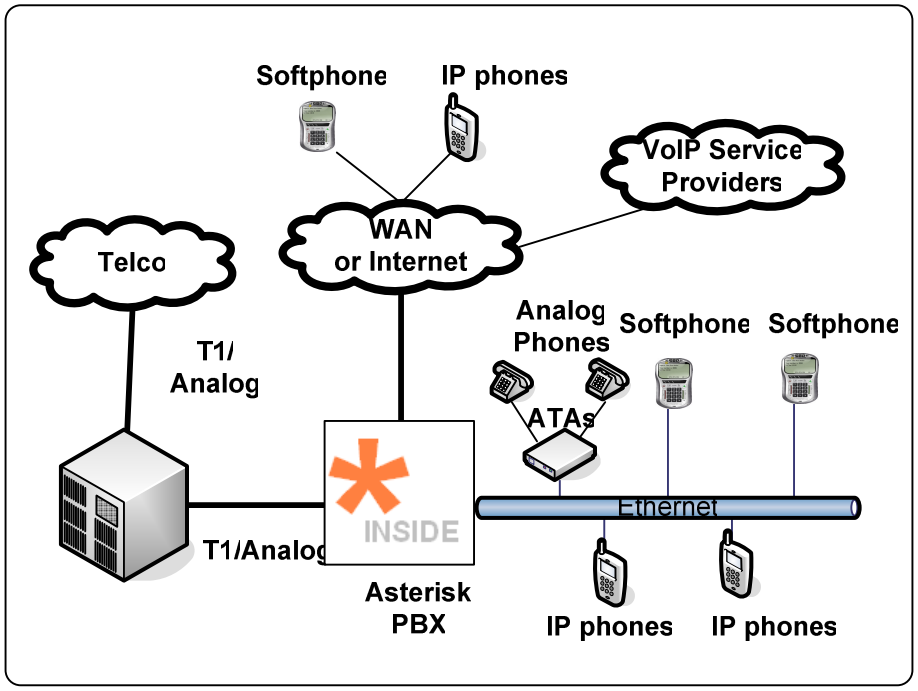


Figure 1.8 Integration with legacy PBX

1.8.3 Toll-Bypass

A very useful application for VoIP is connecting branch offices over the Internet or a WAN. Using an existing data connection allows you to bypass toll charges incurred in telecommunication connections between headquarters and branch offices.

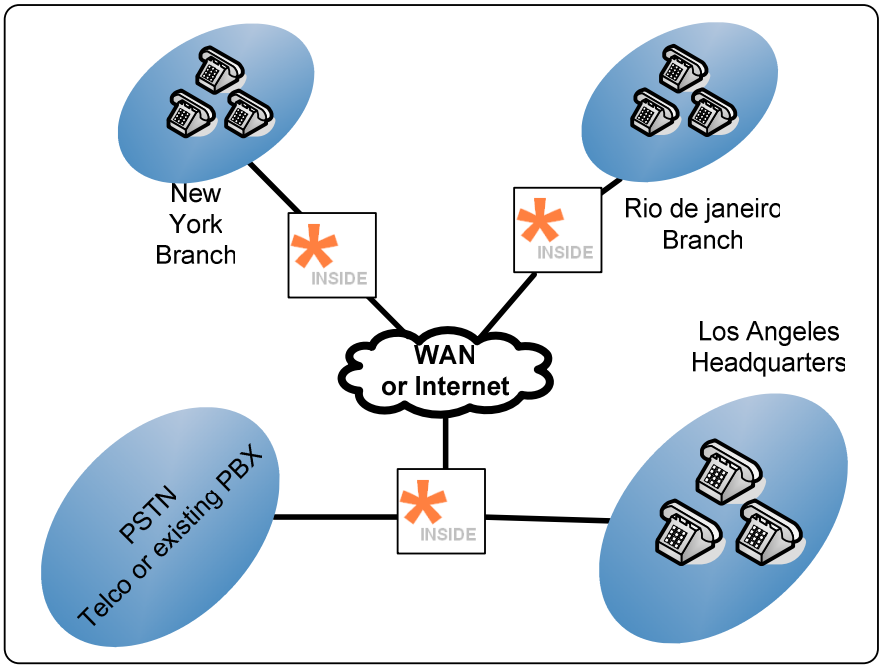


Figure 1.9 Toll Bypass

1.8.4 Application Server (IVR, Conference, Voice Mail)

Asterisk can also be used as an application server for the existing PBX or it can be directly connected to PSTN. Asterisk can do services like voicemail, fax reception, call recording, IVR connected to a database, and audio conferencing server. If you integrate voicemail and fax to existing e-mail you will end up having a unified messaging system, which is usuallyan expensive solution. Using Asterisk as an application server provides extreme cost reduction compared to most other solutions.

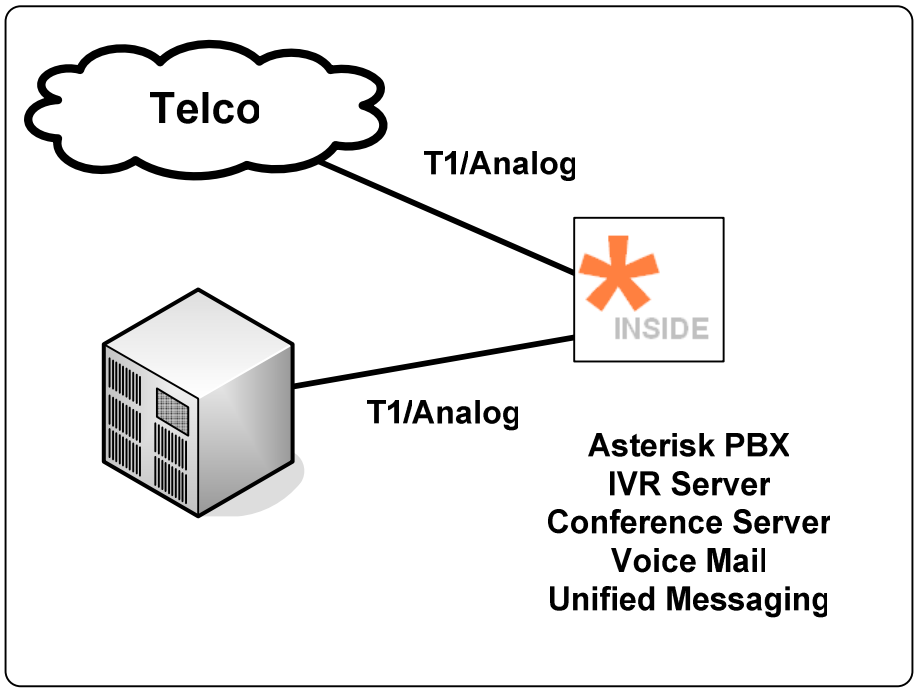


Figure 1.10 Asterisk as an application server

1.8.5 Media Gateway

Most voice over IP service providers use a SIP proxy to host all registering, location and authentication of SIP users. Anyway, they have to send the call to the PSTN directly or route it through a wholesale call termination provider using SIP or H.323 voice over IP connection. Asterisk can act as a B2BUA (back to back user agent) or Media Gateway, substituting very expensive soft switches or media gateways. Compare the price of a four E1/T1 gateway from the main market manufacturers with Asterisk. The Asterisk solution can cost several times less than other solutions and is capable to translate signaling protocols (H.323, SIP, IAX…) and codecs (G.711, G.729…).

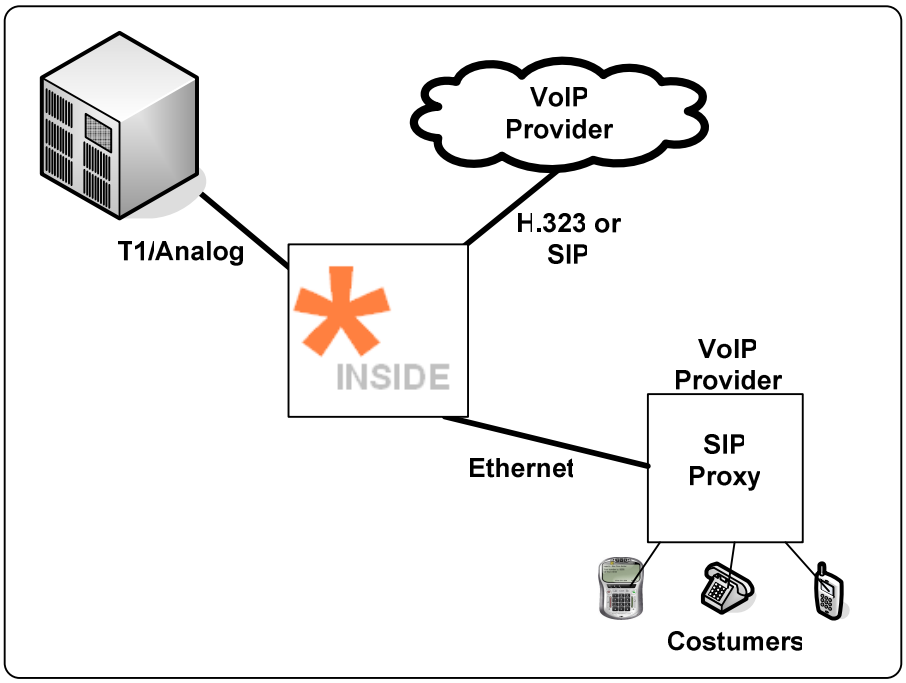


Figure 1.11 Asterisk as a media gateway

1.8.6 Contact Center Platform

A contact center is a very complex solution. It combines several technologies like ACD (automatic call distribution), IVR (interactive voice response), call supervision, and others. Basically, there are three types of contact centers: inbound, outbound, and blended. Inbound contact centers are very sophisticated. Usually they need ACD, IVR, CTI, recording, supervision, and reports. Asterisk has a built in ACD to queue the calls. IVR can be done using AGI (Asterisk Gateway Interface) or internal mechanisms like the application background. CTI (Computer telephony integration) is done using AMI (Asterisk Manager Interface); recording and reporting are built into Asterisk. For an outbound contact center, a predictive or power dialer is one of the main components. Although several dialers are available from the Asterisk open source, it is not hard to build your own for the platform if you so desire. A blended contact center permits simultaneous inbound and outbound operation, saving money by better usage of the agent's time. It's possible to use Asterisk and it's ACD mechanism to implement a blended solution.

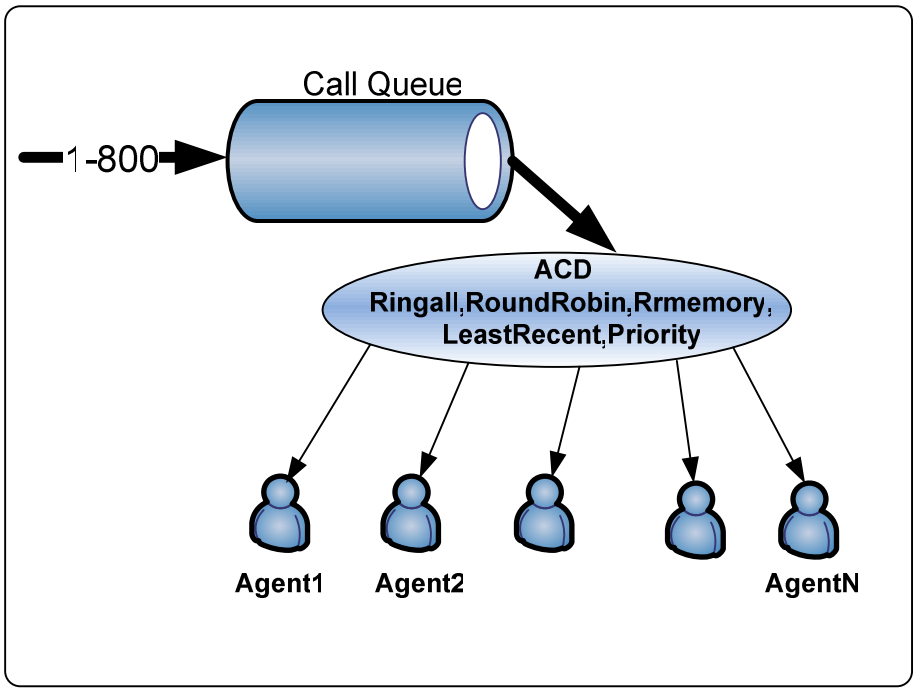


Figure 1-12 Asterisk as a Contact Center Platform

1.9 FINDING INFORMATION AND HELP

Some of the best places to find information about Asterisk are:

1.9.1 The official page about Asterisk

http://www.asterisk.org
http://www.asterisk.org/support/get-started
http://forums.digium.com/

1.9.2 Asterisk wikipedia

http://www.voip-info.org
www.voip-info.org/wiki-Asterisk

1.9.3 Asterisk mailing lists

http://www.asterisk.org/support/mailing-lists

1.9.4 Interesing sites

www.asteriskguru.com (Several tutorials)
www.sineapps.com (News about Asterisk)

1.10 SUMMARY

Asterisk is an Open Source software licensed according to the GPL that enables an ordinary PC to act as a powerful IP PBX platform. It was created by Mark Spencer from Digium, who sells the interface cards for Asterisk. Hardware is open source too andoriginated in the Zapata project developed by Jim Dixon. The Asterisk architecture has the following main components:

- **CHANNELS:** Analog, digital, or voice over IP.
- **PROTOCOLS:** Communications protocol can be SIP, H323, MGCP and IAX and they are responsible for signaling the calls.
- **CODECs:** Translate digital formats of voice allowing compressions, packet loss concealment, silence suppression and comfort noise generation. Asterisk does not support silence suppression.
- **APPLICATIONS:** Responsible for the Asterisk PBX functionality. Conference, voicemail, and fax are examples of Asterisk applications.

Asterisk can be used in several scenarios: from a small IP PBX to a sophisticated contact center.

1.11 QUESTIONS

1. Mark the correct answers. Asterisk has four basic architectural components.

a) Channels
b) Protocols
c) Agents
d) Phones
e) Codecs
f) Applications

2. If necessary, you can create an Asterisk PBX with four trunks and eight phones using three TDM400 cards. The first one with four FXO channels and the other two with four FXS channels each. This affirmative is:

a) False
b) True

3. A FXS channel generates a dialing tone, while a FXO channel receives a dialing tone from the PSTN or another PBX. The affirmative is:

a) False
b) True

4. Mark the correct answers. Asterisk allows the use of the following features:

a) IVR (Interactive Voice Response)
b) ACD (Automatic Call Distribution)
c) IP Phones
d) Analog phones
e) Digital phones from any vendor

5. To play music on hold, Asterisk needs an external player like a MP3 or CD player. The affirmative is:

a) False
b) True

6. This technology is responsible for automatic answering of costumers. Usually plays a "prompt" and wait an option dialed by the user. In some cases can be integrated with a database to provide information by the telephone using text-to-speech technology. It is named _____.

a) CTI
b) IVR
c) DAC
d) Unified Messaging

7 – An E1 trunk supports ___ voice channels while a T1 trunk supports ___ voice channels.

a) 12, 24
b) 30, 24
c) 12, 12
d) 30, 23

8 – In traditional PBXs, usually ACD, IVR and voicemail are included in the PBX together with their respective licensing. This affirmative is:

a) False
b) True

9 – It is possible to connect several branches using voice over IP, thereby reducing long distance toll rates imposed by telephony companies. Asterisk can act at a branch as:

a) The PBX for all users
b) The media gateway connection to an existing PBX
c) The only thing possible is to use IP phones or ATAs connected to a centralized Asterisk.
d) Resilience and robustness are not important when you connect IP phones.

10 – Asterisk can be used as a contact center platform. What are the three main types of contact centers?

a) External
b) Internal
c) Inbound
d) Outbound
e) Blended

Chapter 2

Downloading and installing Asterisk

In this Chapter, we will cover downloading, installation and configuration of Asterisk.

2.1 OBJECTIVES

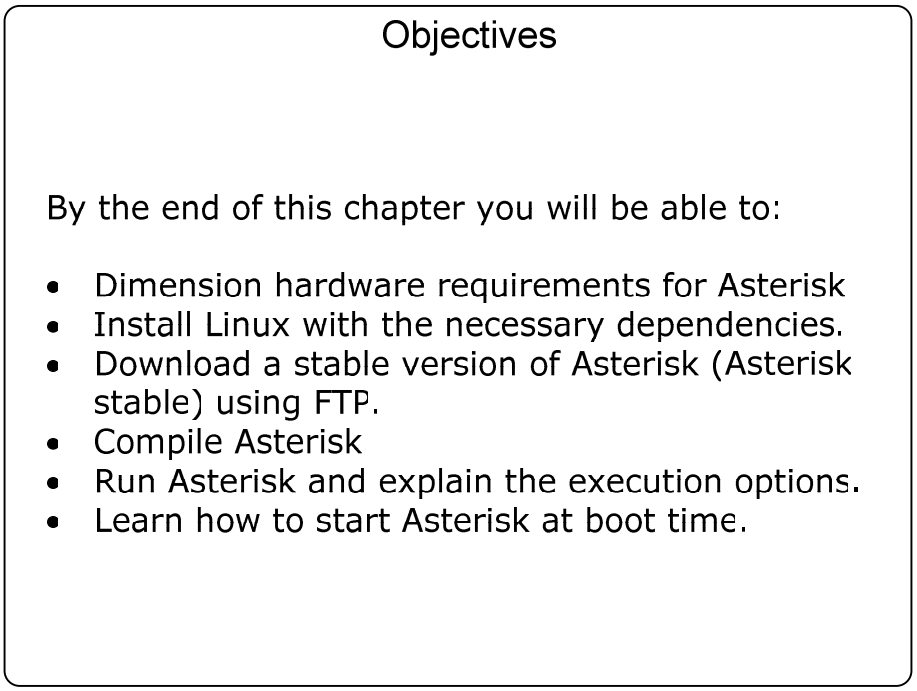


Figure 2.1 Objectives

2.2 INTRODUCTION

Now you will learn how to prepare your system for an Asterisk installation. Asterisk runs in multiple operating systems, but we chose to keep things simple and start with a single operating system: Linux. We chose the Debian distribution because the dependencies are easy to install, this distribution is stable, and has a low footprint. Other distributions can be used as well.

2.3 MINIMUM HARDWARE

Asterisk is CPU intensive because it uses the PC processor to process voice instead of dedicated DSPs. The most important parameter for CPU dimensioning is the number of simultaneous calls. If you are building a

complex system with high load, it is important to keep these concepts in mind. To build a simple system, a Pentium class processor with more than 300 Mhz and 256 Mbytes of RAM memory is fine. Asterisk does not require too much space on disk (around 100 Mbytes of code, including compiled and source). This should allow enough space for voice-mail and custom prompts.

If you are using VoIP, no other hardware is necessary. You can use free softphones from counterpath or SJLABS to connect to VoIP providers. A good list of providers can be found in http://www.voipcharges.com.

Caution:
Some Asterisk applications, like meet-me and music on hold, require a clock source. Usually, the clock source of Asterisk is the TDM interface card. If your system does not use a TDM interface card, you will have to load ztdummy to provide a clock.

2.3.1 Hardware Assembling

The asterisk hardware does not need to be sophisticated. You don't need an expensive video board or too many peripherals. USB, serial and parallel ports should be disabled to avoid consuming unnecessary interrupts. A good network interface card is important. Take special care if you are using TDM interface cards. Some cards use a 3.3 volts PCI bus and it is not easy to find mother boards for them.

2.3.2 IRQ sharing

Telephony interface cards like X100P can generate large quantities of interruptions. Serving these interruptions takes processor time. The drivers can't do this processing if you have another device using the same interruption. In a single CPU system, you should avoid IRQ sharing between devices. We recommend the use of a dedicated hardware for running Asterisk. Don't forget to disable any foreign or unnecessary hardware. Some hardware can be disabled in the motherboard bios setup.

Once you have started your computer, see your assigned interrupts in /proc/interrupts.

```
#
cat /proc/interrupts
CPU0
0: 41353058 XT-PIC timer
1: 1988 XT-PIC keyboard
2: 0 XT-PIC cascade
3: 413437739 XT-PIC wctdm <-- TDM400
4: 5721494 XT-PIC eth0
```

```
7: 413453581 XT-PIC wcfxo <-- X100P
8: 1 XT-PIC rtc
9: 413445182 XT-PIC wcfxo <-- X100P
12: 0 XT-PIC PS/2 Mouse
14: 179578 XT-PIC ide0
15: 3 XT-PIC ide1
NMI: 0
ERR: 0
```

Above, you can see three Digium boards, each in their own IRQ. If this is your case, go ahead and install the hardware drivers. If this is not your case, go back and try something else to avoid IRQ sharing.

2.4 CHOOSING AN OPERATING SYSTEM

Asterisk was initially developed to run on Linux. However, it can also run on BSD Unix or Mac OS X. If you are new to Asterisk, try using Linux first since it is much easier. If you were to try BSD Linux or Mac OS X, I forewarn you that you this operating system presents many challenges to run Digium hardware.

2.4.1 Linux distribution

Several Linux distributions were successfully tested with Asterisk (e.g. Fedora, Redhat, SuSe, Debian, Gentoo, and others). Choose yours. We have chosen Debian Sarge 3.1 and this distribution can be downloaded from http://www.us.debian.org/CD/netinst/#netinst-stable.

2.4.2 Necessary packages

Required by Asterisk:

- readline, readline-devel (required before 1.2)
- bison (Required before 1.2)
- openssl, openssl-dev
- termcap
- ncurses-devel
- zlib-devel

Required by zaptel

- Kernel sources

Caution: The zaptel packages are necessary to compile some Asterisk applications like meetme(). If you compiled Asterisk before

zaptel, you will have to recompile Asterisk to compile meetme() and other dependent applications.

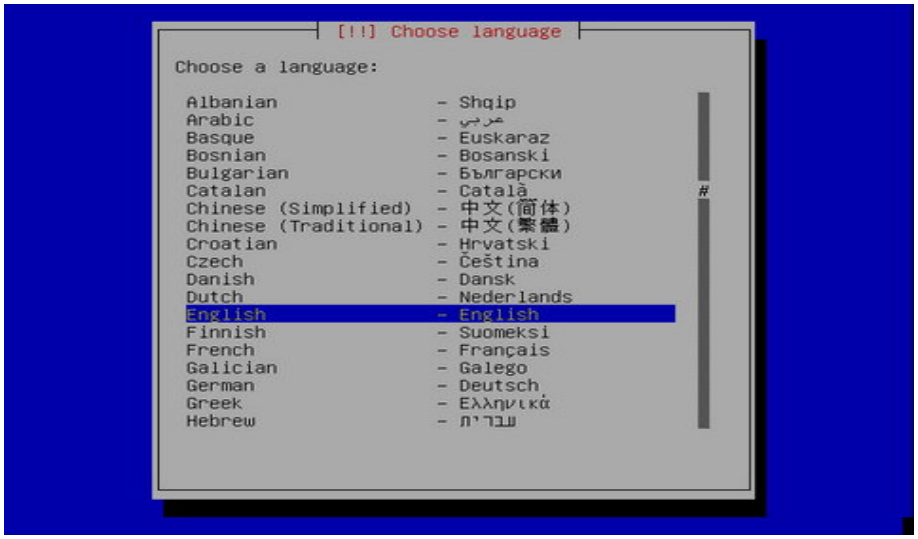
2.5 INSTALLING LINUX PREPARED FOR ASTERISK

We used Debian with Kernel 2.6 to install Asterisk. We have chosen this distribution because it's popular, easy to work with, and it is supported by Digium. The installation steps are described below. You can download Debian Sarge version 3.1 from www.debian.org.

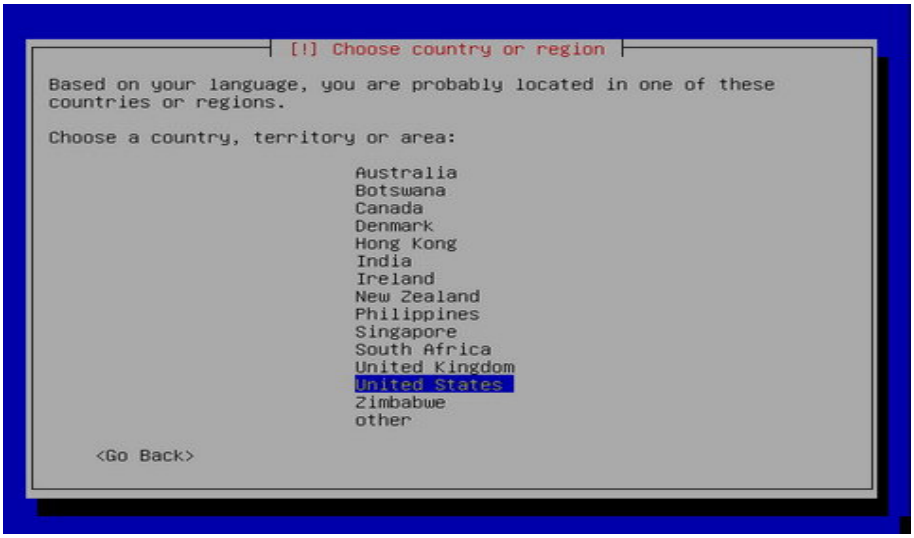
Caution: This installation will format your PC. All your disk data will be erased. Please make sure to backup all your data before you start.

Step 1: Put the CD in the CD-ROM drive and boot your PC. Use the option linux26 to boot with linux kernel version 2.6.

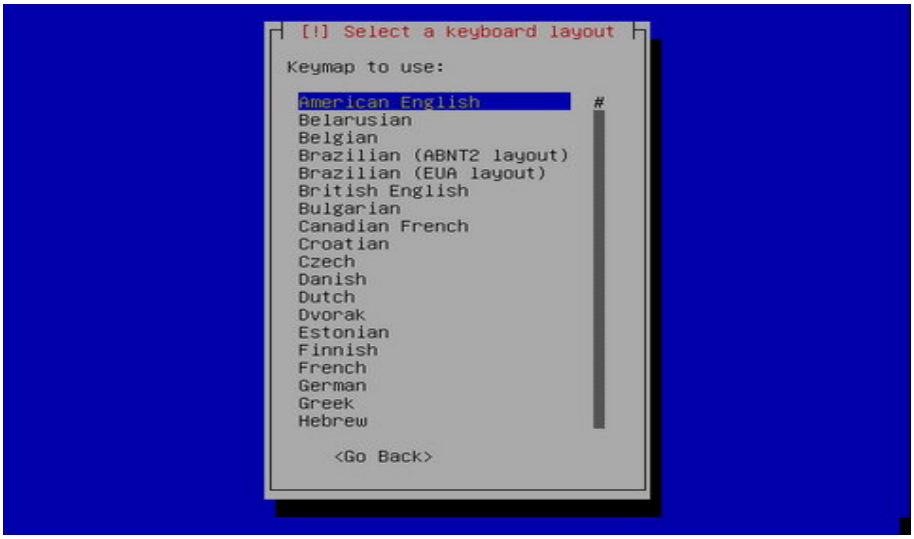
Step 2: Select the language



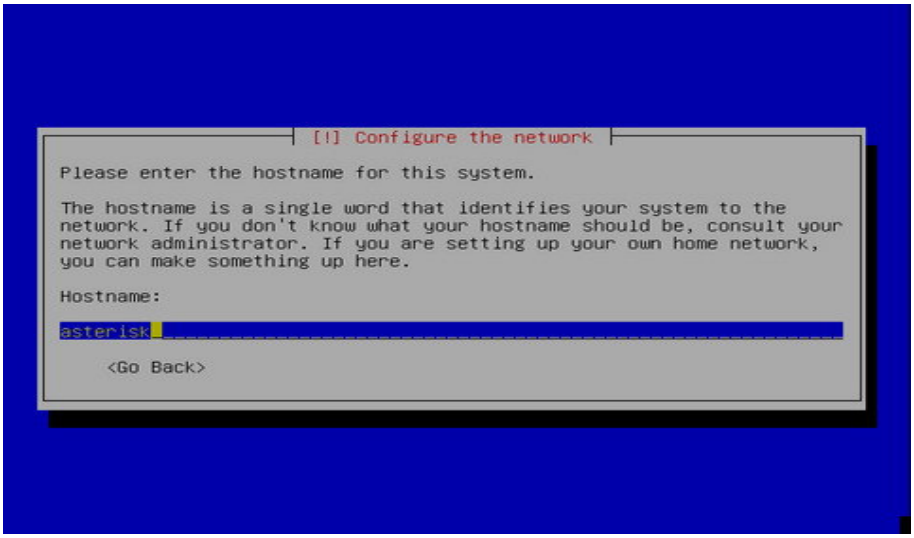
Step 3: Select the country



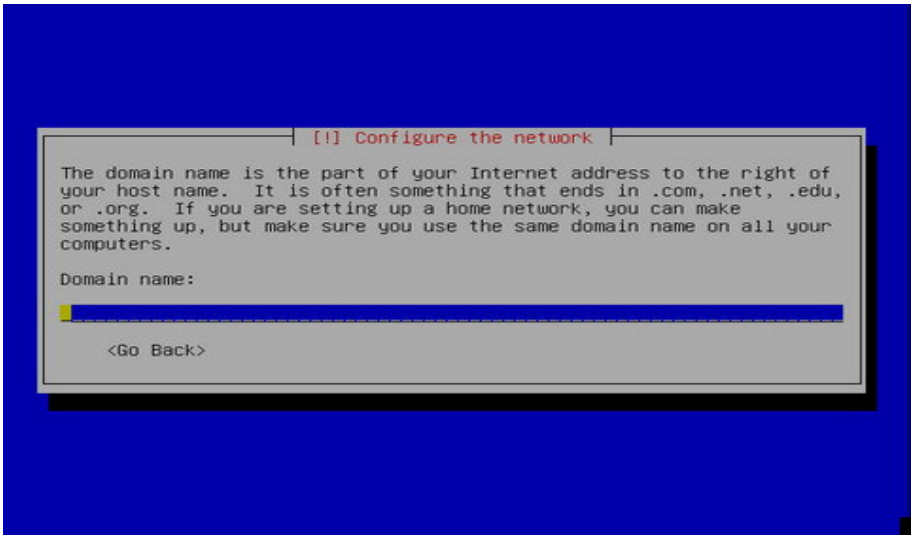
Step 4: Choose the keyboard layout.



Step 5: Type the host name (e.g. asterisk)

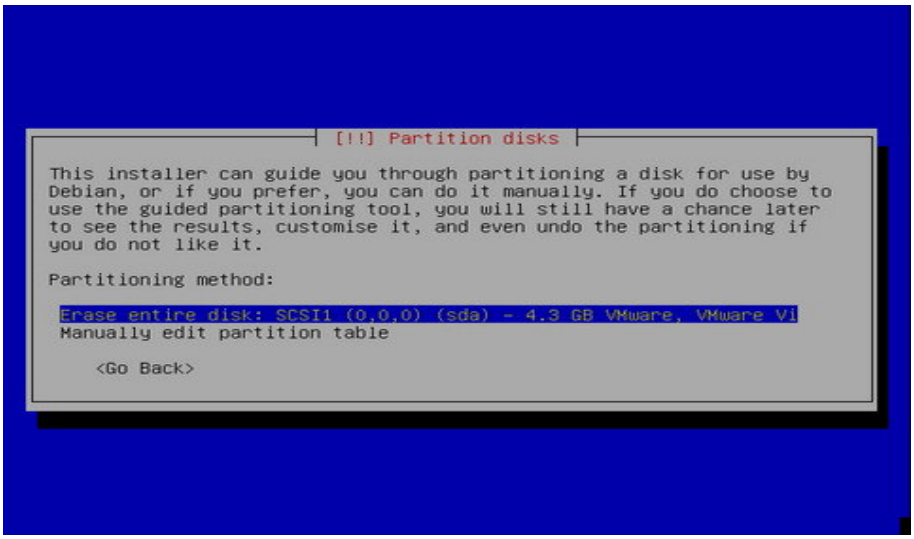


Step 6: Type the domain for this equipment (e.g. astersikguide.com).

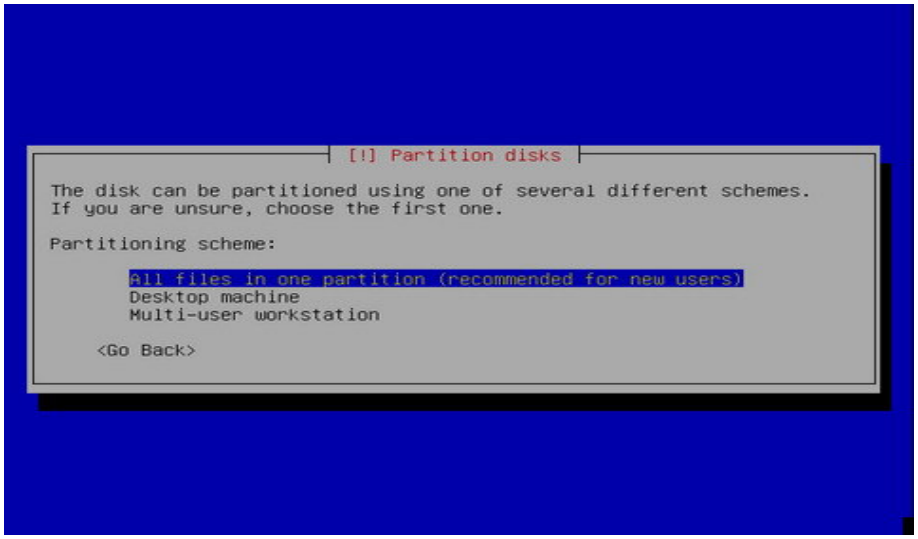


Step 7: Now you will partition the disks. Please confirm that you erased the entire disk.

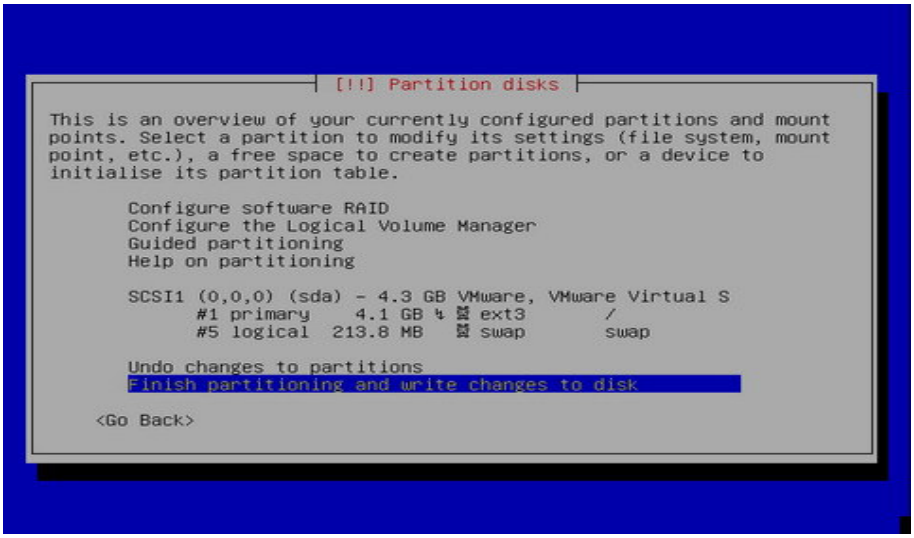
Caution!
All your data will be erased. Backup your data before proceeding.



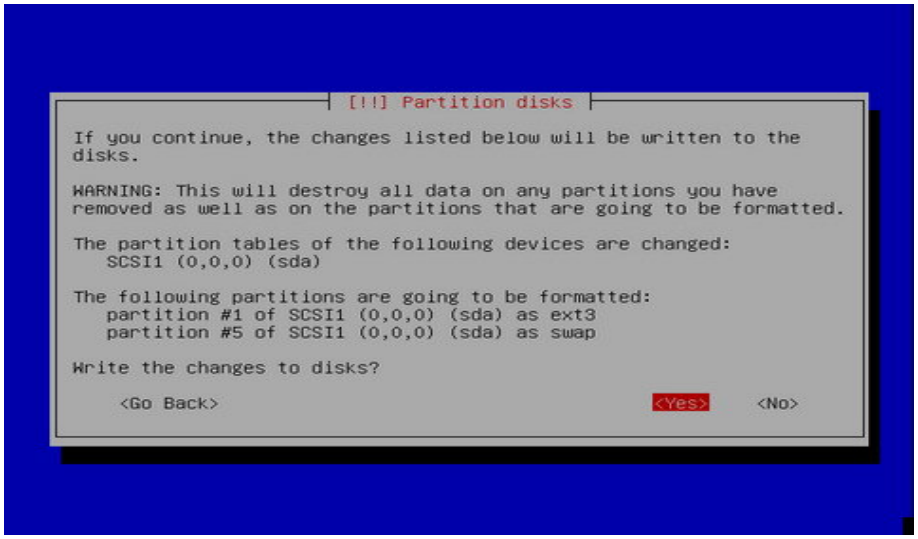
Step 8: Choose "all files in one partition"



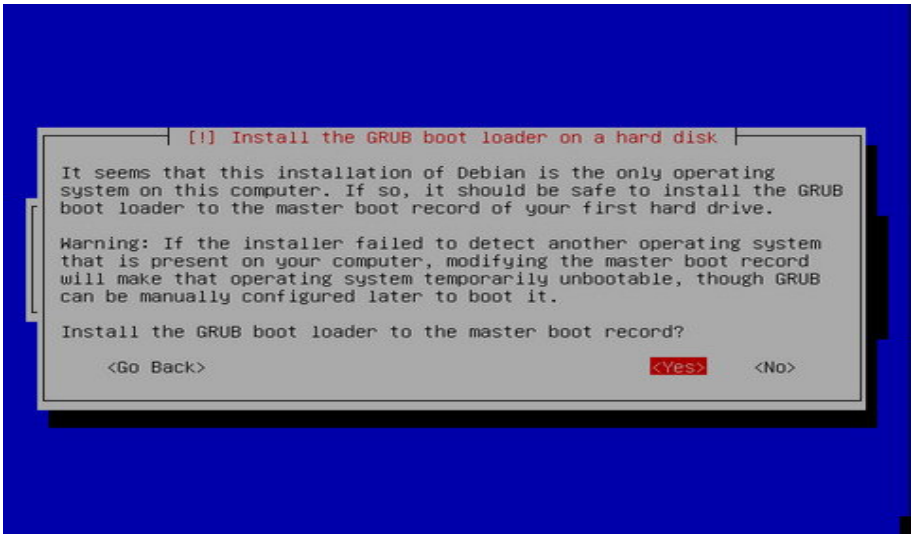
Step 9: Choose "finish partitioning and write changes to disk"



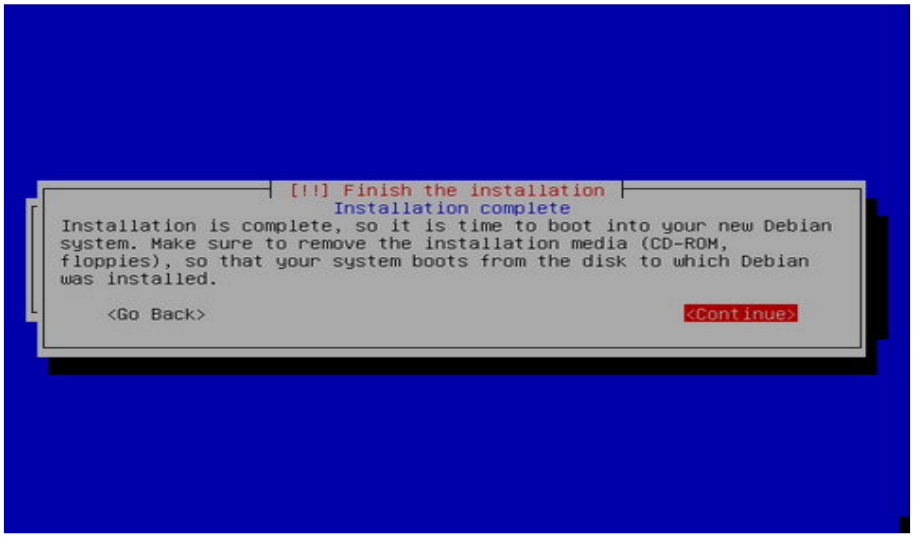
Step 10: Confirm again.



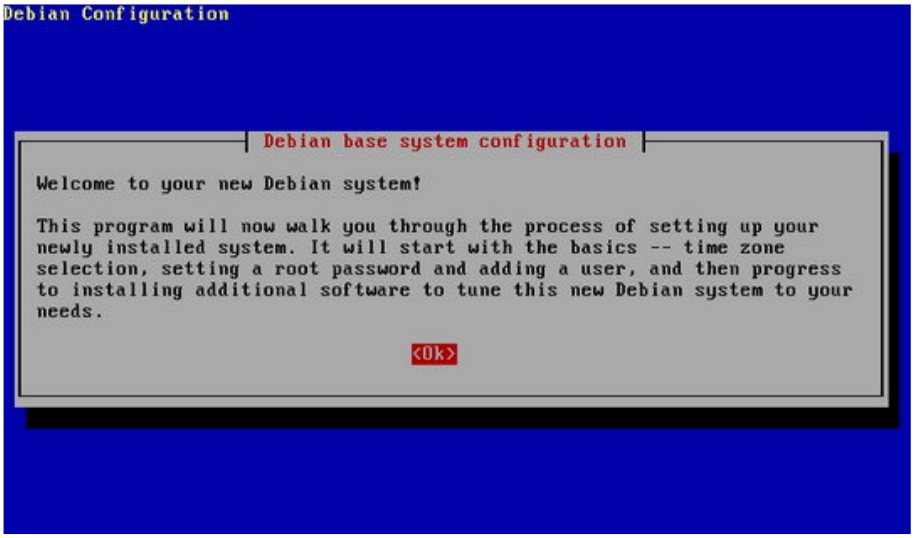
Step 11: Accept GRUB boot loader installation



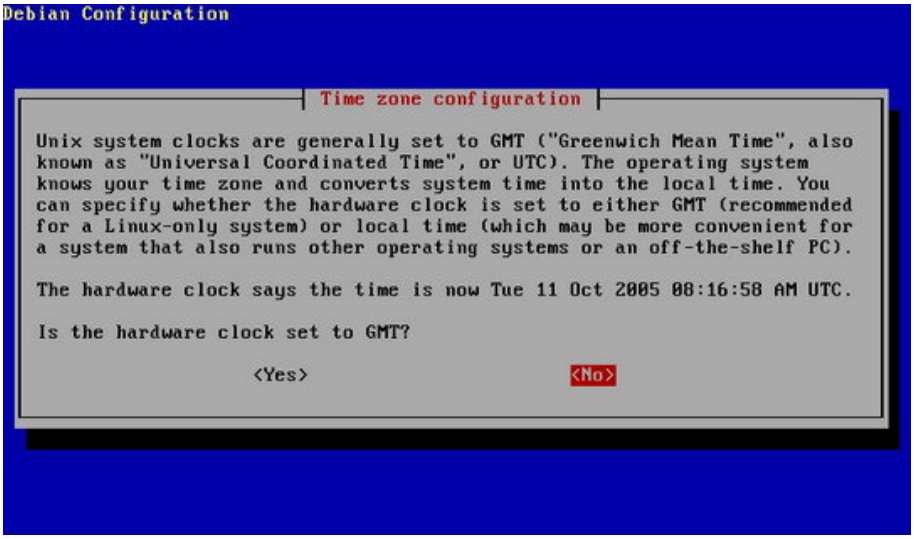
Step 12: Finish the installation



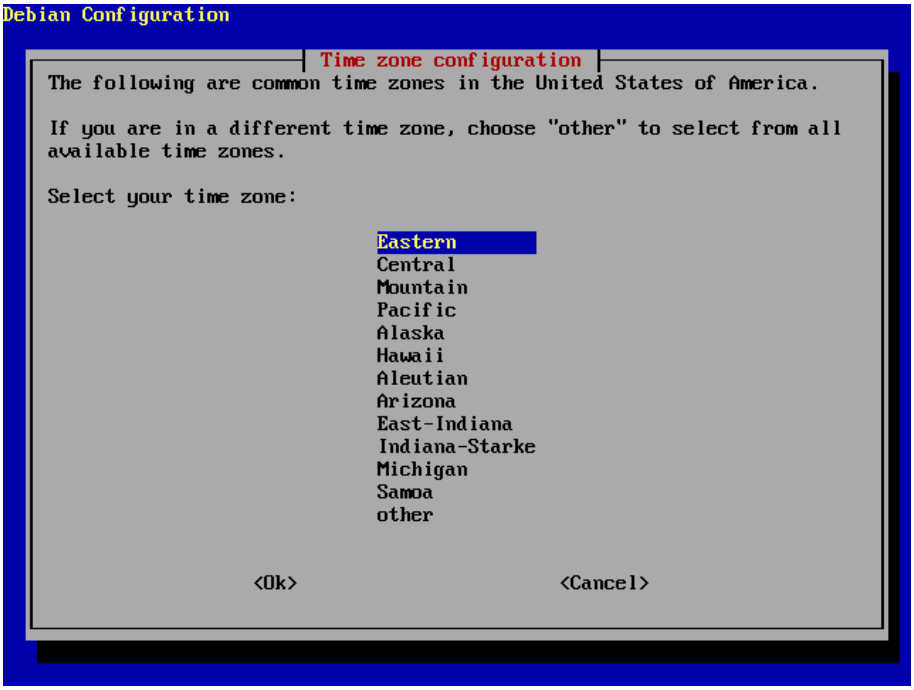
Step 13: Remove the CD, reboot the PC and click OK when the screen depicted below appears.



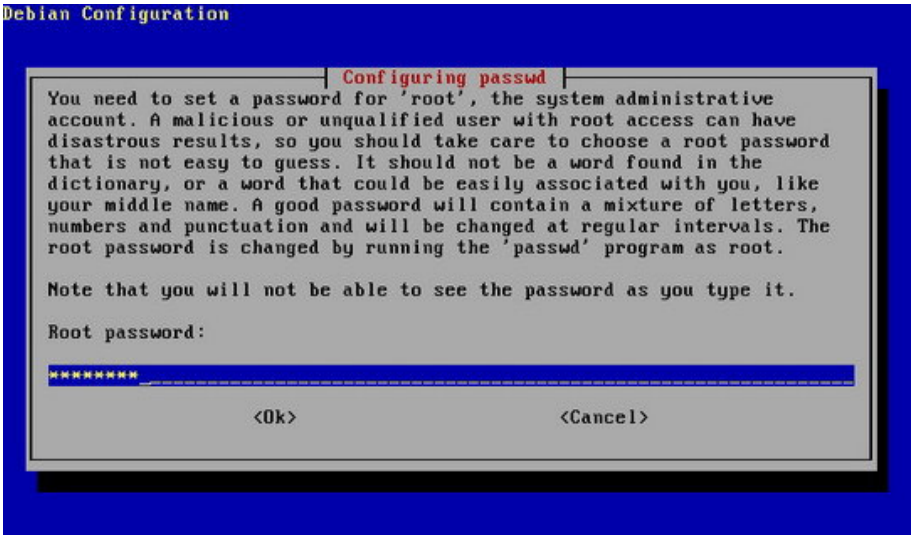
Step 14: Select "No" just in case the hardware clock is not set to GMT.



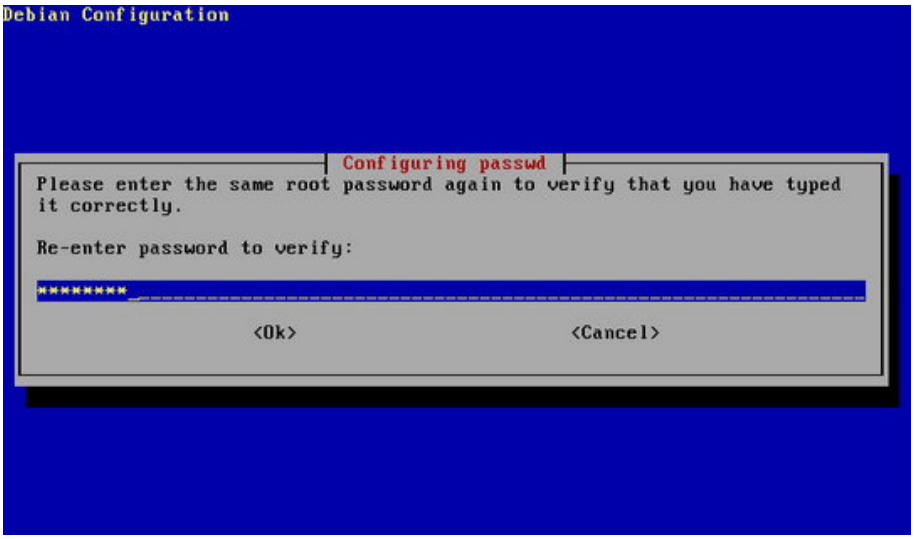
Step 15: Select the appropriated Time zone



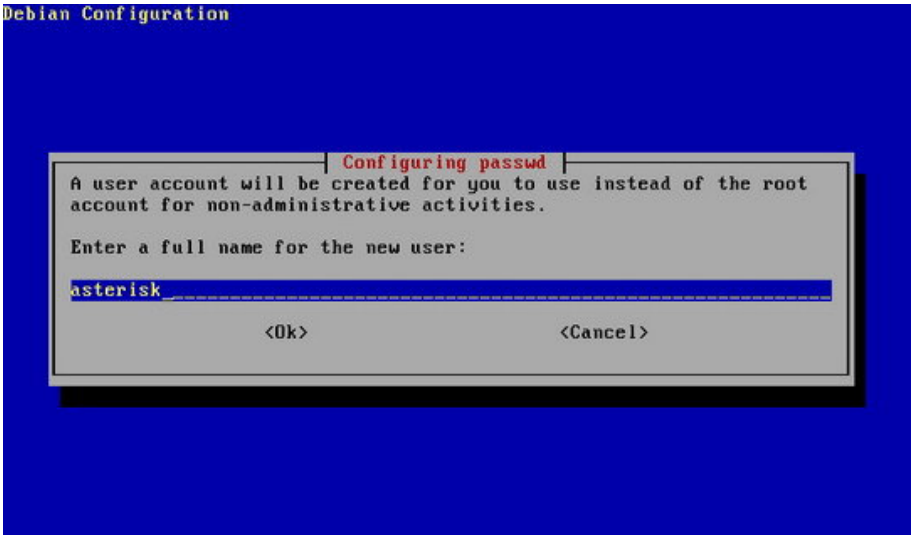
Step 16: Type “asterisk” as the root password.



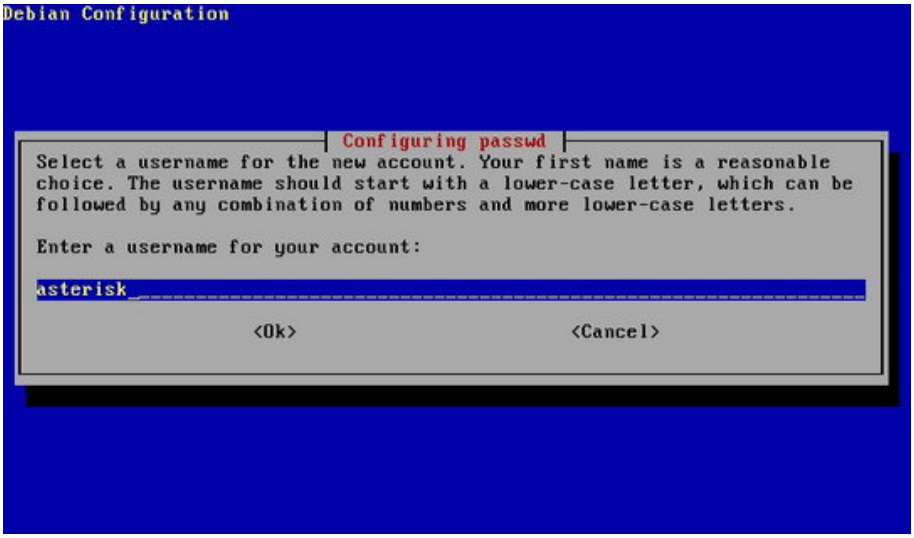
Step 17: Re-enter for confirmation



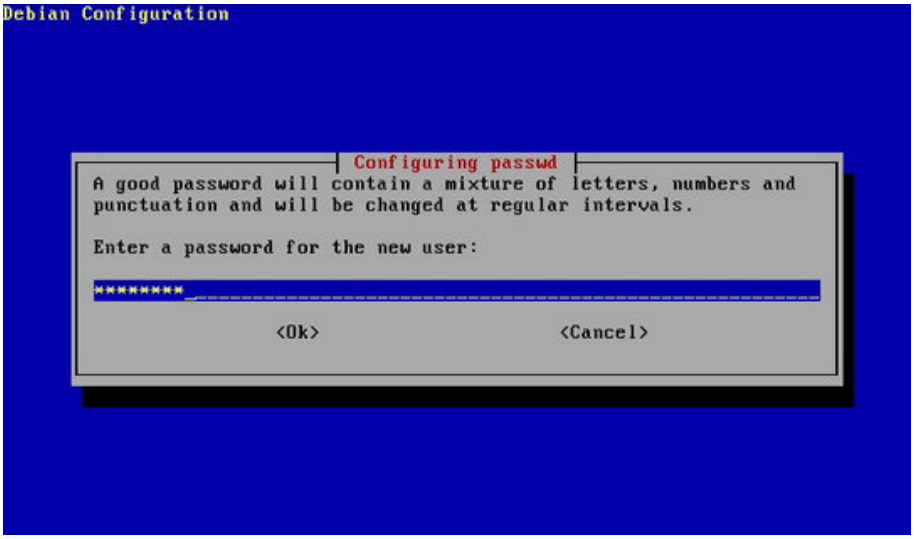
Step 18: Create a user named “asterisk”.



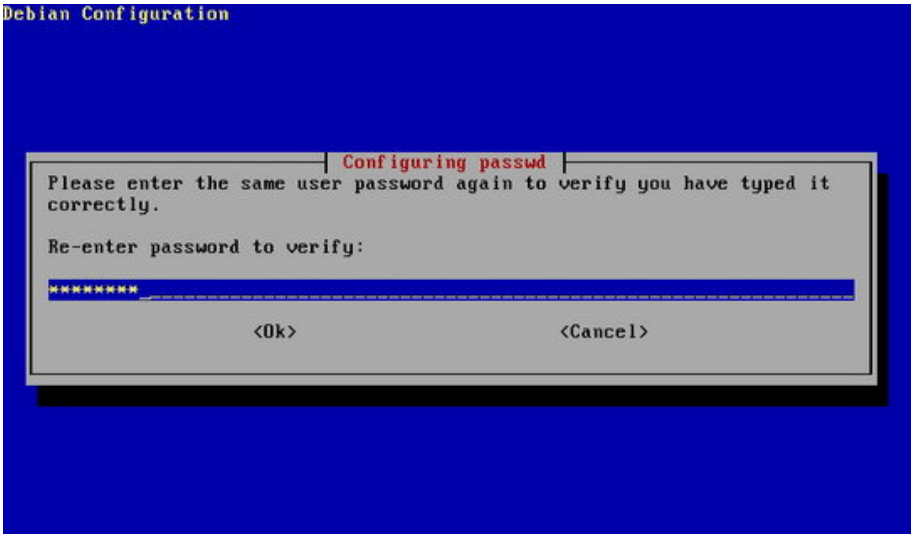
Step 19: Enter the username “asterisk” again



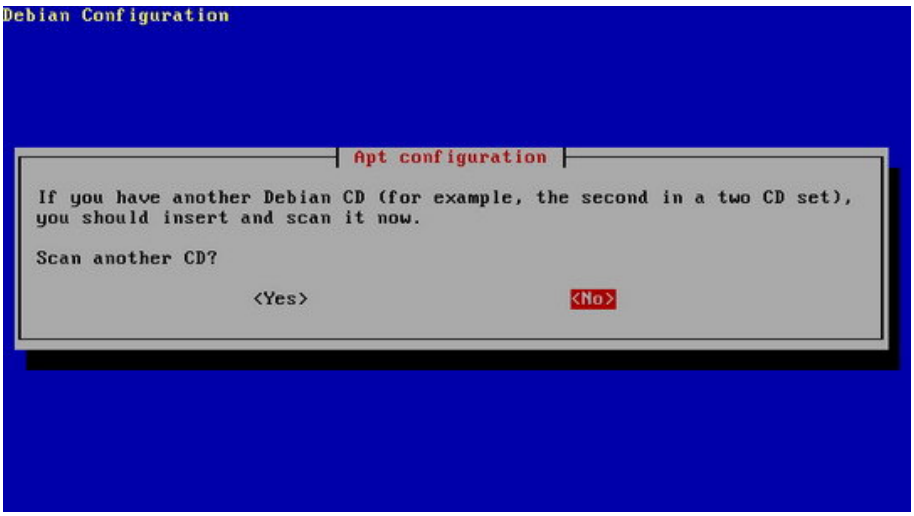
Step 20: Type "asterisk" as the password for user "asterisk"



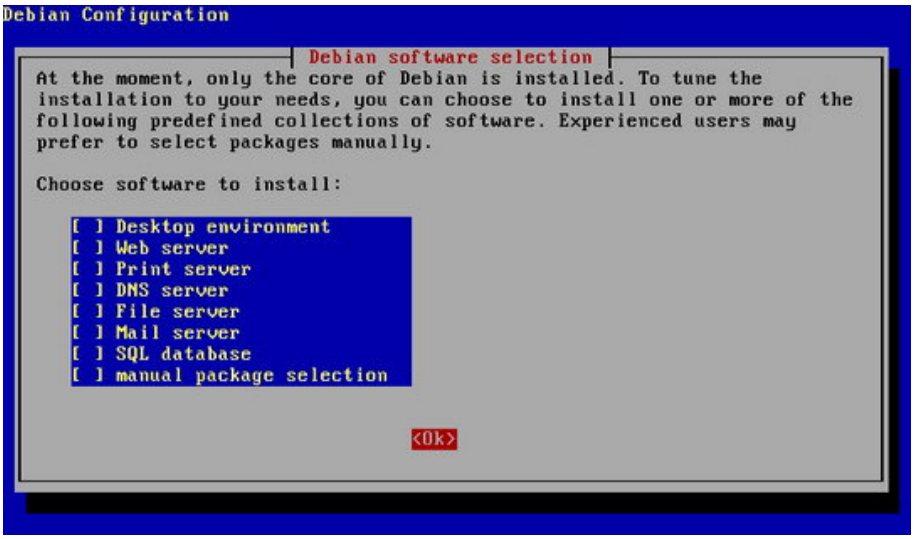
Step 21: Re-enter password for verification



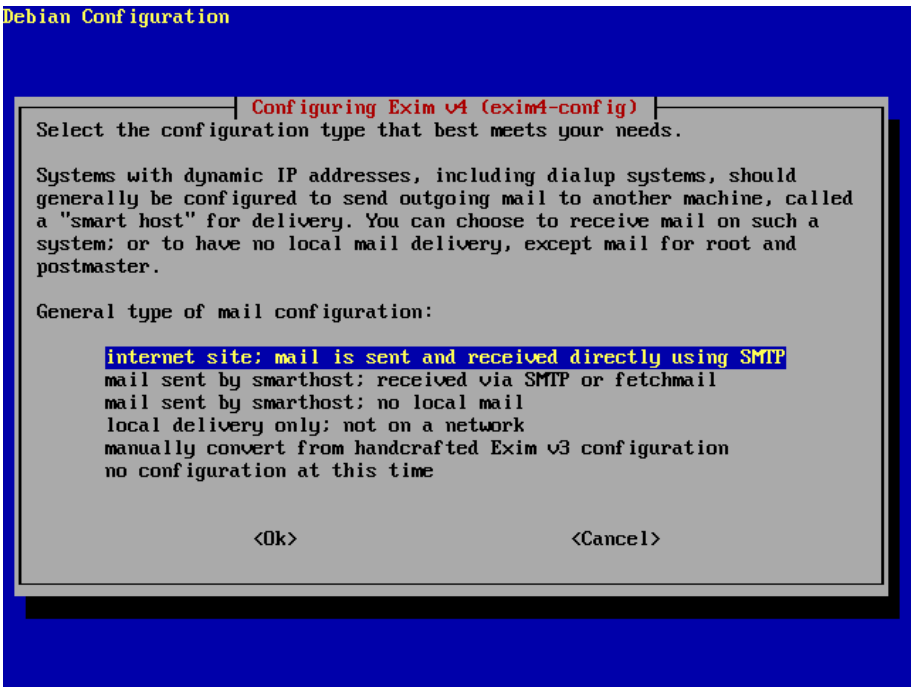
Step 22: Answer “no” for the question “scan another CD?”



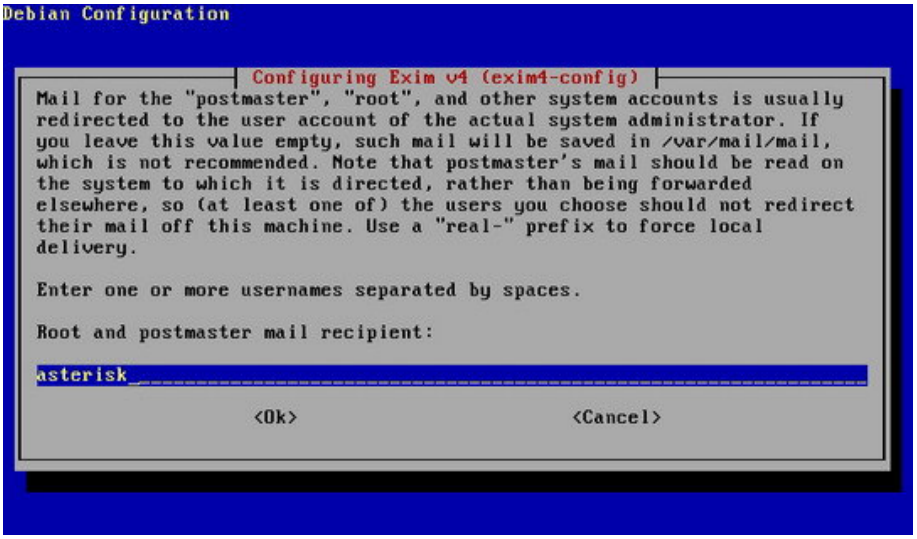
Step 23: We will use this server just for Asterisk. Leave all options blank.



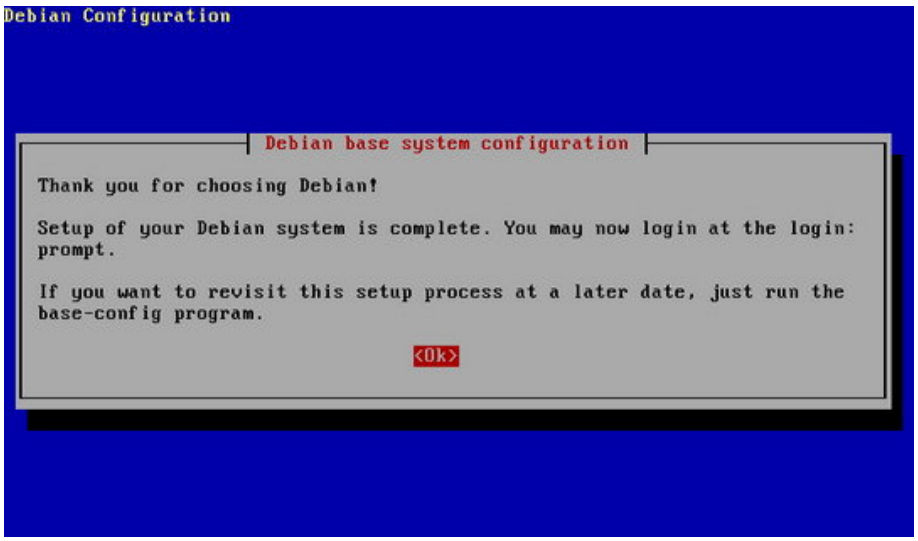
Step 24: Choose internet site; mail is sent and received using SMTP. This is necessary for integration of voice-mail and e-mail.



Step 25: Use the asterisk account for the mail server



Step 26: End



2.6 Preparing the Debian system for Asterisk

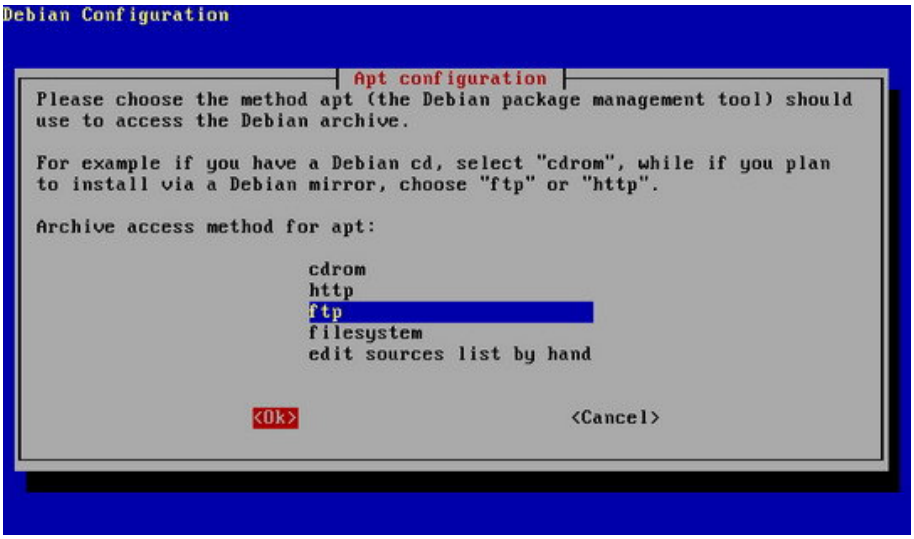
The Debian installation is now complete. Let's now install the packages necessary for the subsequent compilation of Asterisk and Zaptel drivers. We will first indicate to Debian where the packages are going to be downloaded from. This is done by using the apt-setup utility.

Step 1: Log in as root

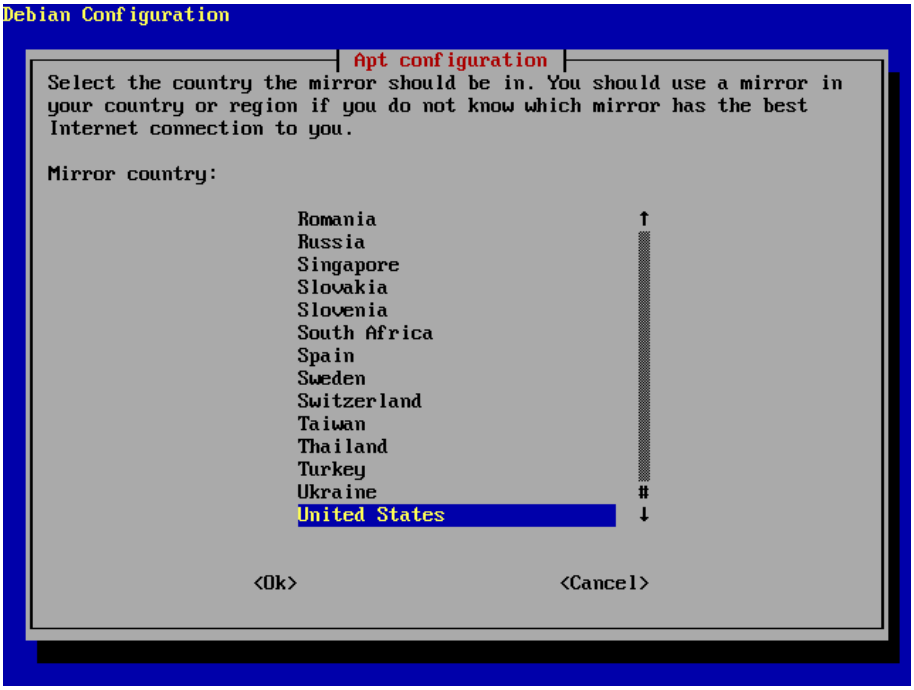
Step 2: Add a source to download the packages (/etc/apt/sources.list)

```
#apt-setup
```

Step 3: Select FTP.

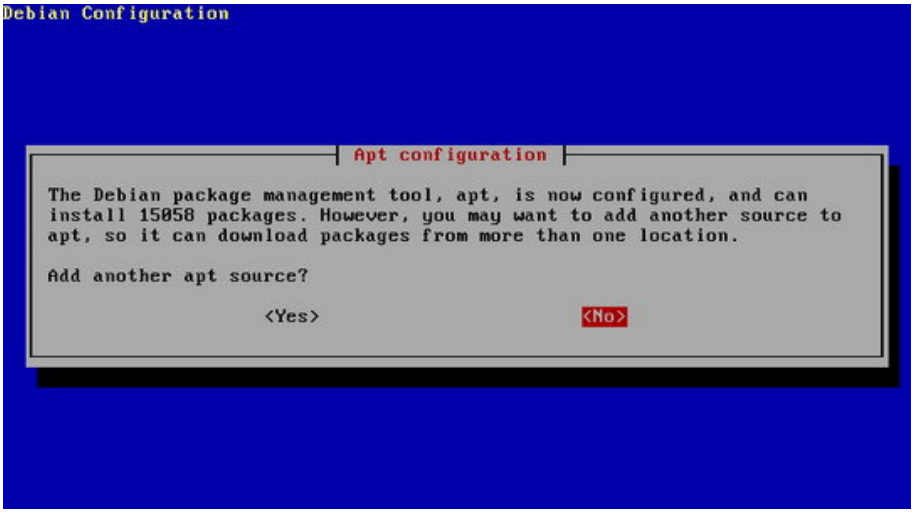


Step 4: Select the appropriate country



Step 5: Choose the appropriate mirror

Step 6: Select <no> to another apt source.



Step 7: To install the kernel headers for zaptel compilation, type:

```
#apt-get install kernel-headers-`uname –r`
#ln -s /usr/src/kernel-headers-`uname -r` /usr/src/linux
```

Step 8: To install the necessary packages, type:

```
#apt-get install bison openssl libssl-dev libasound2-dev libc6-dev libnewt-dev libncurses5-dev zlib1g-dev gcc g++ make
```

2.7 OBTAINING AND COMPILING ASTERISK

Linux is now installed. The next step is the Asterisk installation.

2.7.1 Obtaining Asterisk sources

To obtain Asterisk sources and zaptel drivers you should download them from Digium.com. We will use the wget utility to download them. Create a directory /usr/src to receive the files. You should consult www.asterisk.org to verify which are the latest versions.

Where ´x´ is the latest software revision, type:

```
# cd /usr/src
#wget http://ftp.digium.com/pub/zaptel/zaptel-1.4.x.tar.gz
#wget http://ftp.digium.com/pub/libpri/libpri-1.4.x.tar.gz
#wget http://ftp.digium.com/pub/asterisk/asterisk-addons-1.4.x.tar.gz
#wget http://ftp.digium.com/pub/asterisk/asterisk-1.4.x.tar.gz
```

Uncompress the files using:

```
# tar –xzvf asterisk-1.4.x.tar.gz
# tar –xzvf libpri-1.4.x.tar.gz
# tar –xzvf asterisk-addons-1.4.x.tar.gz
# tar –xzvf zaptel-1.4.x.tar.gz
```

2.7.2 Compiling Zaptel drivers

You will need to compile the Zaptel modules. The commands './configure' and 'make menuselect' were added on version 1.4. The last one allows you to select which utilities and modules to build.

The following commands will do it:

```
cd /usr/src/zaptel-1.4.x/
make clean
./configure
make menuselect
make install
make install-udev              ;Use to create zaptel devices using udev daemon
make config                    ;Use to start zaptel at boot time
update-rc.d zaptel defaults 99       ;Use to start zaptel at boot time
```

Use make menuselect to install only the necessary modules.

'make menuselect' screenshot

2.7.3 Compiling Asterisk

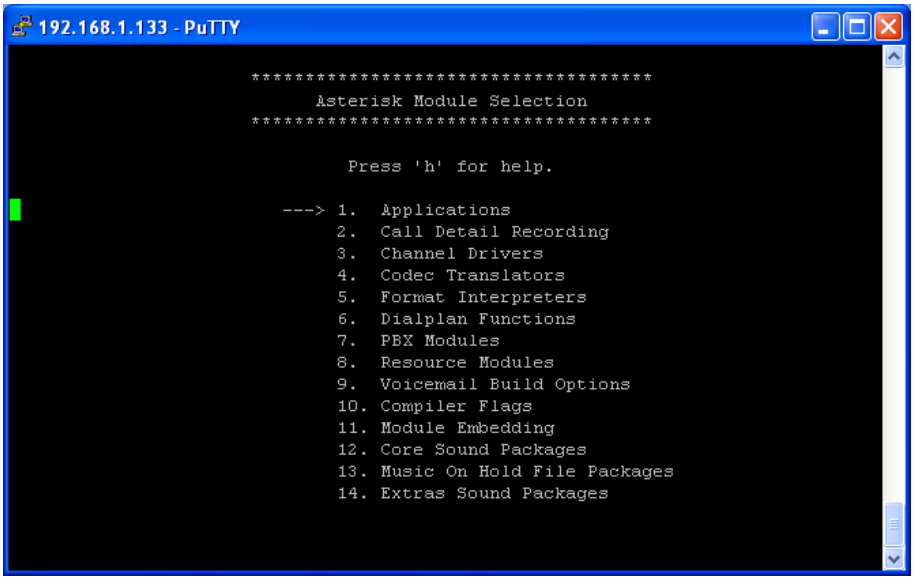
If you have compiled software before, compiling Asterisk will be an easy task. Run the following commands to compile and install Asterisk. Once again you can choose which applications and modules to build using 'make menuselect'.

```
cd /usr/src/libpri-1.4.x
make clean
make
make install
```

```
cd /usr/src/asterisk-1.4.x
make clean
./configure
make menuselect
make
make install
make samples                ;use to create sample configuration files
make config                 ;to start asterisk at boot time
```

Use make menuselect to install only the necessary modules.

'make menuselect' screenshot:



2.8 STARTING AND STOPPING ASTERISK

With this minimal configuration, it's possible to start Asterisk successfully.

```
/usr/sbin/asterisk –vvvgc
```

Use the CLI command **stop now** to shutdown Asterisk.

```
CLI>stop now
```

2.8.1 Asterisk runtime options

The Asterisk starting process is very simple. If Asterisk is run without any parameters, it is launched as a daemon.

```
/sbin/asterisk
```

You can access Asterisk console by executing the command below. Please note that more than one console process can be run at the same time.

```
/sbin/asterisk -r
```

2.8.2 Available runtime options for Asterisk

You can show the available runtime options using 'asterisk –h'

```
debian:/usr/src/asterisk-1.4.1# asterisk -h
Asterisk 1.4.1, Copyright (C) 1999 - 2006, Digium, Inc. and others.
Usage: asterisk [OPTIONS]
Valid Options:
   -V                Display version number and exit
   -C <configfile>   Use an alternate configuration file
   -G <group>        Run as a group other than the caller
   -U <user>         Run as a user other than the caller
   -c                Provide console CLI
   -d                Enable extra debugging
   -f                Do not fork
   -F                Always fork
   -g                Dump core in case of a crash
   -h                This help screen
   -i                Initialize crypto keys at startup
   -I                Enable internal timing if Zaptel timer is available
   -L <load>         Limit the maximum load average before rejecting new calls
   -M <value>        Limit the maximum number of calls to the specified value
   -m                Mute the console from debugging and verbose output
   -n                Disable console colorization
   -p                Run as pseudo-realtime thread
   -q                Quiet mode (suppress output)
   -r                Connect to Asterisk on this machine
   -R                Connect to Asterisk, and attempt to reconnect if disconnected
   -t                Record soundfiles in /var/tmp and move them where they belong after
they are done.
   -T                Display the time in [Mmm dd hh:mm:ss] format for each line of output
to the CLI.
   -v                Increase verbosity (multiple v's = more verbose)
   -x <cmd>          Execute command <cmd> (only valid with -r)
```

2.9 Installation Directories

Asterisk is installed on several directories. These directories can be modified in the asterisk.conf file.

asterisk.conf

```
[directories]
astetcdir => /etc/asterisk
astmoddir => /usr/lib/asterisk/modules
astvarlibdir => /var/lib/asterisk
astdatadir => /var/lib/asterisk
astagidir => /var/lib/asterisk/agi-bin
astspooldir => /var/spool/asterisk
astrundir => /var/run
astlogdir => /var/log/asterisk

; Changing the following lines may compromise your security.
;[files]
;astctlpermissions = 0660
;astctlowner = root
```

```
;astctlgroup = apache
;astctl = asterisk.ctl
;[options]
;internal_timing = yes
```

2.10 LOG FILES AND LOG ROTATION

Asterisk PBX logs their messages on the /var/og/asterisk directory. The file that controls the logs is the logger.conf.

```
; Logging Configuration
;
; In this file, you configure logging to files or to
; the syslog system.
;
; "logger reload" at the CLI will reload configuration
; of the logging system.

[general]
; Customize the display of debug message time stamps
; this example is the ISO 8601 date format (yyyy-mm-dd HH:MM:SS)
; see strftime(3) Linux manual for format specifiers
;dateformat=%F %T
;
; This appends the hostname to the name of the log files.
;appendhostname = yes
;
; This determines whether or not we log queue events to a file
; (defaults to yes).
;queue_log = no
;
; This determines whether or not we log generic events to a file
; (defaults to yes).
;event_log = no
;
;
; For each file, specify what to log.
;
; For console logging, you set options at start of
; Asterisk with -v for verbose and -d for debug
; See 'asterisk -h' for more information.
;
; Directory for log files is configures in asterisk.conf
; option astlogdir
;
[logfiles]
;
; Format is "filename" and then "levels" of debugging to be included:
;    debug
;    notice
;    warning
;    error
;    verbose
;    dtmf
;
```

```
; Special filename "console" represents the system console
;
; We highly recommend that you DO NOT turn on debug mode if you are simply
; running a production system.  Debug mode turns on a LOT of extra messages,
; most of which you are unlikely to understand without an understanding of
; the underlying code.  Do NOT report debug messages as code issues, unless
; you have a specific issue that you are attempting to debug.  They are
; messages for just that -- debugging -- and do not rise to the level of
; something that merit your attention as an Asterisk administrator.  Debug
; messages are also very verbose and can and do fill up logfiles quickly;
; this is another reason not to have debug mode on a production system unless
; you are in the process of debugging a specific issue.
;
;debug => debug
console => notice,warning,error
;console => notice,warning,error,debug
messages => notice,warning,error
;full => notice,warning,error,debug,verbose

;syslog keyword : This special keyword logs to syslog facility
;
;syslog.local0 => notice,warning,error
;
```

There are some console commands associated to the logger process.

```
CLI> logger list channels
Channel                          Type     Status    Configuration
-------                          ----     ------    -------------
/var/log/asterisk/messages       File     Enabled    - Warning Notice Error
                                 Console  Enabled    - Warning Notice Error
```

```
CLI> logger rotate
  == Parsing '/etc/asterisk/logger.conf': Found
Asterisk Event Logger restarted
Asterisk Queue Logger restarted
```

You can control the log rotation using the logrotate daemon. Edit the file /etc/logrotate.d and include the content below to start rotating the log files.

/etc/logrotate.d/asterisk.

```
/var/log/asterisk/messages /var/log/asterisk/*log {
   missingok
   rotate 5
   weekly
   create 0640 asterisk asterisk
   postrotate
       /usr/sbin/asterisk -rx 'logger reload'
   endscript
}
```

More information about logrotate can be obtained using:

```
#man logrotate
```

2.11 STARTING ASTERISK WITH A NON-ROOT USER

It is safer to execute Asterisk with a non-root user. In case of a security failure or a buffer overflow attack, running Asterisk within an environment with less privileges to the user limits the possible actions from an intruder.

To change Asterisk running user:

1) Edit the file: vi /etc/init.d/asterisk

2) Uncomment the following lines:

```
AST_USER="asterisk"
AST_GROUP="asterisk"
```

3) To change user rights in Asterisk folders type:

```
cd /
chown --recursive asterisk:asterisk /etc/asterisk
chmod --recursive u=rwX,g=rX,o= /etc/asterisk
chown --recursive asterisk:asterisk /var/lib/asterisk
chown --recursive asterisk:asterisk /var/log/asterisk
chown --recursive asterisk:asterisk /var/run/asterisk
chown --recursive asterisk:asterisk /var/spool/asterisk
chown --recursive asterisk:asterisk /dev/zap
chmod --recursive u=rwX,g=rX,o= /var/lib/asterisk
chmod --recursive u=rwX,g=rX,o= /var/log/asterisk
chmod --recursive u=rwX,g=rX,o= /var/run/asterisk
chmod --recursive u=rwX,g=rX,o= /var/spool/asterisk
chmod --recursive u=rwX,g=rX,o= /dev/zap
```

4) Test changes with /etc/init.d/asterisk

2.12 ASTERISK INSTALLATION NOTES

2.12.1 Production Systems

If Asterisk is installed in a production environment, you should pay attention to the system design. A server has to be optimized in such a way that telephony systems have priority over other system processes. Asterisk should not run together with processor intensive software like X-Windows for example. If you need to run CPU intensive processes like, for example, a huge database, use a separate server. Generally speaking, Asterisk is sensible to hardware performance variations. Thus, try using Asterisk in a hardware environment that does not require more than 40% of CPU utilization.

2.12.2 Network tips

If you plan to use IP phones, it is important that you pay attention to your network. Voice protocols are very good and resistant to latency and even jitter. However, if you use a poorly configured local area network, the voice quality will suffer. It is only possible to guarantee a good voice quality using QoS (Quality of Service) in switches and routers. If you use voice over the Internet (Without QoS), voice quality can be defined as best effort. Voice in a local area network tends to be good, but even in a LAN environment, if you have 10 Mbps hubs with too many collisions, you will end up having distorted or crappy voice. Follow these recommendations to ensure the best possible voice quality:

- Use QoS end-to-end if possible or economically feasible. With QoS end-to-end, the voice quality is perfect. No excuses!
- Avoid using 10/100 Mbps hubs for voice in a production environment. Collisions can impose jitter to the network. Full duplex 10/100 Mbps are preferred because there are no collisions.
- Use VLANs to separate unnecessary broadcasts of the voice network. You don't want a virus destroying you voice network with ARP broadcasts.
- Educate users about expectations in a voice network. Without QoS, don't state that the voice will be perfect since in most cases it won't. Cell phone voice quality is what is expect for most cases in this scenario.

2.13 SUMMARY

In this chapter you learned the minimum hardware requirements, how to download, how to install, and how to compile Asterisk. Asterisk should be executed with a non-root user for security reasons. You should check your network environment before starting a production environment.

2.14 QUESTIONS

1. What's the minimal Asterisk hardware configuration?

---

2. Telephony interface cards for Asterisk usually have some DSPs (Digital Signal Processors) built in and do not need a lot of CPU resources from the PC.

a) True
b) False

3. If you want perfect voice quality, you need to implement end-to-end QoS (Quality of Service).

a) True
b) False

4. It's possible to have good voice quality with 100 Mbps switches in a non congested local area network.

a) True
b) False

5. List the necessary packages for Asterisk and the Zaptel compilation.

_____

_____

6. If you don't have a TDM interface card, you will end up needing a clock source for synchronization. The ztdummy driver is responsible for this role by using USB as a clock source (Kernel 2.4). This is necessary because some applications like _____ and _____ require a time reference.

7. When you install Asterisk, it's better to leave GUIs uninstalled because Asterisk is sensible to performance variations. GUIs stole a lot of CPU cycles.

a) True
b) False

8. Asterisk configuration files are located in the _____ directory.

9. To install Asterisk sample files you need to type the following command:

_____

_____

10. Why is it important to start Asterisk with a non-root user?

_____

_____

Chapter 3

First Steps

In this chapter you will learn how to perform a basic Asterisk configuration. The main objective here is for you to see the PBX running, to be able to dial between extensions, to dial to a message being played, and to dial to a single analog or SIP trunk. The idea behind this chapter is to make sure that your Asterisk is up and running as soon as possible. After working in this chapter, you will have enough background to prepare for the subsequent chapters, where we will get into configuration details.

3.1 OBJECTIVES

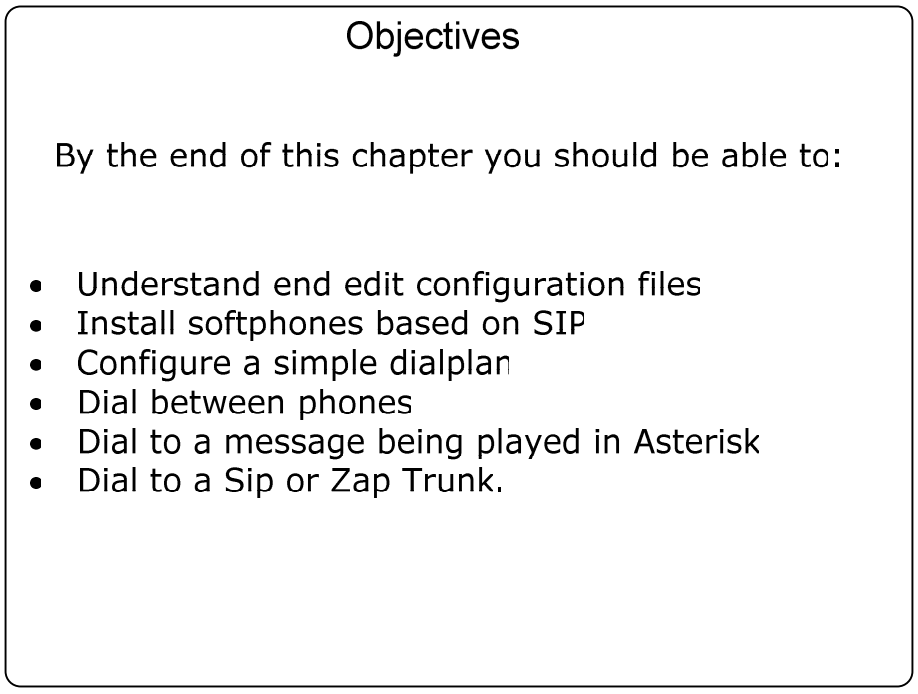


Figure 3.1 Objectives

3.2 UNDERSTANDING THE CONFIGURATION FILES

Asterisk is controlled by text configuration files located in /etc/asterisk. The file format is similar to the Windows ".ini" files. A semicolon is used as a remark character, the signs "=" and "=>" are equivalent and the spaces are ignored.

```
;
; The first line without a comment should be the session title.
```

```
;
[Session]
Key = value; Variable designation
[Session 2]
Key => value; Object declaration
```

Asterisk interprets "=" and "=>" in the same way. Differences in syntax are used to distinguish between objects and variables. Prefer "=" when you want to declare a variable and "=>" to designate an object. The syntax is the same between all files but there are three types of grammar, as discussed below.

3.3 GRAMMARS

Grammar	Object is created:	Conf. File	Example
Simple Group	All in the same line	extensions.conf	**exten**=>4000,1,Dial(SIP/4000)
Option Inheritance	Options are defined first, object inherit the options	zapata.conf	[channels] context=default signalling=fxs_ks group=1 **channel => 1**
Complex Entity	Each entity receives a context	sip.conf, iax.conf	**[cisco]** type=friend secret=mysecret host=10.1.30.50 context=trusted **[xlite]** type=friend secret=xlite host=dynamic

3.3.1 Simple Group

The simple group format used in extensions.conf, meetme.conf and voicemail.conf is the most basic grammar. Each object is declared with options in the same line.

Example:

```
[Session]
Object 1 => op1,op2,op3
Object 2=> op1b,op2b,op3b
```

In this example, object 1 is created with options op1, op2, op3 while object 2 is created with options op1, op2 and op3.

3.3.2 Object options inheritance grammar

This format is used by zapata.conf and agents.conf where a lot of options are available and most interfaces and objects share the same options. Typically there are one or more sections with objects and channels declarations. Options to the object are **declared above** the object and can be changed to another object. Although this concept is hard to understand, it is very easy to use.

Example:

```
[Session]
op1 = bas
op2 = adv
object=>1
op1 = int
object => 2
```

The first two lines configure the value of option op1 and op2 to "bas" and "adv" respectively. When the object 1 is instanced, it is created using option 1 as "bas" and option 2 as "adv". After defining object 1 we have changed the option 1 to "int". Then we create object 2 with option 1 as "int" and option 2 as "adv".

3.3.3 Complex entity object

This format is used by iax.conf, sip.conf as well as other configuration files where there are numerous entities with many options. Typically, this format does not share a large volume of common configurations. Each entity receives a context. Sometimes there are reserved contexts like [general] for global configurations. Options are declared in the context declarations.

Example:

```
[entity1]
op1=value1
op2=value2
[entity2]
op1=value3
op2=value4
```

The entity [entity1] has values "value1" and "value2" for options op1 and op2, respectively. The entity [entity2] has values "value3" and "value4" for options op1 and op2.

3.4 CONFIGURING A PSTN INTERFACE

To connect to the PSTN you will need an interface FXO (Foreign Exchange Office) and a telephone line. You can use an existing PBX extension too. You can obtain a telephony interface card with a FXO interface from several manufacturers. In this example we will show you how to install a zaptel board.

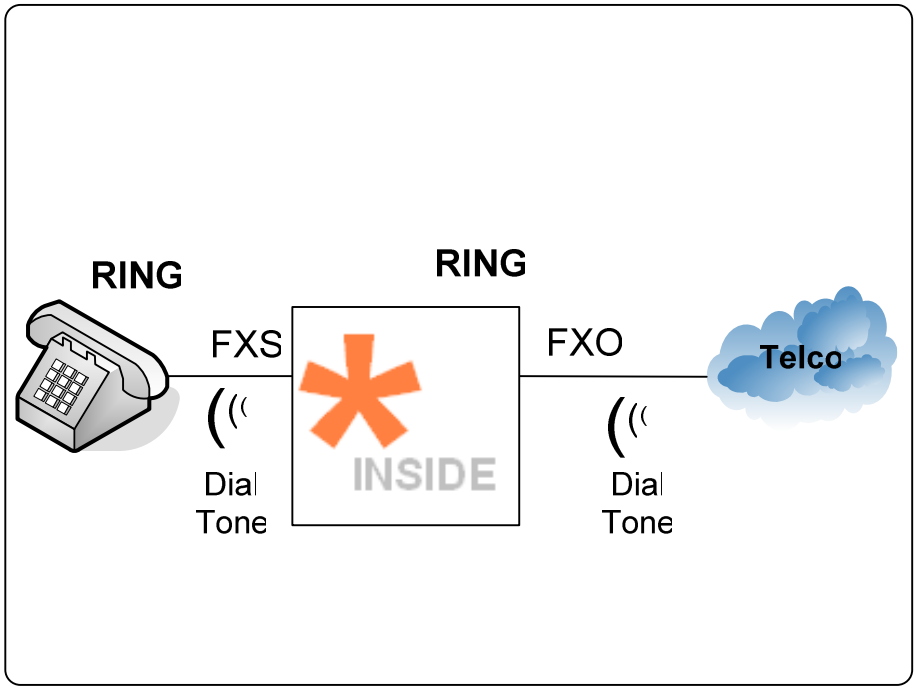


Figure 3.1 FXO end FXS Interfaces

Tip: There are other FXO interfaces available. A X100P clone can be found in the market for a very small price. These boards are old 56K fax/modems based on Motorola and Intel chipsets. The chipsets known to work are:
- Motorola 68202-51
- Intel 537PU
- Intel 537 PG
- Intel Ambient MD3200

There are no guaranties that this board will work. Use at your own risk. Some present eco problems and low sound volume. If you don't want to take risks, Digium boards are a very good choice.

3.4.1 Installing a X100P

Before installing a X100P in your computer, disable all unnecessary or unused hardware from your motherboard. This will help to avoid problems with shared interrupts. In order to properly install the X100P, you will have to plug the card into a PCI slot and edit two configuration files:

- "zaptel.conf" in the /etc directory - the board configuration.
- **"zapata.conf"** in the /etc/asterisk directory - this configures the Zapata channel driver.

Don't be concerned at this point to understand all the details of the configuration files. We will have an entire chapter about Zapata channels. At this moment we will show an excerpt of the file. Save a copy of the file /etc/zaptel.conf to /etc/zaptel.conf.old before you begin editing.

zaptel.conf

```
fxsks=1
loadzone = br
defaultzone=br
channels=1
```

zapata.conf

```
[channels]
context=default
signalling=fxs_ks
group=1
channel => 1
```

After finishing editing the files, load the zaptel drivers as shown below.

```
modprobe zaptel
modprobe wcfxo
ztcfg -vvvvvv
asterisk -vvvvvgc
```

3.5 SIP IP PHONES CONFIGURATION

Let's now configure the SIP phones. The idea is to configure a simple PBX. In subsequent chapters you will have an entire SIP session with all the details. SIP is configured in the /etc/asterisk/sip.conf directory and has all the parameters related to SIP phones and VoIP providers. SIP clients have to be configured before you can make and receive calls.

3.5.1 General section

The SIP file is read from top down. The first section contains the global parameters [general]. The main options are:

- allow/disallow: Defines which codecs can be used.
- bindaddr: Address to be bond to Asterisk SIP listener. If you set it up as 0.0.0.0 (default) it will bind to all interfaces.
- context: Sets the default context for all clients unless changed in the client section.
- bindport: SIP UDP port to listen.
- maxexpirey: Maximum time to register (seconds)
- defaultexpirey: Default time to register (seconds)
- register: Registers Asterisk to another Host.

Example:

```
[general]
bindport = 5060
bindaddr = 10.1.30.45
context = default
disallow = all
allow = ulaw
maxexpirey = 120
defaultexpirey = 80
```

3.5.2 Clients section

After finishing the general sections, it is time to set up the SIP clients. I would like to remind the reader, again, that we will have an entire SIP chapter later in the book. For now, let's concentrate on the basics and leave the details for later.

- [name]: When a SIP device connects to Asterisk, it uses the username part of the SIP URI to find the peer/user.
- type: Configures the connection class. Options are peer, user, and friend.
- peer: Asterisk sends calls to a peer.
- user: Asterisk receives calls from a user.
- friend: Both at same time.
- host: IP address or host name. The most common option is "dynamic", used when the host registers to Asterisk.
- secret: Password to authenticate peers and users.

Example:

```
[cisco]
type=friend
secret=mysecret
host=10.1.30.50
context=trusted

[xlite]
type=friend
secret=xlite
host=dynamic
defaultip=10.1.30.17
```

3.6 DIAL PLAN INTRODUCTION

Dial plan is Asterisk's heart. It defines how Asterisk handles each and every call to the PBX. It consists of extensions that make an instruction list for Asterisk to follow. Instructions are fired by digits received from the channel or application. In order to configure Asterisk successfully, it is crucial to understand the dial plan.

Most of the dial plan is contained in the extensions.conf file at the /etc/asterisk directory. This file uses the simple group grammar and has four major concepts:

-
- Extensions
- Priorities
- Applications
- Contexts

Let's now create a basic dial plan. In subsequent sections of this book I will devote two full chapters exclusively to dial plan. If you installed the sample files (make samples), the extensions.conf already exists. Save it with another name and start with a blank file.

3.6.1 Extensions

The dial plan is a set of defined extensions. An extension is a string that will trigger an event when a call is made. Extensions can be either literal or pattern.

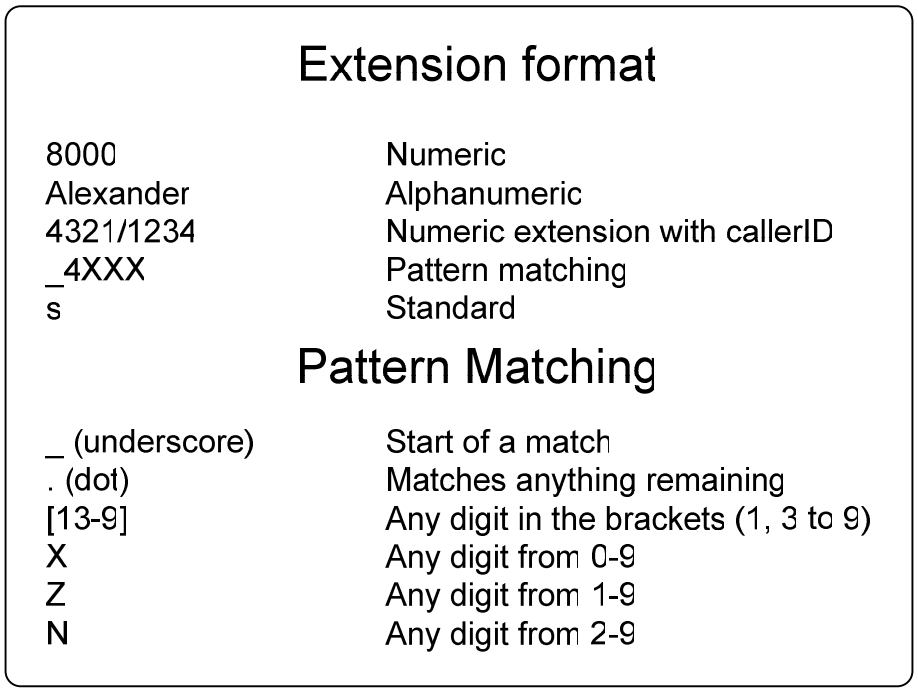


Figure 3.2 Extensions format

Example:

```
exten=>8580,1,Dial(SIP/8580,20)
exten=>8580,2,hangup()
```

The instruction "exten" describes the next step for the call. 8580 is the set of digits received (called number). The numbers "1" and "2" are priorities that define the execution order. Dialing "8580" will ring the SIP IP phone registered as "8580". If the call is unanswered in 20 seconds it will hang-up.

Extension Syntax:

```
exten=> number (name), {priority|label{+|-}offset}[(alias)],application
```

The extension command "exten=" is followed by an extension number or name, a comma, a priority, another comma and, finally, the application. Extension is the matching address of the call (the phone number). Priorities are used to execute the steps in order of priority. Application is the action to be taken (dial, playback, hangup). Each action has a different application.

3.6.2 Priorities

Priorities are numbered steps for execution in each dialed extension. Each priority calls a specific application. Usually, the numbers start with "1" and increment by 1 to each line in the extension definition. In version 1.2, it is

common to use the "n" priority as next, instead of manually assigning the numbers. If numbers are not sequential, execution is aborted. After version 1.2 it became possible to use labels and aliases. We will see more about this under the topic 'Advanced examples' in Chapter 8.

3.6.3 Applications

Applications play an important role in Asterisk. They handle voice channels, playing tones, the acceptance of digits called to the PBX, as well as call hang-up. Applications can be called with options that affect their behavior. You can use '**core show applications'** in the Asterisk's command line interface to show available applications. When you build your first dial plan you will start to understand what applications are appropriated.

```
CLI>core show applications
```

3.6.4 Contexts

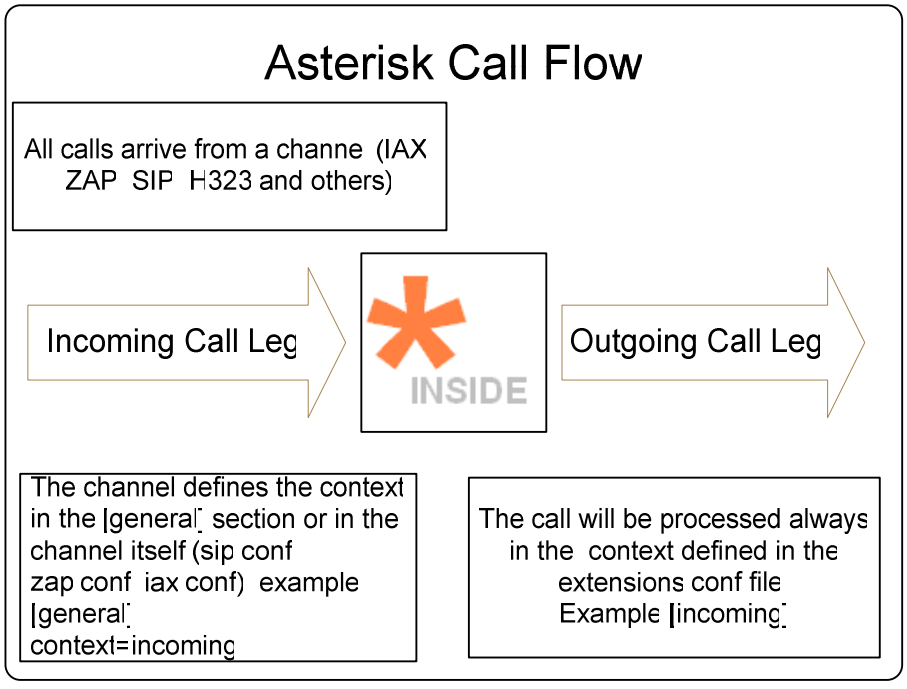


Figure 3.3 Asterisk call flow

Contexts play an important role in Asterisk's dial plan configuration and security. Contexts define a scope, which allows separation of the dial plan into different parts. It is important to understand that contexts are intimately bond to channels. When a Asterisk receives a phone call, the call is processed within its incoming context section. The incoming context is always defined by the channel configuration file (iax.conf, sip.conf, zap.conf…).

Let's suppose, for example, that you have two classes of users: "managers" and "guests". Suppose now that want to play different messages for "guests" and "managers" when they dial "9000". You can accomplish this by defining the incoming context in the channel files (sip.conf, iax.conf, zap.conf).

Example:

In the example below, when Mary dials 9000, she receives the message "youareaguest". When John dials the same number, he receives a different message: "youareamanager".

sip.conf

```
[john]
context=managers
host=dynamic
…
[mary]
context=guests
host=dynamic
…
```

extensions.conf

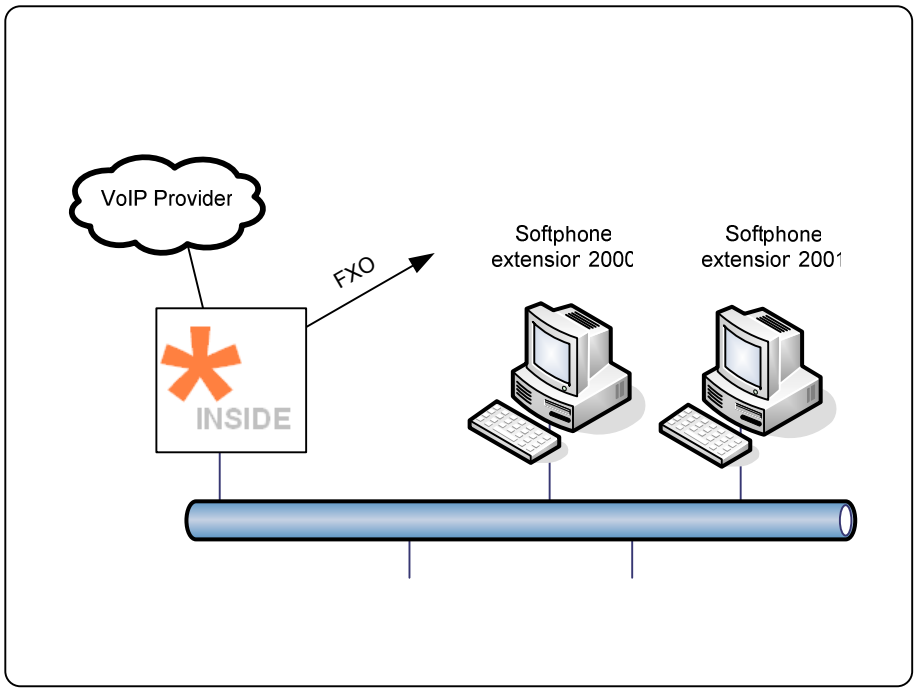
```
[managers]
exten=>9000,1,Playback(youareamanager)
[guests]
exten=>9000,1,Playback(youareaguest)
```

By understanding this concept you can create several features in Asterisk. Different contexts can be used to serve different companies and different classes of users within the same Asterisk configuration. Contexts may determine, for example, who can call long distance and who cannot.

Contexts receive a name inside brackets ([]). All instructions after the definition are part of the context. To start a new context, simply insert the new context. A context ends when another starts.

There are two special contexts in the extensions.conf file. The context [globals] is used to define variables whereas the context [general] is used to define some general options. We will look into these special contexts in chapter 8.

3.6.5 Creating a testing environment



Figur3.4 VoIP Lab

For this lab we recommend that you use a FXO interface card. To create a lab like the one above, you may use a Pentium class PC with 256MB RAM for Asterisk and two other computers for the soft phone. You may use virtual machines (VMWare, VirtualPC) if you wish to use a single computer. Although voice quality will be poor, this is acceptable for testing purposes. We also recommend using the X-Lite softphone from counterpath (www.counterpath.com), although other free soft phones (e.g. SJPhone - www.sjlabs.com) may be used as well. It's possible that the X-Lite softphone does not exist anymore by the time you read this material; you will certainly be able to find others by searching the Internet. You may also use a SIP Analog Telephony Adapter or even an SIP IP Phone. Really, it all depends on your budget.

Step 1: Edit the sip.conf file and add the extension 2000 configuration.

```
[general]
bindport=5060
bindaddr=10.1.30.45 ; put here the IP address of your server
context=default
allow=all

[2000]
type=friend
secret=2000
host=dynamic
canreinvite=no
```

Step 2: Repeat step 1 for extension 2001.

Step 3: Configure the softphone for extension 2000.

a) Run the installation program.
b) Press "next" in the first screen.
c) Accept the licensing agreement.
d) Accept the next screens. In other words NEXT->NEXT->FINISH.
e) Get into the X-LITE menu by pressing the icon illustrated below.

f) In the next screen, select "system settings".

g) Choose "sip proxy".

h) Choose Default.

i) Complete the following fields.

```
Display Name: 2000
Username: 2000
Authorization User: 2000
Password: 2000
Domain/Realm: Asterisk IP Address
SIP Proxy: Asterisk IP Address
```

j) Close X-LITE and open again.

k) Confirm that the phone was registered to Asterisk with the **sip show peers** command.

Step 4: Repeat the configuration for extension 2001 in the other softphone.

3.7 CREATING A BASIC DIAL PLAN

Now we are ready to start creating our first dial plan. We will use examples to explain the dial plan. Let's move on step-by-step.

3.7.1 Basic example

In this example, Asterisk will receive a call, play a sound and hang-up.

Edit the extensions.conf file to include these entries:

```
[incoming]
exten=>s,1,answer()
exten=>s,2,playback(goodbye)
exten=>s,3,hangup()
```

Priority 1 calls the answer() application. Asterisk handles the line and sets up the call. After answering the line, it goes to the next priority.

Priority 2 calls the playback application() to play a sound from goodbye.gsm file.

Finally, **priority 3** hangs up the call.

Example explanation:

A call received by a FXO interface is sent to the incoming context, defined in the channel configuration file (zapata.conf). When an incoming call arrives it is processed in the "s" extension of the incoming context. We have three priorities, each priority calling an application.

The **special extension "s"** is used to start processing the incoming call when the number that dialed to Asterisk is not known (e.g. incoming call by an analog line).

If we are to answer a call, it is better to know the application that will do it. The **answer() application** is used to answer a channel in the ringing state. Some applications require the answer() application before processing the call.

The **playback() application** is used to play a message from a sound file previously recorded. When the application playback() is being executed, any digits pressed are ignored. The syntax is playback(filename). It plays the file with .gsm or .wav extension from the default sounds directory (/var/lib/asterisk/sounds).

The **hangup() application** does exactly what the name says. It disconnects the active channel. You should use it at the end of the call.

3.7.2 A more thorough example

Let's improve upon our first example by introducing two new applications: background() and goto(). The key for interactive systems based on Asterisk

is the background() application. It allows you to play a sound file while waiting for digits to be pressed. When this occurs, the sound that is playing stops and the execution goes to the digits pressed. We will look at the background application in depth in subsequent chapters.

Syntax for Background() application:

```
exten=>extension, priority, background(filename)
```

Other useful application is goto(). As the name implies, it jumps from the current context, extension and priority to a specific context, extension and priority.

Syntax for Goto() application:

```
exten=>extension, priority,goto(context,extension, priority)
```

Valid syntaxes for goto() are:

```
Goto(context,extension,priority)
Goto(extension,priority)
Goto(priority)
```

This is an example of a small company with three departments: tech support, sales, and training. Let's create an interactive system that allows the user to select the department called. Firstly, we will play a greeting like, "press 1 for tech support, 2 for training, and 3 for sales". In this simple example we won't treat invalid digits. After you select a department, the system will play a message like "you were directed to … department" and transfer the execution to the related context.

```
[incoming]
exten=>s,1,Answer()
exten=>s,2,Background(greeting)
exten=>s,3,hangup()
exten=>1,1,playback(support)
exten=>1,2,goto(support,s,1)
exten=>2,1,playback(training)
exten=>2,2,goto(training,s,1)
exten=>3,1,playback(sales)
exten=>3,2,goto(sales,s,1)
```

Step by step explanation:

When someone calls the telephone line connected to Asterisk via the FXO interface card (configured to [incoming] context in zapata.conf), the extension "s" in the [incoming] context is triggered. The extension "s"

answers the call using the background application and plays a greeting message, waiting for digits to be pressed. If the user presses "1", the system goes to extension "1" and plays a message "You were directed to tech support". Next, extension "2" sends the execution to [support] context for further processing.

3.7.3 Bridging channels using Dial() application

We can improve our example by adding the dial() command. Instead of sending the execution to another context, we will transfer the call to an extension, directly using the dial() application

```
[incoming]
exten=>s,1,Answer()
exten=>s,2,Background(greeting)
exten=>s,3,hangup()
exten=>1,1,playback(support)
exten=>1,2,Dial(SIP/8000)
exten=>2,1,playback(training)
exten=>2,2,Dial(ZAP/1)
exten=>3,1,playback(sales)
exten=>3,2,Dial(IAX2/8002)
```

By comparing with the first example, we've just created a shortcut. Instead of sending the call to another context, we are now sending the call to the final channel destination. For "support", we will direct the call to a SIP phone identified by the "8000" number. For "training" we will send the call directly to TDM (analog or digital) to the channel identified as "1". Finally, for sales we will send the call to an IAX2 phone identified by the number "8002".

At this point you should have a clear understanding of the use of several applications like answer(), background(), goto(), hangup() and playback(). If this is not clear, please read again until you feel comfortable with the content. You will need this background from now on.

Once you understand the basics of extensions, priorities and applications, it will be easy to create a simple dial plan. These concepts will be explored in greater depth later in the book, and you will see that the dial plan will become more powerful.

3.8 Labs

These labs will create a small PBX capable to dial between extensions, to the PSTN, or to a VoIP provider. We will configure call reception using auto attendant.

Lab instructions:

- Extensions will range from 2000 to 2100
- To dial the PSTN you will need to dial 9 first
- To dial external call via VoIP provider you will need to dial 1 first
- 8000 should be used to record the message for the auto-attendant.

3.8.1 Calling between phones

Step 1: Edit the extensions.conf file inside the [default] section with the following commands.

```
[default]
exten=>2000,1,Dial(SIP/2000)
exten=>2001,1,Dial(SIP/2001)
```

Step 2: Reload the asterisk extensions

```
CLI>dialplan reload
```

Step 3: Test dialing between 2000 and 2001 softphones.

3.8.2 Calling PSTN using the zaptel interface card (FXO)

Step 1: Edit extensions.conf

```
[default]
exten=>0,1,Dial(ZAP/1,20,r)
```

Step 2: Reload Asterisk extensions

```
CLI>dialplan reload
```

Step 3: Test dialing 0. You will receive an external dial tone, then dial the destination number.

3.8.3 Auto-attendant

Step 1: Edit the extensions.conf to include an interface to record the message for the auto-attendant.

```
[default]
exten=>8000,1,Wait(2)
exten=>8000,2,Record(menu:gsm)
```

```
exten=>8000,3,Wait(2)
exten=>8000,4,Playback(menu)
exten=>8000,5,Hangup()
```

Step 2: Reload the extensions and dial 8000. Record a message like "Welcome to the XYZ company, dial the extension number". Press "#" at the end to finish recording. You will hear the recorded message when playback is executed.

Step 3: Edit the extensions.conf to include the receiving menu.

```
[default]
exten=>s,1,Background(menu)
exten=>s,2,Dial(SIP/2000)
exten=>2000,1,Dial(SIP/2000)
exten=>2001,1,Dial(SIP/2001)
```

Step 4: Dial from your cell phone (or another external phone) to the line connected to Asterisk. You will hear an auto-attendant message. Dial 2000 to be transferred to the 2000 extension.

3.9 SUMMARY

In this chapter you learned that configuration files are stored in the /etc/asterisk directory. To use Asterisk, it is necessary, in the first place, to configure the channels (e.g. sip, zapata, iax). There are three different grammars for configuration files, simple group, object inheritance, and complex entity. The dial plan is created in the extensions.conf file and is a set of contexts and extensions. In the dial plan each extension triggers an application and you learned to use playback, background, dial, goto, hangup and answer applications.

3.10 QUESTIONS

1. These are channel configuration files

   a) zaptel.conf
   b) zapata.conf
   c) sip.conf
   d) iax.conf

2. It is important to define a context in the channel configuration file, because this will define the incoming context for a call. In the extensions configuration file (extensions.conf) a call from this channel will be processed in the matching incoming context.

a) True
b) False

3. The main differences between the playback() and background() applications are (choose two):

a) Playback simply plays a prompt, but does not wait for digits.
b) Background simply plays a prompt, but does not wait for digits.
c) Background plays a message and waits for digits to be pressed
d) Playback plays message and waits for digits to be pressed

4. When a call gets into Asterisk using a telephony interface card (FXO) this call is handled in the special extension:

a) '0'
b) '9'
c) 's'
d) 'i'

5. Valid formats for the goto() application are (choose three):

a) Goto(context,extension, priority)
b) Goto(priority, context, extension)
c) Goto(extension,priority)
d) Goto(priority)

6. An extension cannot be defined as (choose all correct answers):

a) An alphanumeric literal
b) A numeric literal
c) A pattern beginning with a "_" (dot) character
d) A pattern starting with a "_" (underscore) character

7. A pattern _7[1-5]XX matches (choose all correct answers):

a) 7100
b) 7600
c) 7630
d) 7230

8. An incoming context for a zaptel compatible telephony interface is defined in the _____ configuration file:

a) zaptel.conf
b) zapata.conf
c) asterisk.conf
d) modules.conf

9. In the Options Inheritance grammar used by zapata.conf you:

a) Define the object in a single line
b) Define options first and declare the objects below the defined options
c) Define a context for each object

10. Priorities must be consecutive!

a) False
b) True

Page intentionally left empty

Chapter 4

Analog and digital channels

In this section, we will show you how to configure zapata and unicall channels for connecting to the PSTN. Starting with a revision of telephony concepts, you will learn the general principles to understand and configure analog, E1, and T1 lines.

4.1 OBJECTIVES

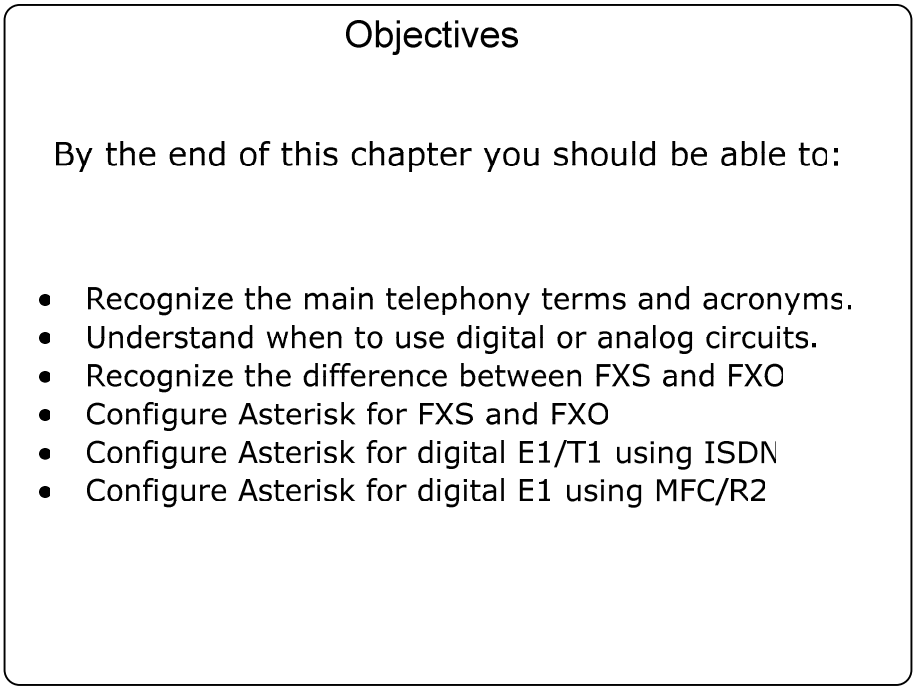


Figure 4.1 Objectives

4.2 TELEPHONY BASICS

Most analog implementations use a pair of cooper lines named tip and ring. When a loop is closed, the phone receives the dial tone from the telecom switch (or the private PBX). The most frequently used signaling is loop-start. There are other, less common kinds of signaling such as ground start, which is used in several countries.

There are three types of signaling:

- Supervision signaling
- Address signaling

- Information signaling

4.2.1 Supervision signaling

The main supervision signaling are on-hook, off-hook, and ringing.

On-Hook – When a user puts the phone on-hook, the PBX interrupts and does not allow the electric current to pass. In this state, the circuit is named on-hook. In this position, only the ringer is active.

Off-Hook – Before starting a phone call, the phone needs to pass to the off-hook state. Removing the handset from the hook closes the loop and indicates to the PBX that the user intends to make a call. Upon receiving this indication, the PBX generates a dial tone, indicating to the user that it is ready to accept the destination address (phone number).

Ringing – When a user calls another phone it generates a voltage to the ringer that warns the other user about a call reception.

Signaling varies by country, with different tones for different countries. You can personalize Asterisk tones to your country by modifying the indications.conf file.

```
[br]
description=Brazil
ringcadance=1000,4000
dial=425
busy=425/250,0/250
ring=425/1000,0/4000
congestion=425/250,0/250,425/750,0/250
callwaiting=425/50,0/1000
```

4.2.2 Address Signaling

You can use two kinds of signaling for dialing. The first and most common is dtmf (dual tone multi-frequency) and the other is pulse dialing (used in old rotary dial phones). Phones have a keypad for dialing. Associated with each button are two frequencies: one high and one low. In the case of dtmf signaling, the combination of these tones indicates what digit is being pressed. MFC/R2 uses a multi-frequency tone different from DTMF.

4.2.3 Information signaling

Information signaling shows the call progress and their different events.

- Dial tone

- Busy Tone
- Ringback
- Congestion
- Invalid number
- Confirmation tone

4.3 PSTN INTERFACES

As in the case of old PBXs, it is often required to connect the Asterisk PBX to the PSTN. Let us show you how to do it.

There are usually three options for telephone lines.

- **Analog:** The most common form for home and small business, usually delivered with a metallic pair of cooper lines.
- **Digital**: Used when many lines are necessary. A digital line is usually delivered by a CSU/DSU or a Fiber multiplexer. The end user connector is usually a RJ45. In some countries, E1 lines are delivered using two coaxial BNC connectors; in this case you will need a balloon to connect to the Rj45 jack in the telephony board.
- **SIP:** This option is recent and the telephone line is delivered using a data connection with SIP signaling (VoIP). This is a good option to use with Asterisk since you will not need to buy a telephony card. Phone calls will be delivered directly to the Ethernet port. Another advantage is that you may be able to free resources from your CPU by avoiding codec transcoding.

4.4 ANALOG FXS, FXO AND E&M INTERFACES

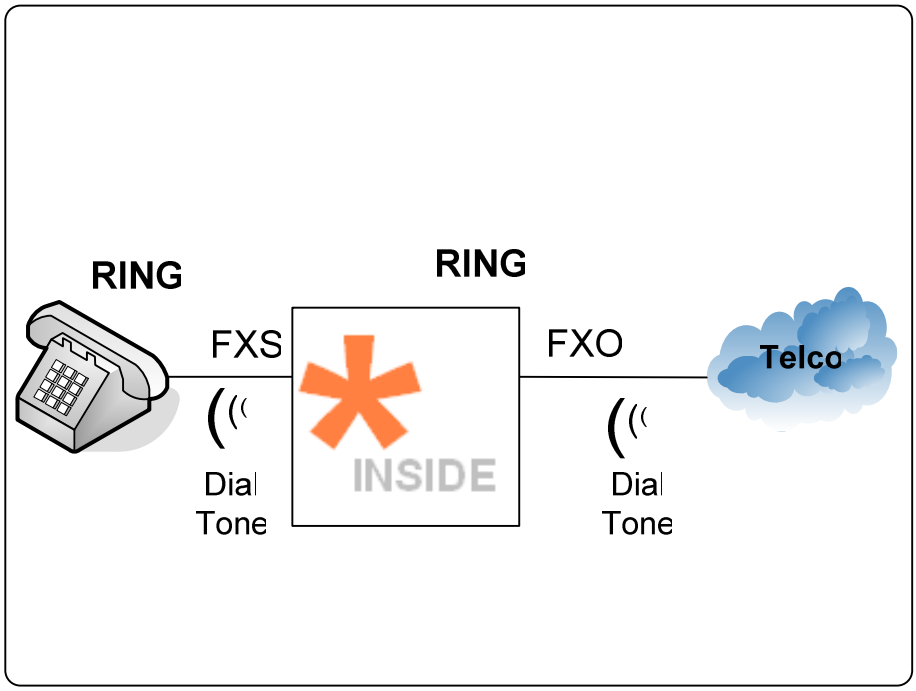


Figure 4.2 FXS and FXO interfaces

4.4.1 FX Interfaces (Foreign eXchange)

FX interfaces are analog. The term "Foreign eXchange" is applied to access trunks to a PSTN central office (CO).

FXO (Foreign eXchange Office)

The FXO interface is used to connect to a CO (Central Office) or another PBX's extension. It communicates directly with a telephone line coming from the PSTN. Another option is to connect the FXO interface to an existing PBX, allowing communication between Asterisk and the legacy PBX. When you connect Asterisk to a PBX port and deliver a remote extension using VoIP, this is often referred as an off-promises extension (OPX). A FXO interface receives a dial tone.

FXS (Foreign eXchange Station)

The FXS interface feeds an analog phone, modem or fax. The FXS provides the dial tone and power for a phone.

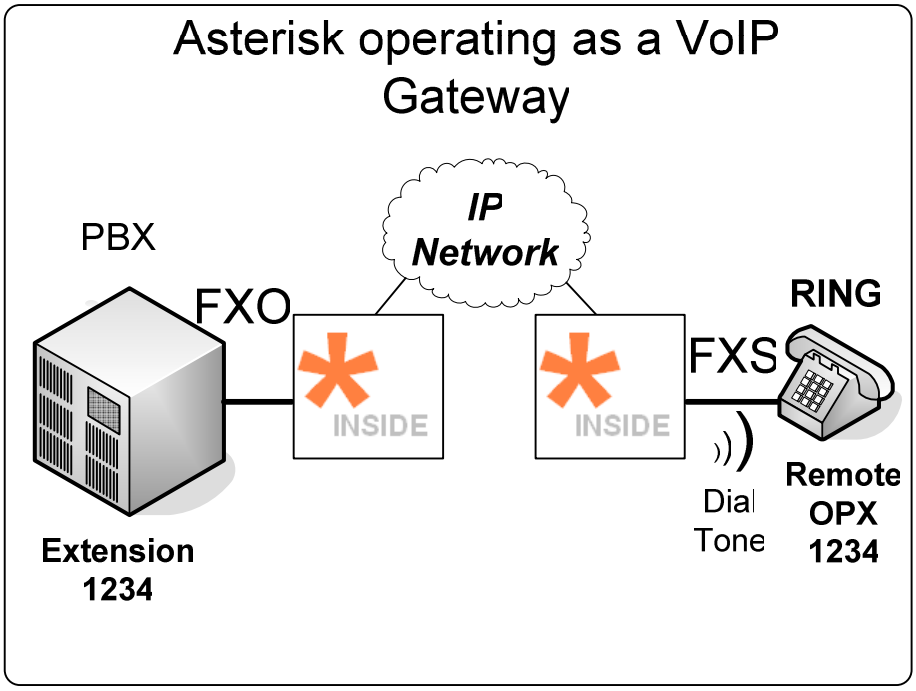


Figure 4.3 Asterisk operating as a VoIP gateway

4.4.2 Trunk signaling

- Loop-Start
- Ground-Start
- Kewlstart

The use of kewlstart signaling in Asterisk is almost default. Kewlstart is not a signaling itself, but it adds intelligence to the circuit by monitoring what is happening at the other side. Kewlstart is based in loop-start. Most switches do not support this feature, which is used to get the hang-up notification.

- **Loopstart:** Used in most analog lines. It allows the telephone to indicate "on-hook" and "off-hook" and the switch to indicate "ring" and "no-ring". Probably is what you have at home. The name comes from the fact that the line is always open. When you close the loop, the switch provides to you the dial tone. An incoming call is signaled by a 100V ringing voltage over the open pair.

- **Groundstart**: Similar to Loopstart. When you want to make a call, one side of the line is short-circuited. When the switch identifies this state, it reverses the voltage through the open pair, and then the loop is closed. This way the line first becomes occupied before being offered to the caller.

- **Kewlstart**: Adds intelligence to the circuits, allowing monitoring of the other side. Kewlstart incorporates many advantages from loop-start.

4.5 E1/T1 DIGITAL LINES

4.5.1 from analog to digital lines

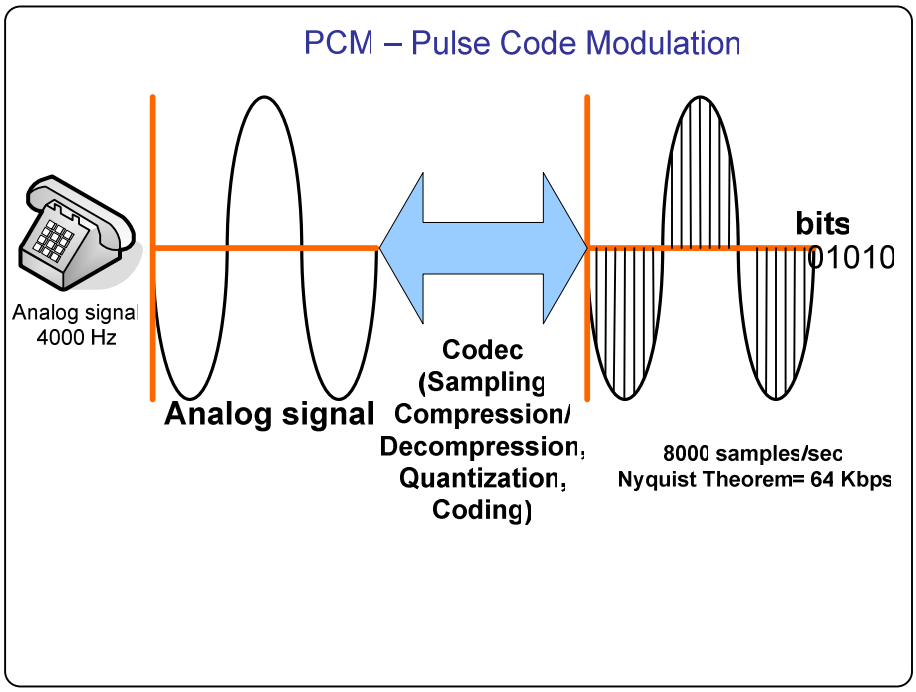


Figure 4.4 from analog to digital voice

The analog signal is sampled 8000 times per second to create a digital version of the analog voice. This encoding is known as PCM (pulse code modulation). In the USAUSA and Japan, the signal is encoded using μlaw (referred in Asterisk as ulaw). In the rest of the world, the encoding is alaw.

4.5.2 Time Division Multiplexing

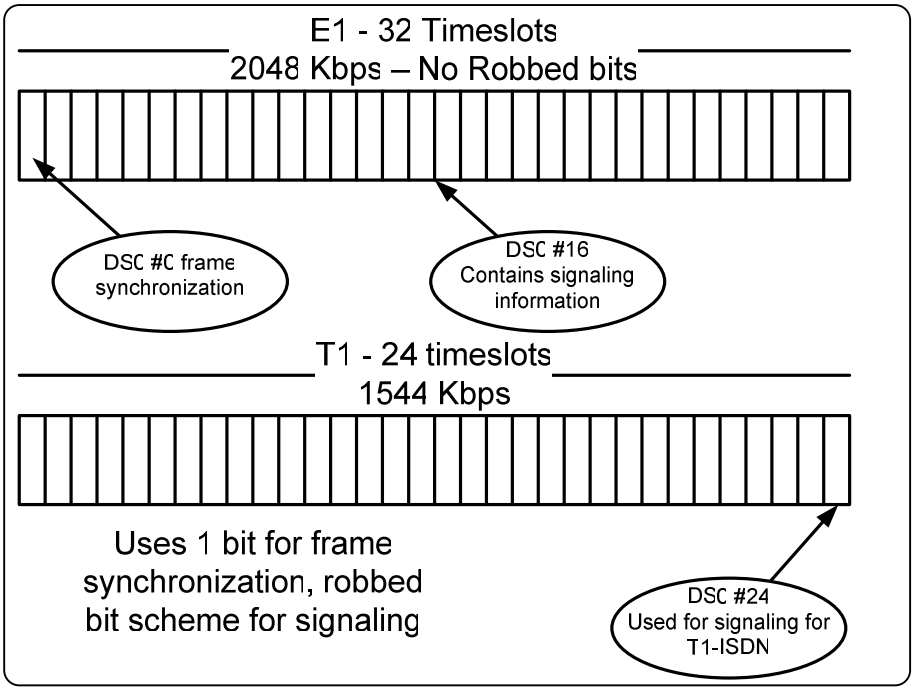


Figure 4.5 T1/E1 Digital Trunks

Analog lines make sense when you need just a few channels. When you want a large number of circuits, the phone company will usually provide you with a digital trunk.

Using TDM (time division multiplexing), it is possible to stuff multiple channels into a single data connection. A digital trunk is actually a data circuit and the voice is transported in digital format using PCM. Each timeslot uses 64 Kbps of bandwidth to transport a single voice channel.

In USAthe USA, the most common digital trunk is T1 with 24 available lines, whereas in Europe and Latin America E1 trunks have 30 lines. Some companies provide fractional T1/E1 with fewer channels.

Robbed bit signaling

Sometimes a T1 trunk uses a robbed bit scheme where one bit is borrowed for signaling. In T1 trunks, the data or voice channel is transmitted with 56 Kbps on each timeslot. As you can see, using the robbed bit, T1 does not loose two slots for synchronizing and signaling.

4.5.3 T1/E1 Line code

T1 and E1 are actually data circuits and have a data coding. This coding determines the way the bits are interpreted.

For T1s the most common line code is B8ZS for layer 1 (physical layer), and ESF (extended super frame) in layer 2 (datalink layer).

For E1s, the most common line code is HDB3 for layer 1 and CCS for layer 2.

The easiest way to know how your digital trunk is configured is to ask this information to the Telco. You will need this information to configure the zaptel.conf file.

4.5.4 T1/E1 Signaling

It is important to understand that T1/E1 lines can be delivered using different kinds of signaling like:

- T1 with robbed bit signaling
- T1 with ISDN signaling
- E1 with MFC/R2 (CAS - Channel Associated Signaling)
- E1 with ISDN signaling

ISDN (integrated services digital network) is becoming very common. . It is a digital voice network standardized by the ITU (International Telecommunications Union) in 1984. With ISDN, you have two kinds of channels:

- Bearer channels
  - Voice
  - Data
- Data channels
  - Out of band signaling
  - LAPD signaling
  - Q.931

Usually, an ISDN line is provided using two physical means:

- BRI (basic rate interface)
  - Known as 2B+D
  - Two bearer (64K) channels and a data (16K) channel
  - BRI uses a pair of cooper wires with 148Kbps.
- PRI (primary rate interface)
  - Delivered using a T1/E1 trunk
  - 23B+D for T1s

- 30B+D for E1s

Sometimes, E1 lines use a CAS signaling scheme named **MFC/R2**. It was defined by the ITU as Q.421/Q441. It is very popular in Latin America and Asia. Several telephony companies in these countries customized MFC/R2 to their needs. Hence, you will need to know the correct country variation in order to make it work.

4.6. ASTERISK TELEPHONY CHANNELS SETUP

To configure a telephony interface card, several steps are necessary. In this chapter, we will show three of the most common scenarios:

- Analog connection using FXS and FXO
- Digital connection using ISDN
- Digital connection using MFC/R2

4.6.1 Example #1 - One FXO, one FXS installation.

In this example, we will use a Digium TDM400 telephony interface card with one FXS and one FXO module. The eequired steps ate listed below:

1. Install the TDM400 board
2. Configure the Zaptel.conf file
3. Load the interface driver
4. Execute zttest to verify interrupts
5. Execute ztcfg to configure the driver
6. Configure the channel ZAP in zapata.conf file
7. Load Asterisk

Step 1: Install the TDM 400 Board

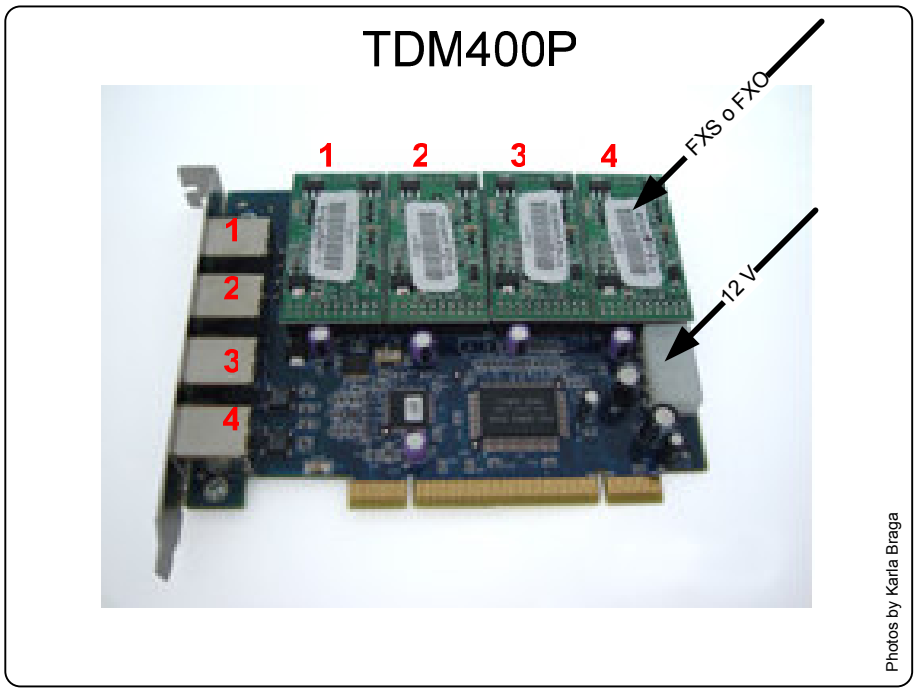


Figure 4.6 TDM400 card

The TDM400P contains FXS and FXO modules. Connect the modules FXS (S100M – green) and FXO (X100M – red) to the TDM400 if it is not already connected. FXS modules need additional power to provide ringing voltages. If you are using FXS, it will be necessary to connect the card directly to the power supply using the 12-volt connector (similar to the hard disk). Use static electricity protection equipment to handle the card to avoid damage.

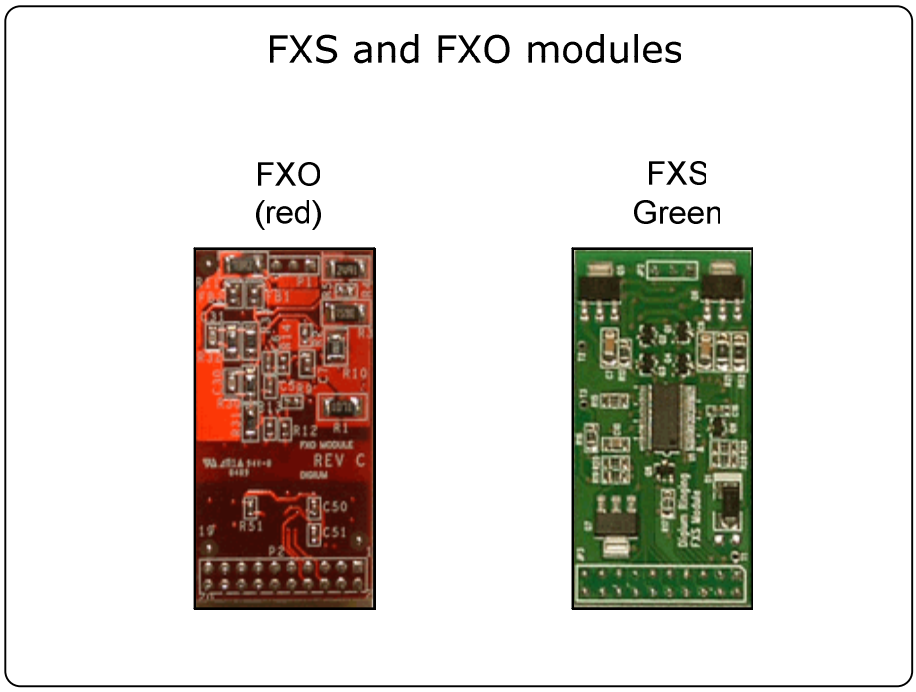


Figure 4.7 FXS Module (X100M - green), FXO module (S100M – red)

Step 2: zaptel.cfg configuration

Zaptel.cfg has to be edited to configure the zaptel driver. The file, zaptel.conf is in the /etc directory.

```
fxsks=1      # FXO port, certify that the red modules is in position 1.
fxoks=2      # FXS port
defaultzone=us
loadzone=us
```

Note: Analog signaling is a bit confusing, it is always the inverse of the card. FXS cards are signaled with FXO whereas FXO cards are signaled with FXS. Asterisk talks to these devices as if it was on the opposite side.

Step 3: Loading kernel drivers

Now you have to load the zaptel module and the related card kernel driver.

Digium drivers table

Card	Driver	Description
TE410P	wct4xxp	4xE1/T1-3.3V PCI
TE405P	wct4xxp	4xE1/T1-5V PCI
TDM400P	wctdm	4 FXS/FXO
T100P	wct1xxp	1 T1

E100P	wctlxxp	1 E1
X100P	wcfxo	1 FXO

```
modprobe zaptel
modprobe wctdm
```

Step 4: Using the zttest utility

An important utility is zttest. It is used to verify interrupt misses in the zaptel card. Audio quality problems are often related to interrupt conflicts.

To verify that your zaptel card is not sharing an interrupt with other cards you can use the below command:

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

You can verify the number of interrupt misses using the zttest utility compiled with the zaptel cards. A number below 99.987% indicates possible problems.

Step 5: Using the ztcfg utility to configure the driver

Zaptel has an unusual system to load the drivers. You first configure the zaptel.conf, and then you apply those configurations to the zaptel driver using ztcfg.

In this case, ztcfg is used to configure the signaling for the FX interfaces. To see the results you can append "-vvvvv" to the command for verbose.

```
#
/sbin/ztcfg -vv
Zaptel Configuration
======================
Channel map:
Channel 01: FXS Kewlstart (Default) (Slaves: 01)
Channel 02: FXO Kewlstart (Default) (Slaves: 02)
2 channels configured.
```

If the channels were loaded successfully, you will see an output similar to the one shown above. Users often wrongly configure zaptel.conf with inverted signaling between channels. If this happens, you will see a message like the one shown below:

```
ZT_CHANCONFIG failed on channel 1: Invalid argument (22)
Did you forget that FXS interfaces are configured with FXO signalling
and that FXO interfaces use FXS signalling?
```

After successfully configuring the hardware, you can proceed to Asterisk configuration.

Step 6: zapata.conf configuration file

It sounds strange, but after configuring the zaptel card, you just configured the card itself. Zaptel can be used for other purposes like routing and SS7. To use it with Asterisk it is time to configure Asterisk Zapata **channels**. Every channel in Asterisk has to be defined, SIP channels are defined in sip.conf, TDM channels are defined in zapata.conf. It will create the logical TDM channels to be used in your dial plan.

```
signalling=fxs_ks;
group=1;                channel group
context=incoming  ;     context
channel => 1;           channel number
signalling=fxo_ks;      FXO signaling for FXS interfaces
group=2;                channel group
context=extensions;     context
channel=> 2             channel number
```

4.6.2 Example #2 – Two T1 or E1 channels using ISDN

Required steps:

1. TE205P or TE210P installation
2. zaptel.conf file configuration
3. zaptel driver loading
4. zttest utility
5. ztcfg utility
6. zapata.conf file configuration
7. Asterisk load and testing

Step 1: TE205P installation

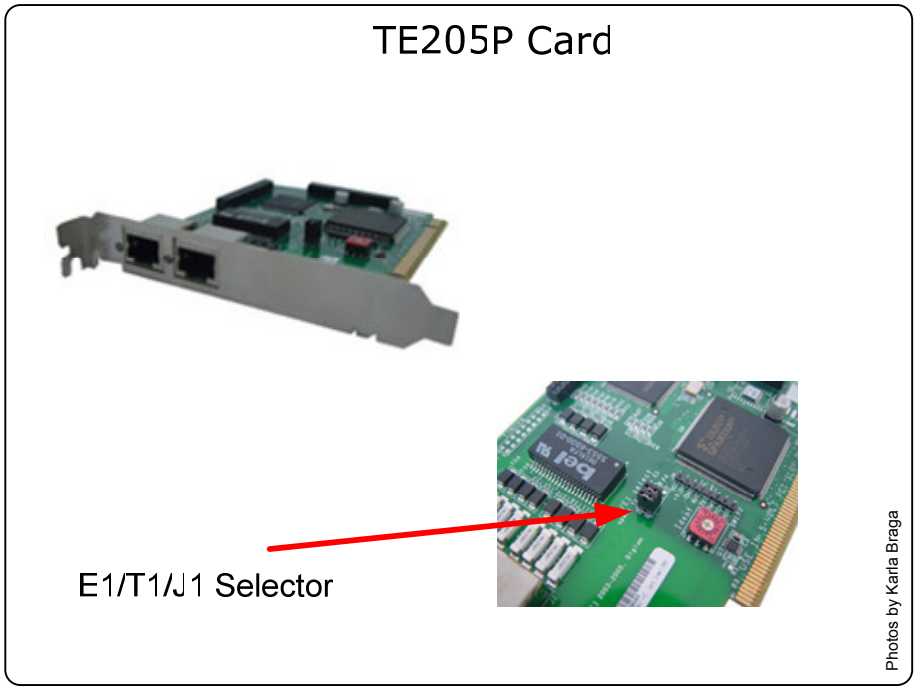


Figure 4.8 TE205P Card

To begin with, it is important to understand the differences between the TE205P and TE210P cards. The TE210P card uses a 64 bits bus powered by 3.3 volts found almost only in server's motherboards. Be careful if you specify this board; make sure your hardware supports the 64 bit, 3.3V bus. The TE205P card uses a 5V PCI which is common in desktop computers. We have chosen the TE205P interface card with two spans for this example because it is easier to reduce it to one span card or to expand it to the four spans card.

Step 2: zaptel configuration file

The configuration of TDM digital cards is a little bit different from the the configuration of their analog counterparts. First, we will need to configure the board spans and then the channels. Spans are numbered sequentially depending on the recognizing order of the boards. In other words when you have more than one interface card, it is hard to know what span belongs to each board.

TIP: Configure this normally. After Asterisk loads, take the cable out of the board and put it back. You will see a message that would look like *"primary span X UP"*. I believe this is the simplest procedure to determine which span belongs to which interface.

Example #1 (2xT1)

```
span=1,1,0,esf,b8zs
span=2,0,0,esf,b8zs
bchan=1-23
dchan=24
bchan=25-47
dchan=48
defaultzone=us
loadzone=us
```

Example #2 (2xE1)

```
span=1,1,0,ccs,hdb3,crc4 # not always necessary, consult Telco.
span=2,0,0,ccs,hdb3,crc4
bchan=1-15, 17-31
dchan=16
bchan=33-47, 49-63
dchan=48
defaultzone=br
loadzone=br
```

Step 3: Loading kernel drivers

Digium drivers table

Card	Driver	Description
TE410P	wct4xxp	4xE1/T1-3.3V PCI
TE405P	wct4xxp	4xE1/T1-5V PCI
TE210P	wct2xxp	2XE1/T1-3.3V PCI
TE205P	wct2xxp	2xE1/T1-5V PCI
TDM400P	Wctdm	4 FXS/FXO
T100P	wct1xxp	1 T1
E100P	Wctlxxp	1 E1
X100P	Wcfxo	1 FXO

```
modprobe zaptel
modprobe wct2xxp
```

Step 4: Using zttest to check missing interrupts

You can verify the number of interrupt misses using the zttest utility compiled with the zaptel cards. A number below 99.987% indicates possible problems. You will find zttest in /usr/src/asterisk.

```
#./zttest
Opened pseudo zap interface, measuring accuracy...
99.987793% 100.000000% 100.000000% 100.000000% 100.000000% 100.000000%
100.000000%
100.000000% 100.000000% 100.000000% 100.000000% 100.000000% 100.000000%
100.000000% 100.000000%
100.000000% 100.000000% 100.000000% 100.000000% 99.987793% 100.000000%
100.000000% 100.000000%
```

```
100.000000% 100.000000% 100.000000%
--- Results after 26 passes ---
Best: 100.000000 -- Worst: 99.987793 -- Average: 99.999061
```

Step 5: Using the ztcfg utility

This is the correct output for ztcfg for one fractional E1 (15 ports) span and two FXO ports.

```
#./ztcfg -vvvv

Zaptel Configuration
======================

SPAN 1: CCS/HDB3 Build-out: 0 db (CSU)/0-133 feet (DSX-1)

Channel map:

Channel 01: Clear channel (Default) (Slaves: 01)
Channel 02: Clear channel (Default) (Slaves: 02)
Channel 03: Clear channel (Default) (Slaves: 03)
Channel 04: Clear channel (Default) (Slaves: 04)
Channel 05: Clear channel (Default) (Slaves: 05)
Channel 06: Clear channel (Default) (Slaves: 06)
Channel 07: Clear channel (Default) (Slaves: 07)
Channel 08: Clear channel (Default) (Slaves: 08)
Channel 09: Clear channel (Default) (Slaves: 09)
Channel 10: Clear channel (Default) (Slaves: 10)
Channel 11: Clear channel (Default) (Slaves: 11)
Channel 12: Clear channel (Default) (Slaves: 12)
Channel 13: Clear channel (Default) (Slaves: 13)
Channel 14: Clear channel (Default) (Slaves: 14)
Channel 15: Clear channel (Default) (Slaves: 15)
Channel 16: D-channel (Default) (Slaves: 16)
Channel 32: FXS Kewlstart (Default) (Slaves: 32)
Channel 33: FXS Kewlstart (Default) (Slaves: 33)

18 channels configured.
```

Step 5: zapata.conf channels configuration

Example #1 (2xT1)

```
callerid="John Mac Enroe"<(555)555-1111>
switchtype=national
signalling =pri_cpe
group = 1
channel => 1-23
group =2
channel => 25-47
```

Example #2 (2xE1)

```
callerid=”Flavio Eduardo” <4830258580>
switchtype=euroisdn
signalling = pri_cpe
group = 1
channel => 1-15;17-31
group =2
channel => 32-46;48-62
```

4.6.3 Useful commands to verify the channels

zap show status:

```
vtsvoffice*CLI> zap show status
Description                         Alarms     IRQ        bpviol     CRC4
Digium Wildcard E100P E1/PRA Card 0 OK         0          0          0
Wildcard X100P Board 1              OK         0          0          0
Wildcard X100P Board 2              RED        0          0          0
```

pri show span:

```
vtsvoffice*CLI> pri show span 1
Primary D-channel: 16
Status: Provisioned, Up, Active
Switchtype: EuroISDN
Type: CPE
Window Length: 0/7
Sentrej: 0
SolicitFbit: 0
Retrans: 0
Busy: 0
Overlap Dial: 0
T200 Timer: 1000
T203 Timer: 10000
T305 Timer: 30000
T308 Timer: 4000
T313 Timer: 4000
N200 Counter: 3
```

zap show channels:

```
vtsvoffice*CLI> zap show channels
   Chan Extension  Context         Language   MusicOnHold
 pseudo            fax             en         default
      1            entrada         en         default
      2            entrada         en         default
      3            entrada         en         default
      4            entrada         en         default
      5            entrada         en         default
      6            entrada         en         default
      7            entrada         en         default
      8            entrada         en         default
```

```
      9            entrada         en         default
     10            entrada         en         default
     11            entrada         en         default
     12            entrada         en         default
     13            entrada         en         default
     14            entrada         en         default
     15            entrada         en         default
     32            default         en         default
     33            fax             en         default
```

zap show channel x:

```
vtsvoffice*CLI> zap show channel 1
Channel: 1*CLI>
File Descriptor: 21
Span: 1
Extension:
Dialing: no
Context: entrada
Caller ID: 4832341689
Calling TON: 33
Caller ID name:
Destroy: 0
InAlarm: 0
Signalling Type: PRI Signalling
Radio: 0
Owner: <None>
Real: <None>
Callwait: <None>
Threeway: <None>
Confno: -1
Propagated Conference: -1
Real in conference: 0
DSP: no
Relax DTMF: no
Dialing/CallwaitCAS: 0/0
Default law: alaw
```

debug pri span x: This command enables a detailed debugging of ISDN calls. It is a very important command when you think that something is not correct. You can detect digits being misdialed and other problems. Below we present the example of a debugging procedure for a successful call. Refer to this example if you need to compare to a call with problems. One tip is using core set verbose=0 to receive just the ISDN q.931 messages.

```
-- Making new call for cr 32833
> Protocol Discriminator: Q.931 (8)  len=57
> Call Ref: len= 2 (reference 65/0x41) (Originator)
> Message type: SETUP (5)
> [04 03 80 90 a3]
> Bearer Capability (len= 5) [ Ext: 1  Q.931 Std: 0  Info transfer capability:
Speech (0)
>                              Ext: 1  Trans mode/rate: 64kbps, circuit-mode
(16)
```

```
>                                Ext: 1  User information layer 1: A-Law (35)
> [18 03 a9 83 81]
> Channel ID (len= 5) [ Ext: 1  IntID: Implicit, PRI Spare: 0, Exclusive
Dchan: 0
>                         ChanSel: Reserved
>                        Ext: 1  Coding: 0   Number Specified   Channel Type: 3
>                        Ext: 1  Channel: 1 ]
> [28 0e 46 6c 61 76 69 6f 20 45 64 75 61 72 64 6f]
> Display (len=14) @h@>[ Flavio Eduardo ]
> [6c 0c 21 80 34 38 33 30 32 35 38 35 39 30]
> Calling Number (len=14) [ Ext: 0  TON: National Number (2)  NPI:
ISDN/Telephony Numbering Plan (E.164/E.163) (1)
>                             Presentation: Presentation permitted, user number
not screened (0) '4830258590' ]
> [70 09 a1 33 32 32 34 38 35 38 30]
> Called Number (len=11) [ Ext: 1  TON: National Number (2)  NPI:
ISDN/Telephony Numbering Plan (E.164/E.163) (1) '32248580' ]
> [a1]fice*CLI>
> Sending Complete (len= 1)
< Protocol Discriminator: Q.931 (8)  len=10
< Call Ref: len= 2 (reference 65/0x41) (Terminator)
< Message type: CALL PROCEEDING (2)
< [18 03 a9 83 81]
< Channel ID (len= 5) [ Ext: 1  IntID: Implicit, PRI Spare: 0, Exclusive
Dchan: 0
<                         ChanSel: Reserved
<                        Ext: 1  Coding: 0   Number Specified   Channel Type: 3
<                        Ext: 1  Channel: 1 ]
-- Processing IE 24 (cs0, Channel Identification)
< Protocol Discriminator: Q.931 (8)  len=9
< Call Ref: len= 2 (reference 65/0x41) (Terminator)
< Message type: ALERTING (1)
< [1e 02 84 88]
< Progress Indicator (len= 4) [ Ext: 1  Coding: CCITT (ITU) standard (0) 0: 0
Location: Public network serving the remote user (4)
<                                 Ext: 1  Progress Description: Inband
information or appropriate pattern now available. (8) ]
-- Processing IE 30 (cs0, Progress Indicator)
< Protocol Discriminator: Q.931 (8)  len=64
< Call Ref: len= 2 (reference 5720/0x1658) (Originator)
< Message type: SETUP (5)
< [04 03 80 90 a3]
< Bearer Capability (len= 5) [ Ext: 1  Q.931 Std: 0  Info transfer capability:
Speech (0)
<                                 Ext: 1  Trans mode/rate: 64kbps, circuit-mode
(16)
<                                 Ext: 1  User information layer 1: A-Law (35)
< [18 03 a1 83 82]
< Channel ID (len= 5) [ Ext: 1  IntID: Implicit, PRI Spare: 0, Preferred
Dchan: 0
<                         ChanSel: Reserved
<                        Ext: 1  Coding: 0   Number Specified   Channel Type: 3
<                        Ext: 1  Channel: 2 ]
< [1c 15 91 a1 12 02 01 bc 02 01 0f 30 0a 02 01 01 0a 01 00 a1 02 82 00]
< Facility (len=23, codeset=0) [ 0x91, 0xa1, 0x12, 0x02, 0x01, 0xbc, 0x02,
0x01, 0x0f, '0', 0x0a, 0x02, 0x01, 0x01, 0x0a, 0x01, 0x00, 0xa1, 0x02, 0x82,
0x00 ]
< [1e 02 82 83]
```

```
< Progress Indicator (len= 4) [ Ext: 1  Coding: CCITT (ITU) standard (0) 0: 0
Location: Public network serving the local user (2)
<                               Ext: 1  Progress Description: Calling
equipment is non-ISDN. (3) ]
< [6c 0c 21 83 34 38 33 32 32 34 38 35 38 30]
< Calling Number (len=14) [ Ext: 0  TON: National Number (2)  NPI:
ISDN/Telephony Numbering Plan (E.164/E.163) (1)
<                            Presentation: Presentation allowed of network
provided number (3) '4832248580' ]
< [70 05 c1 38 35 38 30]
< Called Number (len= 7) [ Ext: 1  TON: Subscriber Number (4)  NPI:
ISDN/Telephony Numbering Plan (E.164/E.163) (1) '8580' ]
< [a1]
< Sending Complete (len= 1)
-- Making new call for cr 5720
-- Processing Q.931 Call Setup
-- Processing IE 4 (cs0, Bearer Capability)
-- Processing IE 24 (cs0, Channel Identification)
-- Processing IE 28 (cs0, Facility)
Handle Q.932 ROSE Invoke component
-- Processing IE 30 (cs0, Progress Indicator)
-- Processing IE 108 (cs0, Calling Party Number)
-- Processing IE 112 (cs0, Called Party Number)
-- Processing IE 161 (cs0, Sending Complete)
> Protocol Discriminator: Q.931 (8)  len=10
> Call Ref: len= 2 (reference 5720/0x1658) (Terminator)
> Message type: CALL PROCEEDING (2)
> [18 03 a9 83 82]
> Channel ID (len= 5) [ Ext: 1  IntID: Implicit, PRI Spare: 0, Exclusive
Dchan: 0
>                        ChanSel: Reserved
>                       Ext: 1  Coding: 0   Number Specified   Channel Type: 3
>                       Ext: 1  Channel: 2 ]
> Protocol Discriminator: Q.931 (8)  len=14
> Call Ref: len= 2 (reference 5720/0x1658) (Terminator)
> Message type: CONNECT (7)
> [18 03 a9 83 82]
> Channel ID (len= 5) [ Ext: 1  IntID: Implicit, PRI Spare: 0, Exclusive
Dchan: 0
>                        ChanSel: Reserved
>                       Ext: 1  Coding: 0   Number Specified   Channel Type: 3
>                       Ext: 1  Channel: 2 ]
> [1e 02 81 82]
> Progress Indicator (len= 4) [ Ext: 1  Coding: CCITT (ITU) standard (0) 0: 0
Location: Private network serving the local user (1)
>                               Ext: 1  Progress Description: Called equipment
is non-ISDN. (2) ]
< Protocol Discriminator: Q.931 (8)  len=5
< Call Ref: len= 2 (reference 5720/0x1658) (Originator)
< Message type: CONNECT ACKNOWLEDGE (15)
< Protocol Discriminator: Q.931 (8)  len=9
< Call Ref: len= 2 (reference 65/0x41) (Terminator)
< Message type: PROGRESS (3)
< [1e 02 84 82]
< Progress Indicator (len= 4) [ Ext: 1  Coding: CCITT (ITU) standard (0) 0: 0
Location: Public network serving the remote user (4)
<                               Ext: 1  Progress Description: Called equipment
is non-ISDN. (2) ]
-- Processing IE 30 (cs0, Progress Indicator)
```

```
< Protocol Discriminator: Q.931 (8)  len=5
< Call Ref: len= 2 (reference 65/0x41) (Terminator)
< Message type: CONNECT (7)
> Protocol Discriminator: Q.931 (8)  len=5
> Call Ref: len= 2 (reference 65/0x41) (Originator)
> Message type: CONNECT ACKNOWLEDGE (15)
NEW_HANGUP DEBUG: Calling q931_hangup, ourstate Active, peerstate Connect
Request
> Protocol Discriminator: Q.931 (8)  len=9
> Call Ref: len= 2 (reference 65/0x41) (Originator)
> Message type: DISCONNECT (69)
> [08 02 81 90]
> Cause (len= 4) [ Ext: 1  Coding: CCITT (ITU) standard (0) 0: 0   Location:
Private network serving the local user (1)
>                  Ext: 1  Cause: Unknown (16), class = Normal Event (1) ]
< Protocol Discriminator: Q.931 (8)  len=5
< Call Ref: len= 2 (reference 65/0x41) (Terminator)
< Message type: RELEASE (77)
NEW_HANGUP DEBUG: Calling q931_hangup, ourstate Null, peerstate Release
Request
> Protocol Discriminator: Q.931 (8)  len=9
> Call Ref: len= 2 (reference 65/0x41) (Originator)
> Message type: RELEASE COMPLETE (90)
> [08 02 81 90]
> Cause (len= 4) [ Ext: 1  Coding: CCITT (ITU) standard (0) 0: 0   Location:
Private network serving the local user (1)
>                  Ext: 1  Cause: Unknown (16), class = Normal Event (1) ]
NEW_HANGUP DEBUG: Calling q931_hangup, ourstate Null, peerstate Null
NEW_HANGUP DEBUG: Destroying the call, ourstate Null, peerstate Null
< Protocol Discriminator: Q.931 (8)  len=9
< Call Ref: len= 2 (reference 5720/0x1658) (Originator)
< Message type: DISCONNECT (69)
< [08 02 82 90]
< Cause (len= 4) [ Ext: 1  Coding: CCITT (ITU) standard (0) 0: 0   Location:
Public network serving the local user (2)
<                  Ext: 1  Cause: Unknown (16), class = Normal Event (1) ]
-- Processing IE 8 (cs0, Cause)
NEW_HANGUP DEBUG: Calling q931_hangup, ourstate Disconnect Indication,
peerstate Disconnect Request
> Protocol Discriminator: Q.931 (8)  len=9
> Call Ref: len= 2 (reference 5720/0x1658) (Terminator)
> Message type: RELEASE (77)
> [08 02 81 90]
> Cause (len= 4) [ Ext: 1  Coding: CCITT (ITU) standard (0) 0: 0   Location:
Private network serving the local user (1)
>                  Ext: 1  Cause: Unknown (16), class = Normal Event (1) ]
< Protocol Discriminator: Q.931 (8)  len=5
< Call Ref: len= 2 (reference 5720/0x1658) (Originator)
< Message type: RELEASE COMPLETE (90)
NEW_HANGUP DEBUG: Calling q931_hangup, ourstate Null, peerstate Null
NEW_HANGUP DEBUG: Destroying the call, ourstate Null, peerstate Null
```

4.7 ZAPATA.CONF CONFIGURATION OPTIONS

There are several options in the zapata.conf file. A description of all options would be boring and counterproductive. Let us detail the main option groups available for easy understanding.

4.7.1 General options (channel independent)

context: Defines the incoming context.

```
context=default
```

channel: Defines channel or channel range. Each channel definition will inherit options defined before the declaration. Channels can be identified individually or in the same line with comma separation. Ranges can be defined using "-".

```
Channel=>1-15
Channel=>16
Channel=>17,18
```

group: Allows channels to be treated as a group. If you dial a group number instead of channel number, the first channel available is used. If channels are phones, when you call a group all phones will ring simultaneously. With commas, you can specify more than one group for the same channel.

```
group=1
group=3,5
```

language: Turns on the internationalization and configures a language. This feature will configure system messages for a specific language. English is the only language with complete prompts available from the standard installation.

musiconhold: Select music on hold class

4.7.2 ISDN options

switchtype: Is dependent on the PBX or switch used. In Europe and Latin America, EuroISDN is common. In the USA the most commonly used is National or manufacturer-dependent PBX.

5ess: Lucent 5ESS
euroisdn: EuroISDN
national: National ISDN
dms100: Nortel DMS100

4ess: AT&T 4ESS
Qsig: Q.SIG

```
switchtype = EuroISDN
```

pridialplan: Necessary for some switches that require a dialplan specification. This option is ignored for most equipment. Valid options are private, national, international, and unknown.

```
pridialplan = unknown
```

prilocaldialplan: Necessary for some switches, usually EuroISDN.

```
prilocaldialplan = unknown
```

overlapdial: Overlap dialing is used when you pass digits after the connection is established, usually block mode numbering (overlapdial=no) or digit mode (overlapdial=yes). Block mode is commonly used by operators.

signaling: Configures signaling type for the subsequentchannels. These parameters should correspond to the ones in the zaptel.conf file. Correct choices are based on the available channel. For ISDN you could choose two types.

- **pri_cpe:** Used when the device is a CPE (Customer Promises Equipment) and usually referred as cpe, client, user or slave. This is the simplest and most used form of signaling. Sometimes, when you try to connect to a private PBX, it is common practice that the PBX be configured as a cpe too. In this case, use pri_net signaling in Asterisk.

- **pri_net:** Used when Asterisk connects to a private PBX configured as CPE. The signaling is often referred as Host, Master or Network.

4.7.3 CallerID options

There are many CallerID options. Some can be disabled although most are enabled by default.

usecallerid: Enables or disables the CallerID transmission for the subsequent channels (Yes/No).

Note: If your system needs two rings before answering, try disabling this feature. It should answer immediately.

---

hidecallerid: It hides the CallerID. (Yes/No)

calleridcallwaiting: enables CallerID receiving during a call waiting indication. (Yes/No)

callerid: Configures a CallerID string for a specific channel. The caller can be configured with "asreceived" in trunk interfaces to pass the CallerID forward.

```
callerid = "Flavio Eduardo Gonçalves" <48 30258500>
```

Note: Only PRI lines can transmit callerID. Most Telcos mandate that you configure your correct callerID. If you do not pass the CallerID often, you should not be able to dial out over the Telco. You will receive calls even with an incorrect CallerID.

4.7.4 Audio quality options

These options adjust certain Asterisk parameters that affect audio quality in the Zapata channels.

echocancel: Disables or enables echo cancellation. You should keep this feature enabled. It accepts "yes" or the number of taps.

Explanation: How echo canceling works?

Most echo canceling algorithms operate by generating multiple copies of a received signal, each delayed by a small interval. This little flow is named "tap". The number of taps determines the echo delay that can be cancelled. These copies are delayed, adjusted and subtracted from the original signal. The trick is to adjust the delayed signal exactly to what is necessary to remove echo.
echocancelwhenbridged: Enables or disables the echo canceller during a pure TDM call. This is usually not necessary.

rxgain: Adjusts the audio reception gain to either increase or decrease reception volume (-100% to 100%).

txgain: Adjusts audio transmission gain to either increase or decrease the transmission volume (-100% to 100%).

Example:

```
echocancel=yes
```

```
echocancelwhenbridged=yes
txgain=-10%
rxgain=10%
```

4.7.5 Billing options

These options change the way call information is recorded in the CDR (call detail records) database.

amaflags: Configures the AMA flags affecting categorization of CDR. It accepts these values:

- **billing**
- **documentation**
- **omit**
- **default**

accountcode: Configures an account code for a specific channel. It can contain any alphanumeric value, usuallyI the department or user name.

```
accountcode=finance
amaflags=billing
```

4.7.6 Call progress options

These items are used to emulate an existing signaling in digital lines to monitor call progress. Since analog channels do not pass these options, they need to be emulated for them.

busydetect: Asterisk analyzes the audio coming through the line during a call or a dial attempt in order to recognize busy signals in analog lines like FXO, FXS e E+M.

callprogress: Enable this feature if you want Asterisk to monitor the call state and detect busy, ring and active line.

```
callprogress=no
busydetect=yes
```

4.7.7 Options for phones connected to FXS interfaces

Adsi (Analog Display Services Interface): It is a set of telecom standards. It was used by some telcos to offer services such as ticket buying, for example.

cancallforward: Enables or disables call forwarding (*72 to enable and *73 to disable).

immediate: In *immediate* mode, instead of providing adial tone, the channel jumps immediately to the "s" extension in the defined context. It is used to create hotlines.

threewaycalling: Enables or disables three-way conferencing.

transfer: Enables or disables transfers using the flash key. To use this option, threewaycalling has to be set as "yes".

mailbox: Warns the user about voicemail messages available for him/her. It can be an audible sign or a visual indicator (in case the telephone supports this feature). The argument is the mailbox number.

4.7.8 Options for FXO trunks.

cidsignalling: CallerID signaling for analog lines

- bell = bell202 as used in the USA
- v23 = v23 as used in the United Kingdom
- dtmf = DTMF as used in Denmark, the Netherlands, and Brazil

```
cidsignalling=bell
```

cidstart: It defines what starts the CallerID, polarity reversal, or ring.

- ring
- polarity

```
cidstart=ring
```

4.8 MFC/R2 CONFIGURATION

MFC/R2 is used in several countries in Latin America, China, Africa, and some European countries. ISDN is superior and preferred if you have it available in your area.

4.8.1 Understanding the problem

The board used to signal MFC/R2 is the same used to signal ISDN. Digium has an incomplete implementation of R2 signaling inside the Zapata channel.

Unfortunately according to Mark Spencer, R2 in zapata is far from being implemented. To use MFC/R2 signaling you will have to use the driver developed by Steve Underwood and available at www.soft-switch.org. Recently, some manufacturers released telephony interface cards with R2 support in DSPs (in the USA, Aculab; in Brazil, khomp and digivoice can be used). In this chapter, we will describe how to implement MFC/R2 using the unicall channel driver.

4.8.2 Understanding the MFC/R2 protocol

The MFC/R2 protocol combines in-band signaling. Address signaling is forwarded using a set of tones and out-of-band signaling and channel signaling is transmitted in the timeslot 16.

Line Signaling (ITU-T Q.421)

In timeslot 16, each voice channel uses four ABCD bits to signal their states and call controlling. Bits C and D are rarely used. In some countries, they can be used for metering (pulse metering for billing). In a normal conversation, we have the two sides operating, the caller side and the called side. Signaling from the caller side is referred as forward signaling. The called side uses backward signaling. We will designate Af and Bf for forwarding signaling and Ab and Bb for backward signaling.

State	ABCD forward	ABCD backward
Idle/Released	1001	1001
Seized	0001	1001
Seize Ack	0001	1101
Answered	0001	0101
ClearBack	0001	1101
ClearFwd (Before clear-back)	1001	0101
ClearFwd (disconnection confirmation)	1001	1001
Blocked	1001	1101

MFC/R2 was defined by the ITU. Unfortunately, several countries customized the standard to their own needs. Therefore, there are variations in standard between countries.

Inter-register signals (ITU-T Q.441)

MFC/R2 signaling uses a combination of two tones. The table below shows the ITU standard.

Signal group I (Forward)

Signal	Description	Forward signal

1	Digit 1	I-1
2	Digit 2	I-2
3	Digit 3	I-3
4	Digit 4	I-4
5	Digit 5	I-5
6	Digit 6	I-6
7	Digit 7	I-7
8	Digit 8	I-8
9	Digit 9	I-9
10	Digit 0	I-10
11	Country code indicator, outgoing half-echo suppressor required	I-11
12	Country code indicator, no echo suppressor required	I-12
13	Test call indicator	I-13
14	Country code indicator, outgoing half-echo suppressor inserted	I-14
15	Not used	I-15

Signal group II (Forward)

Signal	Description	Forward signal
1	Subscriber without priority	II-1
2	Subscriber with priority	II-2
3	Maintanance equipment	II-3
4	Spare	II-4
5	Operator	II-5
6	Data Transmission	II-6
7	Subscriber or operator without forward transfer facility	II-7
8	Data transmission	II-8
9	Subscriber with priority	II-9
10	Operator with forward transfer facility	II-10
11	Spare	II-11
12	Spare	II-12
13	Spare	II-13
14	Spare	II-14
15	Spare	II-15

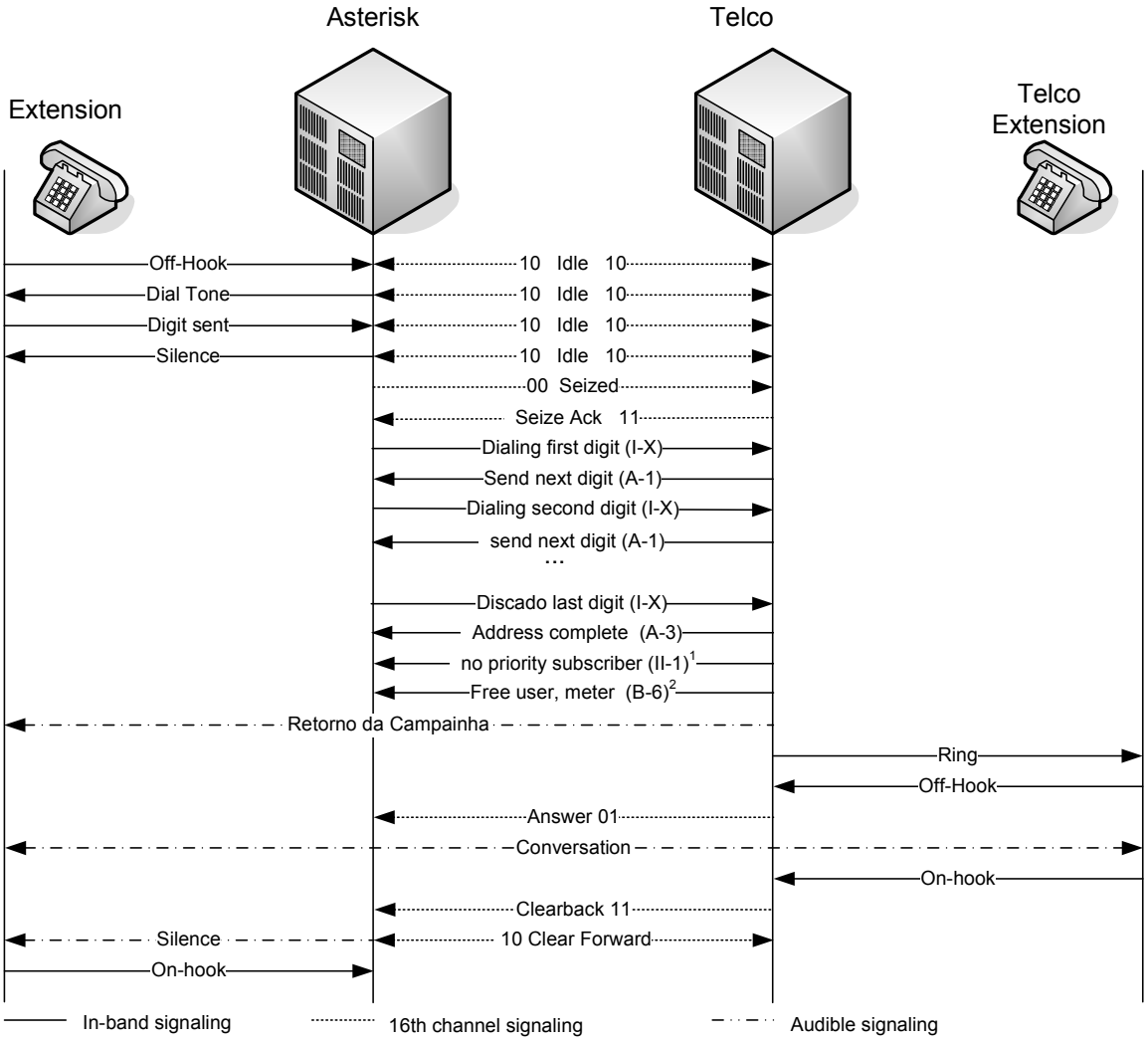
Signal group A (backwards)

Signal	Description	Backwards signal
1	Send next digit (n+1)	A-1
2	Send last but one digit (n-1)	A-2
3	Address complete, changeover to reception of Group B signals	A-3
4	Congestion in the national network	A4
5	Send calling party’s category	A5
6	Address complete, charge, set-up speech conditions	A6
7	Send last but two digit (n-2)	A7
8	Send last but three digit (n-3)	A8
9	Spare	A9
10	Spare	A10
11	Send country code indicator	A11
12	Send language or discrimination digit	A12
13	Send nature of circuit	A13
14	Request information on use of echo suppressor	A14
15	Congestion in an international exchange or at its output	A15

Signal group B (backwards)

Signal	Description	Signal backwards
1	Spare	B1
2	Send special information tone	B2
3	Subscriber's line busy	B3
4	Congestion (after changeover group A to B)	B4
5	Unallocated number	B5
6	Subscriber's line free, charge	B6
7	Subscriber's line free, no charge	B7
8	Subscriber's line out of order	B8
9	Spare	B9
10	Spare	B10
11	Spare	B11
12	Spare	B12
13	Spare	B13
14	Spare	B14
15	Spare	B15

4.8.3 MFC/R2 sequence



The sequence above illustrates a call originating from an Asterisk's extension to a terminal in the PSTN. The PSTN drops the call and ends the communication.

4.8.4 The unicall driver

The unicall driver was developed by Steve Underwood and has a GPL license agreement. It is not part of Asterisk, is not supported by Digium, and is very similar to zaptel channel.

The resources used by a Zapata channel are:

```
PSTN->ZAPTEL Card>ZAPTEL Driver->LIBPRI->CHAN_ZAP->ASTERISK
```

In the unicall driver, the sequence is a little bit different.

```
PSTN->PLACA ZAPTEL->DRIVER ZAPTEL->LIMFCR2->LIBUNICALL->CHAN_UNICALL->ASTERISK
```

4.8.5 MFC/R2 configuration

Assuming now that the installation and configuration of the zaptel driver is already complete, let us start with the zaptel.conf configuration. The digits after the channels put the channel in blocked mode (verify what are the digits for blocked mode in your country the most common are 1101 and 1001).

```
# MFC/R2 usually does not use CRC4
span=1,1,0,cas,hdb3
cas=1-15:1101 ;Depends on the R2 variant (ABCD bits of blocked mode)
dchan=16
cas=17-31:1101
span-2,0,0,cas,hdb3
cas=33-47:1101
dchan=48
cas=49-63:1101
loadzone=br
defaultzone=br
```

To put bits in blocked mode, it is important to signal to the operator that the server is not ready. The card is connected, but not ready to receive calls.

Execute ZTCFG.

```
bash# ztcfg -v

Zaptel Configuration
==================

SPAN 1: CAS/HDB3 Build-out: 0 db (CSU)/0-133 feet (DSX-1)
SPAN 2: CAS/HDB3 Build-out: 0 db (CSU)/0-133 feet (DSX-1)

62 channels configured.
```

4.8.6 Libraries installation and configuration

To install R2, it is necessary to download and compile the following libraries: spandsp, libmfcr2, and libunicall, available at ftp.soft-switch.org.

spandsp

```
#cd /usr/src
#wget http://www.soft-switch.org/downloads/spandsp/spandsp-0.0.2pre25/spandsp-
0.0.2pre25.tar.gz
#tar -xzvf spandsp-0.0.2pre25.tar.gz
#cd spandsp-0.0.2
# ./configure --prefix=/usr
#make
#make install
```

libmfcr2

```
#cd /usr/src
#wget http://www.soft-switch.org/downloads/unicall/unicall-0.0.3pre9/libmfcr2-
0.0.3.tar.gz
#tar -xzvf libmfcr2-0.0.3.tar.gz
#cd libmfcr2-0.0.3.tar.gz
# ./configure --prefix=/usr
#make
#make install
```

libsupertone

```
#cd /usr/src
#wget http://www.soft-switch.org/downloads/unicall/unicall-
0.0.3pre9/libsupertone-0.0.2.tar.gz
#tar -xzvf libsupertone-0.0.2.tar.gz
#cd libsupertone-0.0.2
# ./configure --prefix=/usr
#make
#make install
```

libunicall

```
#cd /usr/src
#wget http://www.soft-switch.org/downloads/unicall/unicall-
0.0.3pre9/libunicall-0.0.3.tar.gz
#tar -xzvf libunicall-0.0.3.tar.gz
#cd libunicall-0.0.3.tar.gz
# ./configure --prefix=/usr
#make
#make install
```

4.8.7 Integrating Unicall to Asterisk

Download chan_unicall related files.

```
#wget http://www.soft-switch.org/downloads/unicall/unicall-0.0.3pre9/asterisk-
1.1.x/chan_unicall.c
#wget http://www.soft-switch.org/downloads/unicall/unicall-0.0.3pre9/asterisk-
1.1.x/channels_Makefile.patch
#wget http://www.soft-switch.org/downloads/unicall/unicall-0.0.3pre9/asterisk-
1.1.x/unicall.conf.sample
```

Copy the following file to Asterisk's compilation structure.

```
# cp chan_unicall.c channels_makefile.patch /usr/src/asterisk/channels
```

Apply the patch to the Asterisk channels.

```
#cd /usr/src/asterisk/channels
#patch < channels_makefile.patch
```

Next, recompile Asterisk.

```
cd /usr/src/asterisk/
make clean
make
make install
```

4.8.8 Unicall channel configuration

You need to configure the unicall channels in the same way you had to do it with the Zapata channels. Edit the unicall.conf file, syntax and attributes are very similar to zapata.conf.

```
;
; Unicall telephony channel driver
;
; Sample configuration file
;
; $Id: unicall.conf.sample,v 1.1 2005/05/28 11:17:02 steveu Exp $
;
[channels]
;
; Default language
;
language=br
;
; Default context
;
context=default
;
; Whether or not to use caller ID
;
usecallerid=yes
;
; Whether or not to hide outgoing caller ID
;
hidecallerid=no
;
; Whether or not restrict outgoing caller ID (will be sent as ANI only, not available for the
user)
; Mostly use with FXS ports
;
;restrictcid=no
;
; Support Caller*ID on Call Waiting
;
callwaitingcallerid=yes
;
; Support three-way calling
```

```
;
threewaycalling=yes
;
; Support flash-hook call transfer (requires three way calling)
;
transfer=yes
;
; Support call forward variable
;
cancallforward=yes
;
; Whether or not to support Call Return (*69)
;
callreturn=yes
;
; Enable echo cancellation
; Use either "yes", "no", or a power of two from 32 to 256 if you wish
; to actually set the number of taps of cancellation.
;
echocancel=yes
;
; Generally, it is not necessary (and in fact undesirable) to echo cancel
; when the circuit path is entirely TDM.  You may, however, reverse this
; behavior by enabling the echo cancel during pure TDM bridging below.
;
echocancelwhenbridged=yes
;
; 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.  Value may be "yes", "no", or a number of
; milliseconds to delay before training (default = 400)
;
;echotraining=yes
;echotraining=800
;
; If you are having trouble with DTMF detection, you can relax the
; DTMF detection parameters.  Relaxing them may make the DTMF detector
; more likely to have "talkoff" where DTMF is detected when it
; shouldn't be.
;
;relaxdtmf=yes
;
; You may also set the default receive and transmit gains (in dB)
;
rxgain=0.0
txgain=0.0
;
; Logical groups can be assigned to allow outgoing rollover.  Groups
; range from 0 to 31, and multiple groups can be specified.
;
group=1
;
; 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 and dialing *8#.  For simple offices, just
; make these both the same
;
callgroup=1
pickupgroup=1
;
; Specify whether the channel should be answered immediately or
; if the simple switch should provide dialtone, read digits, etc.
;
immediate=no
;
; CallerID can be set to "asreceived" or a specific number
; if you want to override it.  Note that "asreceived" only
; applies to trunk interfaces.
;
callerid=asreceived
```

```
;
; AMA flags affects the recording of Call Detail Records.  If specified
; it may be 'default', 'omit', 'billing', or 'documentation'.
;
;amaflags=default
;
; Channels may be associated with an account code to ease
; billing
;
;accountcode=lss0101
;
; For fax detection, uncomment one of the following lines.  The default is *OFF*
;
;faxdetect=both
;faxdetect=incoming
;faxdetect=outgoing
;faxdetect=no
;
; Select which class of music to use for music on hold.  If not specified
; then the default will be used.
;
;musiconhold=default
;
;protocolclass=fx
;protocolvariant=ls,hk
;protocolend=co
;group = 3
;channel => 280-283

;protocolclass=fx
;protocolvariant=ls,hk
;protocolend=cpe
;group = 4
;channel => 284

;
; Set up E1s 2 and 3 to work in China MFC/R2 mode. A maximum of 20 ANI digits
; will be accepted. 7 DNIS digits are expected. MFC/R2 uses the E1s in CAS mode,
; so time slot 16 of each E1 must be skipped when allocating the channels.
;
;loglevel=255
protocolclass=mfcr2
; For MFC/R2 an optional fourth parameter for the variant is composed of bits,
; which must be OR'ed together, as follows:
;
;   1:  Play progress tones. These are usually handled by the far end switch, but
;       may need to be sent as audio through the channel on some systems.
;   2:  Play disconnect tone. The disconnect tone is usually handled by the far end
;       switch, but may need to be sent as audio through the channel on some systems.
;   4:  Play ringback tone. The ringback tone is usually generated by something
;       downstream of the MFC/R2 software, but may need to be generated here on some
;       systems.
;   8:  Get ANI after DNIS. The usual behaviour for incoming calls is to get the
;       calling party category and the ANI as soon as possible, and to get the DNIS
;       afterwards. This doesn't work on all systems, so the option to reverse the
;       behaviour is provided.
;   16: Use immediate accept. Most variants of MFC/R2 offer a way to go directly to
;       the call accepted state, bypassing the use of group B and II tones. This option
;       enables the use of that feature for incoming calls.
;
protocolvariant=br,20,4
protocolend=cpe
group = 1
channel => 1-15
;skip time slot 16
channel => 17-31
channel => 33-47
;skip time slot 48
channel => 49-63
```

Let us take a closer look at the configurations described above:

```
protocolvariant=country,ANI-digits,DNIS-digits
```

Country codes table:

Argentina	"ar"
Bahrain	"bh"
Bolivia	"bo"
Brazil	"br"
Chile	"cl"
China	"cn"
Colombia landlines	"co-land"
Colombia cellular	"co-cell"
Czech	"cz"
Honduras	"hn"
India	"in"
Indonesia	"id"
Korea	"kr"
Malaysia	"my"
Mexico	"mx"
Panama	"pa"
Philipinnes	"ph"
Singapore	"sg"
Thailand	"th"

ANI and DNIS

ANI or Automatic Number Identification is the caller's number. DNIS or Dialed Number Identification Service is the number called or, in other words, the number dialed.

When a call is received, usually the last four numbers are passed to the PBX in a process named DID (direct inward dial). The ANI number is actually the CallerID.

ANI will have the caller's extension when dialing. DNIS in this case will contain the call destination.

It is important that these parameters be configured correctly. Some switches send just the last four digits while others send the complete number.

In the example below we will use DNIS=4 and ANI=20. In other words, we will receive just the last four digits identifying the extension. The CallerID may have 20 digits.

```
protocolvariant=br,20,4
```

4.8.9 Unicall Troubleshooting

Start Asterisk with verbose 15.

```
asterisk –vvvvvvvvvvvvvvv&
asterisk – r
```

While Asterisk is loading, you will see messages similar to the ones shown below:

```
Apr 11 06:45:07 WARNING[24876]: Unicall/1 event Far end unblocked
Apr 11 06:45:07 VERBOSE[24876]: Asterisk Ready.
    -- Unicall/1 far unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/1 event Local end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/1 local unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/2 event Far end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/2 far unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/2 event Local end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/2 local unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/3 event Far end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/3 far unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/3 event Local end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/3 local unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/4 event Far end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/4 far unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/4 event Local end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/4 local unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/5 event Far end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/5 far unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/5 event Local end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/5 local unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/6 event Far end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/6 far unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/6 event Local end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/6 local unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/7 event Far end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/7 far unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/7 event Local end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/7 local unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/8 event Far end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/8 far unblocked
Apr 11 06:45:07 WARNING[24876]: Unicall/8 event Local end unblocked
Apr 11 06:45:07 VERBOSE[24876]:     -- Unicall/8 local unblocked
```

This message indicates that channels passed from the blocked state (11) to the idle state (10). Local end unblocked means that the Asterisk R2 channel is unblocked and ready to work. Far end unblocked means that the Telco is ready to work. If one side remains blocked, the line can be not activated.

UC show channels: You can show channel states using:

```
vtsvoffice*CLI>UC show channels
Channel Extension  Context         Status     Language   MusicOnHold
      1            e1-incoming     Idle                  default
      2            e1-incoming     Idle                  default
      3            e1-incoming     Idle                  default
      4            e1-incoming     Idle                  default
      5            e1-incoming     Idle                  default
      6            e1-incoming     Idle                  default
```

All channels should be in the "idle" state. If the console command did not appear, check unicall installation.

To debug a call using MFC/R2 you will need to edit unicall.conf file and take off the ";" before "loglevel=255" line and restart Asterisk. Use verbose "0" to have just the unicall debugging without other messages. Below is an example that you can use to compare to your call. The call below was completed and released normally. The callerID (ANI) is 1149295000 and the dialed number (DNIS) is 0154830258576.

```
Apr 12 08:33:49 WARNING[4417]: MFC/R2 UniCall/1 Call control(1)
Apr 12 08:33:49 WARNING[4417]: MFC/R2 UniCall/1 Make call
Apr 12 08:33:49 WARNING[4417]: MFC/R2 UniCall/1 Making a new call with CRN 32769
Apr 12 08:33:49 WARNING[4417]: MFC/R2 UniCall/1 0001  ->      [1/       1/Idle          /Idle         ]
Apr 12 08:33:49 WARNING[4417]: Unicall/1 event Dialing
Apr 12 08:33:50 WARNING[4417]: MFC/R2 UniCall/1      <- 1101  [1/      40/Seize         /Idle         ]
Apr 12 08:33:50 WARNING[4417]: MFC/R2 UniCall/1 0 on  ->      [2/      40/Group I       /Idle         ]
Apr 12 08:33:51 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:51 WARNING[4417]: MFC/R2 UniCall/1 0 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:51 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:51 WARNING[4417]: MFC/R2 UniCall/1 1 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:51 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:51 WARNING[4417]: MFC/R2 UniCall/1 1 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:51 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:51 WARNING[4417]: MFC/R2 UniCall/1 5 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1 5 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1 4 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1 4 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1 Calling party category 0x0
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1 1 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /Category     ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1 1 off ->      [2/      40/Group I       /Category     ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /Category     ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1 1 on  ->      [2/      40/Group I       /Category     ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
Apr 12 08:33:52 WARNING[4417]: MFC/R2 UniCall/1 1 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1 1 on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1 1 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1 4 on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1 4 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1 9 on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1 9 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:53 WARNING[4417]: MFC/R2 UniCall/1 2 on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1 2 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1 9 on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1 9 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1 5 on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1 5 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:54 WARNING[4417]: MFC/R2 UniCall/1 0 on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1 0 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1 0 on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
```

```
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1 0 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1 0 on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1 0 off ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1 E on  ->      [2/      40/Group I       /ANI          ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /End of ANI   ]
Apr 12 08:33:55 WARNING[4417]: MFC/R2 UniCall/1 E off ->      [2/      40/Group I       /End of ANI   ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /End of ANI   ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1 8 on  ->      [2/      40/Group I       /End of ANI   ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1 8 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1 3 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1 3 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1 0 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1 0 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:56 WARNING[4417]: MFC/R2 UniCall/1 2 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1 2 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1 5 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1 5 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1 8 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1 8 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:57 WARNING[4417]: MFC/R2 UniCall/1 5 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:58 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:58 WARNING[4417]: MFC/R2 UniCall/1 5 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:58 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:58 WARNING[4417]: MFC/R2 UniCall/1 7 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:58 WARNING[4417]: MFC/R2 UniCall/1      <- 1 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:33:58 WARNING[4417]: MFC/R2 UniCall/1 7 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:33:58 WARNING[4417]: MFC/R2 UniCall/1      <- 1 off [2/      40/Group I       /DNIS         ]
Apr 12 08:33:58 WARNING[4417]: MFC/R2 UniCall/1 6 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:34:01 WARNING[4417]: MFC/R2 UniCall/1      <- 3 on  [2/      40/Group I       /DNIS         ]
Apr 12 08:34:01 WARNING[4417]: MFC/R2 UniCall/1 6 off ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:34:01 WARNING[4417]: MFC/R2 UniCall/1      <- 3 off [2/      40/Group I       /DNIS         ]
Apr 12 08:34:01 WARNING[4417]: MFC/R2 UniCall/1 1 on  ->      [2/      40/Group I       /DNIS         ]
Apr 12 08:34:01 WARNING[4417]: MFC/R2 UniCall/1      <- 5 on  [2/      40/Group II      /Category     ]
Apr 12 08:34:01 WARNING[4417]: MFC/R2 UniCall/1 1 off ->      [2/      40/Group II      /Category     ]
Apr 12 08:34:01 WARNING[4417]: MFC/R2 UniCall/1      <- 5 off [2/      40/Group II      /Category     ]
Apr 12 08:34:01 WARNING[4417]: Unicall/1 event Alerting
Apr 12 08:34:04 WARNING[4417]: MFC/R2 UniCall/1      <- 0101  [1/     200/Await answer  /Category     ]
Apr 12 08:34:04 WARNING[4417]: Unicall/1 event Connected
Apr 12 08:34:25 WARNING[4417]: MFC/R2 UniCall/1      <- 1101  [1/     400/Answered      /Category     ]
Apr 12 08:34:25 WARNING[4417]: MFC/R2 UniCall/1 Far end disconnected(cause=Normal Clearing [16]) - state
0x400
Apr 12 08:34:25 WARNING[4417]: Unicall/1 event Far end disconnected
Apr 12 08:34:25 WARNING[4417]: CRN 32769 - far disconnected cause=Normal Clearing [16]
Apr 12 08:34:25 WARNING[4417]: MFC/R2 UniCall/1 Call control(6)
Apr 12 08:34:25 WARNING[4417]: MFC/R2 UniCall/1 Drop call(cause=Normal Clearing [16])
Apr 12 08:34:25 WARNING[4417]: MFC/R2 UniCall/1 Clearing fwd
Apr 12 08:34:25 WARNING[4417]: MFC/R2 UniCall/1 1001  ->      [1/     800/Clear back    /Category     ]
Apr 12 08:34:28 NOTICE[4417]: Peer '2222' is now TOO LAGGED!
Apr 12 08:34:30 WARNING[4417]: MFC/R2 UniCall/1      <- 1001  [1/     800/Clear fwd D   /Idle         ]
Apr 12 08:34:30 WARNING[4417]: MFC/R2 UniCall/1 Call disconnected(cause=Normal Clearing [16]) - state 0x800
Apr 12 08:34:30 WARNING[4417]: Unicall/1 event Drop call
Apr 12 08:34:30 WARNING[4417]: MFC/R2 UniCall/1 Call control(7)
Apr 12 08:34:30 WARNING[4417]: MFC/R2 UniCall/1 Release call
Apr 12 08:34:30 WARNING[4417]: MFC/R2 UniCall/1 Destroying call with CRN 32769
Apr 12 08:34:30 WARNING[4417]: Unicall/1 event Release call
```

4.9 ZAP CHANNEL FORMAT.

ZAP channels use the following format in the dial plan:

```
Zap/[g]<identifier>[c][r<cadence>]

<identifier>- Physical channel numeric identifier
[g] – Group identifier
[c] – Answer confirmation. A number is not considered until the callee press
“#”
[r] – customized ringing
[cadence] Integer from 1 to 4
```

Examples:

```
zap/2 - channel 2
zap/g1 - First available channel in group 1
```

4.10 UNICALL CHANNEL FORMAT

Unicall channels use the following format in the dial plan:

```
Unicall/[g]<identifier>[c][r<cadence>]

<identifier>- Physical channel numeric identifier
[g] – Group identifier
[c] – Answer confirmation; A number is not considered until the callee press
“#”
[r] – customized ringing
[cadence] Integer from 1 to 4
```

Example:

```
Unicall/2   - Channel 2
Unicall/g1  - First available channel in group 1
```

4.11 QUESTIONS

1 – Supervision signaling includes:

a) On-hook
b) Off-hook
c) Ringing
d) Dtmf

2 – Information signaling includes:

a) Dtmf
b) Dial tone
c) Invalid number
d) Ringback

e) Congestion
f) Busy
g) Pulse

3 – There are two types of analog interfaces available for Asterisk, FXS and FXO. Mark the correct answers.

a) FXS – Foreign Exchange Station can be connected directly to the company PBX extension port.
b) FXO – Foreign Exchange Office can be connected to the public switched telephony network.
c) FXS – Foreign Exchange Station provides dial tone and can be connected to a standard analog phone.

4 – About T1 and E1 signaling, mark the correct affirmations.

a) E1 is a digital signaling that use 1.544 Mbits/s bandwidth
b) T1 is often used in Latin America and Europe
c) It is possible to use 30 channels for an E1 trunk and 23 channels for a T1 trunk in an ISDN PRI configuration.
d) ISDN is an example of CCS signaling while MFC/R2 is an example of CAS signaling.

5 – MFC/R2 signaling is supported by a third-party driver developed by Steve Underwood available in www.soft-switch.org.

a) False
b) True

6 – To configure zaptel hardware you should first edit the _____ file:

a) zaptel.conf
b) zapata.conf
c) unicall.conf
d) serial.conf

7 – The zaptel hardware is independent of Asterisk. In the zapata.conf, you configure Asterisk channels and not the hardware itself.

a) False
b) True

8 – When using a TDM400 with a ___ port is necessary to connect the PC power source to the card using a specific connector (similar to the used to power the hard disk).

a) FXO
b) FXS
c) E+M
d) ISDN

9 – Echo, pops and noise in a zaptel card are often related to the:

a) Asterisk compilation
b) Cable problems
c) PCI Interrupt conflicts
d) Electromagnetic interference

10 – R2 signaling defined by ITU is standardized in the whole world and there are no variations to the standard county dependent.

a) True
b) False

Chapter 5

Voice over IP with Asterisk

In this chapter, we will learn the basics of VoIP applied to the Asterisk scenario.

5.1 OBJECTIVES

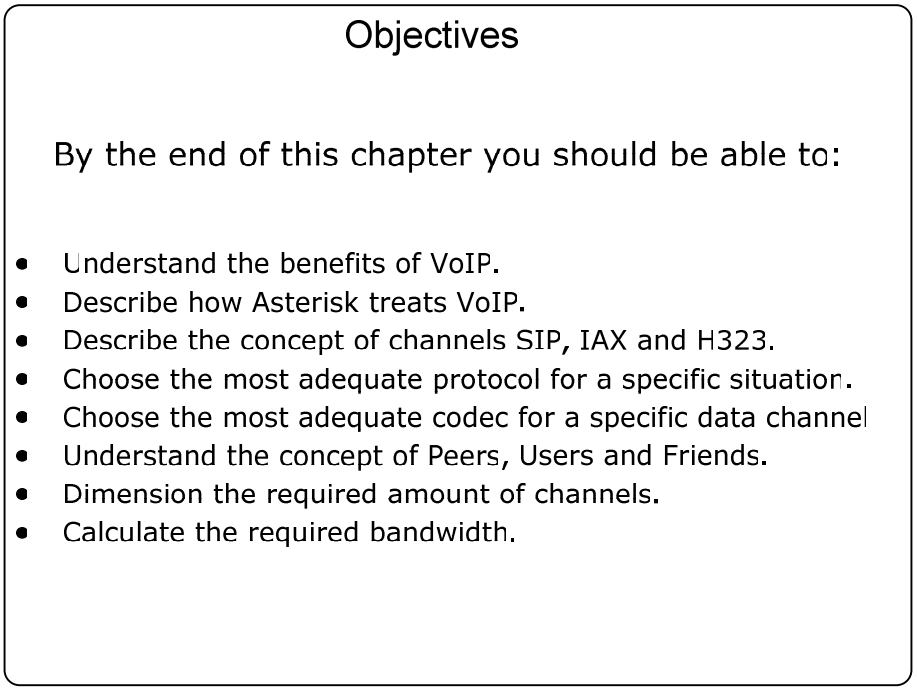


Figure 5.1 Objectives

5.2 INTRODUCTION

Voice over IP is growing quickly in the telephony market. The convergence paradigm is changing the way we communicate, reducing costs and enhancing the way we trade information. Voice is just the beginning of the full multimedia communication era including voice, video, and presence. Fiber optics will be one of the best substitutes for gasoline. In the future, we won't transport people to the work, we will transport work to people, because it is cleaner, faster, and cheaper. VoIP is just the beginning of this revolution. Right now, the VoIP concept is being extended to IMS (Internet Multimedia Systems), and you will hear much more about IMS in the near future.

5.3 VoIP BENEFITS

5.3.1 Convergence

The main benefit of VoIP is the combination of data and voice networks to reduce costs (convergence). However, analyzing just voice minute costs may not be enough to justify the adoption of VoIP. Minute costs sold by telcos are quickly becoming cheaper and is important to consider this before adopting VoIP.

5.3.2 Infrastructure costs

On the other hand, the use of a single network infrastructure reduces the costs associated with additions, removals, and changes. The pervasive IP brings VoIP technology to several new devices like cell phones, PDAs, embedded systems, and laptops.

5.3.3 Open Standards

Finally, the open standards upon which VoIP is built are bringing freedom to choose from different vendors and this single benefit makes the costumer a king instead of a subordinate of telcos and pbx manufacturers.

5.3.4 Computer Telephony Integration

Telephony is a lot older than computing. Telephony PBXs are circuit switch based. Sometimes a computer makes a supervision of the switch. With VoIP, telephony is from the ground up created based in computer standards. This makes the use of Computer Telephony applications a lot cheaper and easier than in the old model. You can quickly create a long list of telephony applications based on Asterisk. You can develop IVRs, ACDs, CTI, Dialers, screen popups, and other applications in a fraction of time required for traditional PBXs.

5.4 ASTERISK VoIP ARCHITECTURE

Asterisk's architecture is shown below. Asterisk treats all VoIP protocols as channels. You can use any codec or any protocol. The most important concept is that Asterisk bridges any channel one to another. Then, you can translate signaling protocols like H.323, SIP and IAX, one to another and even with different codecs. As an example, you can translate a call from a SIP phone in the local area network using the G.711 codec to a H323 connection to you VoIP provider using the G.729 codec.

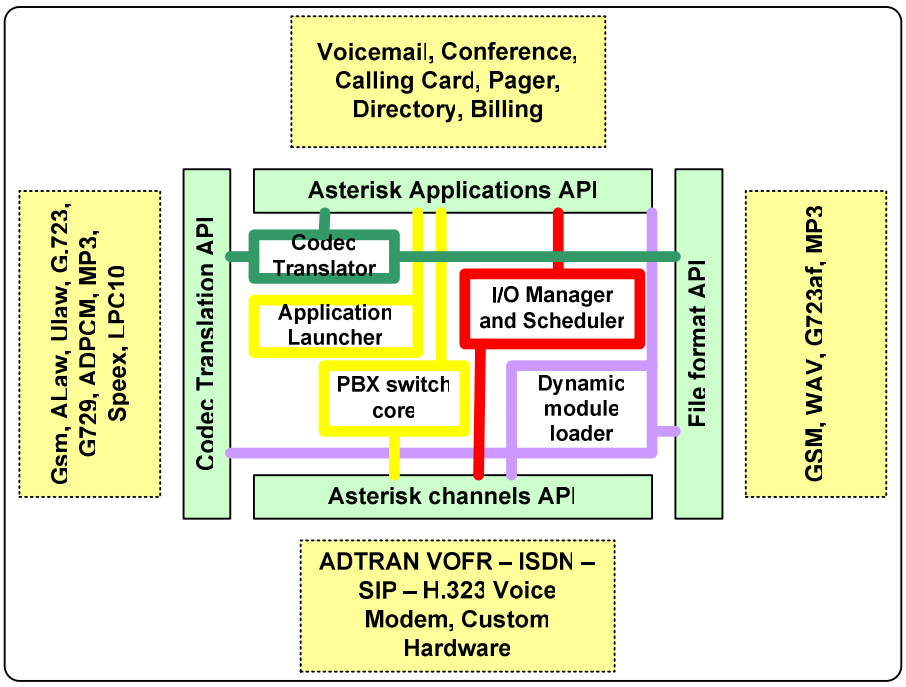


Figure 5-2 Asterisk architecture

In the next chapters, we will explain details of the SIP and IAX architecture. H.323 is not part of Asterisk, but it is available as an add-on. We will not cover H.323 in this book.

5.5 VoIP protocols and the OSI (ISO Open Systems Interconnect) model

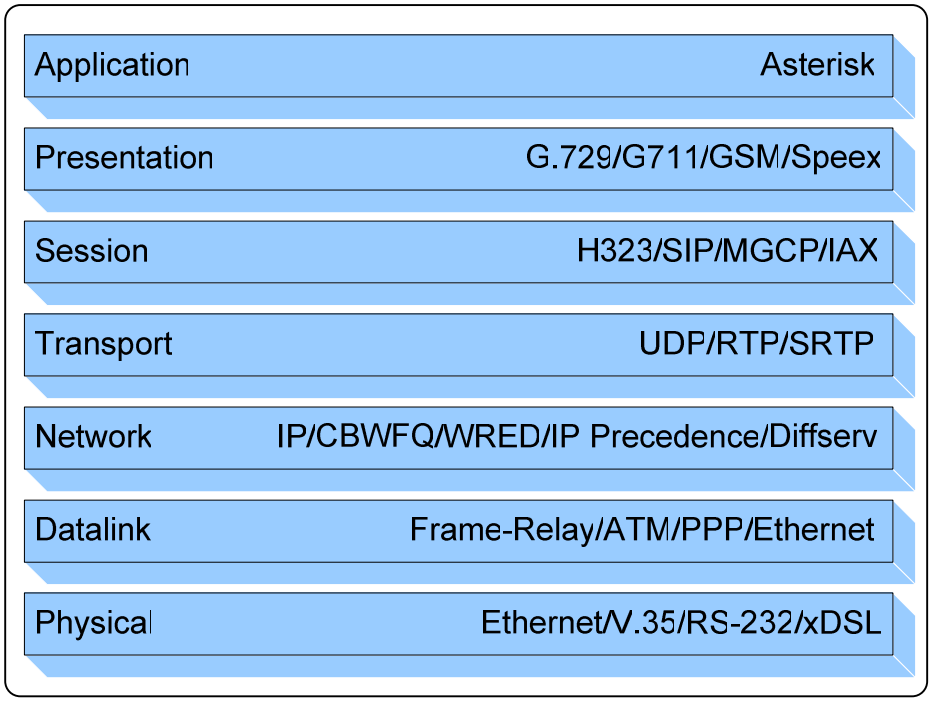


Figure 5.3 VoIP in the OSI model.

As you can see above, VoIP corresponds to a set of different protocols working together. Different OSI layers are present in VoIP communication. The figure above will help you to understand the role of each protocol and their relationship to one another.

is the first four layers represent just a data network like the Internet you have in your business or home. You can use some QoS protocols like "diffserv" or "cbwfq" to prioritize voice packets and enhance voice quality. Most VoIP protocols use RTP (real time protocol) as the transport protocol of choice.

In the session layer, protocols are responsible for setting up and closing the calls. H.323 is one of the oldest and mature protocols in this area. SIP in now pervasive in the VoIP provider market, putting aside the H.323. Signaling protocols use TCP or UDP to transport the packets.

In the presentation layer, we have the codecs transforming the multimedia stream from one format to another with different characteristics.

Example:

SIP:

SIP uses UDP or TCP in port 5060 to transport signaling. RTP transports the audio stream using ports 1000 to 2000 in Asterisk (as defined in rtp.conf). A call can be, for example coded in g.711. A softphone in the application layer will use the lower layers to communicate.

H.323 uses TCP in ports 1720 and 1719 to transport signaling. RTP transports audio usually in UDP ports 16383 to 32768. A call can be coded in g.729 for example.

5.5 How to choose a protocol

5.5.1 SIP - Session Initiated Protocol

SIP is an IETF (Internet Engineering Task Force) open standard, largely defined in RFC 3261. Most modern VoIP providers use SIP and it is becoming the most popular VoIP standard. The strength of SIP is to be an IETF based standard. SIP is light, if compared to older H.323. The main weakness of SIP is NAT traversal, a challenge to most SIP VoIP providers. IETF did not create SIP with billing in mind, but for open communications between peers. Billing is usually a concern for VoIP providers.

5.5.2 IAX – Inter Asterisk eXchange

IAX is an open protocol defined by Digium and it is currently a draft. You can download it from www.ietf.org/internet-drafts/drafts-guy-iax00.txt. IAX is an all-in one protocol since it transports signaling and media through the same UDP port (4569). Mark Spencer developed IAX as a binary protocol for reduced bandwidth. The main strengths of IAX are reduced bandwidth usage (it does not use RTP) while at the same time being very easy for NAT and firewall traversal since it uses only one UDP port (4569). If a traditional PBX manufacturer were to have created IAX it would probably have marketed the protocol as the "best thing since ice cream"; in some situations IAX in trunk mode can reduce voice bandwidth use by one third. This protocol was in its version 2 when I wrote this book.

5.5.3 MGCP – Media Gateway Control Protocol

MGCP is a protocol used in conjunction with H.323, SIP and IAX. Its greatest advantage is scalability. It is configured in the call agent instead of the gateways. This simplifies the configuration process and permits centralized management. The asterisk implementation is not complete and it seems that not many people use it.

5.5.4 H.323

H.323 is largely being used in VoIP. It is one of the first VoIP protocols and is essential for connecting older VoIP infrastructures based in gateways. H.323 is still the standard in the gateway market, although the market is slowly migrating to SIP. H.323's strengths includes large market adoption and maturity. H.323's weaknesses are related to the complexity of implementation and standard bodies associated costs.

5.5.5 Protocol comparison table

Protocol	Standard body	It is used for:
IAX2	IETF draft	Asterisk trunks IAX2 phones Connection to IAX service providers
SIP	IETF standard	SIP Phones Connection to SIP service providers
MGCP	IETF/ITU standard	MGCP Phones Currently does not support connecting to a MGCP gateway or service provider
H.323	ITU standard	H.323 Phones H.323 gateways Currently does not support being a gatekeeper, but can connect to an external gatekeeper.
SCCP	Cisco Proprietary	Cisco Phones

5.6 Peers, Users and Friends

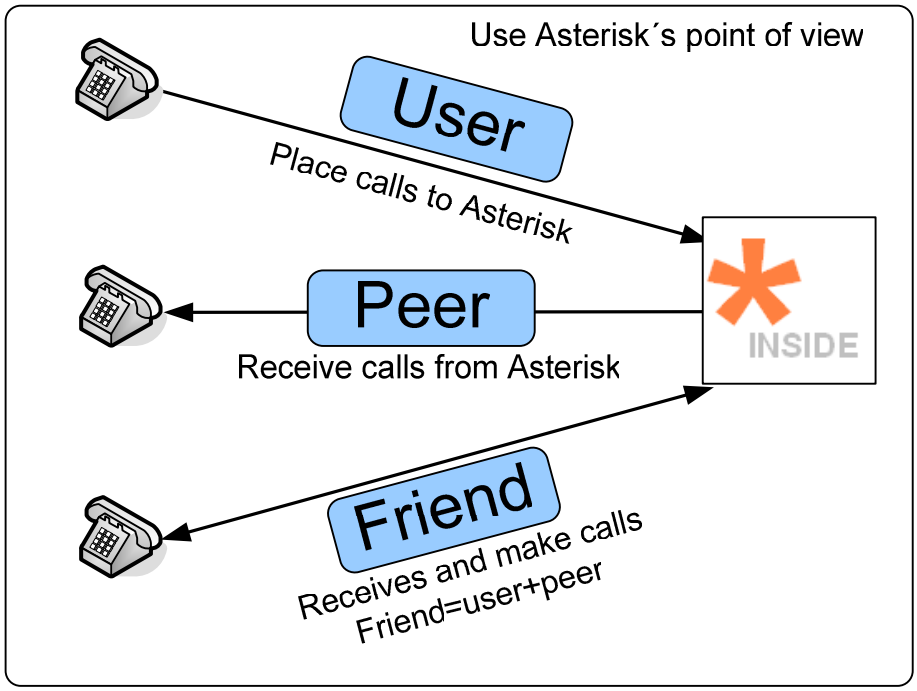


Figure 5-4 Users, Peers and Friends

There are three kinds of SIP and IAX clients. The first one is "user". Users can make calls to an Asterisk server, but they cannot connect to receive calls from this server. The second one is a "peer". You can make calls to a peer, but you will not receive calls from them. Usually a server or a device will require both concepts at the same time. A "friend" is a shortcut to a "user" +

"peer". A phone would probably fall into this category, since it needs to make and receive calls.

5.7 CODECS AND CODEC TRANSLATION

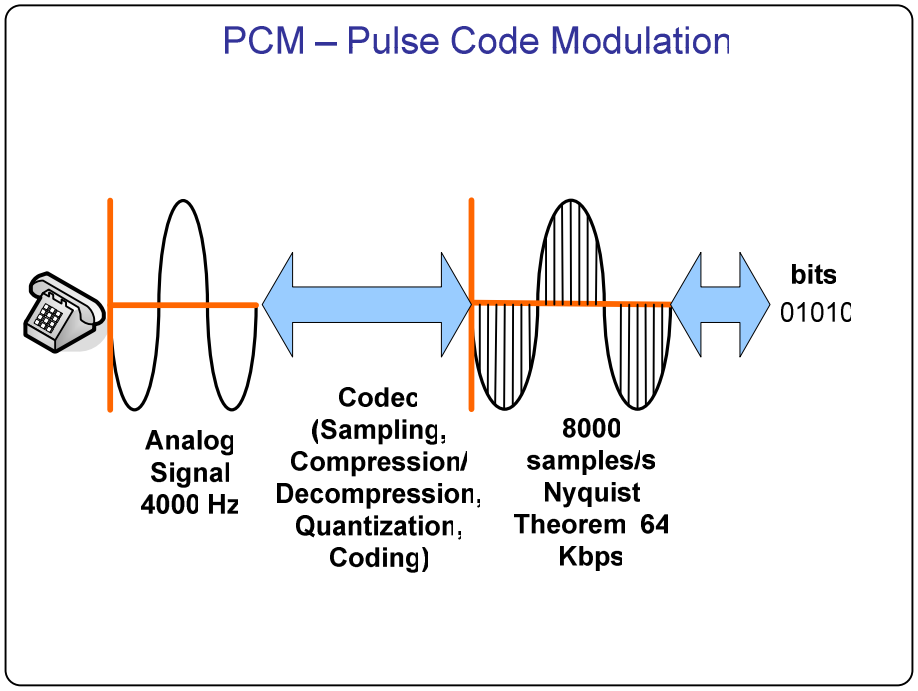


Figure 5-5 Digital voice processing

You will use a codec to convert voice from analog to digital signal. Codecs differ from one another in aspects like sound quality, compression rate, bandwidth, and computing requirements. Services, phones, and gateways usually support several of them. G.729, a very popular codec, requires licensing.

Asterisk supports the following codecs:

- **GSM:** 13 Kbps
- **iLBC:** 13.3 Kbps
- **ITU G.711:** 64 Kbps
- **ITU G.723.1**: 5.3/6.3 Kbps
- **ITU G.726:** 16/24/32/40 Kbps
- **ITU G.729:** 8 Kbps

- **Speex** - 2.15 to 44.2 Kbps
- **LPC10** - 2.5 Kbps

Asterisk permits translation between codecs. There are cases where this is not possible, such as the case of g.723, which is supported only in pass-thru mode. Translating from one codec to another consumes many resources from the CPU. Thus, avoid this altogether whenever possible.

5.8 How to choose a Codec

Codec selection depends in several aspects like:

- Sound quality
- Licensing costs
- CPU processing consumption
- Bandwidth requirements
- Packet loss concealment
- Availability for Asterisk and phone devices

A table comparing the most popular codecs is shown below: The quality of these codecs is considered "toll", or in other words, similar to PSTN.

Codec	g.711	g.729A (20 ms)	iLBC (30 ms)	GSM 06.10 RTE/LTP
bandwidth (Kbps)	64	8	13.33	13
Costs	Free	US$10.00 (per channel)	Free	Free
Resistance to Frame Erasure[1]	No mechanism	3%	5%	3%
Complexity MIPS [2]	~0.35	~13	~18	~5

[1] Resistance to packet loss refers to the rate when MOS is next to 0.5 worst from peak quality for the specific codec.
[2] Complexity refers to quantities in millions of instructions per second spent to code and decode the codec using a reference design in a Texas Instruments DSP (TMS320C54x). There is a direct relationship between the processor frequency and MIPS, but it is not possible to draw a precise relationship between hardware platforms that are so different. Use this table just for comparison.

5.9 OVERHEAD CAUSED BY PROTOCOL HEADERS

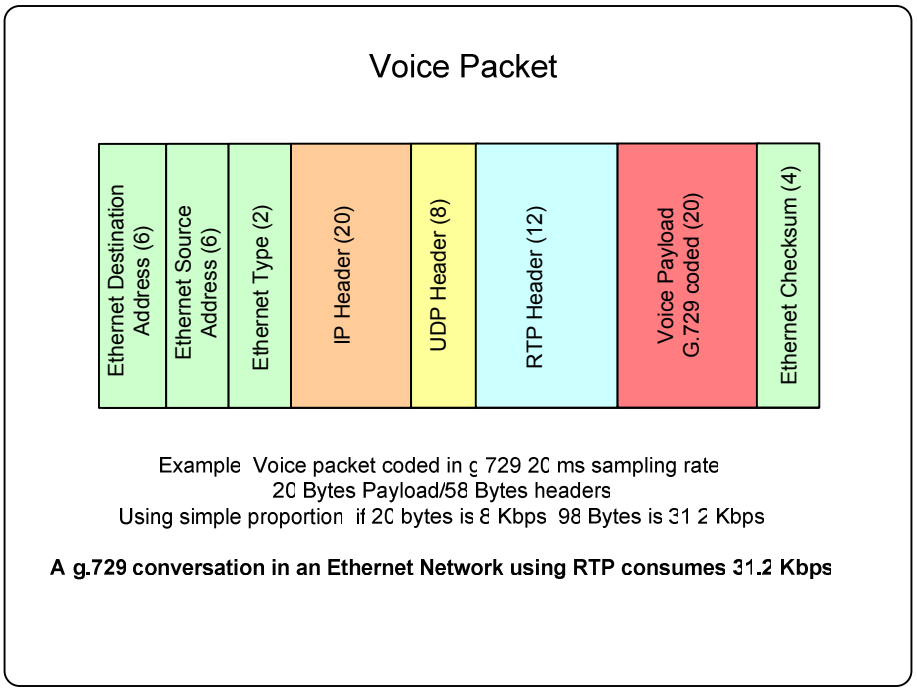


Figure 5.6 overhead caused by network headers.

In spite of codecs making little use of bandwidth, we have to take into consideration the overhead caused by protocol headers like Ethernet, IP, UDP and RTP. Observing this, we could say that bandwidth depends on headers used. If we are in an Ethernet network, the bandwidth requirement is higher then in a PPP Network because the PPP header is shorter than the Ethernet one. Let us look through some examples:

Codec g.711 (64 Kbps)

- **Ethernet** (Ethernet+IP+UDP+RTP+G.711) = 95.2 Kbps
- **PPP** (PPP+IP+UDP+RTP+G.711) = 82.4 Kbps
- **Frame-Relay** (FR+IP+UDP+RTP+G.711) = 82.8 Kbps

Codec G.729 (8 Kbps)

- **Ethernet** (Ethernet+IP+UDP+RTP+G.729) = 31.2 Kbps
- **PPP** (PPP+IP+UDP+RTP+G.729) = 26.4 Kbps
- **Frame-Relay** (FR+IP+UDP+RTP+G.729) = 26.8 Kbps

You can easily calculate other bandwidth requirements using the following website's calculator.

http://www.packetizer.com/voip/diagnostics/bandcalc.html

5.10 TRAFFIC ENGINEERING

A main issue in the design of voice over IP networks is dimensioning the number of lines and required bandwidth to a specific destination like a remote office or a service provider. It is also important to dimension the number of Asterisk's simultaneous calls (main parameter for Asterisk's dimensioning).

5.10.1 Simplifications

The first and widely used simplification is to estimate the number of calls by user type. Example:

Business PBXs (one simultaneous call for each five extensions)
Residential users (one simultaneous call for each sixteen users)

Example #1

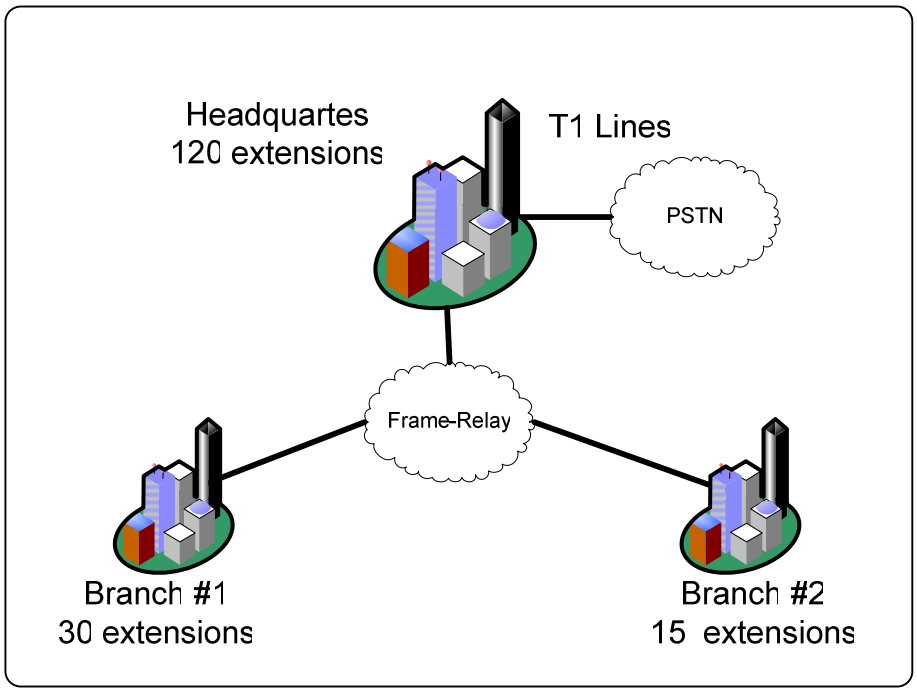


Figure 5.7 VoIP design example

The company has a headquarter with 120 extensions and two branches, the first one with 30 extensions and the second one with 15 extensions. Our objective is to dimension the number of E1 trunks in the headquarters and the bandwidth required for the Frame-Relay network.

1.a Number of T1 lines
Total number of extensions using T1 lines: 120+30+15=165 lines

Using one trunk for each five extensions for business use

Total number lines = 33 or approximately 2xT1 lines

1.b Bandwidth requirements

We choose the g.729 codec because of bandwidth requirements, sound quality and medium CPU consumption.

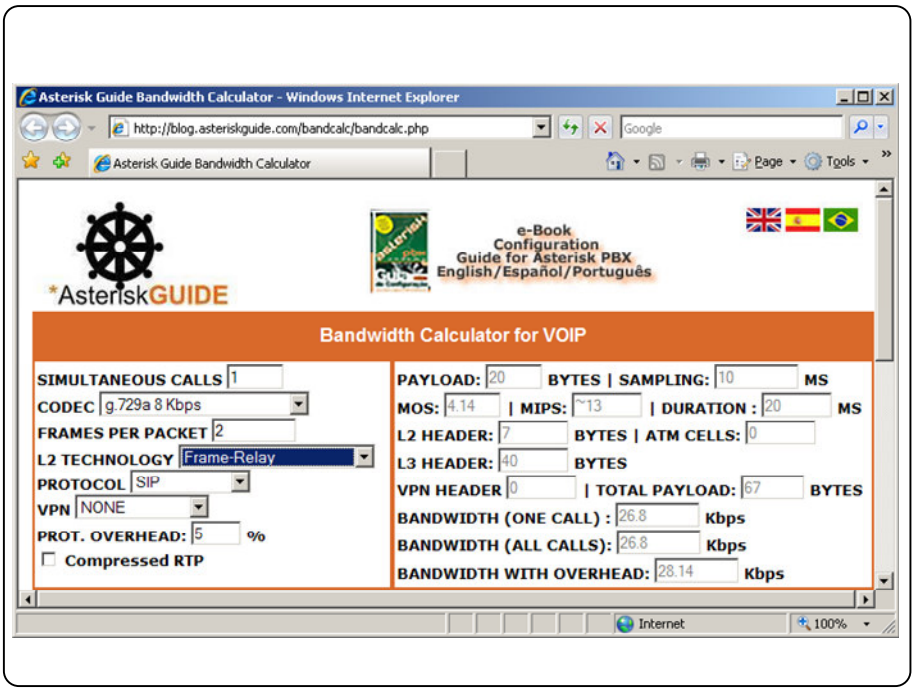


Figure 5.8 VoIP bandwidth requirement (g.729, Frame-relay)

With one trunk for each five extensions:

Required bandwidth for branch #1 (Frame-relay): 26.8*6=160.8 Kbps
Required bandwidth for branch #2 (Frame-relay): 26.8*3= 80.4 Kbps

5.10.2 Erlang B method

1.a Number of VoIP simultaneous calls

Sometimes, simplifications are not the best methods to use. When you have previous data, you can adopt a more scientific approach. Agner Karup Erlang (Copenhagen Telephone Company, 1909) has developed a formula to calculate lines in a trunk group between two cities. We will use his formula.

Erlang is a traffic measurement unit commonly used in telecom. It is used to describe a traffic volume of one hour.

Example: 20 calls in an hour with 5 minutes of conversation average.

You calculate the number of Erlangs as below:

Traffic minutes in the hour: 20 x 5 = 100 minutes
Hour of traffic inside one hour: 100/60 = 1.66 Erlangs

You can get these measures from a call logger and use it to design your network to calculate the number of lines required. Once the number of lines is known, it is possible to calculate the bandwidth requirements.

Erlang B is the most used method to calculate the number of lines in a trunk group. It assumes that calls arrive randomly (Poisson distribution) and blocked calls are immediately cleared. This method requires that you know the Busy Hour Traffic (BHT). You can obtain the BHT from a call logger or by a simplification:

BHT=17% of the call minutes of one day.

Another important variable is GoS (Grade of Service). GoS defines the probability of blocking calls by line shortage. You can arbitrate this parameter. It is usually 0.05 (5% calls lost) or 0.01 (1% calls lost).

Example#1:

Using the same example from 5.10.1 let us give you some data about traffic patterns. From the call logger we discovered these data.

Data from call logger (Call minutes and BHT):

- Headquarters to Branch#1 = 2000 minutes, BHT = 300 minutes
- Headquarters to Branch#2 = 1000 minutes, BHT = 170 minutes
- Branch#1 to Branch#2 = 0, BHT=0

Let's arbitrate GoS=0.01

Headquarters to branch#1 - BHT=300 minutes/60 = 5 Erlangs
Headquarters to branch#2 – BHT=170 minutes/60 = 2.83 Erlangs

Using an Erlang Calculator (www.erlang.com)

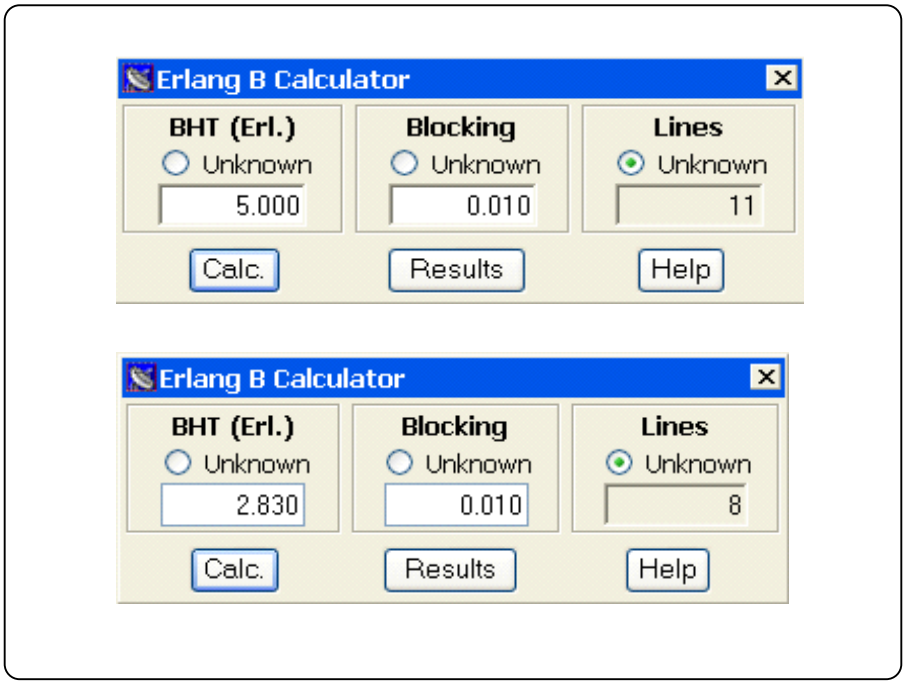


Figure 5.9 Using ErlangB calculator

For the Headquarters to branch#1, 11 lines are required.
For the Headquarters to branch#2, 8 lines are required
1.b Bandwidth Required

We are using a WAN where packet loss is rare. We will choose the g.729 codec because of its good sound quality and data compression (8 Kbps).

Selected codec: g.729
Datalink layer: Frame-Relay

Estimated voice bandwidth for Branch#1: 26.8x11 =294.8 Kbps
Estimated voice bandwidth for Branch#2: 26.8x8 =214.40 Kbps

5.11 Reducing the bandwidth required for VoIP

There are three methods to reduce the bandwidth required for VoIP calls:

- RTP header compression
- IAX Trunked
- VoIP payload

5.11.1 RTP Header Compression

In Frame-Relay and PPP networks, you can use RTP header compression. RTP header compression was defined in RFC 2508. It is an IETF standard

available in several routers. Be cautious, however, as some routers require a different feature set for this resource to be available.

The impact of using RTP header compression is fabulous. It reduces the bandwidth required in our example from 26.8 Kbps per voice conversation to 11.2 Kbps: a 58.2% reduction!

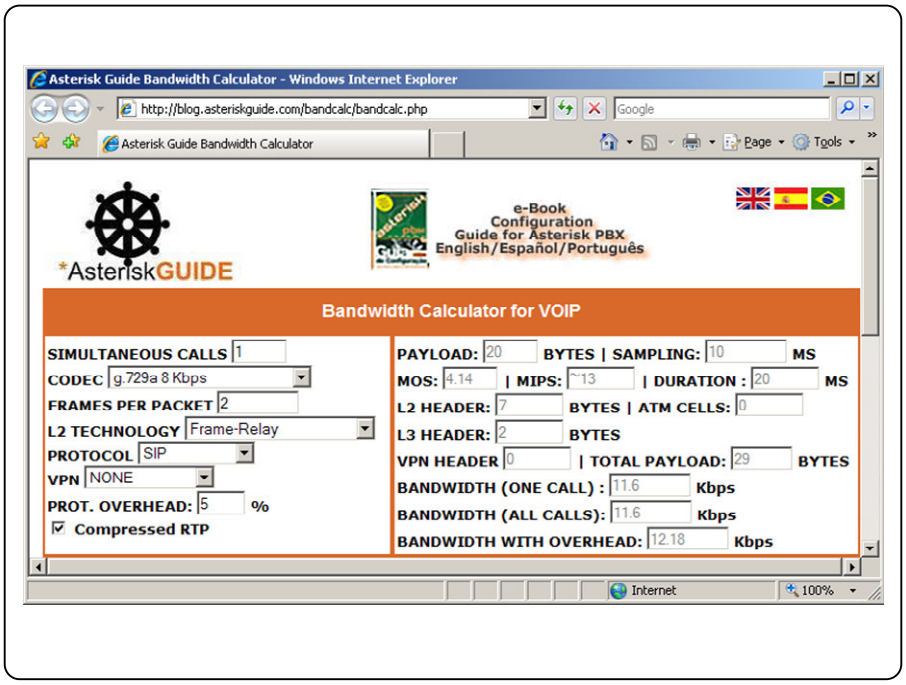


Figure 5.10 VoIP bandwidth using cRTP

5.11.2 IAX2 trunk mode

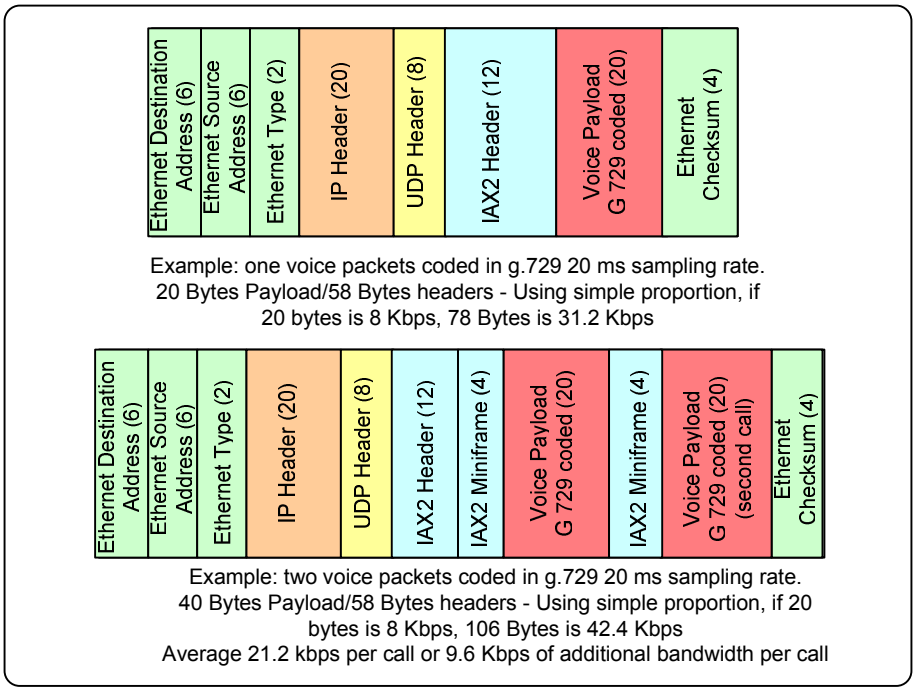


Figure 5.11 IAX2 trunk mode bandwidth reduction

If you are connecting two Asterisk servers, you can use the IAX2 protocol in the trunk mode. This revolutionary technology does not need any special routers and it can be applied to any kind of data link. IAX2 trunk mode reuses the same headers from the second call and over. Using g.729 and a PPP link, the first call will consume 30 Kbps of bandwidth, but the second call will use the same header as the first and reduce the necessary bandwidth for the additional call to 9.6 Kbps. A study from John Todd from Loligo can be seen in http://www.voip-info.org/wiki-Asterisk+bandwidth+iax2.

We can calculate the required bandwidth in trunk mode as follows:

Branch#1 (11 calls)
Bandwidth = 31.2 + (11-1)* 9.6 Kbps = 127.2 Kbps.

Branch#2 (8 calls)
Bandwidth = 31.2 + (8-1)* 9.6 Kbps = 98.4 Kbps.

The first call uses 31.2 Kbps and the next 9.6 and so on.

5.11.3 Increasing the Voice Payload

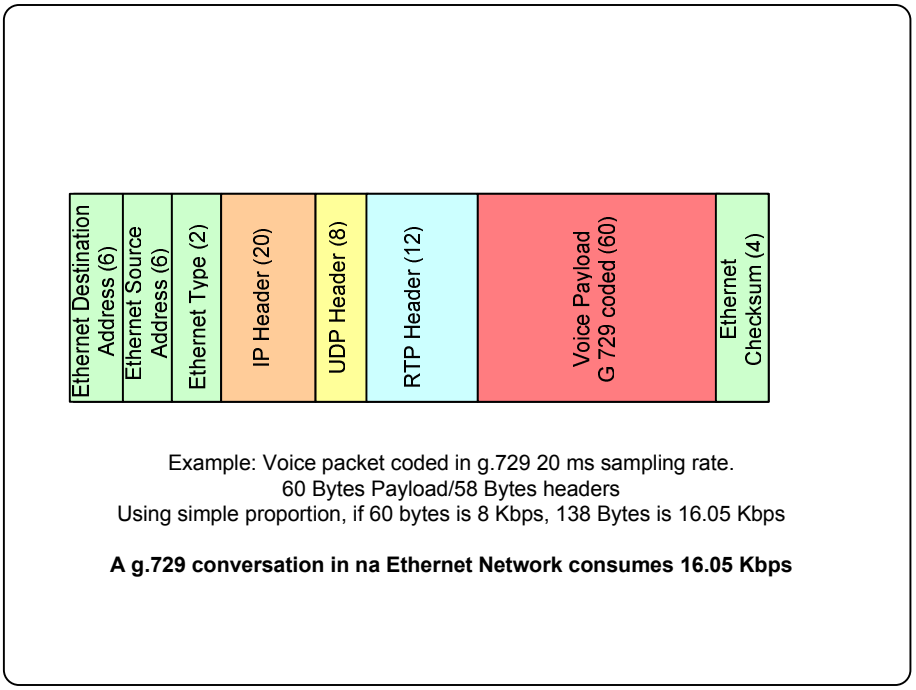


Figure 5.11 increasing the voice payload to reduce bandwidth

It was a commonly used scheme in gateways over the Internet. Using a bigger payload, you sacrifice latency in favor of reduced bandwidth. Now in the Asterisk version 1.4 you can change the RTP Packetization appending the frame size to the codec in the allow instruction.

Example:

```
allow=ulaw:30
```

The permitted values are:

Name	Min	Max	Default	Increment
g723	30	300	30	30
gsm	20	300	20	20
ulaw	10	150	20	10
alaw	10	150	20	10
g726	10	300	20	10
ADPCM	10	300	20	10
SLIN	10	70	20	10
lpc10	20	20	20	20
g729	10	230	20	10
speex	10	60	20	10
ilbc	30	30	30	30

5.12 SUMMARY

In this chapter, you have learned that Asterisk treats voice over IP using channels. It supports SIP, IAX, H.323, MGCP and Skinny protocols. You compared and learned how to choose a signaling protocol and a codec for VoIP channels. IAX2 is more bandwidth efficient and can traverse NAT easily. SIP is the most supported protocol by third-party phone and gateways vendors. The H.323 protocol is the oldest one and it should be used to connect to legacy VoIP infrastructure. In section 5.11 we have learned how to design and dimension a VoIP network.

5.13 QUESTIONS

1. Please, list at least four benefits of voice over IP.

_____

_____

_____

2. Convergence is the integrations of voice, data and video in a single network and their main benefit is the cost reduction in the implementation and maintenance of separate networks.

a) False
b) True

3. Asterisk cannot use simultaneously resources from PSTN and VoIP because the codecs are not compatible.

a) False
b) True

4. Asterisk is a SIP proxy with integration to other protocols

a) False
b) True

5. Using the OSI reference model, SIP, H.323 and IAX2 are in the _____ layer.

a) Presentation
b) Application

c) Physical
d) Session
e) Datalink

6. SIP is the most adopted protocol for IP phones and is an open standard ratified by IETF.

a) False
b) True

7. H.323 is an inexpressive protocol with very few applications, abandoned by the market, which is moving to SIP.

a) False
b) True

8. IAX is a proprietary Digium protocol. In spite of its small adoption by phone vendors, IAX is excellent when you need:

a) Reduce bandwidth usage
b) Video media format
c) NAT traversal
d) Protocols standardized by IETF or ITU.

9. "Users" can receive calls from Asterisk.

a) False
b) True

10. Mark true affirmations concerning codecs.

a) G711 is the equivalent to PCM it uses 64 Kbps of bandwidth.
b) G.729 is free for commercial use and it uses 8 Kbps of bandwidth
c) GSM is growing because it uses approximately 13 Kbps and does not need a license.
d) G711 u-law is common in US while a-law is common in Europe and Latin America.
e) G.729 is light and uses very few CPU resources in their coding/decoding process.

Chapter 6

The IAX Protocol

In this chapter, we will learn a few things about the IAX protocol, its strengths and weaknesses. Details like trunk mode and the interconnection of two Asterisk servers will also be covered.

6.1 OBJECTIVES

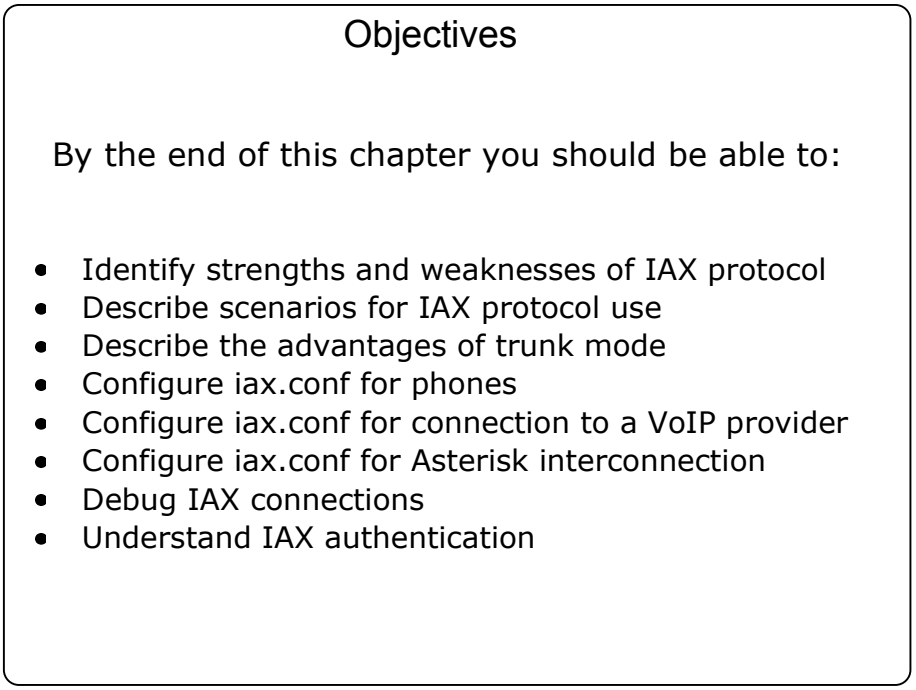


Figure 6.1 Objectives

6.2 INTRODUCTION

All references in this document correspond to IAX version 2. The Inter-Asterisk eXchange Protocol provides media transport and signaling for voice and video. It was created primarily for voice, but it can also accommodate video and other multimedia streams. IAX was inspired in other VoIP protocols like SIP and MGCP. IAX, instead of using two separate protocols for signaling and media, unified them to make an unique protocol. IAX does not use RTP for media transport. Instead, it embeds the media within itself. The main objectives of IAX design were:

- Reduce the bandwidth required for media transport and signaling
- Provide NAT transparency
- Be able to transmit the dial plan information
- Support efficient use of paging and intercom.

6.3 How it works?

IAX is a peer-to-peer signaling and media protocol. The signaling protocol is similar to SIP, but it does not use RTP.

The IAX basic approach is to multiplex the multimedia streams over a single UDP connection between two hosts. The greatest benefit of this approach is that it becomes simple to traverse connections over NAT, generally present in ADSL devices (this is an important feature for VoIP providers). IAX uses a single port, UDP 4569 by default, and then uses a 15 bit call number to multiplex all streams.

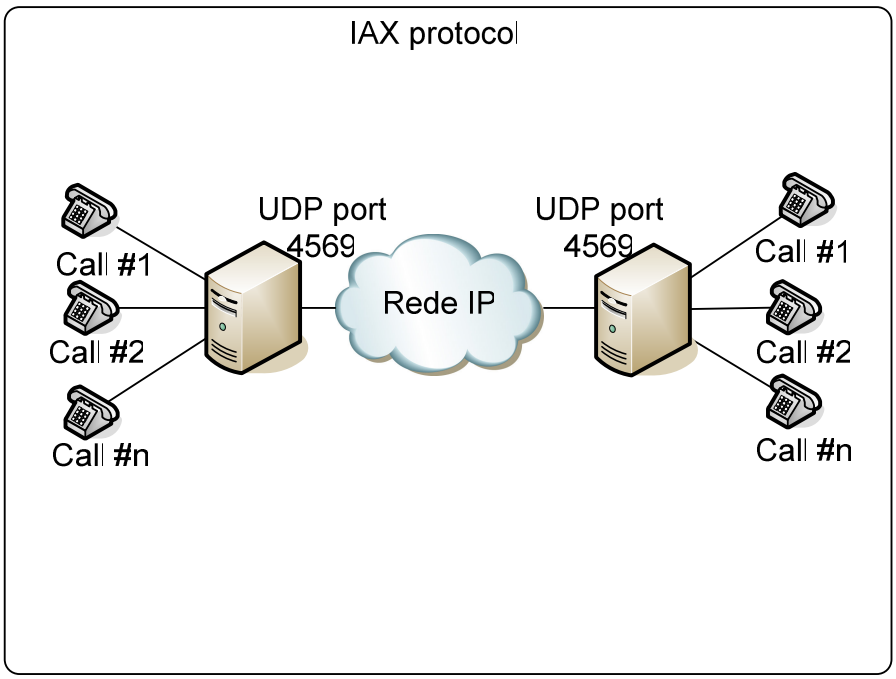


Figure 6.2: Multiple calls over a single UDP connection

The IAX protocol uses registration and authentication processes similar to the SIP protocol. A description of the protocol can be found in http://www.ietf.org/internet-drafts/draft-guy-iax-01.txt

6.4 BANDWIDTH USAGE

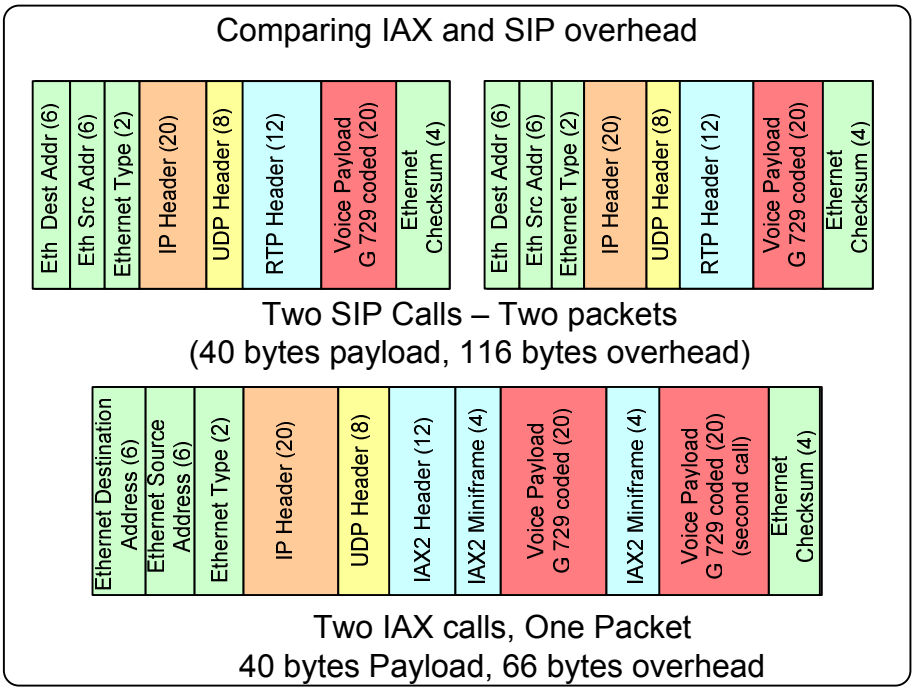


Figure 6.3 – Comparing IAX and SIP headers overhead

The bandwidth used in a VoIP network is affected by several factors. Codecs and headers are the most important factors. The IAX protocol has an astonishing feature called trunk mode, whereby it multiplexes several calls using single header. A bandwidth study was published by John Todd from loligo inc. and it can be found in http://www.voip-info.org/wiki-Asterisk+bandwidth+iax2. It is reproduced below for convenience.

```
Testing results:

G.711 (ulaw)
 one call:   164333.75 bps/94.26 pps   ( 82.1 kbps)
 two calls:  296171.60 bps/101.46 pps  (148.0 kbps)

Thus:
For every additional call: 131837 bps (65.9 kbps)
Est. IP/IAX2 overhead (1 call): 32495 bps (16.0 kbps)
Raw number of calls per megabit: 15

ILBC: see note below
 one call:   56134.91 bps/67.45 pps    (28.0 kbps)
 two calls:  98679.11 bps/102.41 pps   (49.3 kbps)

Thus:
For every additional call: 42544 bps (21.2 kbps)
Est. IP/IAX2 overhead (1 call): 13590 bps ( 6.7 kbps)
Raw number of calls per megabit: 47
```

```
G.729
 one call:   60124.33 bps/101.26 pps   (30.0 kbps)
 two calls:  79496.23 bps/102.85 pps   (39.7 kbps)

Thus:
For every additional call: 19372 bps ( 9.6 kbps)
Est. IP/IAX2 overhead (1 call): 40752 bps (20.3 kbps)
Raw number of calls per megabit: 103


GSM
 one call:   70958.16 bps/102.13 pps   (35.4 kbps)
 two calls: 100455.23 bps/102.63 pps   (50.2 kbps)

Thus:
For every additional call: 29497 bps (14.7 kbps)
Est. IP/IAX2 overhead (1 call): 41461 bps (20.7 kbps)
Raw number of calls per megabit: 68

LPC10
 one call:   43855.44 bps/89.94  pps   (21.9 kbps)
 two calls:  56059.18 bps/100.81 pps   (28.0 kbps)

 Thus:
  For every additional call:       12203 bps ( 6.1 kbps)
  Est. IP/IAX2 overhead (1 call):  31561 bps (15.8 kbps)
  Raw number of calls per megabit: 164

(SPEEX):
 one call:   74817.18 bps/101.06 pps   (37.4 kbps)
 two calls: 109692.68 bps/102.18 pps   (54.8 kbps)

Thus:
For every additional call: 34875 bps (17.4 kbps)
Est. IP/IAX2 overhead (1 call): 39941 bps (19.9 kbps)
Raw number of calls per megabit: 57
```

Conclusions:

- Bandwidth usage is influenced by factors such as simultaneous calls, layer 2 protocol, full, or half duplex network.

- A measurement of 75 Kbps is reported as 37.5 Kbps because the network is full duplex, but the stat is taken in the server interface driver that accounts all packets received and transmitted.

- The data on ILBC protocol is incorrect because ILBC sends 30 ms frames and the trunk mode uses 20 ms timing. You have to change trunk timing to get better ILBC results.

6.5 CHANNEL NAMING

6.5.1 The format of an IAX channel name used for outbound channels is:

```
IAX/[<user>[:<secret>]@]<peer>[:<portno>][/<exten>[@<context>][/<options>]
```

<user>	UserID on remote peer, or name of client configured in iax.conf
<secret>	The password. Alternatively it can be the filename for an RSA key without the trailing extension (.key or .pub), and enclosed in square brackets
<peer>	Name of server to connect to
<portno>	Port number for connection
<exten>	Extension in the remote Asterisk server
<context>	Context in the remote Asterisk server
<options>	The only option available is 'a' meaning 'request autoanswer'

6.5.2 Outbound channels example:

IAX2/8590:secret@myserver/8590@default	Call the 8590 extension in myserver. It uses 8590:secret as the name/password pair
IAX2/iaxphone	Call "iaxphone"
IAX2/judy:[judyrsa]@somewhere.com	Call somewhere.com using judy as the username and a RSA key for authentication

6.5.3 The format of an incoming IAX channel is:

```
IAX2/[<username>@]<host>]-<callno>
```

<username>	Username if known
<host>	Host connecting
<callno>	Local call number

6.5.4 Incoming channel example:

IAX2[flavio@8.8.30.34]/10	It is a call number 10 from IP address 8.8.30.34 using flavio as user.

IAX2[8.8.30.50]/11	It is a call number 11 from IP address 8.8.30.50.

6.6 USING IAX

You can use IAX in several ways. In this section, we will show you how to configure IAX for several scenarios like:

- Connecting a soft-phone using IAX
- Connecting IAX to a VoIP provider using IAX
- Connecting two servers using IAX
- Connecting two servers using IAX in trunk mode
- Debugging an IAX connection
- Using RSA pair keys for authentication

6.6.1 Connecting a soft-phone using IAX

Asterisk supports IP phones based on IAX like snom and IAXy, as well as soft-phones like idefisk. The process for soft-phones, ATAs, and hard-phones is very similar. To configure an IAX device, you need to edit the iax.conf file in /etc/asterisk directory.

We will use as an example the IDEFISK software (www.asteriskguru.com); it is full-featured. and free

Step 1: Make a backup of the original iax.conf file using:

```
#cd /etc/asterisk
#mv iax.conf iax.conf.backup
```

Step 2: Start editing a new iax.conf file:

```
[general]
bindport=4569
bindaddr=8.8.1.4
bandwidth=high          ; Very important parameter, it changes the codecs
available
disallow=all
allow=ulaw
jitterbuffer=no
forcejitterbuffer=no
tos=lowdelay
autokill=yes

[guest]
type=user
context=guest
callerid="Guest IAX User"
```

```
; Trust Caller*ID Coming from iaxtel.com
;
[iaxtel]
type=user
context=default
auth=rsa
inkeys=iaxtel

;
; Trust Caller*ID Coming from iax.fwdnet.net
;
[iaxfwd]
type=user
context=default
auth=rsa
inkeys=freeworlddialup

;
; Trust callerid delivered over DUNDi/e164
;
;
;[dundi]
;type=user
;dbsecret=dundi/secret
;context=dundi-e164-local

[2003]
type=friend
context=default
secret=senha
host=dynamic
```

In the edition below, I have preserved default (non-commented) lines of the sample file. The following parameters were modified:

```
bandwidth=high
```

This line affects codec selection. Using the “high” setting allows the selection of a high bandwidth and a high quality codec such as g.711 defined by the “ulaw” keyword. If you keep the default parameter, you will not be able to choose “ulaw”. In this case, Asterisk will give you the message “no codec is available” for the configuration below.

```
disallow=all
allow=ulaw
```

In the commands described above, we have disabled all codecs and enabled just ulaw. In LANs, I am confident that you will prefer to use ulaw, because it is not processor-intensive and saves CPU cycles. Even using more bandwidth, this codec is preferable because in LANs you usually have a hundred megabits Ethernet or even a Gigabit. A voice call using ulaw uses

100 kilobits per second of bandwidth from your network and it is a very light use for today's high-speed LANs. In WAN or Internet networks you will usually not enable "ulaw", trading some available CPU cycles of voice compression for better bandwidth use. The codecs GSM, g.729, and iLBC provide a good compression factor.

```
[2003]
type=friend
context=default
secret=senha
host=dynamic
```

In the above commands, we have defined a friend named [2003]. The context is the default (in the first labs we always use the default context to avoid confusion; this context will be fully explained in chapter 9). The line "host=dynamic" provides for a dynamic registration of the phone's IP address.

Step 3: Download and install idefisk software from the following URL:

http://www.asteriskguru.com/tools/idefisk/idefisk133_installer.exe

Note: URLs frequently change. Please resort to "googling" if you cannot find the file in this specific URL. You can choose other soft-phones for the lab as well.

Step 4:

Configure an Asterisk account by clicking the right button over the idefisk's task bar icon, or use ALT+A.

You should see a screen similar to the one below:

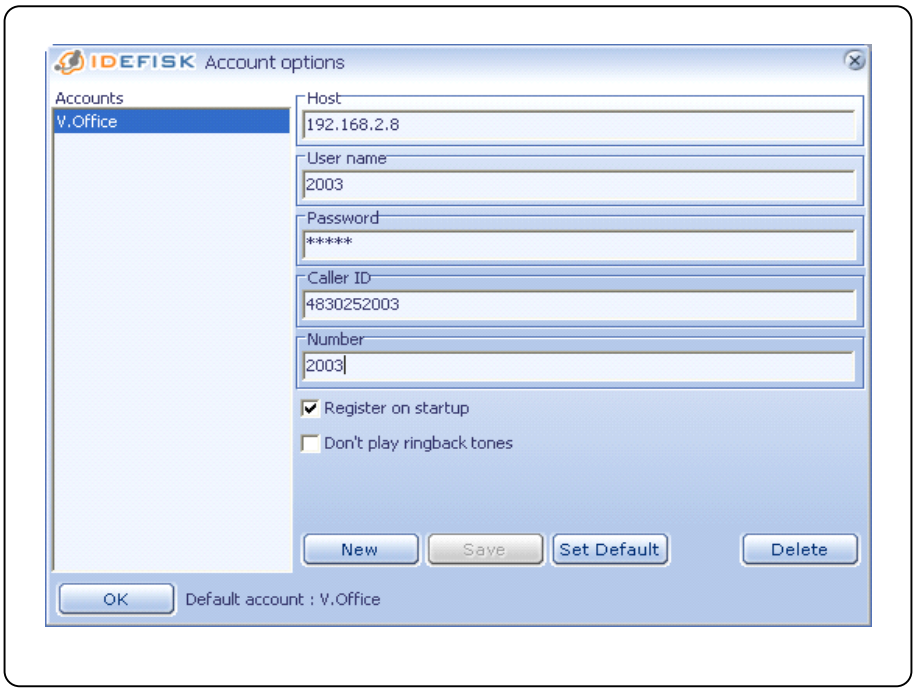


Figure 6.4 Idefisk account options

Step 5: Configure the extensions.conf file to test your IAX device.

```
[default]
exten=>2000,1,Dial(SIP/2000)
exten=>2001,1,Dial(SIP/2001)
exten=>2003,1,Dial(IAX2/2003)
```

Now, you can dial between the SIP phones created in chapter 3 and the IAX phone created in the lab.

6.6.2 Connecting to a VoIP provider using IAX

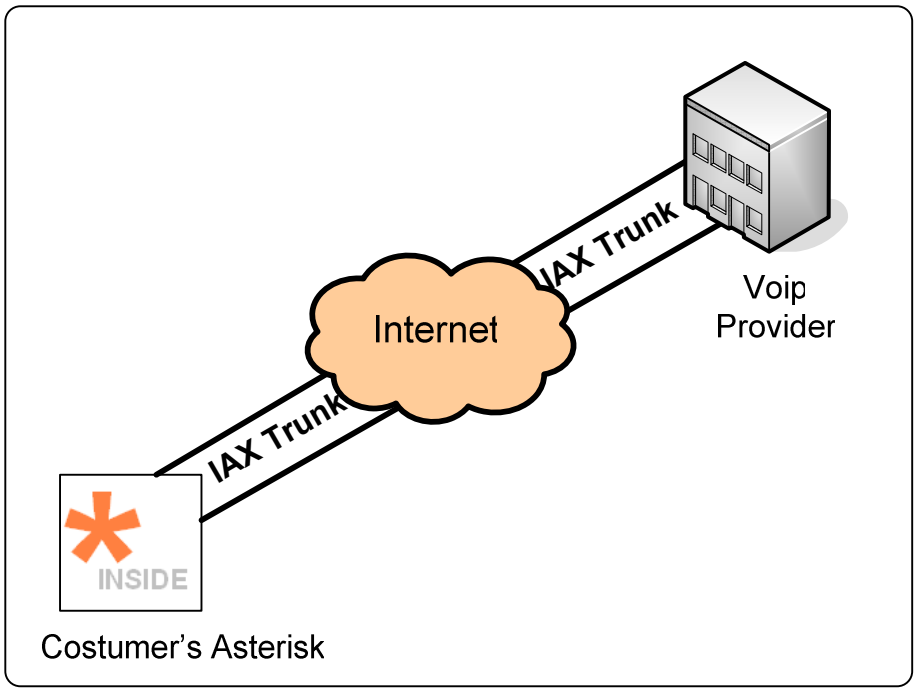


Figure 6.5 Connecting to a VoIP Provider

There are a few VoIP providers that support IAX. You can test a connection using FWD (www.freeworlddialup.com) or iaxtel (www.iaxtel.com). In my opinion, there is a lot of sense in using an IAX provider; IAX can save a lot of bandwidth, is easy to traverse NAT and can authenticate using RSA key pairs.

6.6.3 Connecting to freeworlddialup using IAX

Step 1: Open an account in your favorite provider. If you do not know any provider, use freeworlddialup for this example - it is free.

Step 2: In the services area, enable IAX for your account. Check with your provider for the procedures.

In freeworlddialup you can go to "extra features" and select IAX as shown below.

Figure 6-6 Enabling IAX in freeworlddialup

Notes: Screens and URLs change frequently, please double-check the procedures with your provider.

Step 3: Configure the iax.conf file to register your Asterisk with your provider. Add the following lines to the [general] section of the file.

```
[general]
register=>621538:password@iax2.fwdnet.net/2003
```

In the instructions described above, you registered with your provider using your account and password. In the moment you receive a call, it will be forwarded to the 2003 extension.

```
[621538]          ; Your account number
type=peer
secret=senha      ; Your password
host=iax2.fwdnet.net
```

In the instructions described above, we are created a peer that corresponds to the provider for dialing purposes.

```
[iaxfwd]
type=user
context=default
auth=rsa
inkeys=freeworlddialup
```

Remove the remark of [iaxfwd] section in the iax.conf file. This is necessary for RSA authentication. Using the public key provided by FWD allows you to be sure that the call is being received from the true provider. If anyone else

tries to use the same path, they cannot authenticate because they do not have the corresponding private key.

Step 4: Trying the connection.

To test the connection, let's use the Time service, number 612, from FWD. When you dial 612 in the provider, you will be answered by an IVR with the current time. To accomplish this, it is necessary to edit the extensions.conf file.

```
[default]
exten=>612,1,Dial(IAX2/621538:senha@iax2.fwdnet.net/612,20,r)
```

Go to the Asterisk CLI and issue a reload. To verify if Asterisk is registered with the provider, use the next command.

```
CLI>reload
CLI>iax2 show register
```

Now simply dial 612 in Asterisk.

6.6.4 Connecting two Asterisk servers through an IAX trunk.

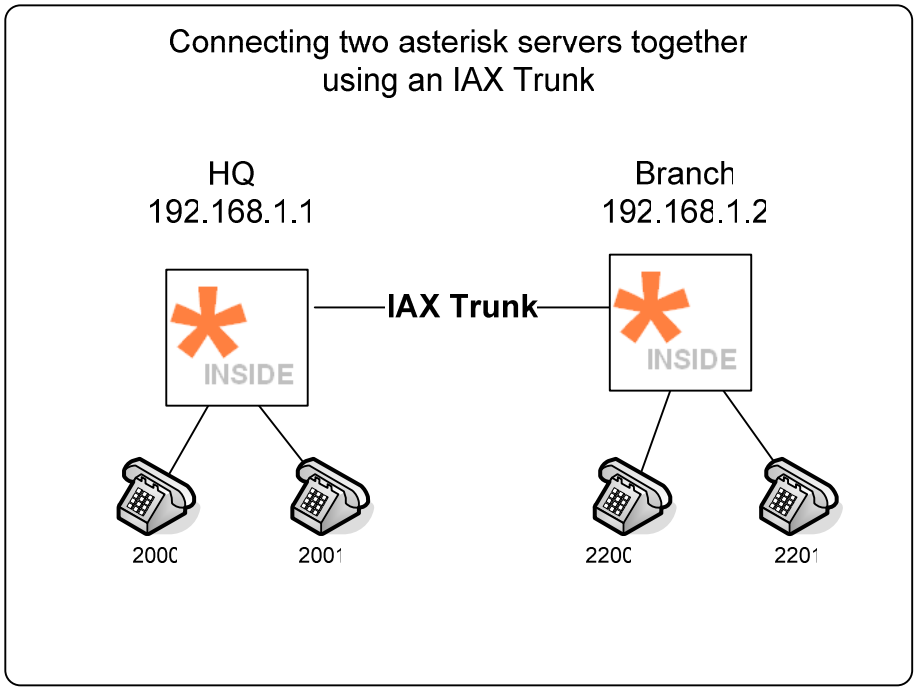


Figure 6-7 Connecting two Asterisk servers using IAX

It is very easy to connect one server to another. You will not need to register because the IP addresses are known. You will just have to create the peers and users in the iax.conf file. All extensions in the HQ site start with 20 followed by two digits (e.g. 2000). In the Branch, all extensions start with 22 followed by two digits (e.g. 2200). We will use trunk and you will need a zaptel timing source to enable this feature.

Step 1: Edit the iax.conf file in the Branch server.

```
[general]
bindport=4569                   ; bindport and bindaddr may be specified
bindaddr=0.0.0.0                ; more than once to bind to multiple
disallow=all
allow=ulaw
;allow=gsm

[Branch]
type=user
context=default
secret=password
host=192.168.2.10
trunk=yes
notransfer=yes

[HQ]
type=peer
```

```
context=default
username=HQ
secret=password
host=192.168.2.10
callerID='HQ'
trunk=yes
notransfer=yes

[2200]
type=friend
auth=md5
context=default
secret=password
host=dynamic
callerid='2000'

[2201]
type=friend
auth=md5
context=default
secret=password
host=dynamic
callerid='2001'
```

Step 2: Configure the extensions.conf file in Branch server

```
[general]
static=yes
writeprotect=no
autofallthrough=yes
clearglobalvars=no
priorityjumping=no

[globals]

[default]
exten=>_20XX,1,dial(IAX2/HQ/${EXTEN},20)
exten=>_20XX,2,hangup

exten=>_22XX,1,dial(IAX2/${EXTEN},20)
exten=>_22XX,2,hangup
```

Step 3: Configure the iax.conf file in the HQ Server

```
[general]
bindaddr=0.0.0.0
bindport=4569
disallow=all
allow=ulaw
allow=gsm

[Branch]
type=peer
context=default
username=Branch
secret=password
```

```
host=192.168.2.9
callerid="Branch"
trunk=yes
notransfer=yes

[HQ]
type=user
secret=password
context=default
host=192.168.2.9
callerid="HQ"
trunk=yes
notransfer=yes

[2000]
type=friend
auth=md5
context=default
secret=password
callerid="2200"
host=dynamic

[2001]
type=friend
auth=md5
context=default
secret=password
callerid="2201"
host=dynamic
```

Step 4: Configure the extensions.conf file in the HQ server.

```
[general]
static=yes
writeprotect=no
autofallthrough=yes
clearglobalvars=no
priorityjumping=no

[globals]

[default]
exten=>_22XX,1,Dial(IAX2/Branch/${EXTEN})
exten=>_22XX,2,hangup

exten=>_20XX,1,Dial(IAX2/${EXTEN})
exten=>_20XX,2,hangup
```

Step 5: Test a call from the 2000 phone in the HQ server to the 2200 phone in the Branch server.

6.7 IAX AUTHENTICATION

Let's analyze now the IAX authentication process from the practical standpoint and help you to choose the best method for each specific requirement.

6.7.1 Incoming connections

When Asterisk receives an incoming connection, the initial information can include a user name (from the field "username=") or not. The incoming connection has an IP address too and Asterisk uses it for authentication as well.

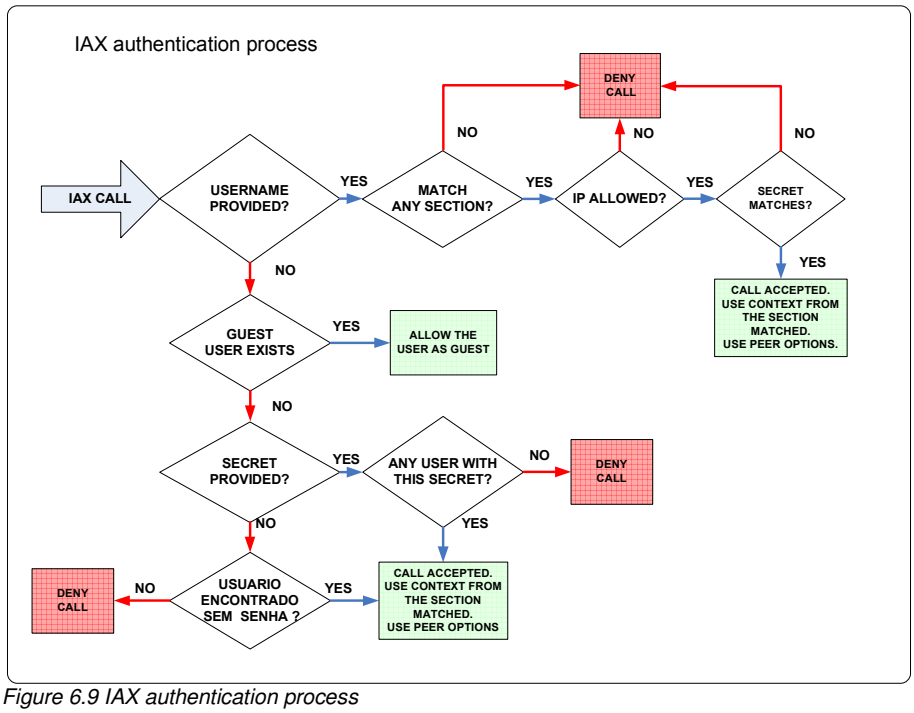


Figure 6.9 IAX authentication process

If a user is provided, Asterisk does the following:

1. Searches the iax.conf for an entry with "type=user" (or "type=friend" with a section name matching the username. If it did not find it, Asterisk refuses the connection.

2. If the entry found has deny/allow configurations, it compares the IP address from the caller to decide whether to accept the call or not depending on the deny/allow clauses.

3. It checks the password (secret) using plaintext, md5, or RSA.

4. It accepts the connection and sends the call to the context specified in the line "context=" from the iax.conf file.

If a username is not provided, Asterisk does the following:

1. Searches for an entry containing "type=user" (or type="friend") in the iax.conf file without a specified secret. It checks deny/allow clauses as well. If an entry is found, the connection is accepted and the section name is used as the user's name.

2. Searches for an entry containing "type=user" (or "type=friend") in the iax.conf file with a secret or RSA key specified. It checks deny/allow clauses. If an entry is found it tries to authenticate the caller using the specified secret if it matches, it accepts the connection. Section name is the user's name.

Let's suppose your iax.conf file has the following entries

```
[guest]
type=user
context=guest

[iaxtel]
type=user
context=incoming
auth=rsa
inkeys=iaxtel

[iax-gateway]
type=friend
allow=192.168.0.1
context=incoming
host=192.168.0.1

[iax-friend]
type=user
secret=this_is_secret
auth=md5
context=incoming
```

If a call has a specified username like:

- guest
- iaxtel
- iax-gateway
- iax-friend

Asterisk will try to authenticate the call using only the corresponding entry in the iax.conf file. If any other names were specified the call would be rejected.

If no user is specified, Asterisk will try to authenticate the connection as guest. However, if guest does not exist, it will try any other connections with a matching secret. In other words, if you don't have a guest section in your iax.conf file, a malicious user could try to guess any matching secret by not specifying the user name. IP addresses deny/allow restrictions apply too.

A good way to avoid secret guessing is to use RSA authentication. Another method is to restrict the IP addresses allowed to call in.

6.7.2 IP address restrictions

permit = <ipaddr>/<netmask> deny = <ipaddr>/<netmask>	Rules are interpreted in sequence and all are evaluated (this concept is different from ACLs usually found in routers and firewalls). Example#1 permit=0.0.0.0/0.0.0.0 deny=192.168.0.0/255.255.255.0 Will deny any packet from 192.168.0.0/24 network Example#2 deny=192.168.0.0/255.255.255.0 permit=0.0.0.0/0.0.0.0 It will permit any packet. The last instruction supersedes the first.

6.7.3 Outbound connections

Outbound connections take authentication information through the following methods:

- IAX2 channel description passed by the dial() application.
- An entry with "type=peer" or type="friend" in the iax.conf file.
- A combination of both methods.

6.7.4 Connecting two Asterisk servers (simplified)

Step 1: Edit the iax.conf file in the Branch server.

```
[general]
bindport=4569                   ; bindport and bindaddr may be specified
bindaddr=0.0.0.0                ; more than once to bind to multiple
disallow=all
allow=ulaw

[Branch]
type=user
context=default
secret=password
host=192.168.2.10
trunk=yes
notransfer=yes

[2200]
type=friend
auth=md5
context=default
secret=password
host=dynamic
callerid='2000'

[2201]
type=friend
auth=md5
context=default
secret=password
host=dynamic
callerid='2001'
```

Step 2: Configure the extensions.conf file in Branch server

```
[general]
static=yes
writeprotect=no
autofallthrough=yes
clearglobalvars=no
priorityjumping=no

[globals]

[default]
exten=>_20XX,1,dial(IAX2/HQ:password@192.168.2.10/${EXTEN},20)
exten=>_20XX,2,hangup
exten=>_22XX,1,dial(IAX2/${EXTEN},20)
exten=>_22XX,2,hangup
```

Step 3: Configure the iax.conf file in HQ Server

```
[general]
bindaddr=0.0.0.0
bindport=4569
disallow=all
```

```
allow=ulaw
allow=gsm

[HQ]
type=user
secret=password
context=default
host=192.168.2.9
callerid="HQ"
trunk=yes
notransfer=yes

[2000]
type=friend
auth=md5
context=default
secret=password
callerid="2200"
host=dynamic

[2001]
type=friend
auth=md5
context=default
secret=password
callerid="2201"
host=dynamic
```

Step 4: Configure the extensions.conf file in HQ server.

```
[general]
static=yes
writeprotect=no
autofallthrough=yes
clearglobalvars=no
priorityjumping=no

[globals]

[default]
exten=>_22XX,1,Dial(IAX2/Branch:password@192.168.2.9/${EXTEN})
exten=>_22XX,2,hangup

exten=>_20XX,1,Dial(IAX2/${EXTEN})
exten=>_20XX,2,hangup
```

Step 5: Test a call from the 2000 phone in the HQ server to the 2200 phone in the Branch server.

6.8 THE IAX.CONF FILE CONFIGURATION

The iax.conf file has several parameters and to discuss each parameter one-by-one would be boring and counterproductive. All parameters can be found in the sample file along with a description. In wiki www.voip-info.org you will

find detailed information about each one. We will show here some of the most important configurations in the general, peers, and users sections.

6.8.1 [General] Section

Server Addresses

bindport = <portnum>	Configures the IAX UDP port. Default is 4569.
bindaddr = <ipaddr>	Use 0.0.0.0 to bind Asterisk to all interfaces or specify the IP address of a specific interface.

Codec selection

bandwidth = [low|medium|high]	High =all codecs Medium=all codecs except ulaw and alaw Low= low bandwidth codecs
allow/disallow = [alaw|ulaw|gsm|g.729| etc.]	Codec selection fine tuning

6.8.2 Jitter buffer

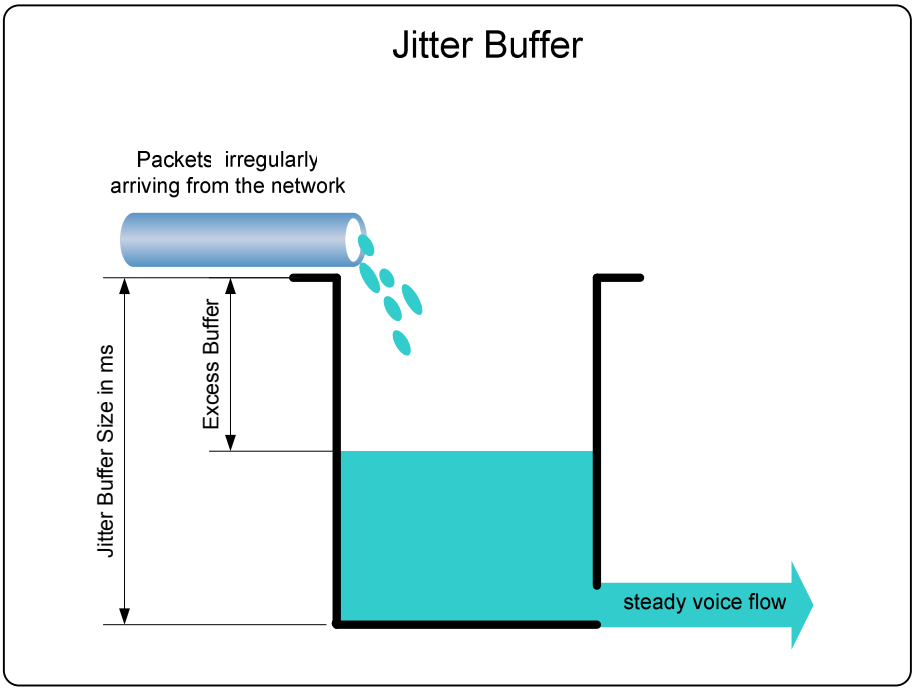


Figure 6.10 Jitter Buffer

Jitter is the delay variation between packets. It is the most important factor that affects voice quality. A Jitter buffer is used to compensate for the delay variation. It sacrifices latency in favor of lower jitter. You can make an analogy between the jitter buffer and a water tank. Both can receive packets or water at irregular intervals, but will deliver in the bottom a regular flow.

A small jitter below 20 ms is usually unperceivable. However, jitter above this level is annoying. The latency or delay should be kept below 150 ms. By creating a jitter buffer, we will sacrifice some delay for a lower jitter. This concept is known as "delay-budget".

You can affect the jitter buffer with these parameters:

- **Jitterbuffer=<yes/no>** – Enables or disables
- **Dropcount=<number>** - Maximum amount of frames that should be delayed in the last two seconds. The recommended setting is 3 (1,5% of dropped frames)
- **Maxjitterbuffer=<ms> -** Usually below 100 ms
- **Maxexcessbuffer=<ms>** - If the network delay improves, the jitter buffer could be oversized. Then Asterisk will try to reduce it.
- **Minexcessbuffer=<ms>** - if the excess buffer decreases until this value, Asterisk starts to increase the buffer size.

6.8.3 Frame tagging

The parameter below marks the IP packet in the type of service field. Routers can read this tag, thereby prioritizing traffic. In the version 1.4, Asterisk is now using DSCP codes for this field (RFC2474).

Allowed values are CS0, CS1, CS2, CS3, CS4, CS5, CS6, CS7, AF11, AF12, AF13,AF21, AF22, AF23, AF31, AF32, AF33, AF41, AF42, AF43 and ef (expedited forwarding)

```
tos=ef
```

6.9 IAX2 DEBUG COMMANDS

iax2 show netstats

```
vtsvoffice*CLI> iax2 show netstats
                          -------- LOCAL ----------------------  -------- REMOTE --------------------
Channel                  RTT  Jit  Del  Lost   %  Drop  OOO  Kpkts  Jit  Del  Lost   %  Drop  OOO  Kpkts
IAX2/8590-1               16   -1    0    -1  -1     0   -1      1   60  110     3   0     0    0      0
```

iax2 show channels

```
vtsvoffice*CLI> iax2 show channels
Channel              Peer             Username    ID (Lo/Rem)  Seq (Tx/Rx)  Lag      Jitter  JitBuf  Format
IAX2/8590-2          8.8.30.43        8590        00002/26968  00004/00003  00000ms  -0001ms  0000ms  unknow
```

iax2 show peers

```
vtsvoffice*CLI> iax2 show peers
Name/Username    Host                 Mask             Port          Status
8584             (Unspecified)   (D)  255.255.255.255  0             UNKNOWN
8564             (Unspecified)   (D)  255.255.255.255  0             UNKNOWN
8576             (Unspecified)   (D)  255.255.255.255  0             UNKNOWN
8572             (Unspecified)   (D)  255.255.255.255  0             UNKNOWN
8571             (Unspecified)   (D)  255.255.255.255  0             UNKNOWN
8585             (Unspecified)   (D)  255.255.255.255  0             UNKNOWN
8589             (Unspecified)   (D)  255.255.255.255  0             UNKNOWN
8590             8.8.30.43       (D)  255.255.255.255  4569          OK (16 ms)
3232             (Unspecified)   (D)  255.255.255.255  0             UNKNOWN
9 iax2 peers [1 online, 8 offline, 0 unmonitored]
```

iax2 debug

Look into this output and identify the beginning and the end of the call. Observe the delay and jitter information obtained using poke and pong packets. Theses packets help to create the output of "iax2 show netstats" command.

```
vtsvoffice*CLI> iax2 debug
IAX2 Debugging Enabled


Rx-Frame Retry[ No] -- OSeqno: 000 ISeqno: 000 Type: IAX     Subclass: REGREQ
   Timestamp: 00003ms  SCall: 26975  DCall: 00000 [8.8.30.43:4569]
   USERNAME         : 8590
   REFRESH          : 60

Tx-Frame Retry[000] -- OSeqno: 000 ISeqno: 001 Type: IAX     Subclass: REGAUTH
   Timestamp: 00009ms  SCall: 00003  DCall: 26975 [8.8.30.43:4569]
   AUTHMETHODS      : 2
   CHALLENGE        : 137472844
   USERNAME         : 8590

Rx-Frame Retry[ No] -- OSeqno: 001 ISeqno: 001 Type: IAX     Subclass: REGREQ
   Timestamp: 00016ms  SCall: 26975  DCall: 00003 [8.8.30.43:4569]
   USERNAME         : 8590
   REFRESH          : 60
   MD5 RESULT       : f772b6512e77fa4a44c2f74ef709e873

Tx-Frame Retry[000] -- OSeqno: 001 ISeqno: 002 Type: IAX     Subclass: REGACK
   Timestamp: 00025ms  SCall: 00003  DCall: 26975 [8.8.30.43:4569]
   USERNAME         : 8590
   DATE TIME        : 2006-04-17  16:03:00
   REFRESH          : 60
   APPARENT ADDRES : IPV4 8.8.30.43:4569
   CALLING NUMBER   : 4830258590
   CALLING NAME     : Flavio

Rx-Frame Retry[ No] -- OSeqno: 002 ISeqno: 002 Type: IAX     Subclass: ACK
```

```
   Timestamp: 00025ms  SCall: 26975  DCall: 00003 [8.8.30.43:4569]
Tx-Frame Retry[000] -- OSeqno: 000 ISeqno: 000 Type: IAX     Subclass: POKE
   Timestamp: 00003ms  SCall: 00006  DCall: 00000 [8.8.30.43:4569]
Rx-Frame Retry[ No] -- OSeqno: 000 ISeqno: 001 Type: IAX     Subclass: ACK
   Timestamp: 00003ms  SCall: 26976  DCall: 00006 [8.8.30.43:4569]
Rx-Frame Retry[ No] -- OSeqno: 000 ISeqno: 001 Type: IAX     Subclass: PONG
   Timestamp: 00003ms  SCall: 26976  DCall: 00006 [8.8.30.43:4569]
   RR_JITTER       : 0
   RR_LOSS         : 0
   RR_PKTS         : 1
   RR_DELAY        : 40
   RR_DROPPED      : 0
   RR_OUTOFORDER   : 0
Tx-Frame Retry[-01] -- OSeqno: 001 ISeqno: 001 Type: IAX     Subclass: ACK
   Timestamp: 00003ms  SCall: 00006  DCall: 26976 [8.8.30.43:4569]

Rx-Frame Retry[ No] -- OSeqno: 000 ISeqno: 000 Type: IAX     Subclass: NEW
   Timestamp: 00003ms  SCall: 26977  DCall: 00000 [8.8.30.43:4569]
   VERSION         : 2
   CALLING NUMBER  : 8590
   CALLING NAME    : 4830258590
   FORMAT          : 2
   CAPABILITY      : 1550
   USERNAME        : 8590
   CALLED NUMBER   : 8580
   DNID            : 8580

Tx-Frame Retry[000] -- OSeqno: 000 ISeqno: 001 Type: IAX     Subclass: AUTHREQ
   Timestamp: 00007ms  SCall: 00004  DCall: 26977 [8.8.30.43:4569]
   AUTHMETHODS     : 2
   CHALLENGE       : 190271661
   USERNAME        : 8590

Rx-Frame Retry[Yes] -- OSeqno: 000 ISeqno: 000 Type: IAX     Subclass: NEW
   Timestamp: 00003ms  SCall: 26977  DCall: 00000 [8.8.30.43:4569]
   VERSION         : 2
   CALLING NUMBER  : 8590
   CALLING NAME    : 4830258590
   FORMAT          : 2
   CAPABILITY      : 1550
   USERNAME        : 8590
   CALLED NUMBER   : 8580
   DNID            : 8580

Tx-Frame Retry[-01] -- OSeqno: 000 ISeqno: 001 Type: IAX     Subclass: ACK
   Timestamp: 00003ms  SCall: 00004  DCall: 26977 [8.8.30.43:4569]
Rx-Frame Retry[ No] -- OSeqno: 001 ISeqno: 001 Type: IAX     Subclass: AUTHREP
   Timestamp: 00063ms  SCall: 26977  DCall: 00004 [8.8.30.43:4569]
   MD5 RESULT      : 57cc5c48affba14106c29439944413a1

Tx-Frame Retry[000] -- OSeqno: 001 ISeqno: 002 Type: IAX     Subclass: ACCEPT
   Timestamp: 00054ms  SCall: 00004  DCall: 26977 [8.8.30.43:4569]
   FORMAT          : 1024

Tx-Frame Retry[000] -- OSeqno: 002 ISeqno: 002 Type: CONTROL Subclass: ANSWER
   Timestamp: 00057ms  SCall: 00004  DCall: 26977 [8.8.30.43:4569]
Tx-Frame Retry[000] -- OSeqno: 003 ISeqno: 002 Type: VOICE   Subclass: 138
   Timestamp: 00090ms  SCall: 00004  DCall: 26977 [8.8.30.43:4569]
Rx-Frame Retry[ No] -- OSeqno: 002 ISeqno: 002 Type: IAX     Subclass: ACK
   Timestamp: 00054ms  SCall: 26977  DCall: 00004 [8.8.30.43:4569]
Rx-Frame Retry[ No] -- OSeqno: 002 ISeqno: 003 Type: IAX     Subclass: ACK
   Timestamp: 00057ms  SCall: 26977  DCall: 00004 [8.8.30.43:4569]
Rx-Frame Retry[ No] -- OSeqno: 002 ISeqno: 004 Type: IAX     Subclass: ACK
   Timestamp: 00090ms  SCall: 26977  DCall: 00004 [8.8.30.43:4569]
Rx-Frame Retry[ No] -- OSeqno: 002 ISeqno: 004 Type: VOICE   Subclass: 138
   Timestamp: 00210ms  SCall: 26977  DCall: 00004 [8.8.30.43:4569]
Tx-Frame Retry[-01] -- OSeqno: 004 ISeqno: 003 Type: IAX     Subclass: ACK
   Timestamp: 00210ms  SCall: 00004  DCall: 26977 [8.8.30.43:4569]
Rx-Frame Retry[ No] -- OSeqno: 003 ISeqno: 004 Type: IAX     Subclass: PING
   Timestamp: 02083ms  SCall: 26977  DCall: 00004 [8.8.30.43:4569]
Tx-Frame Retry[000] -- OSeqno: 004 ISeqno: 004 Type: IAX     Subclass: PONG
   Timestamp: 02083ms  SCall: 00004  DCall: 26977 [8.8.30.43:4569]
```

```
   RR_JITTER       : 0
   RR_LOSS         : 0
   RR_PKTS         : 1
   RR_DELAY        : 40
   RR_DROPPED      : 0
   RR_OUTOFORDER   : 0

Rx-Frame Retry[ No] -- OSeqno: 004 ISeqno: 005 Type: IAX     Subclass: ACK
   Timestamp: 02083ms  SCall: 26977  DCall: 00004 [8.8.30.43:4569]
Rx-Frame Retry[ No] -- OSeqno: 004 ISeqno: 005 Type: IAX     Subclass: HANGUP
   Timestamp: 08693ms  SCall: 26977  DCall: 00004 [8.8.30.43:4569]
   CAUSE           : Dumped Call
```

To turn off debugging, use:

```
vtsvoffice*CLI>iax2 no debug
```

6.10 Summary

In this chapter, we have learned the strengths and weaknesses of the IAX protocol. We demonstrated IAX working in several scenarios, such as soft-phones as well as a trunk between two Asterisk servers. Trunk mode allows you to save bandwidth by carrying more then one call in a single packet. At the end, you learned console commands to check the status and debug the protocol.

6.11 QUESTIONS

1. Two of the main benefits of IAX are bandwidth savings and easier NAT traversal.

a) False
b) True

2. IAX protocols use different UDP ports for signaling and media.

a) False
b) True

3. The bandwidth used by the IAX protocol is the voice payload and the following headers (Mark all that apply):

a) IP
b) UDP
c) IAX
d) RTP
e) cRTP

4. It is important to match the codec payload (20 to 30 ms) with frame synchronization (20ms default) when using trunk mode.

a) False
b) True

5. When IAX is used in trunk mode, just one header is used for multiple calls.

a) False
b) True

6. IAX is the most used protocol to connect to service providers, because it is easier for Nat traversal.

a) False
b) True

7. In an IAX channel as shown below, the option <secret> can be a password or a _____.

```
IAX/[<user>[:<secret>]@]<peer>[:<portno>][/<exten>[@<context>][/<options>]]
```

8. The IAX2 show registry shows information about:

   a) Registered users
   b) Providers to which Asterisk is connected

9. Jitter buffer sacrifices latency to have a steady flow of voice.

   a) True
   b) False

10. RSA keys can be used for IAX authentication. You have to keep the _____ key secret and give to your costumers and partners the matching _____ key.

   a) public, private
   b) private, public
   c) shared, private
   d) public, shared

Paga left intentionally empty

Chapter 7

The SIP Protocol

7.1 OBJECTIVES

By the end of this chapter, you will be able to:

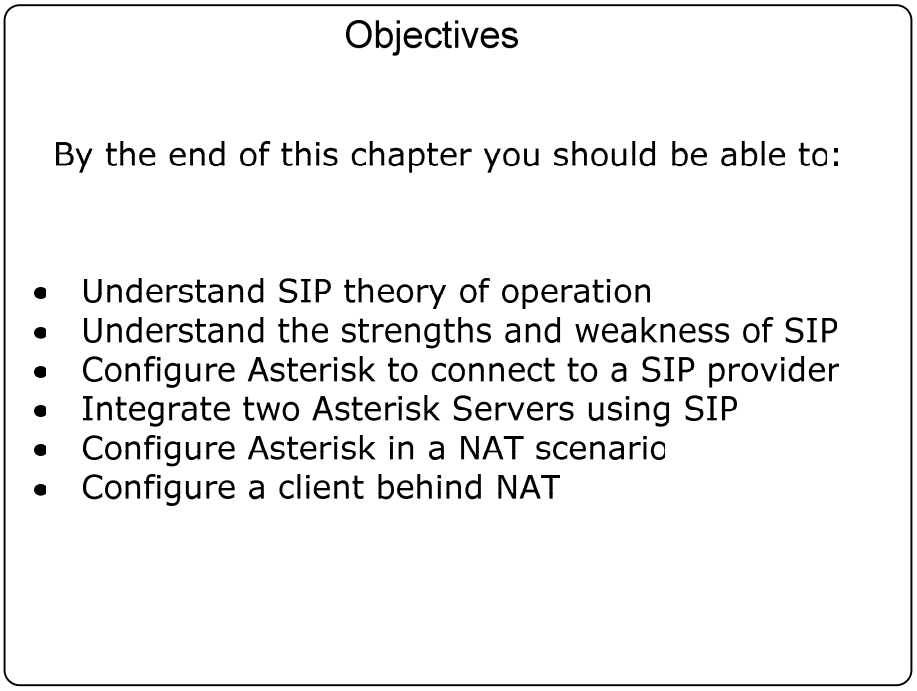


Figure 7.1 Objectives

7.2 OVERVIEW

SIP or Sessions Initiated Protocol is a text-based protocol similar to HTTP and SMTP. It was designed to initialize, keep and terminate interactive communication sessions between users. These sessions may include voice, video, chat, interactive games, and others. It was define by the IETF and is becoming a *de facto* standard for voice communications.

7.2.1 Theory of Operation

SIP is a signaling protocol and it has the following components:

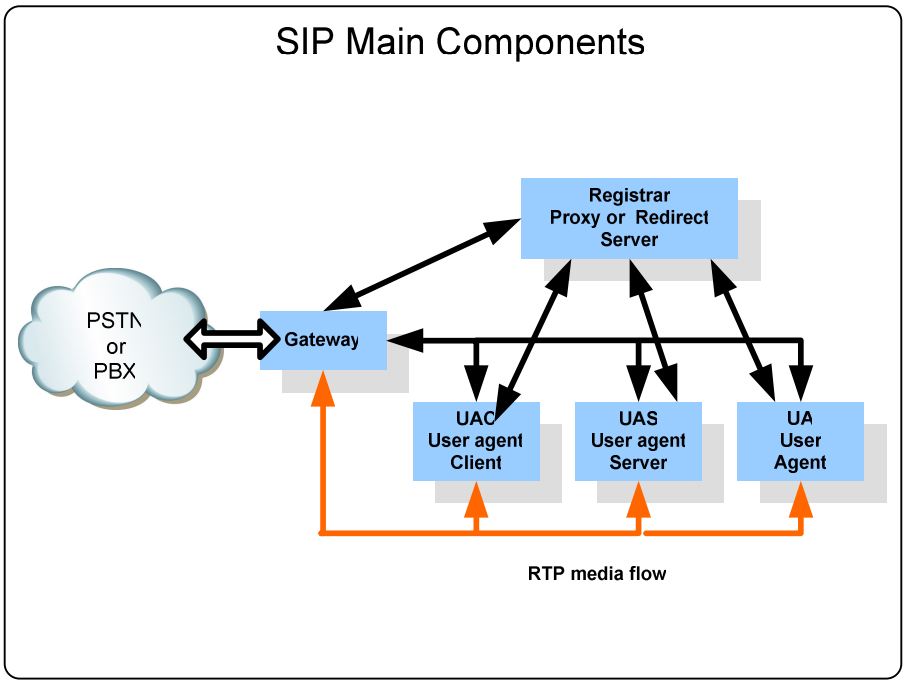


Figure 7.2 SIP Main Components

- **UAC (user agent client)** - The client or terminal that initializes SIP signaling.
- **UAS (user agent server)** – The server that responds to a SIP signaling coming from an UAC.
- **UA (user agent)** – The SIP terminal (phones or gateways that contain both UAC and UAS).
- **Proxy Server** – Receives requests from an UA and transfers to other SIP Proxies if the particular station is not under their administration.
- **Redirect Server** – Receives requests and sends them back to the UA, including destination data, instead of directly forwarding to the destination.
- **Location Server** – Receives requests from an UA and updates the location database with this information.

Generally, the Proxy, Redirect, and Location servers are hosted within the same hardware and use the same piece of software that we call SIP Proxy. The SIP Proxy is responsible for location database maintenance, connection establishment, and session termination.

7.2.2 SIP Register process

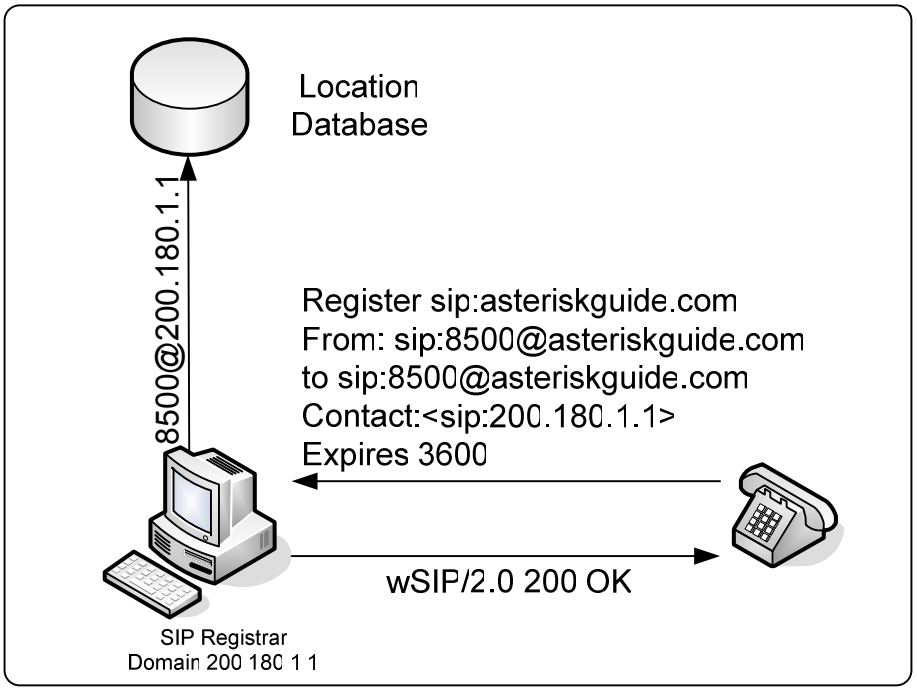


Figure 7.3 SIP Register Process

Before a phone can receive calls, it needs to register to a location database. In the location database, the IP address will be bond to the name. In the example above, extension 8500 will be bound to IP address 200.180.1.1. In the SIP architecture, the registered extension could be flavio@asteriskguide.com as well. You do not need, necessarily, to use phone numbers.

7.2.3 Proxy operation

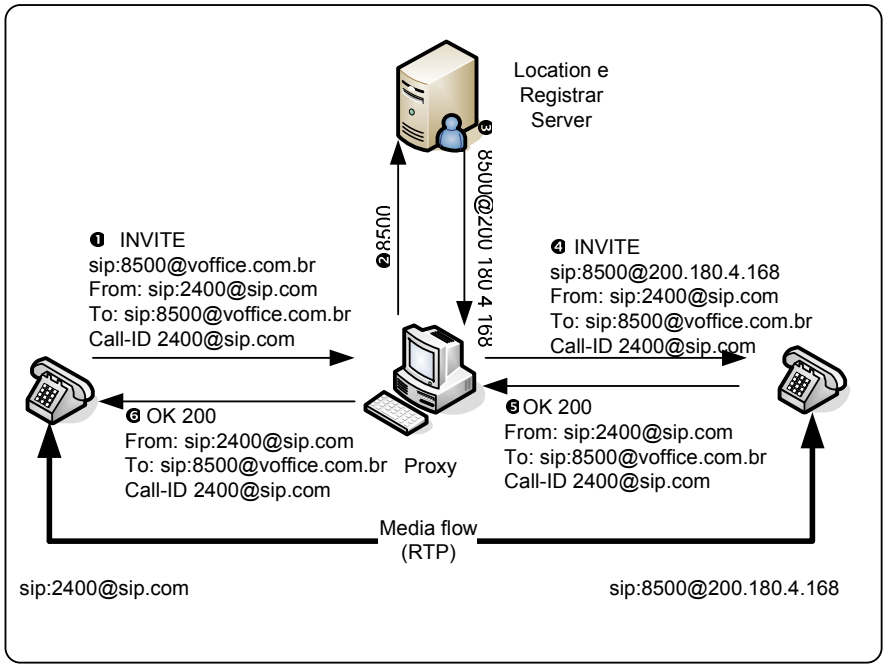


Figure 7.4 SIP proxy Operation

7.2.4 Redirect operation

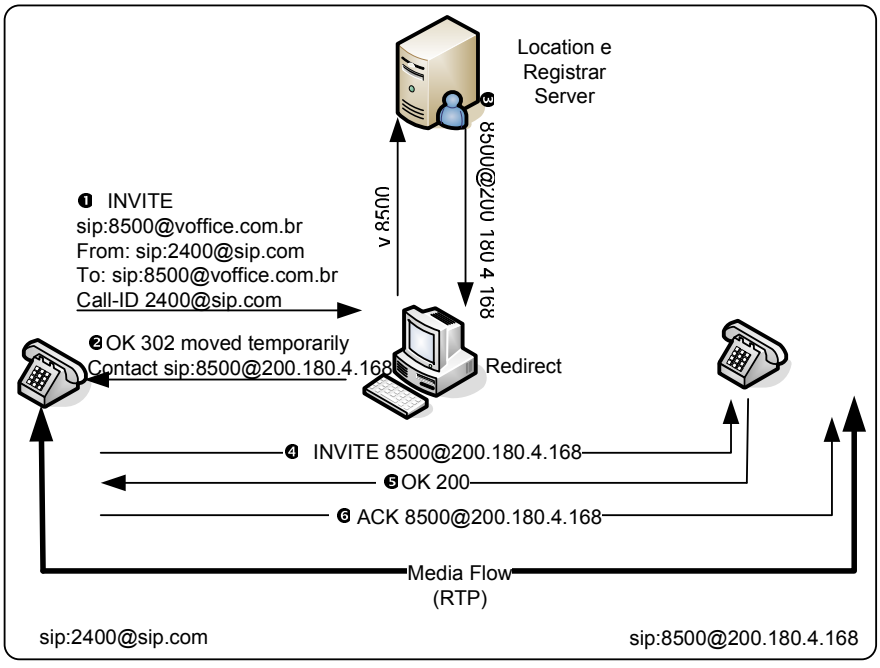


Figure 7.5 SIP Redirect Operation

7.2.5 How Asterisk treats SIP

It is important to understand that Asterisk is not a SIP Proxy neither a SIP redirector. Asterisk can make the role of the registrar and location server, but it connects two UACs to itself. Therefore, Asterisk is considered a Back-to-Back user agent (B2BUA). In other words, it connects two SIP channels, bridging them together. Asterisk has a re-invite mechanism that can make the SIP channels talk to each other directly instead of passing through Asterisk can be used together with a SIP proxy like SER (Sip Express Router, www.iptel.org).

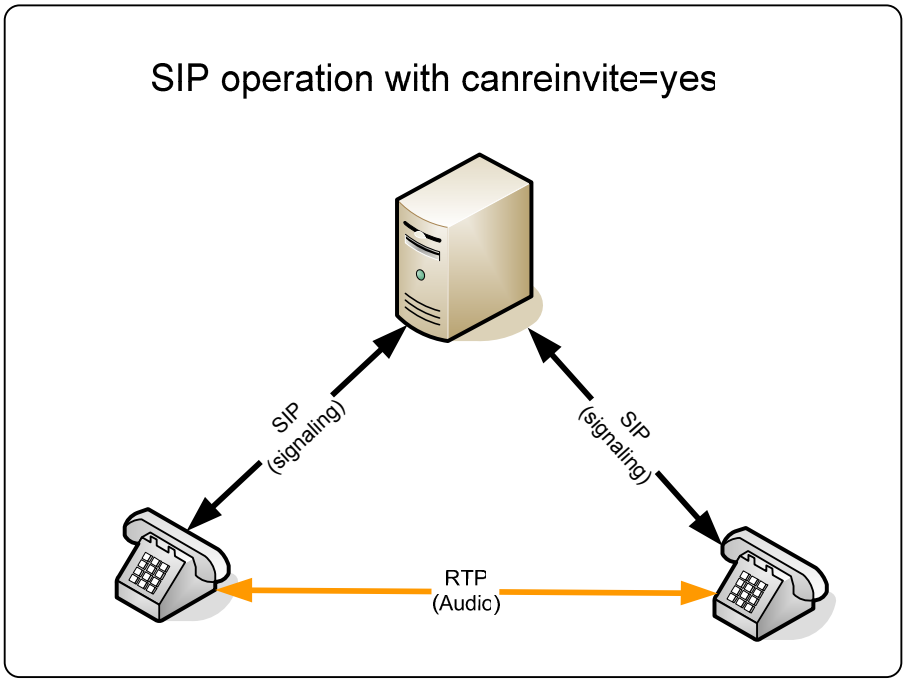


Figure 7.6 Asterisk with canreinvite option set

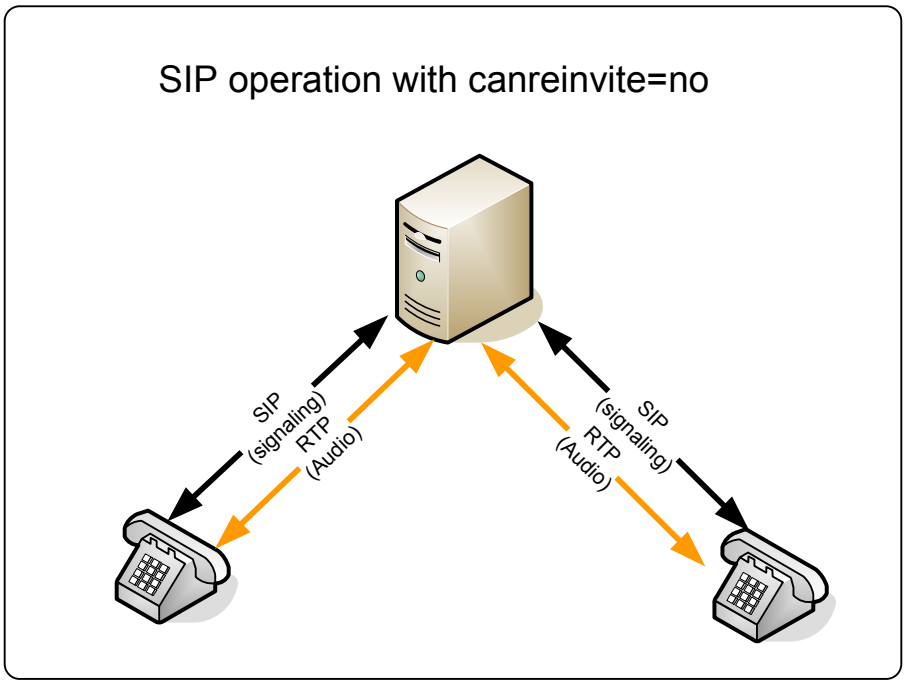


Figure 7.7 Asterisk without canreinvite option

7.2.6 Sip Messages

The basic SIP messages are:

- INVITE – connection establishment
- ACK – Acknowledge
- BYE – connection termination
- CANCEL – connection termination for a non established call
- REGISTER – Register an UAC to a SIP PROXY
- OPTIONS – Can be used to check availability
- REFER – Transfer a SIP call to someone else
- SUBSCRIBE – Subscribe to notification events
- NOTIFY – Send out channel information
- INFO – Send various messages (i.e. DTMF)

- MESSAGE – Send instant messages

The SIP responses are in text format and are easily readable (similar to http messages). The most important responses are:

- 1XX – Information messages (100–trying, 180–ringing, 183–progress)
- 2XX – Successful request complete (200 – OK)
- 3XX – Call redirect, request has to be directed to other place (302 – moved temporarily, 305 use proxy)
- 4XX – Error (403 – Forbidden)
- 5XX – Server Error (500 – Internal Server Error; 501 – Not implemented)
- 6XX – Global Failure (606 – Not acceptable)

Example:

```
INVITE sip:2000@192.168.1.133 SIP/2.0
Via: SIP/2.0/UDP
192.168.1.116;rport;branch=z9hG4bKc0a8017400000063452fafbb00006967000000d2
From: "unknown"<sip:2001@192.168.1.133>;tag=1556140623845
To: <sip:2000@192.168.1.133>
Contact: <sip:2001@192.168.1.116>
Call-ID: 64B4C8EC-FCFC-49E9-98B1-90982EEEBED3@192.168.1.116
CSeq: 2 INVITE
Max-Forwards: 70
User-Agent: SJphone/1.61.312b (SJ Labs)
Content-Length: 335
Content-Type: application/sdp
Proxy-Authorization: Digest
username="2001",realm="asterisk",nonce="6c55905e",uri="sip:2000@192.168.1.133",resp
onse="983c0099eea125d8cdfe93b0ec99f3ec",algorithm=MD5
```

7.2.7 SDP (Session description protocol)

SDP is defined in IETF RFC2327. It is intended for describing multimedia sessions for the purpose of session announcement, session invitation, and other forms of multimedia session initiation. The SDP includes:

- Transport protocol (RTP/UDP/IP)
- Type of media (text, audio, video)
- Media format or codec (H.261 video, g.711 audio, etc.)

- Information needed to receive these media (addresses, ports, etc.)

The example below is the transcription of a SDP describing a call between two phones.

```
v=0
o=- 3369741883 3369741883 IN IP4 192.168.1.116
s=SJphone
c=IN IP4 192.168.1.116
t=0 0
a=setup:active
m=audio 49160 RTP/AVP 3 97 98 8 0 101
a=rtpmap:3 GSM/8000
a=rtpmap:97 iLBC/8000
a=rtpmap:98 iLBC/8000
a=fmtp:98 mode=20
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-11,16
```

7.3 SIP ADVANCED SCENARIOS

In chapter three, we have discussed the basic options to connect a SIP Phone to Asterisk. Now let's move on to more advanced configurations. In the next sections you will learn how to configure Asterisk to connect to a SIP provider, how to connect two Asterisks together using SIP and how to place a call to a SIP provider. All SIP configurations are done in the file "/etc/asterisk/sip.conf"

7.3.1 Connecting Asterisk to a SIP provider

Asterisk is often used to connect to a SIP VoIP provider. VoIP providers usually have better rates for phone call than traditional providers. Another interesting attractive of VoIP providers is the possibility of buying DID numbers in other cities and even foreign countries. These are good reasons to use VoIP for telecommunications. In this section, you will learn how to connect Asterisk to a VoIP provider.

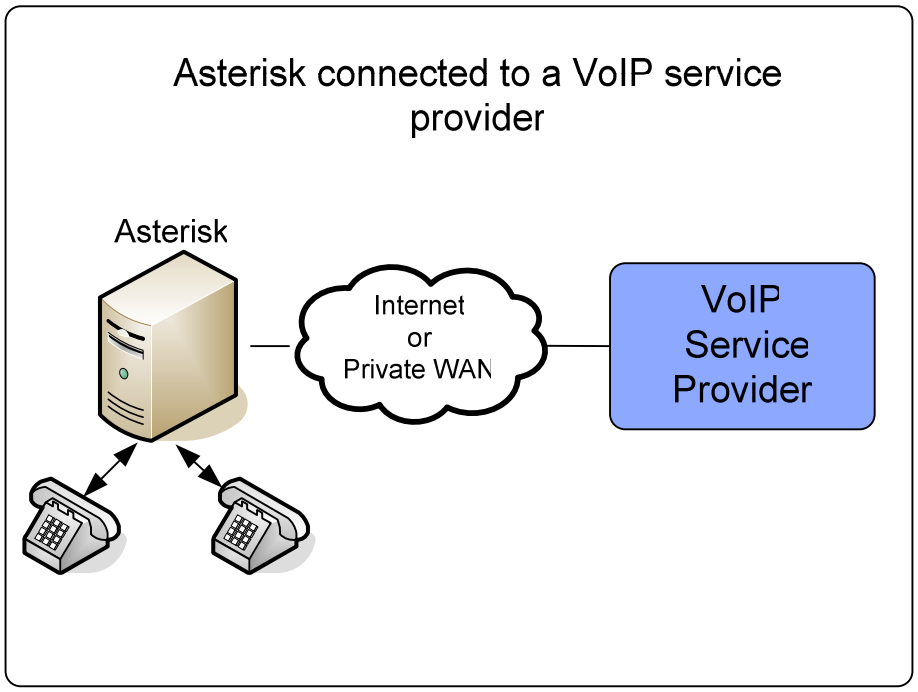


Figure 7.8 Asterisk connected to a VoIP service provider

Three steps are required to connect Asterisk to a SIP provider. Tests can be carried on by establishing an account in your favorite provider.

Step 1: Registering to a SIP provider (sip.conf)

This configuration will allow your provider to locate Asterisk's IP address. In the statement below, we are telling Asterisk to register to a SIP provider www.freeworlddialup.com, and inform the provider Asterisk's IP address. The statement says that you want to receive calls in the extension 4100.

In the [general] section of the sip.conf file enter the line below:

```
register=>621538:password@fwd.fwdnet.net/4100
```

Step 2: Configure the [peer] (sip.conf)

Create an entry of type peer to the desired provider to simplify the asterisk dialing.

```
[fwd]
context=incoming
type=friend
dtmfmode=rfc2833
canreinvite=no
username=621538
secret=secret
host=fwd.pulver.com
```

```
fromuser=621538
fromdomain=fwd.pulver.com
insecure=very
disallow=all
allow=ulaw
```

Step 3: Create a route to the FWD in the dial plan

We will choose the digits 010 as the destination route to FWD. To dial #610000 inside the FWD simply dial 010610000.

```
exten=>_010.,1,Set(CALLERID(num)=621538)
exten=>_010.,n,Set(CALLERID(Name)=”Flavio Gonçalves”)
exten=>_010.,n,Dial(SIP/${EXTEN:3}@fwd)
exten=>_010.,n,Hangup
```

VoIP provider specific options

Below we will go into the details of the options set in sip.conf file for connection to a VoIP provider.

```
register=>621538:password@fwd.fwdnet.net/4100
```

The instruction register in the sip.conf file is used to register to a provider. The register transaction is authenticated with name and secret. You can use a slash (“/”) to provide an extension for incoming calls. Technically speaking, the extension will be placed in the “Contact” header field of the SIP request.

The registering behavior can be controlled by some parameters:

```
registertimeout=20
registerattempts=10
```

To check if register occurred successfully, you can use the console command below:

```
CLI>sip show registry
```

The parameter “username” is used in the authentication digest. The digest is computed using username, secret and realm:

```
username=ip1140623535
```

Host defines the VoIP provider address or name:

```
host=fwd.pulver.com
```

The parameters "Fromuser" and "Fromdomain" are sometimes required to authenticate. Those parameters are used in the SIP "From" header field:

```
fromuser=621538
fromdomain=fwd.pulver.com
```

When you connect to a VoIP provider, credentials are required. After the initial invite, the provider sends to you a message called "407 Proxy Authentication Required" and you provide the credentials in the next INVITE message. For incoming calls, your Asterisk asks for the provider credentials. Obviously, the provider does not have a valid credential to your Asterisk server. When you use "insecure=very", you are telling Asterisk to not send the "407 Proxy Authentication Required" to the provider and to make accept incoming calls.

```
insecure=very
```

7.3.2 Connecting two Asterisk servers together through SIP

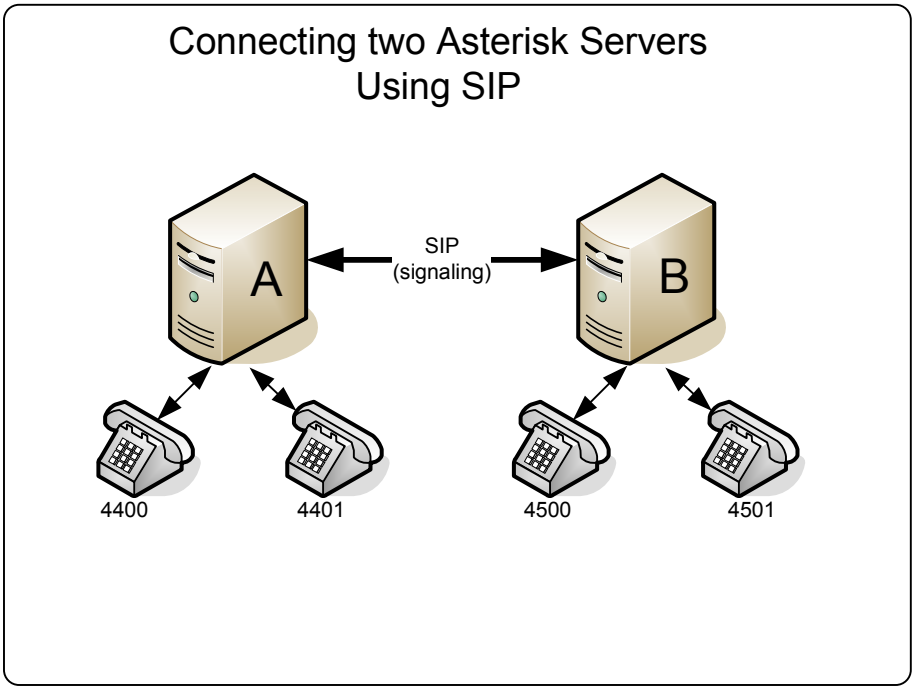


Figure 7.9 Connecting two asterisk server together using SIP

You can use SIP to interconnect two Asterisk boxes. It is important to pay attention to the dial plan before moving on with this configuration. Users usually want to connect other PBXs with minimal effort. The idea now is to use an extension number only to connect to the other PBX.

Step 1: Edit the sip.conf file in server A:

```
[B]
type=user
secret=B
host=A
disallow=all
allow=ulaw
canreinvite=no

[B-out]
type=peer
fromuser=A
username=A
secret=A
host=B
disallow=all
allow=ulaw
canreinvite=no
```

Step 2: Edit the sip.conf file in server B:

```
[A]
type=user
host=B
secret=A
disallow=all
allow=ulaw
canreinvite=no

[A-out]
type=peer
host=A
fromuser=B
username=B
secret=B
disallow=all
allow=ulaw
canreinvite=no
```

Step 3: Edit the extensions.conf file in server A:

```
[default]
exten=_44XX,1,dial(SIP/${EXTEN},20)
exten=_44XX,2,hangup()
exten=_45XX,1,dial(SIP/B-out/${EXTEN})
exten=_45XX,2,hangup()
```

Step 4: Edit the extensions.conf file in server B:

```
[default]
exten=_44XX,1,dial(SIP/A-out/${EXTEN})
exten=_44XX,2,hangup()
```

```
exten=_45XX,1,dial(SIP/${EXTEN})
exten=_45XX,2,hangup()
```

7.3.3 Asterisk domain support

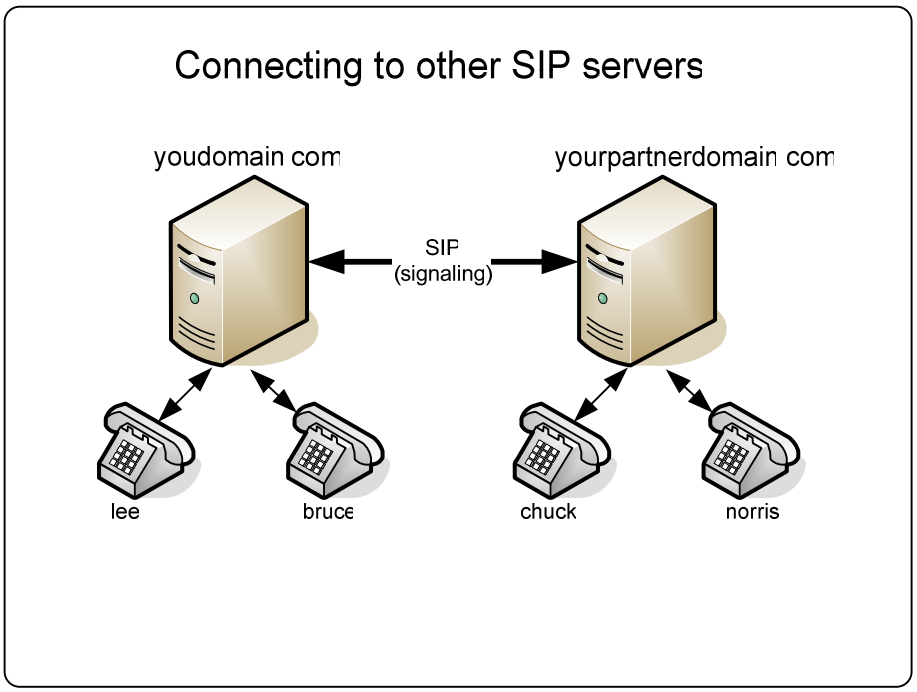


Figure 7.10 connecting other users using a SIP URI.

The SIP protocol follows the Internet architecture. The first thing to do before configuring SIP is to correctly set the DNS Servers. In a SIP environment, you can call a user located in any SIP proxy and other users can call you as well, using your SIP Uniform Resource Identifier (URI). To set a DNS Server for SIP you have to add SRV records to your DNS Server.

```
; SIP server/proxy and its backup server/proxy
sip1.yourdomain.com        21600 IN A    200.180.4.169
sip2.yourdomain.com        21600 IN A    200.175.61.150
;
; DNS SRV records for SIP
_sip._udp.yourdomain.com   21600 IN SRV 10 0 5060 sip1.asteriskguide.com.
_sip._udp.yourdomain.com   21600 IN SRV 20 0 5060 sip2.asteriskguide.com.
```

After configuring the DNS, you can use the URI, which points to a SIP user, SIP phone or telephone extension. A SIP URI looks similar to an email address, for example sip:chuck@yourpartnerdomain.com. Using SIP URI's, no telephone number is needed to make a call from one SIP phone to another. To dial an external user, simply use a statement as the one shown below.

```
exten=4000,1,dial(SIP/chuck@yourpartnerdomain.com)
```

There are some parameters to control domain behavior.

```
srvlookup=yes
```

This parameter enables DNS SRV lookups on outbound calls. Without this parameter, it is no possible to dial calls using sip names based on domain.

```
allowguest=yes
```

The above parameter allows an external invite to be processed without authentication. It processes the call within the context defined in the general section or in the domain statement.

Warning: If you define a context in the general section with access to PSTN, an external user can dial the PSTN over your PBX. casein this case, you will incur any charges. Allow only your own extensions in the context defined in the general section.

```
domain=acme.com,default
```

The domain command allows you to handle more than one domain inside Asterisk. If a call comes from one specific domain, it is directed to a specific context.

```
;autodomain=yes
```

It includes the local IP and hostname in the allowed domains.

```
;allowexternaldomains=no
```

The default is yes. Uncomment the line to disallow calls to outside domains.

7.4 ADVANCED CONFIGURATIONS

7.4.1 Codec configuration

Codec configuration is simple and straightforward. You can set the words "allow" and "disallow" in the [general] section or peer/user section. The best practice is to standardize the codec to avoid transcoding, which is processor intensive. Use the same codec for messages and prompts.

```
[general]
disallow=all
allow=g729
```

7.4.2 DTMF options

In several occasions, you will need to pass digits as if you were operating the voicemail or IVR (interactive voice response) prompts. It is very important to pass DTMF correctly.

The simplest method to pass DTMF is called "inband". It is set in the [general] or peer/user section of the sip.conf file. When you set dtmfmode=inband, DTMF tones will be generated as sounds in the audio channel. The main issue with this method is that when you compress the audio channel using a codec such as g.729, sounds are distorted and DTMF tones are not recognized correctly. If you plan to use dtmfmode=inband, use the g.711 codec (ulaw and alaw).

```
dtmfmode=inband
```

Another way is to use RFC2833. It allows you to pass DTMF tones as named events in the RTP packets. A table of events corresponding to tones is shown below.

Event Codification	
0—9	0--9
*	10
#	11
A—D	15
Flash	16

Table 7.1 – DTMF named Events

```
dtmfmode=rfc2833
```

Finally, you can pass DTMF digits inside SIP packets, instead of RTP packets. This method is defined in the RFC3265 (signaling events) and RFC2976.

```
dtmfmode=info
```

Following the the release of version 1.2, it is now possible to use:

```
dtmfmode=auto
```

It tries to use the RFC2833 and if it is not possible use inband tones.

7.4.3 QoS (quality of service) marking configuration

QoS (quality of service) is a set of techniques responsible for voice quality. QoS is implemented in such a way as to reduce bandwidth, latency, and jitter. The main QoS functions are packet scheduling, fragmentation, and

header compression. QoS is implemented in switches and routers, not by Asterisk itself. However, Asterisk can help routers and switches by marking packets for express delivery. Marking is done using DSCP (differentiated services code point) defined in RFCs 2474 and RFC2475.

```
tos_sip=cs3
tos_audio=ef
tos_video=af41
```

In the 1.4 version 1.4 you can specify different codes for signaling (SIP), audio (RTP), and video (RTP).

7.4.4 SIP authentication

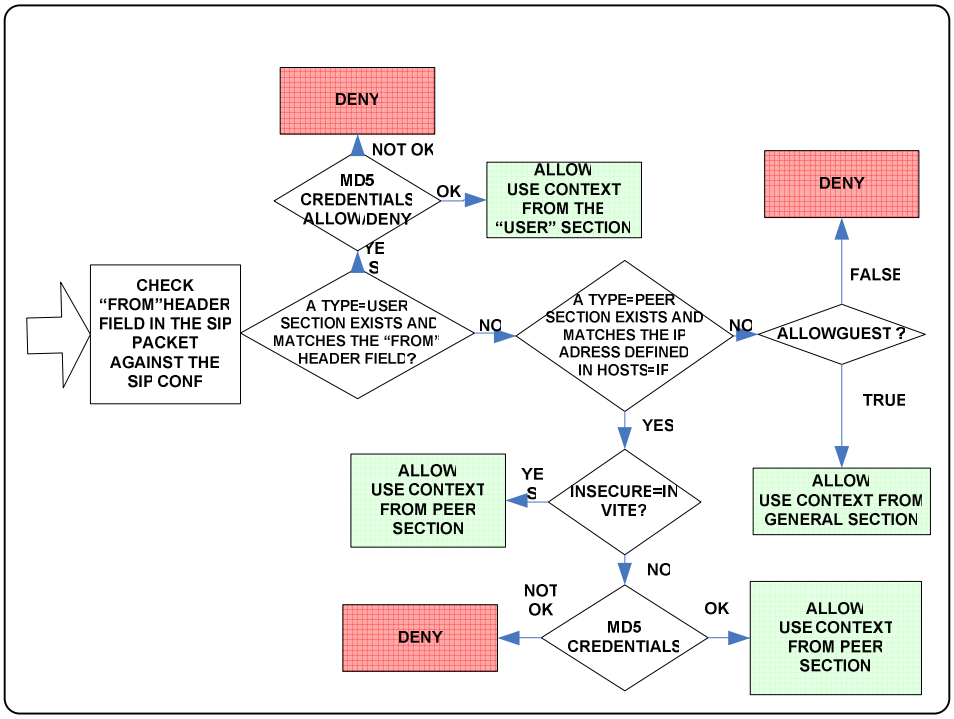


Figure 7.11 SIP authentication

When Asterisk receives a SIP call, it follows the rules described above.

Three parameters play an important role in SIP authentication:

```
allowguest=yes/no
```

This parameter controls if a "user" without a corresponding "peer" can authenticate without a name and secret. We have discussed this parameter before in the domain support section.

```
insecure=invite;port
```

When we use insecure=invite or insecure=very, Asterisk does not generate a message "407 Proxy Authentication Required". Without this message, the user can make a call without authentication. This is often used to connect to VoIP service providers. The calls coming from the VoIP service provider usually are not authenticated.

```
autocreatepeer=yes/no
```

The command above is used when Asterisk is connected to a SIP proxy. It creates a peer dynamically to each call. When this option is enabled, any UAC can connect to the Asterisk Server. It is important to limit the IP connection to the SIP proxy. The SIP proxy, in turn, takes care of access control. Peer configuration is based on the general options as well as the "Contact" header field of the SIP packet.

Warning: Use this with extreme caution as it completely opens Asterisk.

```
secret=senha
```

This parameter configures the secret for authentication. If you do not want to present the secrets in text files, you can use md5secret to put a hash instead of the secret.

To generate the MD5 secret you can use:

```
#echo -n "username:realm:secret" |md5sum
```

Then use the statement below.

```
md5secret=0b0e5d467890....
```

Warning: Do not forget to use the –n parameter; the carriage return will be used in the md5 computation.

```
deny=0.0.0.0/0.0.0.0
permit=192.168.1.0/255.255.255.0
```

The statements above will deny all IP addresses and allow UAC only from the local network (192.168.1.0/24).

7.4.5 RTP options

It is possible to control some RTP parameters.

```
rtptimeout=60
```

Terminate calls without RTP activity for more the 60 seconds when not in hold.

```
rtpholdtimeout=120
```

Terminate calls without RTP activity even on hold (should be bigger then rtptimeout).

7.5 SIP NAT TRAVERSAL

Network Address Translation (NAT) is a feature used by most networks to save Internet IP addresses. Usually, a company receives a small block if IP addresses and end users receive one IP address dynamically when connected to the Internet. NAT solves the addressing problem by mapping internal addresses to external addresses. It keeps in the memory a mapping of internal to external addresses. This mapping is valid for a specific length of time after which the mapping is discarded. The mapping uses IP:port pairs for the internal and external addresses.

There are four kinds of NAT:

- Full Cone
- Restricted Cone
- Port Restricted Cone
- Symmetric

7.5.1 Full Cone

The first NAT, "Full Cone", represents a static mapping from an external IP:port pair to an internal IP:port pair. Any external computer can connect to it using the external IP:port pair. This is the case in non-stateful firewalls implemented with the use of filters.

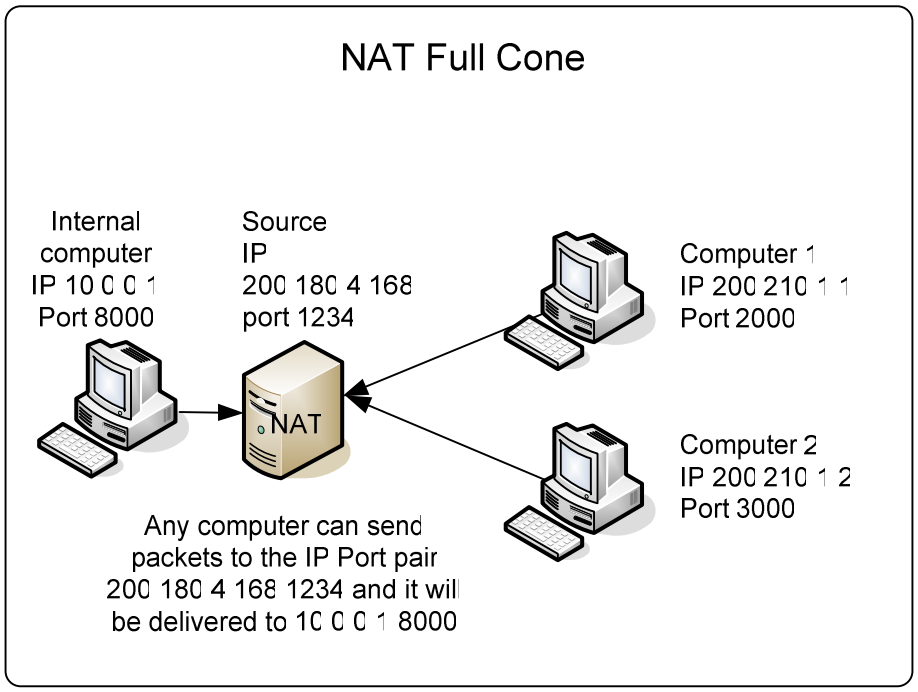


Figure 7.12 NAT Full Cone

7.5.2 Restricted Cone

In the restricted cone scenario, the external ip:port pair is opened only when the internal computer sends data to an outside address. However, the restricted cone NAT blocks any incoming packets from a different address. In other words, the internal computer has to send data to an external computer before it can send data back.

7.5.3 Port Restricted Cone

The port restricted cone firewall is almost identical to the restricted cone. The only difference is that, now, the incoming packet has to come from exactly the same IP and port of the sent packet.

7.5.4 Symmetric

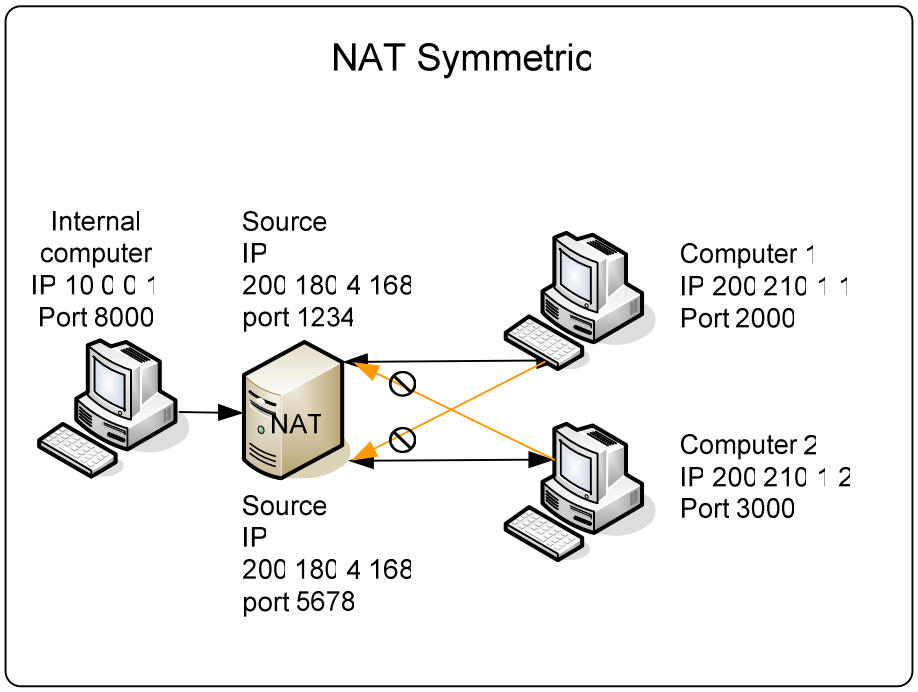


Figure 7.13 NAT Symmetric

The last type of Nat is called symmetric. It is different from the first three in that a specific mapping is done to each external address. Only specific external addresses are allowed to come back by the NAT mapping. It is not possible to predict the external IP:port pair that will be used by the NAT device. With the other three types of NAT, it was possible to use an external server to discover the external IP address to use for communication. With symmetric Nat, even if you can connect to an external server, the discovered address cannot be used for any other device except for this server.

7.5.5 NAT firewall table

	Need to send data before receiving	**It is possible to determine the IP:port pair for returning packets**	**It restricts the incoming packets to the destination IP:port**
Full Cone	**No**	**Yes**	**No**
Restricted Cone	**Yes**	**Yes**	**Only IP**
Port Restricted Cone	**Yes**	**Yes**	**Yes**
Symmetric	**Yes**	**No**	**Yes**

7.5.6 SIP signaling and RTP over NAT

Some of the biggest issues in NAT traversal are that you have to solve two problems. The first one is SIP signaling and the second one is audio (RTP). Most problems of one-way audio are NAT related.

An interesting thing about SIP is that when an UAC sends a packet, it embeds the IP address in the SIP "Contact" header field. Usually this is an internal (RFC1918) address and responses to this packet cannot be routed over the internet back to the UAC. When you put the statement "nat=yes" in the sip.conf file you are telling Asterisk to ignore the address contained in the "Contact" header field of the SIP header and use the source IP address and port in the packet's IP header.

```
nat=yes
```

Now, it is necessary to keep the NAT mapping open. If NAT times out, Asterisk could not send an invite to the UAC. The UAC could send calls but could not receive. We can use the statement below to keep NAT open.

```
qualify=yes
```

Qualify will send a SIP packet using the OPTIONS method regularly. This will help to keep NAT open.

Even with SIP signaling resolved, now we have a challenge to pass RTP from one phone to another. If the user's NAT is of the symmetric type, it is not possible to send packets from one UAC to another directly. In this case, we have to force the RTP thru Asterisk using:

Qualify sends an OPTION each 60 seconds and every 10th second when the host is not reachable. You can use "sip show peers" to see the latency for the peers.

```
canreinvite=no
```

These configurations are appropriate for most cases. However, it is possible to optimize the traffic using advanced techniques like STUN (Simple Traversal of UDP over Nat), which is useful with full cone, restricted cone, port restricted cone, and ALG (Application Layer Gateway). Using these techniques, you do not need to do anything in Asterisk for Nat traversal. Sorry to say but, most firewalls today, even home DSL/Cable routers, are symmetric, making STUN unusable. ALG could solve the problem but it is not supported, not implemented, or buggy in most cases.

7.5.7 Asterisk behind NAT

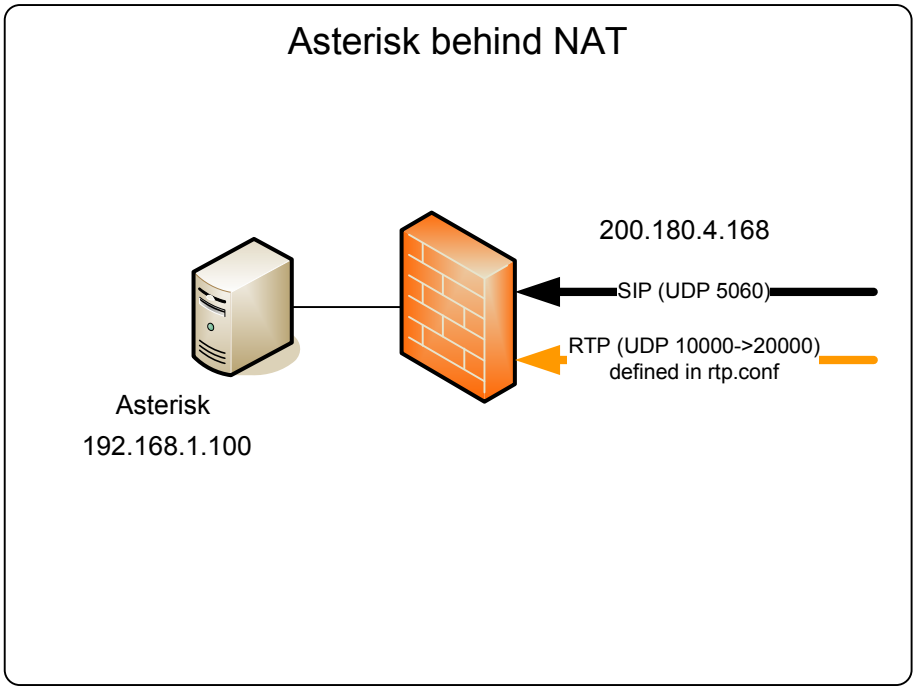


Figure 7.14 Asterisk behind NAT

All scenarios above assume that the Asterisk server have an external (valid) Internet address. Sometimes the Asterisk server is implemented behind a Firewall with NAT. In this case, it is necessary to do some extra configurations.

Step 1: Configure the firewall to redirect statically the UDP port 5060 to the Asterisk server.

Step 2: Configure the firewall to redirect statically the UDP ports from 10000 to 20000. If you want to restrict the number of opened ports, you can edit the rtp.conf file to change the rtp port range. Another way is to use an intelligent firewall that supports the SIP protocol to open dynamically the RTP ports.

```
; RTP Configuration
;
[general]
;
; RTP start and RTP end configure start and end addresses
;
rtpstart=10000
rtpend=20000
```

Step 3: Configure Asterisk to include the external address in the header fields of the SIP packets including SDP (Session Description Protocol). You accomplish this by adding two statements to the sip.conf file.

```
externip=200.180.4.168   ;External IP address
localnet=192.168.1.0/255.255.255.0   ;Internal Network Address
nat=yes
```

The first parameter (EXTERNIP) tells Asterisk to include the external IP address inside the SIP headers for external destinations. The second parameter (LOCALNET) allows Asterisk to differentiate external and internal addresses.

Optionally you can use "externhost" if you use a Dynamic DNS with a DHCP address on the server.

7.6 SIP LIMITATIONS

Asterisk does not support SIP calls over TCP or TLS transport protocols. It uses only UDP.

Asterisk uses the incoming RTP flow to synchronize the outgoing flow. If the incoming flow is interrupted (silence suppression) then music-on-hold will be cut. In other words, you cannot use silence suppression in phones or providers with Asterisk.

7.7 SIP DIAL STRINGS

You can call a SIP destination with different dial strings. You can use:

```
SIP/peer                        ; Need to have a defined peer in sip.conf
SIP/flavio@voffice.com.br       ; By the URI
SIP/[exten@]peer[:portno]
SIP/[user:password@domain/extension
```

Examples:

```
exten=>s,1,Dial(SIP/ipphone)
exten=>s,1,Dial(SIP/info@voffice.com.br)
exten=>s,1,Dial(SIP/192.168.1.8:5060,20)
exten=>s,1,Dial(SIP/8500@sip.com:9876)
```

7.8 SIP CLI COMMANDS

You can show all available SIP console commands using:

```
CLI>help sip
```

7.9 QUESTIONS

1. SIP is a protocol similar to _____ and _____.

   a) IAX
   b) HTTP
   c) H323
   d) SMTP

2. SIP can have sessions of type: (mark all that apply)

   a) Voice
   b) e-mail
   c) Video
   d) Chat
   e) Games

3. Are SIP components: (mark all that apply)

   a) User Agent
   b) Media gateway
   c) PSTN Server
   d) Proxy Server
   e) Registrar Server

4. Before a phone can receive calls, it needs to _____.

5. A SIP server can operate in the PROXY or REDIRECT mode. The difference between them is that in the Proxy mode, all signaling pass by the SIP proxy. In the redirect mode, after discovering the location, the clients signal between themselves.

   a) True
   b) False

6. In proxy mode, the media flow goes through the SIP Proxy.

   a) True
   b) False

7. Asterisk is a SIP Proxy.

a) True
b) False

8. The canreinvite=yes/no option is fundamental. It will define if the media pass inside Asterisk or goes directly from one client to another. It has a major impact in Asterisk scalability.

a) True
b) False

9. Asterisk supports silence suppression in the SIP channels.

a) True
b) False

10. The hardest NAT type to traverse is:

a) Full Cone
b) Restricted Cone
c) Port Restricted Cone
d) Symmetric

Page intentionally left empty

Chapter 8

Introduction to the dial plan

The dial plan is the most important Asterisk configuration. All calls are handled by the dial plan. It is responsible for the PBX operation and behavior. The dial plan controls everything Asterisk does when you dial a number or name. It is controlled by a file named /etc/asterisk/extensions.conf and you will learn how to create this file structure in this chapter.

8.1 OBJECTIVES

By the end of this chapter, you should be able to:

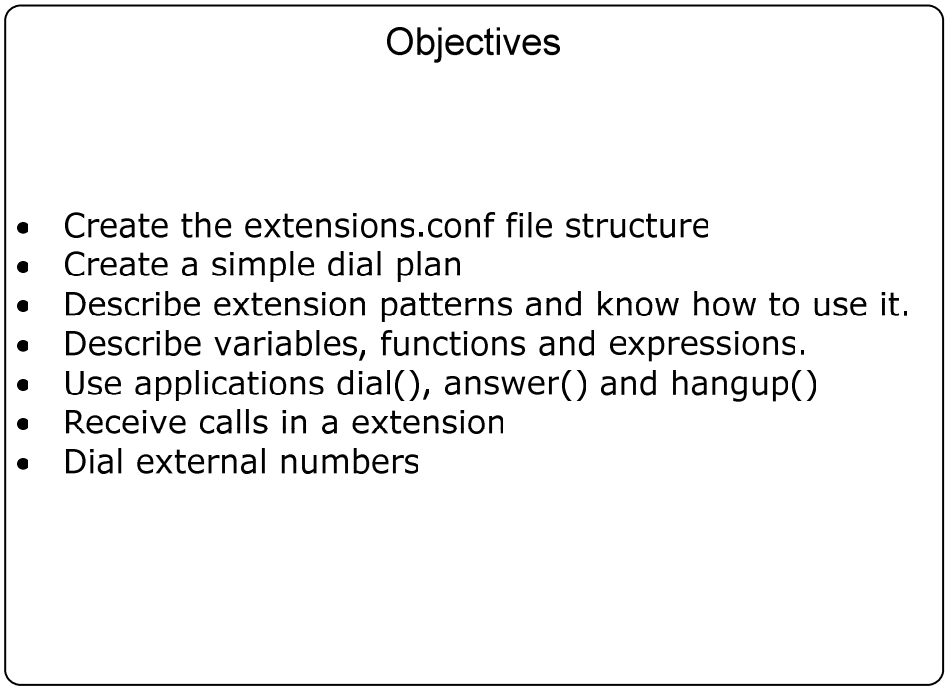


Figure 8.1 Objectives

8.2 EXTENSIONS.CONF FILE STRUCTURE

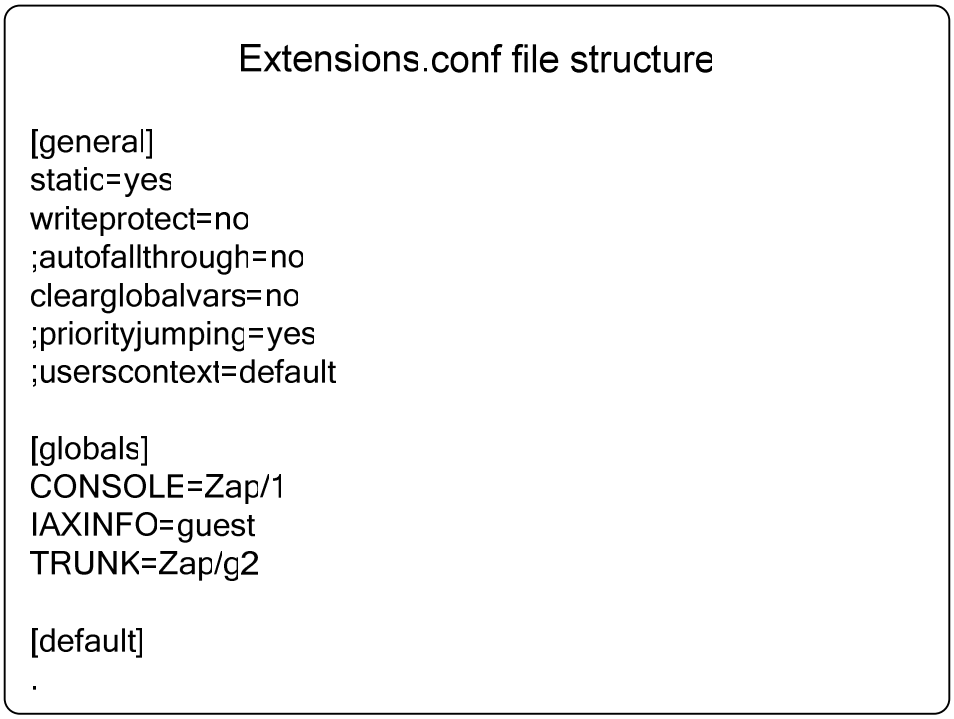


Figure 8.2 Extensions.conf file structure

The extensions.conf file is separated in sections. The first is the [general] section followed by the [globals] section. The beginning of each section starts with its name definition (i.e. [default]) and finishes when another section is created.

8.2.1 [general] section

General section sits at the top of the file. Before starting to configure the dial plan is interesting to know the general options that controls some dial plan behaviors. These options are:

- **static and write protect:** If static=yes and writeprotect=no, you can save dialplan by CLI command 'save dialplan' too.

Warning: If you issue an "save dialplan" command from the CLI you will loose any remarks and comments in the file.

-
- **autofallthrough:** If autofallthrough is set, then if an extension runs out of things to do, it will terminate the call with BUSY, CONGESTION, or HANGUP depending on Asterisk's best guess. This is the default. If

autofallthrough is not set, then if an extension runs out of things to do, Asterisk will wait for a new extension to be dialed. In version 1.4 the default is yes.

- **clearglobalvars**: If clearglobalvars is set, global variables will be cleared and reparsed into an dialplan reload, or Asterisk reload. If clearglobalvars is not set, then global variables will persist through reloads, and even if deleted from the extensions.conf or one of its included files, they will remain set to the previous value.
  -
- **priority jumping:** If priorityjumping is set to 'yes', then applications that support 'jumping' to a different priority based on the result of their operations will do so (this is backwards compatible behavior with pre-1.2 releases of Asterisk). Individual applications can also be requested to do this by passing a 'j' option into their arguments. In version 1.4, this option defaults to 'no'.

8.2.2 [globals] Section

In the [globals] section you will define global variables and their initial values. You can access the variable in the dialplan using ${GLOBAL(variable)}. You can even access variables defined in the linux/unix environment using ${ENV(variable).

Global variables are not case sensitive. A few examples could be:

```
INCOMING>Zap/8&Zap/9
RINGTIME=>3
```

In the example below, you can set and test a global variable in the dialplan.

```
exten=9000,1,set(GLOBAL(RINGTIME)=4)
exten=9000,n,Noop(${GLOBAL(RINGTIME)})
exten=9000,n,hangup()
```

8.3 CONTEXTS

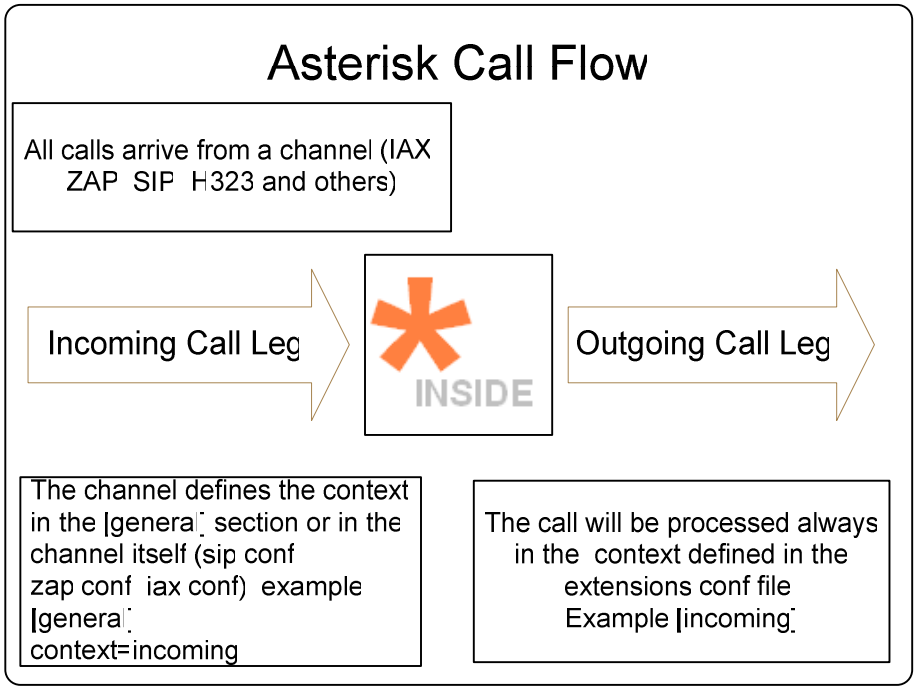


Figure 8.3 Asterisk call flow

Context is the named partition of the dial plan. In chapter three, we have learned the basic concepts of context, extensions and priorities. After the sections [general] and [globals], the dial plan is a set of contexts, each context has several extensions, each extension has several priorities, and each priority calls an application with several arguments. Let us revise the concepts introduced in chapter three.

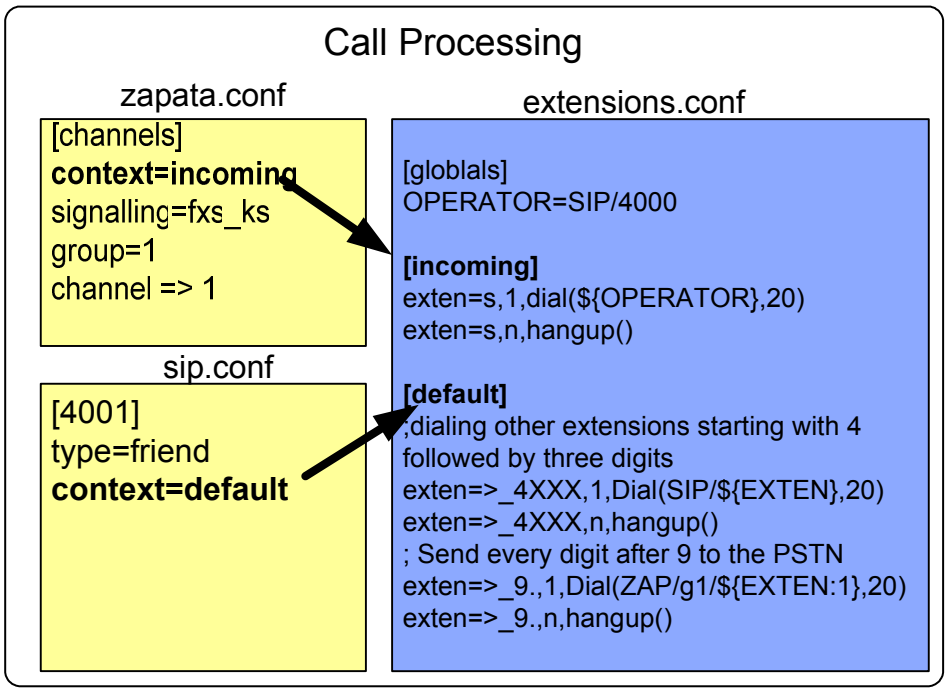


Figure 8.4 Basic Dial Plan

You can build a simple dial plan to reach other phones and the PSTN. However, Asterisk is much more powerful than that. Our objective is to teach you more details of what is possible in the dial plan.

8.4 EXTENSIONS

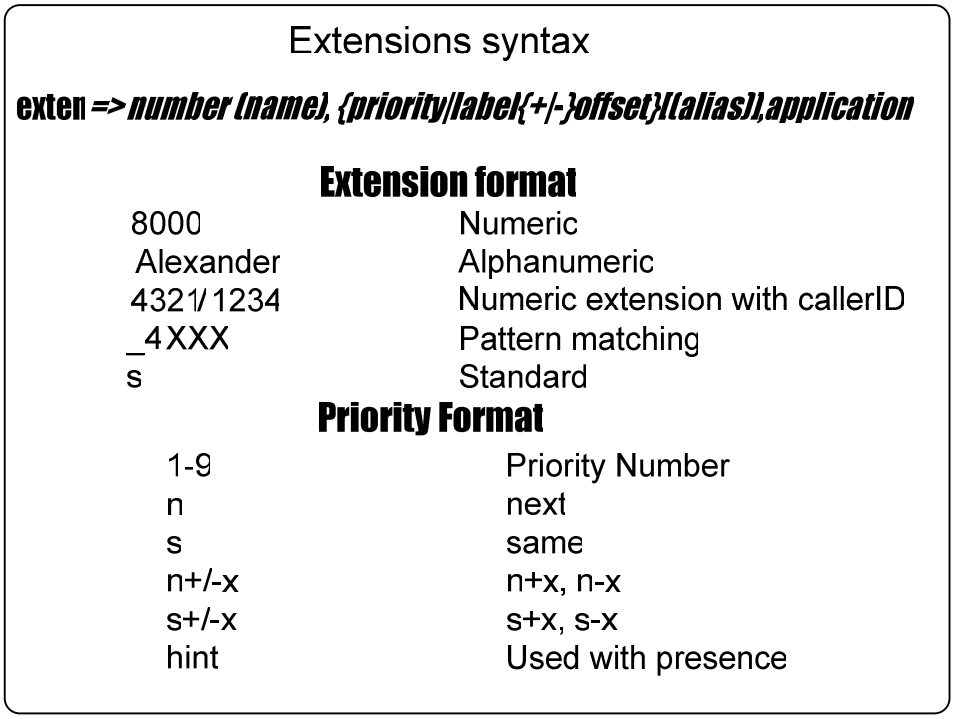


Figure 8.5 Extensions and priorities

Different from the traditional PBX, where extensions are associated with phones, interfaces, menus and so on, in Asterisk an extension is a list of commands to be processed when a specific extension number or name is triggered. The commands are processed in priority order.

An extension can be literal, standard, or special. A standard extension includes only numbers or names and the characters "*" and "#". 12#89* is a valid literal extension. Names can be used for extension matching too. Extensions are case sensitive. However, you cannot create two extensions with the same name with different cases.

When an extension is dialed, the command with the first, priority is executed followed by the command with priority 2 and so on. This happens until the call is disconnected or some command returns the number one indicating failure. What Asterisk does when the last priority is executed is regulated by the parameter autofallthrough. See the [general] section in this chapter.

Example:

```
exten=>123,1,Answer
exten=>123,2,Playback(tt-weasels)
exten=>123,4,Hangup
```

Above you find the list of instructions to be processed when extension "123" is dialed. The first priority is to answer the channel (necessary when the channel is in the ringing state, i.e. FXO channels). The second priority is to playback an audio file called "tt-weasels". The third priority hangs up the channel.

Another option is to handle the call by the CallerID. You can use the "/" character to specify the CallerID to be processed.

Examples:

```
exten=>123/100,1,Answer()
exten=>123/100,2,Playback(tt-weasels)
exten=>123/100,4,Hangup()
```

This example will trigger extension "123" and execute the following options only if CallerID is 100.

This can also be done by using pattern, as described below:

```
exten=>1234/_256NXXXXXX,1,Answer()
```

hint: maps an extension to a channel. It is used to monitor channel state. It is used in conjunction with presence. The phone has to support it.

8.4.1 Pattern Matching

You can use patterns in the dial plan. They are very useful to reduce the dial plan size. Any pattern starts with the "_" character. The following characters can be used to define a pattern.

Pattern Matching

_ (Underscore)	Start of a match
. (dot)	Matches one ore more characters
! (exclamation)	Matches zero ore more characters
[123-7]	Matches any digit in the brackets (1,2,3to 7)
X	Any digit from 0-9
Z	Any digit from 1-9
N	Any digit from 2-9

Examples

Extension	Description
_61XX	Sao Paulo´s Office (6100 - 6199)
_63XX	New York Office (6300-6399)
_62XX	San Francisco Office (6200 - 6299)
_7[1-3]XX	Bangalore Office (7100-7399)
_7[04-9]XX	Beijing Office (7000-7099, 7400-7499)
_9.	Any number staring with 9
_9XXXXXXX	Any number with 8 digits starting with 9

Figure 8.6 Pattern Matching

8.4.2 Standard extensions

Asterisk uses some extension names as standard extensions.

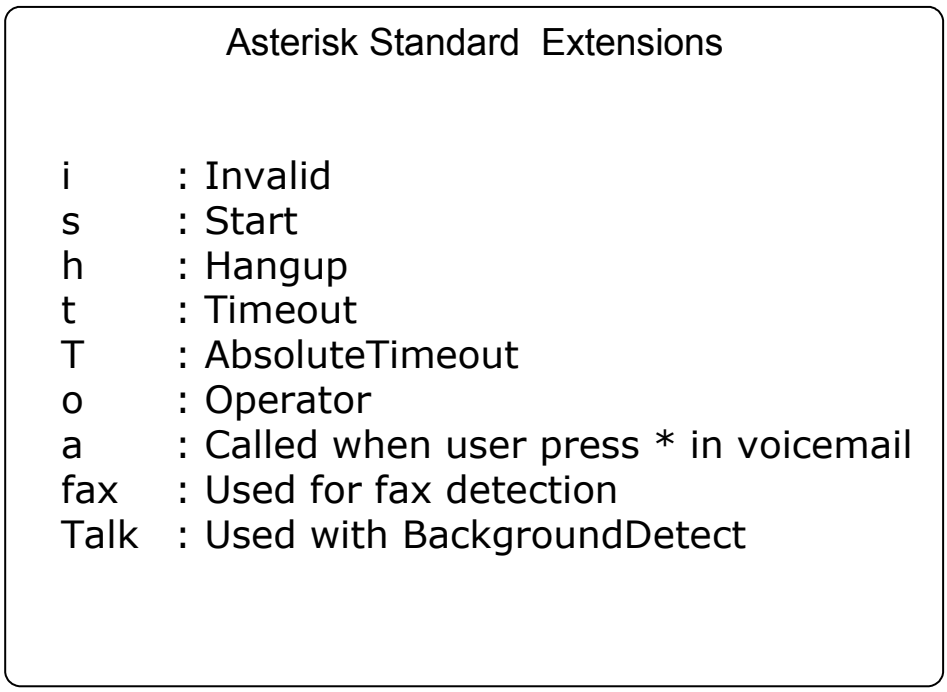


Figure 8.7 Asterisk standard extensions

Description:

s: Start. It is used to handle a call when there is no dialed number. It is useful for FXO trunks and in menu processing.

t: Timeout. It is used when calls stay inactive after a prompt had been played. It is used to hang-up an inactive line as well.

T: AbsoluteTimeout. If you establish a call limit using the AbsoluteTimeout() function. When the call exceeds the limit defined, it will be sent to T extension.

h: Hangup. It is called after the user disconnects the call

i: Invalid. It is triggered when you call an inexistent extension in the context. Using these extensions can affect the content of CDR records. Specifically, the dst that does not contain the number dialed.

o: Operator. It is used to go to operator when the user presses "0" in voicemail.

8.5 VARIABLES

In the Asterisk PBX, variables can be global, channel-specific, and environment-specific. You can use the NoOP() application to see the content of a variable in the console.

It can use a global variable or a channel-specific variable as applications arguments. A variable can be referenced as below where "varname" is the name of the variable.

```
${varname}
```

A variable name can be an alphanumeric string starting with a letter. Global variable names are not case sensitive. However system variables (Asterisk defined are channel defined) are case sensitive. The variable ${EXTEN} is different from ${exten}.

8.5.1 Global variables

Global variables can be configured in the [global] section in the extensions.conf file or using the application:

```
set(Global(variable)=content).
```

8.5.2 Channel variables

Channel variables are configure dusing the "Set()" application. Each channel receives its own variable space. There is no chance of collisions between variables from different channels. A channel specific variable is destroyed when the channel hangs up.

Some of the most used variables are:

${EXTEN} Extension dialed
${CONTEXT} Current context
${CALLERID(name)}
${CALLERID(num)}
${CALLERID(all)} Current callerID
${PRIORITY} Current priority

There are other channel variables. They are all uppercase. You can see the content of several variables using the dumpchan() application. Below is a simple excerpt of dump channel variables.

```
exten=9001,1,dumnpchan()
exten=9001,n,echo()
exten=9001,n,hangup()
```

Dumpchan output:

```
Dumping Info For Channel: SIP/4400-08191828:
================================================================================
Info:
Name=               SIP/4400-08191828
Type=               SIP
UniqueID=           1161186526.0
CallerID=           4400
CallerIDName=       laptop
DNIDDigits=         9001
RDNIS=              (N/A)
State=              Ring (4)
Rings=              0
NativeFormat=       0x4 (ulaw)
WriteFormat=        0x4 (ulaw)
ReadFormat=         0x4 (ulaw)
1stFileDescriptor=  16
Framesin=           0
Framesout=          0
-TimetoHangup=       0
ElapsedTime=        0h0m0s
Context=            default
Extension=          9001
Priority=           1
CallGroup=
PickupGroup=
Application=        DumpChan
Data=               (Empty)
Blocking_in=        (Not Blocking)

Variables:
SIPCALLID=500CEBC0-9483-4CED-B1E4-16D953655CFC@192.168.1.116
SIPUSERAGENT=SJphone/1.61.312b (SJ Labs)
SIPDOMAIN=192.168.1.133
SIPURI=sip:4400@192.168.1.116
```

Tip: A complete list can be found at:

http://www.voip-info.org/wiki/view/Asterisk+Detailed+Variable+List

8.5.3 Environment variables

Environment variables can be used to access variables defined in the operating system. You can set environment variables using the function ENV()

```
${ENV(LANG)}
Set(ENV(LANG))=en_US
```

8.5.4 Application specific variables

Some applications use variables for data input and output. You can set variables before calling the application or retrieve the variable after the application execution.

Example:

The Dial application returns the following variables:

- ${DIALEDTIME} ->This is the time from dialing a channel until it is disconnected.
- ${ANSWEREDTIME} -> This is the amount of time for actual call
- ${DIALSTATUS} This is the status of the call:
  - CHANUNAVAIL
  - CONGESTION
  - NOANSWER
  - BUSY
  - ANSWER
  - CANCEL
  - DONTCALL
  - TORTURE
- ${CAUSECODE} -> Error message for the call

8.5.5 Macro specific variables

Some additional channel variables are available within a macro context:

- ${ARG1}: First macro argument
- ${ARG2}: Second macro argument and so on.
- ${MACRO_CONTEXT}: Context where the macro was called
- ${MACRO_EXTEN}: Extension that triggers the macro
- ${MACRO_OFFSET}: Set by a macro to influence priority after exiting.
- ${MACRO_PRIORITY}: Priority where the macro was triggered

8.6 EXPRESSIONS

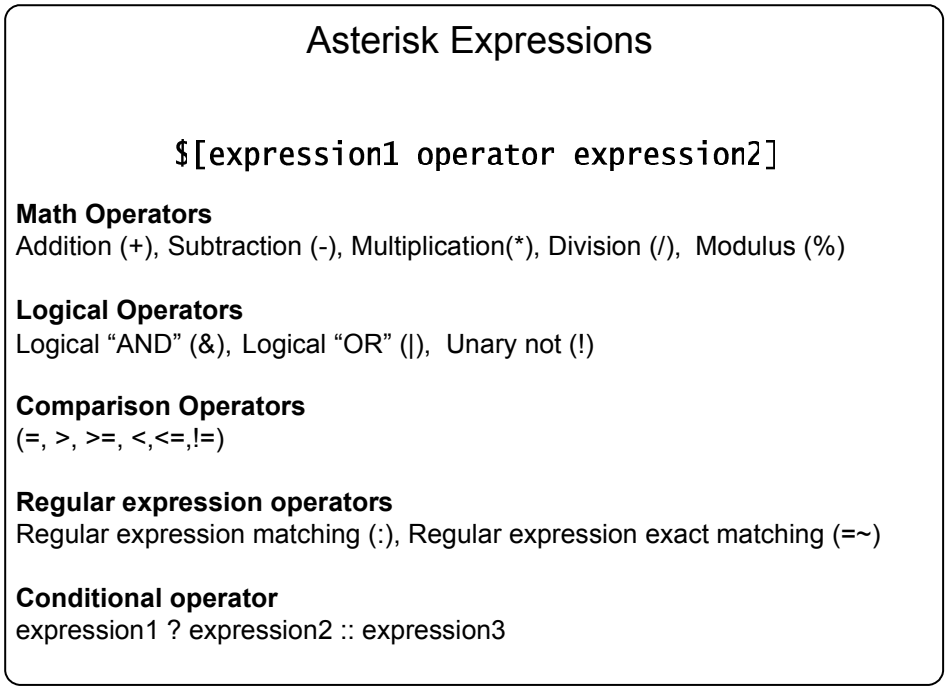


Figure 8.5 Asterisk expressions

Expressions can be very useful in the dial plan. They are used to manipulate strings and perform math and logical operations. The expression syntax is defined below:

```
$[expression1 operator expression2]
```

Let's suppose that we have a variable called "I" and we want to add 100 to the variable.

```
$[${I}+100]
```

When Asterisk finds an expression in the dial plan, it changes the whole expression by the resulting value.

8.6.1 Operators

The following operators can be used to build expressions. It is important to observe operator precedence.

1 – Parentheses "()"
2 – Unary operators "! -"
3 – Regular expression ": =~

4 – Multiplicative operators "* / %"
5 – Additive operators "+ -"
6 – Comparison operators
7 – Logical operators
8 – Conditional operators

Math Operators

- Addition (+)
- Subtraction (-)
- Multiplication(*)
- Division (/)
- Modulus (%)

Logical Operators

- Logical "AND" (&)
- Logical "OR" (|)
- Logical Unary Complement (!)

Regular expression operators

- Regular expression matching (:)
- Regular expression exact matching (=~)

A regular expression is a special text string to describe a search pattern. You may think of regular expressions as wildcards. Regular expressions are used to match a string to a pattern to check matching. If the match succeeds and the regular expression contains at least one match, the first match is returned. Otherwise, the result is the number of characters matched.

Comparison operators

The result of a comparison is 1 if the relation is true or 0 if false.

- **=** equal
- **!=** not equal
- **<** less than
- **>** greater than
- **<=** less or equal than
- **>=** greater or equal than

8.6.2 LAB. Evaluate the following expressions:

Put these expressions in your dial plan and use the NoOP() application to evaluate the expressions. Dial # 9002 and see results in the Asterisk console. Use verbose 15 to show the results.

```
exten=9002,1,set(NAME="FLAVIO")                 ;Set NAME=FLAVIO
exten=9002,n,set(I=4)
exten=9002,n,set(URI="40001@asteriskguide.com")
exten=9002,n,NoOP(${NAME})
exten=9002,n,NoOP(${I})
exten=9002,n,NoOP($[${I}+${I}])
exten=9002,n,NoOP($[${I}=4])
exten=9002,n,NoOP($[${I}=4 & ${NAME}=FLAVIO])
exten=9002,n,NoOP($[${URI} =~ "4[0-9][0-9][0-9][0-9]@."])
exten=9002,n,NoOP($[${I}=4?"MATCH"::"DO NOT MATCH"])
exten=9002,n,hangup
```

8.7 FUNCTIONS

After version 1.2, some applications are being replaced by functions. They allow the processing of some variables in a more advanced way than only expressions. You can see the full list of functions issuing the console command below.

```
CLI>core show functions
```

8.7.1 String length

${LEN(string)} returns the string length

```
Example:
exten=>100,1,Set(Fruit=pear)
exten=>100,2,NoOp(${LEN(Fruit)})
exten=>100,3,NoOp(${LEN(${Fruit})})
```

In the first operation, the system has to show 5 as the result (the number of letters in the word “fruit”). The second returns the number 4 (the number of letters in the word “pear”).

8.7.2 Substrings

```
${string:offset:length }
```

Returns the substring, starting from the positing defined by the “offset” parameter, with the string length defined in the “length” parameter.

If offset is negative, it starts from right to left, beginning at the end of the string.

If length is omitted or negative then it takes the whole string starting by the offset.

Examples:

```
${123456789:1}-returns 23456789
${123456789:-4}-returns 6789
${123456789:0:3}-returns 123
${123456789:2:3}-returns 345
${123456789:-4:3}-returns 678
```

```
exten=>_NXX.,1,Set(areacode=${EXTEN:0:3})
```

Takes the area code from the first three digits.

```
exten=>_516XXXXXXX,1,Dial(${EXTEN:3})
```

It takes all digits from the variable ${EXTEN}, except for the area code

8.7.3 String concatenation

To concatenate two strings, simply write them together.

```
${foo}${bar}
555${number}
${longdistanceprefix}555${number}
```

8.8 APPLICATIONS

To build a dial plan we need to understand the concept of applications. You will use applications to handle the channel in the dial plan. Applications are implemented in several modules. Available applications depend on modules. You can show all Asterisk applications using the console command:

```
CLI>core show applications
```

Alternatively, you can show details of a specific application using:

Example:

```
CLI>core show application dial
```

To build a simple dial plan you need to know a few applications. In the next chapter, we will discuss more advanced examples.

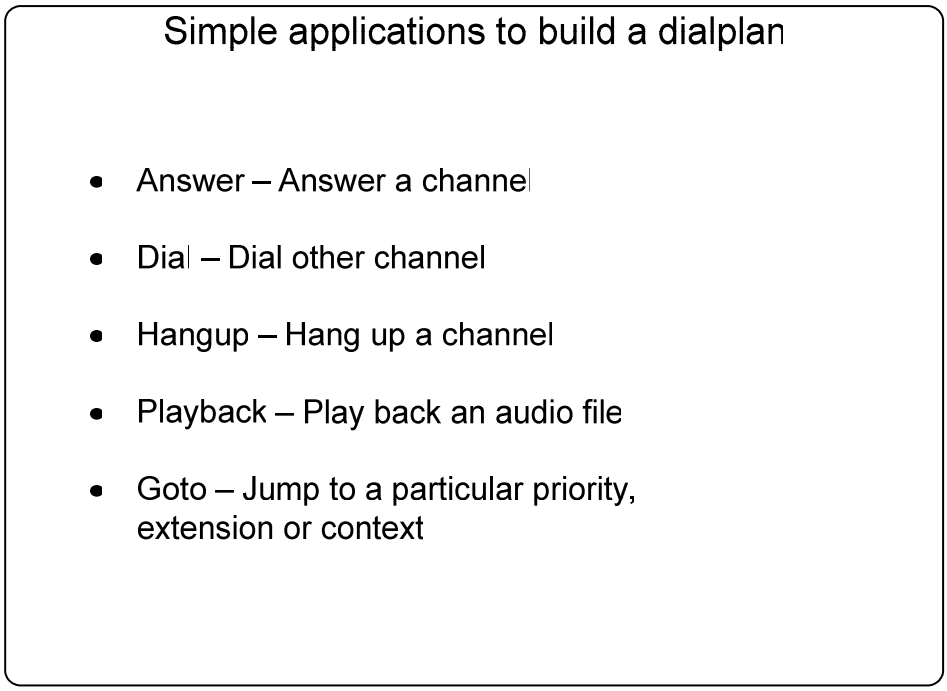


Figure 8.6 Simple applications to build a dialplan

We will use the above applications to create a simple dial plan for two basic PBXs.

8.8.1 Answer application

[Synopsis]
Answers a channel if ringing

[Description]

Answer([delay]): If the call has not been answered, the application will answer it. Otherwise, it has no effect on the call. If a delay is specified, Asterisk will wait this number of milliseconds specified in 'delay' before answering the call.

8.8.2 Dial application

The description below can be obtained issuing show application dial in the dial plan. For easy searching it is being reproduced below.

Dial()

Place a call and connect to the current channel

The syntax for the Dial application is shown below

```
;Dialing a channel
Dial(type/identifier,timeout,options, URL)
```

```
;Dialing to multiple channels
Dial(Technology/resource[&Tech2/resource2...][|timeout][|options][|URL]):
```

This application will place calls to one or more specified channels. As soon as one of the requested channels answers, the originating channel will be answered, if it has not already been answered. These two channels will then be active in a bridged call. All other channels that were requested will then be hung up.

Unless there is a timeout specified, the Dial application will wait indefinitely until one of the called channels answers, the user hangs up, or if all of the called channels are busy or unavailable. Dialplan executing will continue if no requested channels can be called, or if the timeout expires.

This application sets the following channel variables upon completion:

- **DIALEDTIME** - This is the time from dialing a channel until the time that it is disconnected.
- **ANSWEREDTIME** - This is the amount of time for actual call.
- **DIALSTATUS** - This is the status of the call:
  - CHANUNAVAIL
  - CONGESTION
  - NOANSWER
  - BUSY
  - ANSWER
  - CANCEL
  - DONTCALL
  - TORTURE

For the Privacy and Screening Modes, the DIALSTATUS variable will be set to DONTCALL if the called party chooses to send the calling party to the 'Go Away' script. The DIALSTATUS variable will be set to TORTURE if the called party wants to send the caller to the 'torture' script.

This application will report normal termination if the originating channel hangs up, or if the call is bridged and either of the parties in the bridge ends the call.

The optional URL will be sent to the called party if the channel supports it. If the OUTBOUND_GROUP variable is set, all peer channels created by this application will be put into that group (as in Set(GROUP()=...).

Options:

A(x)	Plays an announcement to the called party, using 'x' as the file.
C	Reset the CDR for this call.
d	Allow the calling user to dial a 1-digit extension while waiting for a call to be answered. Exit to that extension if it exists in the current context, or the context defined in the EXITCONTEXT variable, if it exists.
D([called][:calling])	Send the specified DTMF strings *after* the called party has answered, but before the call gets bridged. The 'called' DTMF string is sent to the called party, and the 'calling' DTMF string is sent to the calling party. Both parameters can be used alone.
f	Force the callerid of the *calling* channel to be set as the extension associated with the channel using a dialplan 'hint'. For example, some PSTNs do not allow CallerID to be set to anything other than the number assigned to the caller.
g	Proceed with dialplan execution at the current extension if the destination channel hangs up.
G(context^exten^pri)	If the call is answered, transfer the calling party to the specified priority and the called party to the specified priority+1.Optionally, an extension, or extension and context, may be specified. Otherwise, the current extension is used.
h	Allows the called party to hang-up by sending the '*' DTMF digit
H	Allow the calling party to hang-up by hitting the '*' DTMF digit.
i	Asterisk will ignore any forwarding requests it may receive on this dial attempt.

j	Jump to priority n+101 if all of the requested channels were busy.
L(x[:y][:z])	Limits the call to 'x' ms. Plays a warning when 'y' ms are left. Repeats the warning every 'z' ms. The following special variables can be used with this option: • **LIMIT_PLAYAUDIO_CALLER** yes|no (default yes) Play sounds to the caller. • **LIMIT_PLAYAUDIO_CALLEE** yes|no Play sounds to the callee. • **LIMIT_TIMEOUT_FILE** File to play when time is up. • **LIMIT_CONNECT_FILE** ->File to play when call begins. • **LIMIT_WARNING_FILE** ->File to play as warning if 'y' is defined. The default is to say the time remaining.
m([class])	Provides hold music to the calling party until a requested channel answers. A specific MusicOnHold class can be specified.
M(x[^arg])	Executes the Macro for the *called* channel before connecting to the calling channel. Arguments can be specified to the Macro using '^' as a delimiter. The Macro can set the variable MACRO_RESULT to specify the following actions after the Macro is finished executing. • **ABORT** Hangs-up both legs of the call. • **CONGESTION** Behaves as if line congestion was encountered. • **BUSY** Behaves as if a busy signal was encountered. This will also have the application jump to priority n+101 if the 'j' option is set. • **CONTINUE** Hangs-up the called party and allows the calling party to continue dialplan execution at the next priority. • **GOTO**:<context>^<exten>^<priority> Transfers the call to the specified priority. Optionally, an extension, or extension and priority, can be specified. In addition, PBX services are not run on the peer (called) channel, so you will not be able to set timeouts via the TIMEOUT() function in

	this macro.
n	This option is a modifier for the screen/privacy mode. It specifies that no introductions be saved in the priv-callerintros directory.
N	This option is a modifier for the screen/privacy mode. It specifies that if callerID is present, do not screen the call.
o	Specifies that the CallerID that was present on the *calling* channel be set as the CallerID on the *called* channel. This was the behavior of Asterisk 1.0 and earlier.
O([x])	"Operator Services" mode (Zaptel channel to Zaptel channel only; if specified on non-Zaptel interface, it will be ignored). When the destination answers (presumably an operator services station), the originator no longer has control of their line They may hang-up, but the switch will not release their line until the destination party hangs-up (the operator). Specified without an arg, or with 1 as an arg, the originator hung-up will cause the phone to ring back immediately. With 2 specified, when the "operator" flashes the trunk, it will ring his/her phone back.
p	This option enables screening mode. This is basically privacy mode without memory.
P([x])	Enables privacy mode. Use 'x' as the family/key in the database if it is provided. The current extension is used if a database family/key is not specified.
r	Indicates ringing to the calling party. Passes no audio to the calling party until the called channel has answered.
S(x)	Hangs-up the call after 'x' seconds *after* the called party has answered the call.
t	Allows the called party to transfer the calling party by sending the DTMF sequence defined in features.conf.
T	Allows the calling party to transfer the called party by sending the DTMF sequence defined in features.conf.
w	Allows the called party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in features.conf.

W	Allows the calling party to enable recording of the call by sending the DTMF sequence defined for one-touch recording in features.conf.
K	Allows the called party to enable parking of the call by sending the DTMF sequence defined for call parking in features.conf.
K	Allows the calling party to enable parking of the call by sending the DTMF sequence defined for call parking in features.conf.

Example:

```
exten=_4XXX,1,Dial(SIP/${EXTEN},20,tTm)
```

In the example above, the application will dial to the corresponding SIP channel. Both caller and called could transfer the call (Tt). Music on hold will be heard instead of ring back. If nobody answers in 20 seconds, the extension will go to the next priority.

8.8.3 Dialing between extensions

To enable dialing between extension, we could use the channel variable ${EXTEN} that means the dialed extension.

Example:

If the extension range is between 4000 to 4999 and all extensions use SIP, we could adopt the command below.

```
exten=_4XXX,1,Dial(SIP/${EXTEN})
```

8.8.4 The hang-up application

Hangs-up the calling channel

[Description]

Hangup([causecode]): This application will hang-up the calling channel. If a causecode is given, the channel's hangup cause will be set to the given value.

8.8.5 The Goto application

Jump to a particular priority, extension, or context

[Description]

Goto([[context|]extension|]priority): This application will cause the calling channel to continue dialplan execution at the specified priority. If no specific extension, or extension and context, are specified, then this application will jump to the specified priority of the current extension. If the attempt to jump to another location in the dialplan is not successful, then the channel will continue at the next priority of the current extension.

8.9 BUILDING A DIALPLAN

To build a simple dial plan you need to treat all incoming and outgoing calls. You do this by creating contexts and extensions. In this section, we will show you how to build the most common extensions.

8.9.1 Dialing to an external destination

To dial an external destination you could precede the number dialed with a route. In North America, it is usual to use 9 followed by the number to be dialed externally. If you are using an analog or digital channel to the PSTN the command should look like below.

```
exten=_9NXXXXXX,1,Dial(Zap/1/${EXTEN:1},20,tT)
```

The above line will permit you to dial 9 and the desired number. In the way it is been done you will use the first zapata channel (ZAP/1). If you have several lines and this one is busy, the call will not be completed. In other hand, you could use the line below to choose automatically the first available Zapata channel.

```
exten=_9NXXXXXX,1,Dial(Zap/g1/${EXTEN:1},20,tT)
```

The "g1" parameter will search the first available channel in the group, allowing the use of all channels.

Using the line below you could dial a long distance number.

```
exten=_91NXXNXXXXXX,1,Dial(Zap/g1/${EXTEN:1},20,tT)
```

8.9.2 Dialing 9 to get a PSTN line

If you do not have any restrictions to external dialing you could simplify and use the line below.

```
exten=9,1,Dial(Zap/g1,20,tT)
```

8.9.3 Receiving a call in the operator extension

In the example below, the operator extension is 4000. The PSTN line is connected to a FXO interface. In the Zapata.conf file the context specified is "[incoming]". Any call coming from the PSTN will be routed to the incoming context in the dialplan. This line does not have a DID (direct inward dialing); then, we will have to receive the call in the "s" extension.

```
[incoming]
exten = s,1,Answer()
exten = s,2,Dial(SIP/4000,40,tT)
exten = s,3,Hangup()
```

8.9.4 Receiving a call using DID (direct inward dialing)

If you have a digital line, you will receive the dialed extension. When this is the case, you don't need to forward to the operator. You can forward the call directly to the destination. Suppose your DID range is from 3028550 to 3028599 and the last four numbers are passed in the DID. The configuration should look like the one in the example below.

```
[incoming]
exten => _85[5-9]X,1,Answer()
exten => _85[5-9]X,2,Dial(SIP/${EXTEN},15,tT)
exten => _85[5-9]X,3,Hangup()
```

8.9.5 Playing several extensions simultaneously

You can set Asterisk to dial an extension and if it is not answered to dial several other extension simultaneously. You can do that as below.

```
exten => 0,1,Dial(Zap/1,15,tT)
exten => 0,2,Dial(Zap/1&Zap/2&Zap/3,15)
exten => 0,2,Hangup()
```

In this example when someone dials the operator, in the first place the channel ZAP/1 is tried. If nobody answers after 15 seconds (timeout) the channels ZAP/1, ZAP/2 and ZAP/3 will ring simultaneously for another 15 seconds.

8.9.6 Routing by the Caller ID

In this example you could give different treatments depending on the CallerID (this could be useful for ex-girlfriends, for example!).

```
exten => 8590/4832518888,1,Playback(Ihavemovedtochina)
exten => 8590,1,Dial(Zap/1,20)
exten => 8590,2,Voicemail(u8590)
exten => 8590,102,Voicemail(b8590)
```

In this example, we have added a special rule when, if the Caller ID is 4832518888, you playback a message on the previously recorded file "Ihavemovedtochine". Other calls are accepted as usual.

8.9.7 Using variables in the dial plan

Asterisk can use global and channel variables in the dial plan as arguments for certain applications. Look at the examples below.

```
[globals]
Flavio => Zap/1
Daniel => Zap/2&SIP/pingtel
Anna => Zap/3
Christian => Zap/4

[mainmenu]
exten => 1,1,Dial(${Daniel}&${Flavio})
exten => 2,1,Dial(${Anna}&${Christian})
exten => 3,1,Dial(${Anna}&${Flavio})
```

Using variables makes future changes easier. If you change the variable, all references are changed immediately.

8.10 Building a simple Dial Plan

This section will show you how to build a simple dial plan in the /etc/asterisk/extensions.conf for two very common scenarios.

8.10.1 PBX with 16 SIP extensions and 4 FXO trunks to PSTN.

Consider the following PBX example:

- Digium TDM400 installed and running
- zaptel.conf correctly configured
- zapata.conf correctly configured
- sip.conf correctly configured

- Channel group = 1 (group=1 in zapata.conf)
- FXO context = incoming (context=incoming in zapata.conf)
- SIP channels context = extensions
- Extensions range from 20 to 39
- Dial 0 to Operator
- Operator in the extension 20
-

```
;Operator call reception
[incoming]
exten=s,1,Answer()
exten=s,2,Dial(SIP/20,20,tT)
exten=s,3,Hangup()

[extensions]
;dialing other channels
exten=_[2-3]X,1,Dial(SIP/${EXTEN},20,tT)
exten=_[2-3]X,2,Hangup()

;PSTN call
exten=9,1,Dial(Zap/g1,20,tT)

;Calling the operator
exten=0,1,Dial(SIP/20,20,tT)
```

8.10.2 PBX with one T1 trunk and 50 sip phones

Consider the example below:

- TE110P installed and connected to a PSTN over a T1 trunk
- zaptel.conf correctly configured
- zapara.conf correctly configured
- sip.conf correctly configured
- Channel group =1 (group=1 in zapata.conf)
- T1 context = incoming (context=incoming in zapata.conf)
- SIP channels context= extensions
- Channel numbering from 4000 to 4049
- Operator in the 4000 extension or dialing 9
- External calls routed using DID

```
;Incoming calls
[incoming]
exten=_40[0-4]X,1,Answer()
exten=_40[0-4]X,2,Dial(SIP/${EXTEN},20,tT)
exten=_40[0-4]X,3,Hangup()

[extensions]
;dialing other extensions
exten=_40[0-4]X,1,Dial(SIP/${EXTEN},20,tT)
```

```
exten=_40[0-4]X,2,Hangup()

;external calls
exten=_9.,1,Dial(zap/g1/${EXTEN:1},20,tT)

;Calling the operator
exten=0,1,Dial(SIP/4000,20,tT)
```

8.11 ADDING SOME LOGIC TO YOUR DIAL PLAN

Several new features where introduced in version 1.2. In version 1.0, when an error occurs the logic jumped 101 priorities. This behavior is regulated by the "priorityjumping" parameter in the [general] section of the extensions.conf file. The parameter was introduced in version 1.2 and its default setting was 'yes'. Now, in version 1.4, the default value is 'no'.

The good news is that the dial plan is now more easily readable and can use labels to help the extensions logic. See the example below:

Adding some logic to the dialplan

Example in the 1.0 version

```
exter=2000,1,Dial(SIP/2000)
exter=2000,2,voicemail(u2000)
exter=2000,3,hangup()
exter=2000,102,voicemail(b2000)
exter=2000,103,hangup()
```

Example after 1.2 version.

```
exter=2000,1,Dial(SIP/2000)
exter=2000,n,goto(${DIALSTATUS})
exter=2000,n,hangup()
exter=2000,n(BUSY),voicemail(b2000)
exter=2000,n,hangup
exter=2000,n(NOANSWER),voicemail(u2000)
exter=2000,n,hangup
exter=2000,n(CHANUNAVAILABLE),hangup
exter=2000,n(CANCEL),hangup
exter=2000,n(CONGESTION),hangup
```

Figure 8.6 Adding some logic to the dial plan

You can observe the new constructions after version 1.2. Firstly, the "n" (next) priority appears. Secondly,, the goto jumps to the labels, instead of jumping 101 positions as in the old style. This makes it easier to insert new lines in the extension without been worried about the correct priority number. Thirdly, we could use the Goto application to jump according to the ${DIALSTATUS} variable value. Now you have a fine tuning treatment for the calls.

You can also use the variable ${HANGUPCAUSE} that allows capturing the PRI (q931) cause codes in ISDN channels.

8.12 SUMMARY

In this chapter, you have learned that a dial plan is the main configuration piece of Asterisk. It is composed of a combination of contexts, extensions and priorities. You have learned how to build a simple dial plan for digital as well as analog PBXs.

8.13 QUESTIONS

1. In the [general] section the default value to the option "writeprotect" is 'no'. If you issue a command "save dialplan" in Asterisk's CLI (mark all that apply).

a) Asterisk will overwrite extensions.conf with actual configuration
b) All comments are lost
c) An extensions.conf.bak will be created
d) The option static="yes" should be configure to save the dial plan

2. Usually, the global variables are written in uppercase and the channel variables with only the first letter in uppercase. This is not mandatory, but makes it easier to identify the variable's type

a) True
b) False

3. The 's' extension is used as the starting point in a context. Usually you use the 's" extension in the following cases:

a) In the incoming context for a call without DNIS (dialed number)
b) As a menu staring point called from the background application
c) In the incoming context with DNIS (dialed number)
d) As a starting point directed by the "goto' command

4. Write four situations where contexts could be used

_____

_____

_____

_____

5. To use a variable in the dial plan you should use the following format

a) $[varname]
b) {varname}
c) $(varname)
d) ${varname}

6. The Asterisk variable type could be (mark three)
.

a) Constants
b) Public variables
c) Environment variables
d) Global variables
e) Private variables
f) Channel variables

7. To obtain a string length you could use the function:_____.

8. To concatenate strings it is simply put them together:

${foo}${bar}
555${thenumber}

a) True
b) False

9. Suppose that you are configuring an analog PBX based on Asterisk. Write the necessary instructions to build a dial plan to receive calls in the operator (SIP/4000). If the operator extension is not answered before the timeout, it will have to ring channels SIP/4000 and SIP/4001 simultaneously.

10. Suppose that you are configuring a digital PBX based on Asterisk. Write the necessary instructions to allow the external dialing for long distance numbers.

Page intentionally left empty

Chapter 9

Dial Plan advanced features

Now that you have learned the basics of a dial plan, let us put some spice in the configuration by describing advanced techniques, new applications, and concepts.

9.1 OBJECTIVES

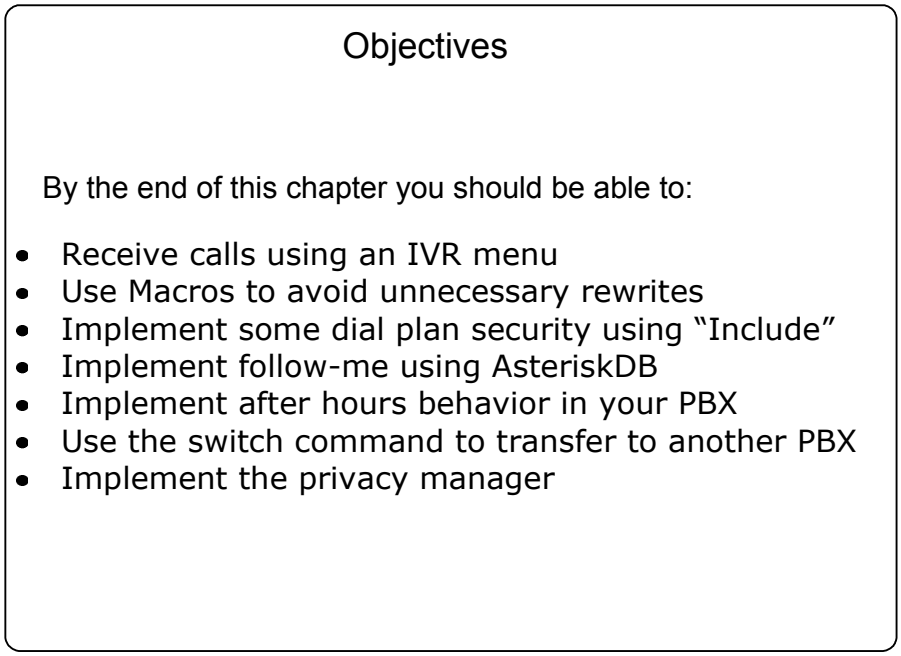


Figure 9.1 Objectives

9.2 RECEIVING CALLS USING AN IVR MENU.

In the last section you have received all calls by using DID or forwarding to the operator. Now you will learn how to implement an IVR menu and how to create an auto-attendant service. Before getting into the specific about how to do it, let's examine some new applications.

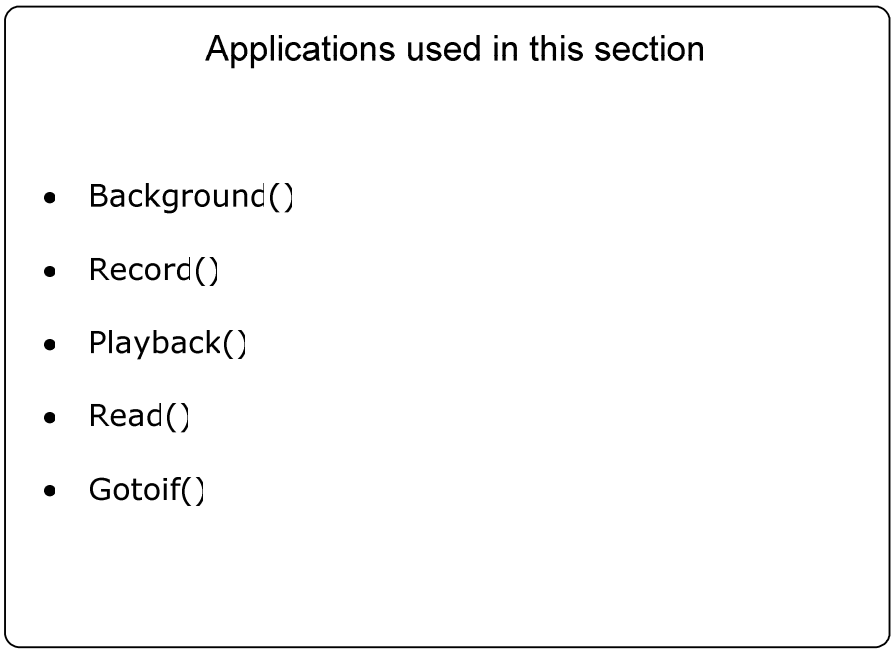


Figure 9.2 Applications used in this section

We simply put the output of the command show application below to make easier for readers. You can obtain these descriptions using "show application *applicationname*".

9.2.1 The Background() application

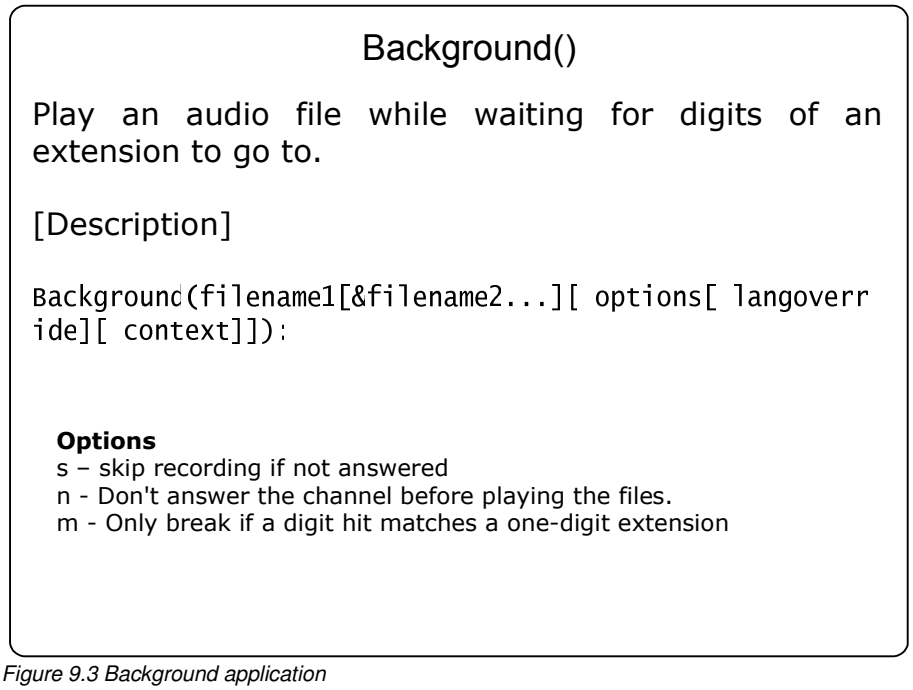


Figure 9.3 Background application

This application will play the given list of files while waiting for an extension to be dialed by the calling channel. To continue waiting for digits after this

application has finished playing files, the WaitExten application should be used. The 'langoverride' option explicitly specifies which language to attempt to use for the requested sound files. If a 'context' is specified, this is the dial plan context that this application will use when exiting to a dialed extension. If one of the requested sound files does not exist, call processing will be terminated.

Options:

- s - Causes the playback of the message to be skipped if the channel is not in the 'up' state (i.e. it hasn't been answered yet). If this happens, the application will return immediately.
- n - Don't answer the channel before playing the files.
- m - Only break if a digit hit matches a one-digit extension in the destination context.

9.2.2 The Record() application

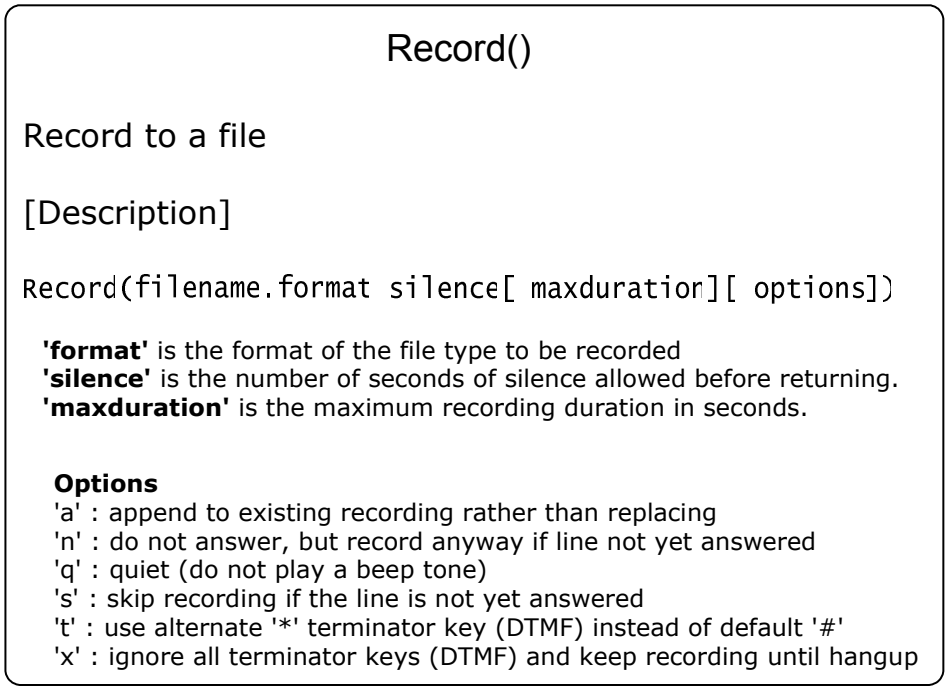


Figure 9.4 Record() application

This application records from the channel into a given filename. If the file exists, it will be overwritten.

- 'format' is the format of the file type to be recorded (wav, gsm, etc).
- 'silence' is the number of seconds of silence allowed before returning.
- 'maxduration' is the maximum recording duration in seconds. If it is missing or zero there is no maximum.
- 'options' may contain any of the following letters:

- 'a' : appends to existing recording rather than replacing
- 'n' : do not answer, but record anyway if line not yet answered
- 'q' : quiet (do not play a beep tone)
- 's' : skips recording if the line is not yet answered
- 't' : use alternate '*' terminator key (DTMF) instead of default '#'
- 'x' : ignore all terminator keys (DTMF) and keep recording until hung-up

If filename contains '%d', these characters will be replaced with a number incremented by one each time the file is recorded.

Use 'core show file formats' to see the available formats on your system.

The user can press '#' to terminate the recording and continue to the next priority.

If the user hangs-up during a recording, all data will be lost and the application will terminate.

9.2.3 The playback application

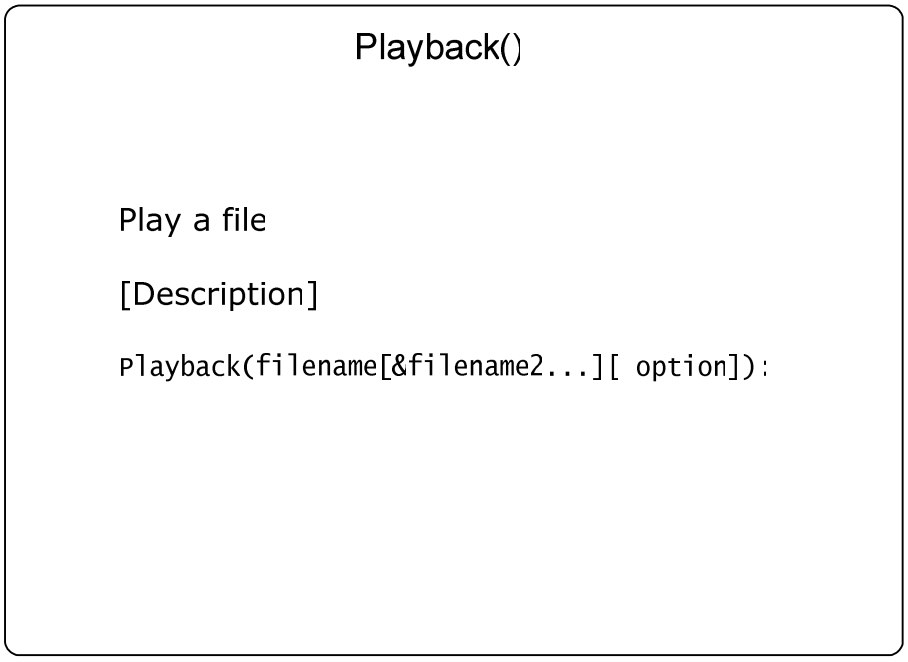


Figure 9.5 Playback() application

Plays back given filenames (do not put extension). Options may also be included following a pipe symbol. The 'skip' option causes the playback of the message to be skipped if the channel is not in the 'up' state (i.e. it hasn't been answered yet). If 'skip' is specified, the application will return immediately should the channel not be off hook. Otherwise, unless

'noanswer' is specified, the channel will be answered before the sound is played. Not all channels support playing messages while still on hook. If 'j' is specified, the application will jump to priority n+101 when the file does not exist, if present.

This application sets the following channel variable upon completion:

PLAYBACKSTATUS The status of the playback attempt as a text string, one of:

- SUCCESS
- FAILED

9.2.4 The read application

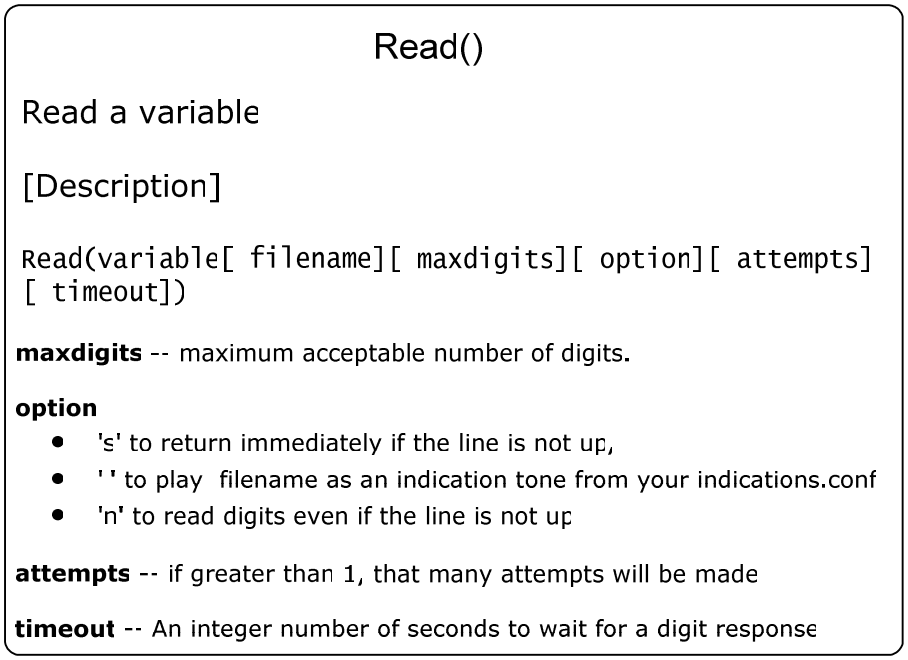


Figure 9.6 Read() application

This application reads a pre-determined number of string digits, a certain number of times, from the user into the given variable.

- filename -- file to play before reading digits or tone with option i
- maxdigits -- maximum acceptable number of digits. Stops reading after maxdigits have been entered (without requiring the user to press the '#' key). Defaults to 0 - no limit - wait for the user press the '#' key. Any value below 0 means the same. Max accepted value is 255.
- option -- options are 's' , 'i', 'n'
    - 's' to return immediately if the line is not up,
    - 'i' to play filename as an indication tone from your indications.conf

    - 'n' to read digits even if the line is not up
- attempts -- if greater than 1, these many attempts will be made in case no data is entered
- timeout -- An integer number of seconds to wait for a digit response. If greater than 0, that value will override the default timeout.

Read should disconnect if the function fails or errors out.

9.2.5 The gotoif application

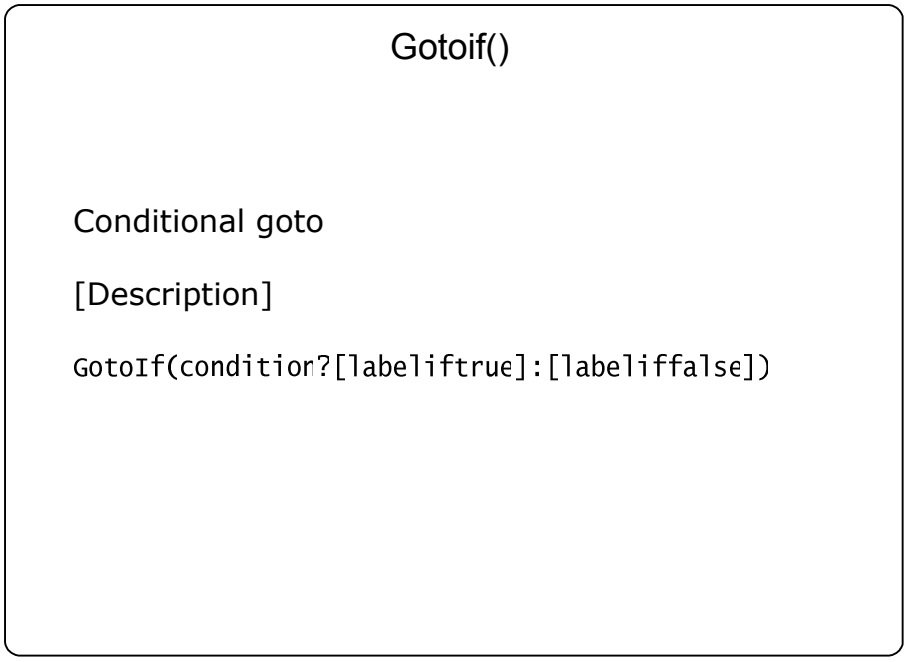


Figure 9.7 Gotoif() application

This application will cause the calling channel to jump to the specified location in the dial plan based on the evaluation of the given condition. The channel will continue at 'labeliftrue' if the condition is true, or 'labeliffalse' if the condition is false. The labels are specified with the same syntax as used within the Goto application. If the label chosen by the condition is omitted, no jump is performed, but execution continues with the next priority in the dial plan.

9.2.6 Important timeout settings

You can set two timeout parameters:

Set(TIMEOUT(digit)=*seconds*) – Interdigit timeout.

Set(TIMEOUT(response)=*seconds*) – Timeout for a full response.

9.2.7 Lab - Building an IVR menu step-by-step.

Let's create an IVR menu with the following functionality:

When dialed, the IVR will playback an audio file with the message "Welcome to the XYZ Corporation; press 1 for sales, 2 for tech support, 3 for training, or wait to speak to a representative. When digit one is pressed , the call is transferred to sales (SIP/4001); else, if digit 2 is pressed, the call is transferred to tech support (SIP/4002); if 3 is pressed, the call is transferred to training (SIP/4003). If no digits are pressed, the call is transferred to the operator (SIP/4000).

Step 1 – Record the prompts

Let's create an extension to record the prompts. To record a prompt, dial from a soft phone to 9005*filename.* When you hear the beep start recording, press "#" to stop recording, you will hear a beep and the system will playback the recorded prompt.

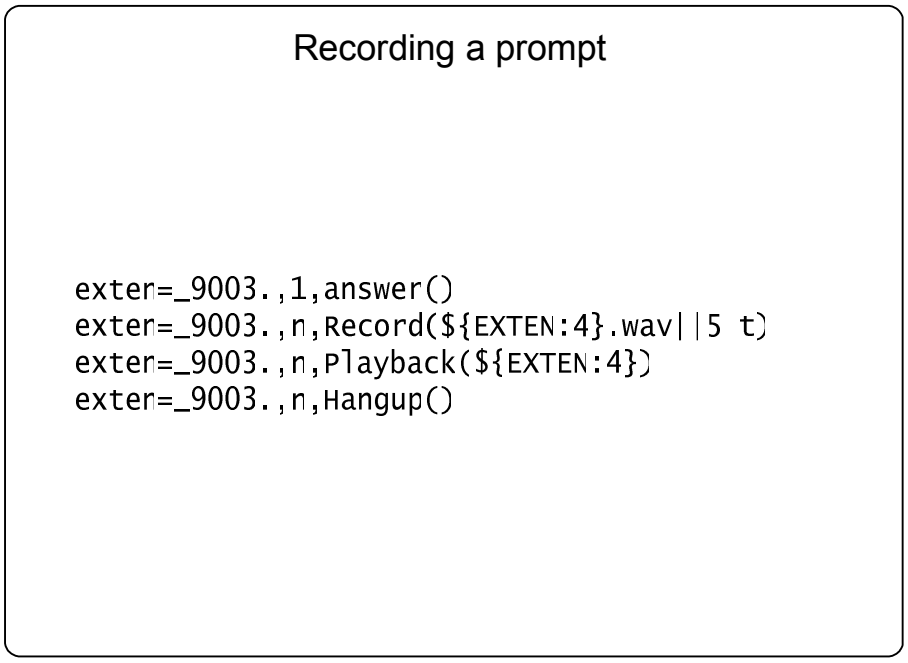


Figure 9.8 Recording a prompt()

Step 2 – Creating the menu logic

Creating the menu logic

```
exten=>9004,1,goto(menu,s,1)
[menu]
exten=>s,1,Background(mainmenu)
exten=>1,1,goto(sales,s,1)
exten=>2,1,goto(techsupport,s,1)
exten=>3,1,goto(training,s,1)
;handling an invalid digit
exten=>i,1,Dial(${OPERATOR})
;handling timeout
exten=>t,1,Dial(${OPERATOR})
[sales]
exten=>s,1,Dial(SIP/4001,20,t)
[techsupport]
exten=s,1,Dial(SIP/4002,20,t)
[training]
exten=s,1,Dial(SIP/4002,20,t)
```

Figure 9.9 Menu logic

When dialing the 9004 extension, processing will jump to the menu in the 's' extension, priority 1.

9.2.8 Matching as you dial

A company setup menu for receiving calls. The background application reads the current context and defines the maximum length for each number for any possible combination.

```
[incoming]
exten=>s,1,Background(welcome)
exten=>1,1,Dial(Zap/1)
exten=>2,1,Dial(Zap/2)
exten=>21,1,Dial(Zap/3)
exten=>22,1,Dial(Zap/4)
exten=>31,1,Dial(Zap/5)
exten=>32,1,Dial(Zap/6)
```

When you dial this company, the welcome message is played first. After that, Asterisk waits for any digit to be dialed.

Dialed number	Asterisk Action
1	Immediately calls Dial(Zap/1)
2	Waits for the timeout, then goes to Dial(Zap/2)

21	Immediately calls (Zap/3)
22	Immediately calls (Zap/4)
3	Waits for the timeout then disconnects
31	Immediately calls Dial(Zap/5)
32	Immediately calls Dial(Zap/6)
4	Immediately disconnects

It is important to avoid ambiguity in the menus. Everybody wants to be answered quickly. For this reason, you should not use numbers 2, 21, and 22.

9.2.9 Lab - Using the Read() application

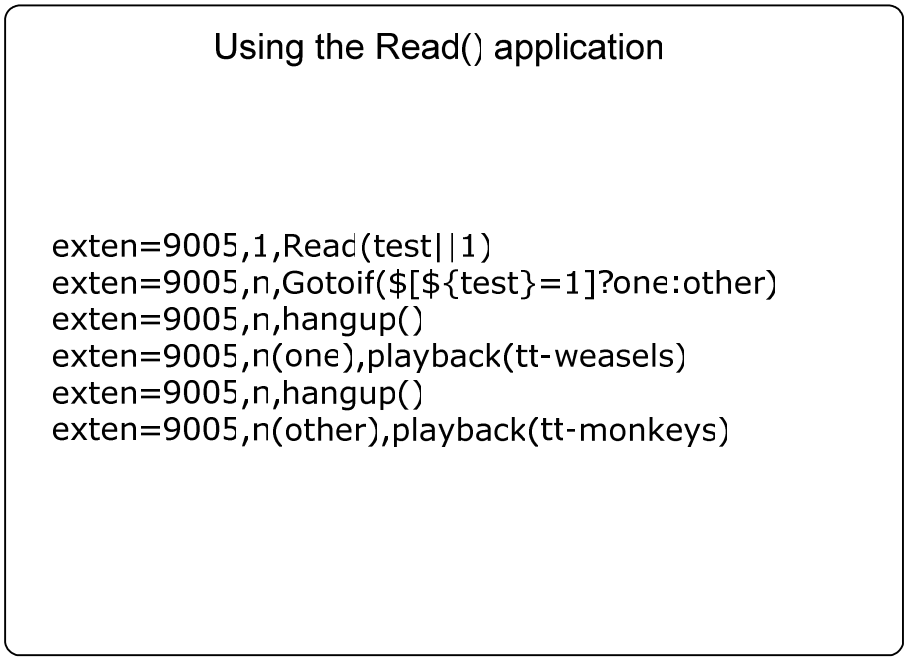


Figure 9.10 Using the read() application()

9.3 CONTEXT INCLUSION

A context can include the contents of another context.

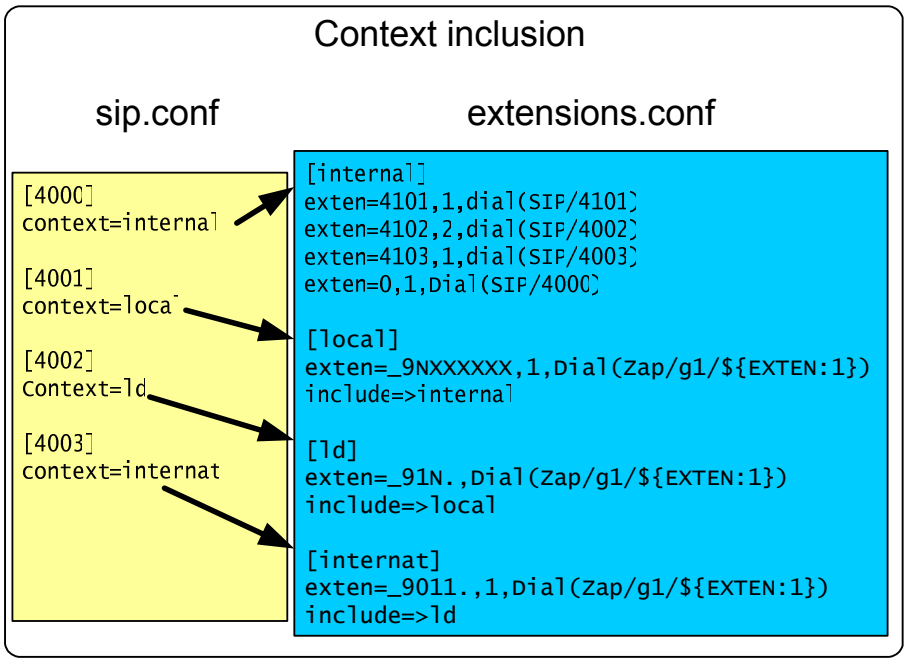


Figure 9.11 Context inclusion

In the example above, any channel can dial any extension in the internal context, but only the 4003 channel can dial international extensions. You can use context inclusion to make it easier the dial plan creation. Using context inclusion, you can control who has access to what extensions.

9.3.1 Context inclusion golden rules

1. **A channel can only dial numbers within the same context as the channel.**

2. **The context where the call is processed is defined in the incoming channel (zapata.conf, iax.conf, sip.conf).**

9.4 USING THE SWITCH STATEMENT

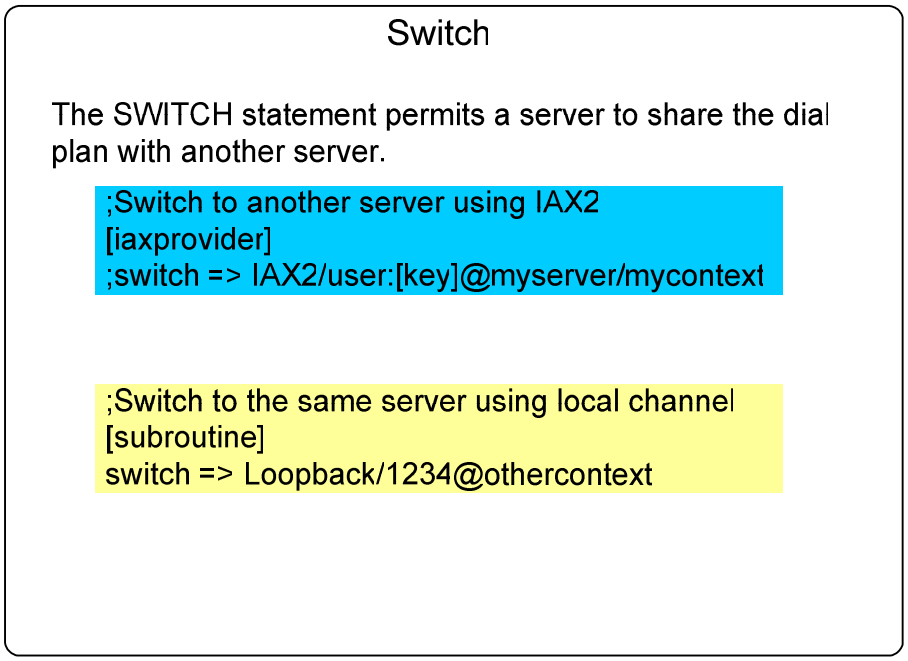


Figure 9.12 Switch statement

You can send the dial plan processing to another server using the switch command. You will need the name and key of the other server. The context is the destination context.

9.5 DIAL PLAN PROCESSING ORDER

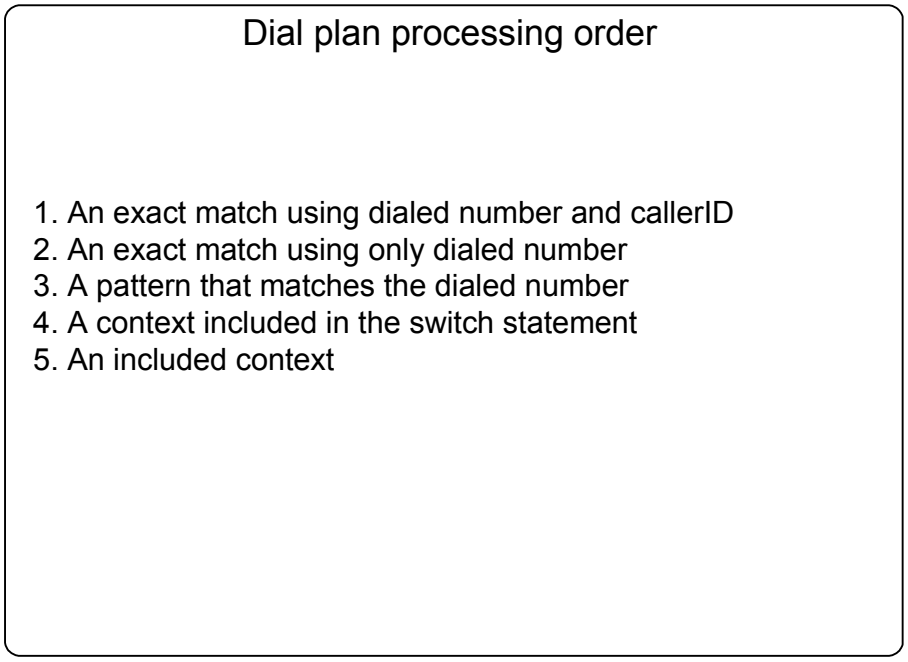


Figure 9.13 Dial plan processing order

When Asterisk receives an incoming call, it looks in the context defined by the channel. In some cases, if more than one pattern matches the dialed number, Asterisk cannot process the call in the exact way you think it should. You can see the matching order using the "dialplan show" CLI command.

Example:

Let's say that you want to dial 912 to route to an analog trunk (ZAP/1) and all other numbers starting with 9 to another analog trunk (ZAP/2). Then you write something like:

```
[example]
exten=>_912.,1,Dial(Zap/1/${EXTEN})
exten=>_9.,1,Dial(Zap/2/${EXTEN})
```

If two patterns match an extension, you can control what extension is processed first, using included contexts. An included context is processed later than a pattern in the same context.

9.6 THE #INCLUDE STATEMENT

Will we use a big file or several files? You can use the #INCLUDE <filename> statement to include other files in your extensions.conf. As an example, we could create a users.conf for local users, services.conf for special services. Be careful to not mix "#INCLUDE <filename>" with "include=>context".

9.7 MACROS

A Macro is an instruction set that executes sequentially. Macros are used to avoid rewriting lines of code, thereby allowing the re-utilization of pieces of code inside the dial plan.

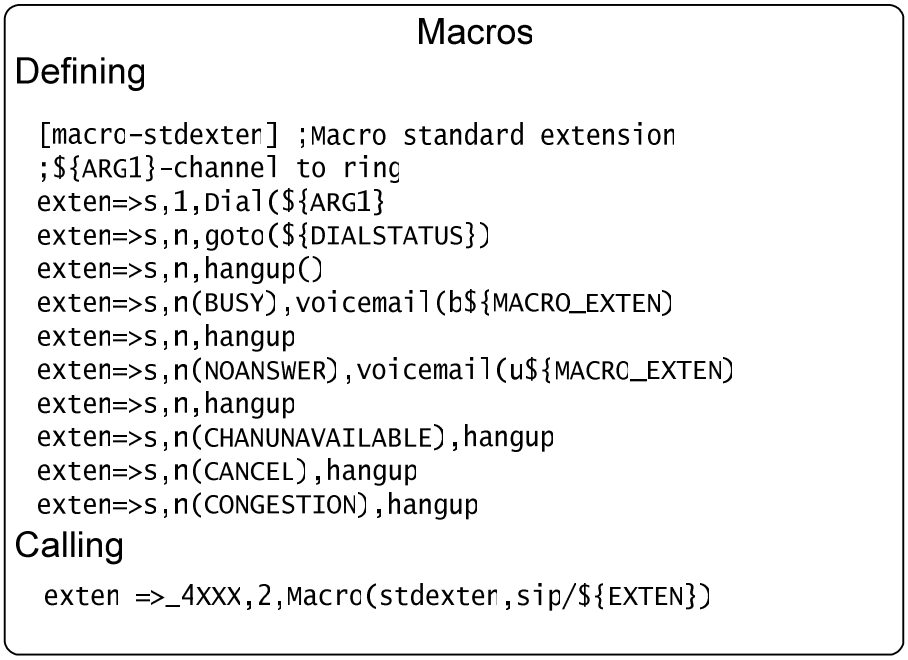


Figure 9.14 Macros

9.7.1 Defining a macro

You can define a macro using [macro-*macroname*]. Within the macro, you may use the specific variables listed below:

- ${ARG1}: First macro argument
- ${ARG2}: Second macro argument and so on.
- ${MACRO_CONTEXT}: Context where macro was called
- ${MACRO_EXTEN}: Extension that triggers the macro

- ${MACRO_OFFSET}: Set by a macro to influence priority after exiting

- ${MACRO_PRIORITY}: Priority where the macro was triggered

9.7.3 Calling a macro

To call a macro you need to use the application:

```
Macro(macroname,arg1,arg2...)
```

To call the macro defined above you should use:

```
exten =>_4XXX,2,Macro(stdexten,sip/${EXTEN})
```

Please check the macros defined in extensions.conf (sample file) to see other examples.

9.8 Implementing Call Forward, Black Lists and DND

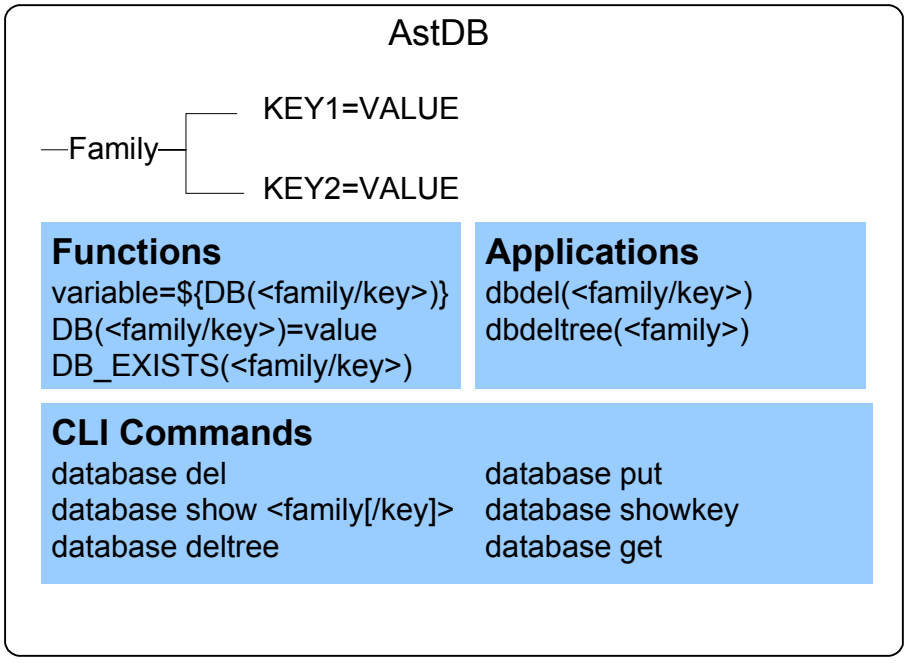


Figure 9.15 Asterisk DB

To implement call forward and black lists we will need some way to store and restore data. Fortunately, Asterisk provides a mechanism for storing and retrieving data from a Berkley DB database (version 1) called AstDB. It is similar to the Windows registry database using the family and keys hierarchical concept. The data persists between Asterisk restarts.

9.8.1 Functions, applications and CLI commands

There are some functions, applications and CLI commands to work with AstDB.

- variable=${DB(<family/key>)}
- DB(<family/key>)=value
- DB_EXISTS(<family/key>)

Examples:

```
exten=_*21*XXXX,1,set(DB(CFBS/${CALLERID(num)}=${EXTEN:4}))
exten=s,1,set(temp=${DB(CFBS/${EXTEN})})
```

Some applications can be used to manipulate Astdb.

- dbdel(<family/key>)
- dbdeltree(<family>)

It is possible to use CLI commands to set and delete keys as well.

- database del
- database put
- database show <family[/key]>
- database showkey
- database deltree
- database get

9.8.2 Implementing Call Forward, DND and blacklists

In this example, you will learn how to implement call forward immediate and call forward on busy. We will use *21* to program call forward immediate and *61* to program call forward on busy status. To cancel the programming, use #21# and #61# respectively.

To populate the database with all the options use:

Call Forward. BlackList, DND

```
[apps]
;call forward immediate
exten=>_*21*XXXX,1,Set(DB(CFIM/${CALLERID(num)})=${EXTEN:4})
exten=>_*21*XXXX,2,Hangup
exten=>*21*,1,DBdel(CFIM/${CALLERID(num)})
exten=>*21*,2,Hangup
;Do not disturb
exten=>_*41*X.,1,Set(DB(dnd/${EXTEN:4})=${EXTEN:4})
exten=>_*41*X.,n,Hangup
exten=>*41*,1,DBdel(dnd/${EXTEN:4})
exten=>*41*,2,Hangup
;call forward on busy status
exten=>_*61*XXXX,1,Set(DB(CFBS/${CALLERID(num)})=${EXTEN:4})
exten=>_*61*XXXX,2,Hangup
exten=>*61*,1,DBdel(CFBS/${CALLERID(num)})
exten=>*61*,2,Hangup
```

Figure 9.16 populating the database

Families used:

- CFIM – Call Forward Immediate
- CFBS – Call Forward on Busy status
- DND – Do Not Disturb

Try populating the database dialing:

- *21*(Destination extension for call forwarding immediate)
- *61*(Destination extension for call forwarding on busy status)
- *41*(Extension to put on not disturb)

Use the CLI command “database show” to see the families, keys and values added.

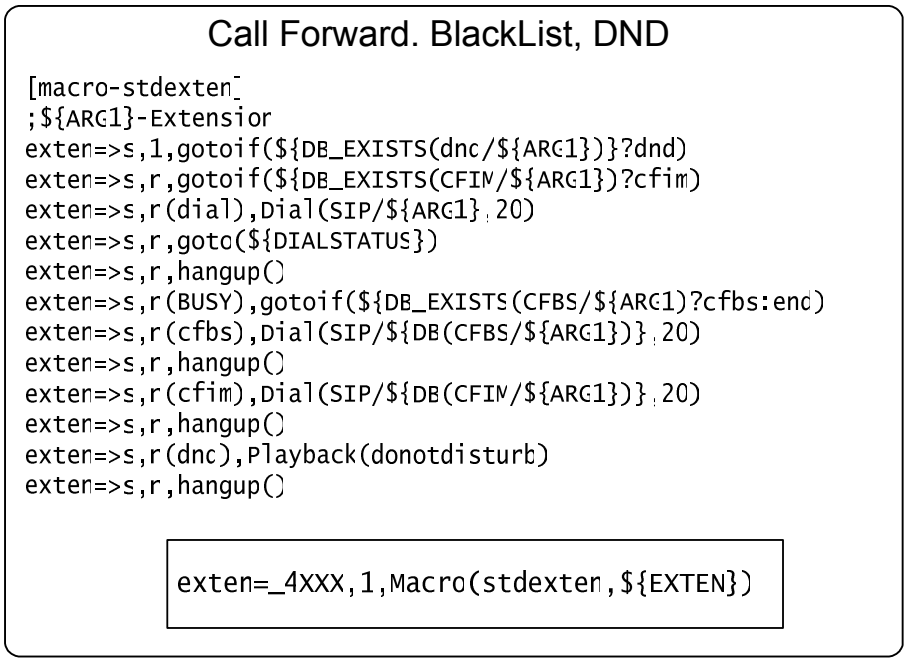


Figure 9.17 handling the calls using a macro

The above macro checks if the database contains the key:value pairs corresponding to CFIM, CFBS or DND and handle them appropriately. The macro exemplified below calls the dialing routine.

```
exten=_4XXX,1,Macro(stdexten,${EXTEN})
```

9.9 USING A BLACKLIST

To create a black list we use the LookupBlacklist() application. The application checks the name/number in the CallerdID. If the number is not found, the applications sets the variable $LOOKUPBLSTATUS to "NOTFOUND". If to the number is found, the application sets the variable to "FOUND". You can use the "j" option in the application to use the old (1.0) behavior, jumping 101 positions if the number/name is found.

Example:

```
[incoming]
exten => s,1,LookupBlacklist(j)
exten => s,2,Dial(SIP/4000,20,tTj)
exten => s,3,Hangup()
exten => s,102,Goto(blocked,s,1)

[blocked]
exten => s,1,Answer()
exten => s,2,Playback(blockedcall)
exten => s,3,Hangup()
```

To insert a number in the blacklist we could use the same resource as before, using *31* followed by the extensions to be blacklisted. To remove from the blacklist you should use #31# followed by the number to be removed.

```
[apps]
exten=>_*31*X.,1,Set(DB(blacklist/${EXTEN}=1})
exten=>_*31*X.,2,Hangup()
exten=>_#31#X.,1,dbdel(blacklist/${EXTEN}:4)
exten=>_#31#X.,2,Hangup()
```

You can also insert the numbers in the blacklist using the console CLI:

```
CLI>database put blacklist <name/number> 1
```

Note: The value associated with the key can be anyone. The blacklist application will search for the key, not the value.

To erase the number from the blacklist you can use:

```
CLI>database del blacklist <name/number>
```

9.10 TIME BASED CONTEXTS

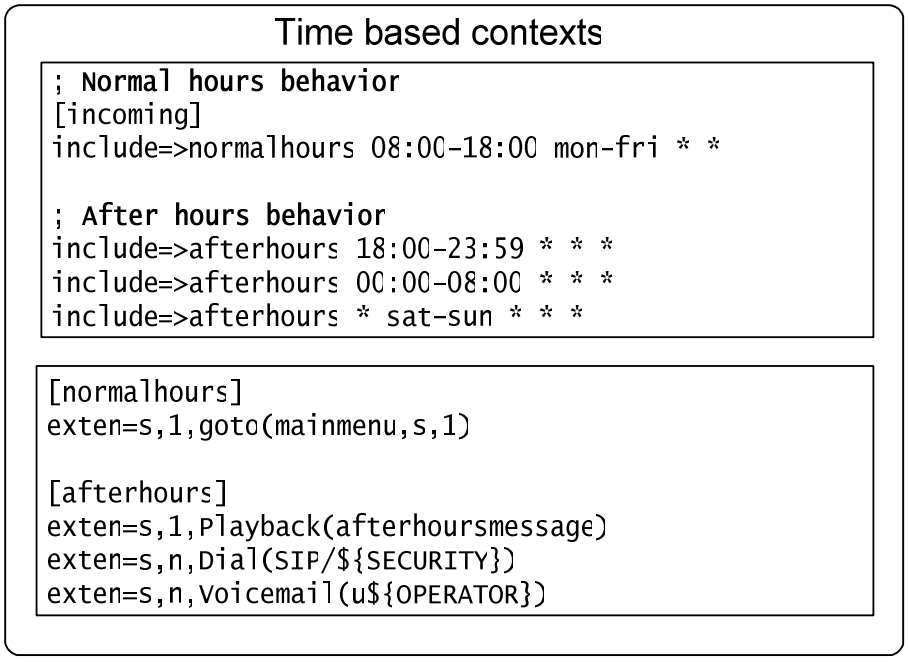


Figure 9.18 Time based contexts

In the above figure, we have a dial plan with three contexts. The [incoming] context is where the calls are usually received. We have included four lines that change behavior depending on the system time, as exemplified below:

```
include => context|<times>|<weekdays>|<mdays>|<months>
```

During regular working hours, processing will be redirected to the mainmenu where it will probably call an IVR to handle the incoming call. If is the call takes place after hours, it will call the security extension defined in the ${SECURITY} variable. If the security does not get the call, it will be sent to the operator's voicemail.

GotoifTime().

The gotoiftime() syntax is shown below.

```
GotoIfTime(<timerange>|<daysofweek>|<daysofmonth>|<months>?[[context|]extension|]pri)
```

It can replace the time-based context and seems easier to understand and read. You can specify the time as follows.

<timerange>=<hour>':'<minute>'-'<hour>':'<minute> |"*"
<daysofweek>=<dayname>|<dayname>'-'<dayname>|"*"
<dayname>="sun"|"mon"|"tue"|"wed"|"thu"|"fri"|"sat"
<daysofmonth>=<daynum>|<daynum>'-'<daynum> |"*"
<daynum>=number from 1 to31
<hour>=number from 0 to 23
<minute>=number from 0 to 59
<months>=<monthname>|<monthname>'-'<monthname>|"*"
<monthname>="jan"|"feb"|"mar"|"apr"|"may"|"jun"|"jul"|"aug"|"sep"|"oct"
|"nov"|"dec"

Names for days and months are not case sensitive.

```
exten=>s,1,GotoIfTime(8:00-18:00|mon-fri|*|*?normalhours,s,1)
```

The statement above transfers the processing to the extension 's' in the normalhours context if the call is between 08:00AM and 06:00PM from Monday to Friday.

9.11 To get a new dial tone use DISA

DISA or "direct inward system access" is a system that allows receiving a second dial tone. It permits users to dial again to another destination. It is often used by technicians when dialing long distance calls for technical

supporting during the weekend. Instead of dialing from their homes directly to the destination, they call the office's DISA number, receive a dial tone, and then call the destination. Long distance charges incur at the company instead of the home phone.

```
DISA(passcode[|context])
DISA(password file)
```

Example:

```
exten => s,1,DISA(no-password|default)
```

With the above statement, the user dials the PBX and even without passing any password, it will receive a dial tone. Any call using DISA will be processed using the default context.

The arguments for this application include a global password or individual passwords inside a file. If no context is specified 'DISA' will be assumed. If you use a password file, the complete path has to be specified. A callerID can be specified for the DISA external dialing too.

Example:

```
numeric-passcode|context|"Flavio" <(48) 30258590>
```

9.12 LIMIT SIMULTANEOUS CALLS

In version 1.2 server, new functions were added. The GROUP() function allows you to count how many active channels you have in one group at the same time.

Example:

You have a branch in Rio de Janeiro where phones follow the pattern "_214X". This location is served by a leased line and 64K was reserved for voice bandwidth. In this case the maximum number of allowed calls is 2 (G.729, 30.2K per call). To limit calls to Rio by two:

```
exten=>_214X,1,set(GROUP()=Rio)
exten=>_214X,n,Gotoif($[${GROUP_COUNT()} > 1]?outoflimit)
exten=>_214X,n,Dial(SIP/${EXTEN})
exten=>_214X,n,hangup
exten=>_214X,n(outoflimit),playback(callsexceedcapacity)
exten=>_214X,n,hangup
```

9.13 LAB - PUTTING IT ALL TOGETHER

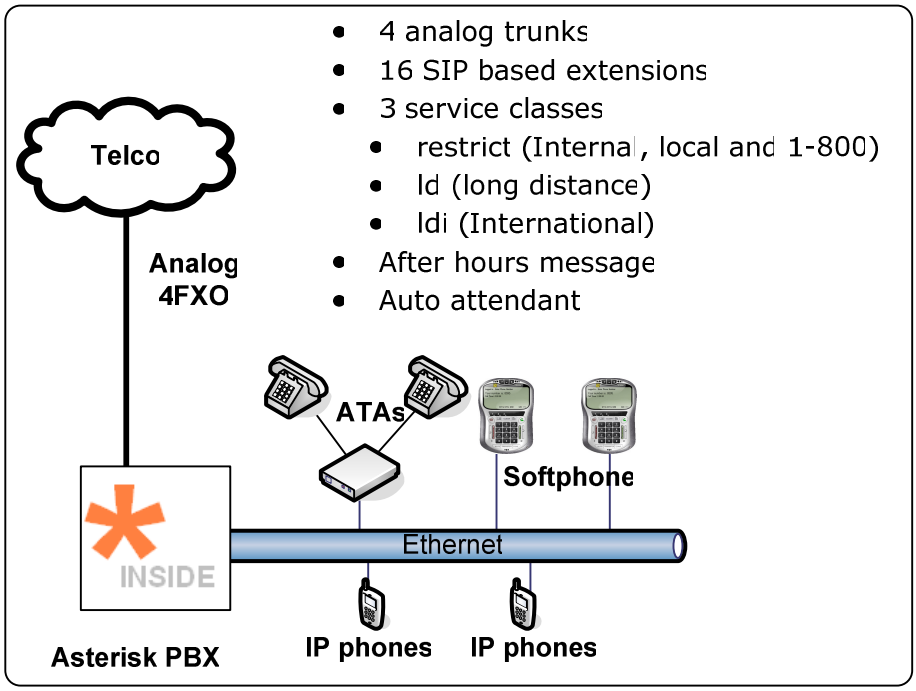


Figure 9.19 Complete PBX example

By now, you should have learned several dial plan concepts. You are probably now a bit confused by too many applications, functions, and concepts. Let's put it all together in a dial plan example.

Let's guide you through the whole PBX configuration for the scenario below.

- 4 analog trunks
- 16 SIP based extensions
- 3 service classes
    - restrict (Internal, local and 1-800)
    - ld (long distance)
    - ldi (International)
- After hours message
- Auto attendant

9.13.1 Step 1 - Configuring channels

Analog trunks (zapata.conf)

To begin with, we will configure the analog trunks in the zapata channel configuration file (zapata.conf). In this case we will use a T400P Digium card with 4 FXO interfaces.

Let's assume that the driver is already loaded and the driver configuration file (zaptel.conf) is correctly configured.

```
signalling=fxs_ks
language=en
context=incoming
group=1
channel => 1-4
```

SIP channels (sip.conf)

We have chosen the dial plan numbering from 2000 to the number 2099. Two codecs will be used: G.729 and G.711 ulaw. The first one will be used for phones using Asterisk over the Internet or WAN and the second one for phones using in the local network. We will arbitrate that, extensions from 2000 to 2039 use 'restrict service class', 2040 to 2059 'ld service class', and 2060 to 2099 'ldi service class'.

```
[general]
disallow=all
allow=gsm
allow=ulaw
bindport = 5060
bindaddr = 0.0.0.0
context = restrict

[2000]
type=friend
username=20
secret=secret
host=dynamic
mailbox=20
context=restrict
canreinvite=yes

[2040]
type=friend
username=20
secret=secret
host=dynamic
mailbox=20
context=ld
```

```
canreinvite=yes
dtmfmode=rfc2833

[2060]
type=friend
username=20
secret=secret
host=dynamic
mailbox=20
context=ldi
canreinvite=yes
dtmfmode=rfc2833
```

9.13.2 Step 2 – Configure the dial plan

Now let's start to configure the extensions.conf.

Define internal extensions and local dialing

```
[restrict]
exten=>_20XX,1,Dial(SIP/${EXTEN},20,t)
exten=>_9XXXXXXX,1,Dial(ZAP/g1/${EXTEN:1},20) ; local calls
exten=>_91800.,1,Dial(ZAP/g1/${EXTEN:1},20); 1-800
```

Define LD (long distance)

```
[ld]
Include=>restrict
exten=>_9NXXNXXXXXX,1,Dial(Zap/g1/${EXTEN:1},20)
```

Define international calls

```
[ldi)
include=> ld
exten=>_901X.,1,Dial(Zap/g1/${EXTEN:1},20)
```

9.13.3 Step 3 - Receive calls using an auto-attendant

To receive calls, use two contexts. The first one is for normal-hours operation, where the call will be received by an auto-attendant. The second one is for after hours, where the caller will receive a message like "you called company XYZ, our normal hours are from 08:00AM to 06:PM; if you know the destination extension number you can try dialing it now or hang-up".

Menus: Normal-hours, After-hours

In the menus below, the system will play a message warning the caller that the company was reached after regular working hours, allowing the caller to

dial the destination extension number (someone may be working after regular working hours).

```
[incoming]
include=>normalhours|08:00-18:00|mon-fri|*|*

include=>afterhours|18:00-23:59|*|*|*
include=>afterhours|00:00-07:59|*|*|*
include=>afterhours|*|sat-sun|*|*

[normalhours]
exten=>s,1,Goto(mainmenu,s,1)

[afterhours]
exten=>s,1,Background(afterhours)
exten=>s,2,hangup()
exten=>i,1,hangup()
exten=>t,1,hangup()
include=>restrict
```

Menus: Main and Sales

During normal working hours the call is answered by an auto-attendant menu. A message like "welcome to XYZ Company, dial 1 for sales, 2 for tech-support, three for training, or the desired extension number".

```
[globals]
OPERATOR=SIP/2060
SALES=SIP/2035
TECHSUPPORT=SIP/2004
TRAINING=SIP/2036

[mainmenu]
exten=> s,1,Background(welcome)
exten=>1,1,Goto(sales,s,1)
exten=>2,1,Goto(techsupport,s,1)
exten=>3,1,Goto(training,s,1)
exten=>i,1,Playback(Invalid)
exten=>i,2,hangup()
exten=>t,1,Dial(${OPERATOR},20,Tt)
include=>restrict

[sales]
exten=>s,1,Dial(${SALES},20,Tt)

[techsupport]
exten=>s,1,Dial(${TECHSUPPORT},20,Tt)

[training]
exten=>s,1,Dial(${TRAINING},20,Tt)
```

With all these statements, the functionality of your dialing plan is now ready. In the next section, we will demonstrate how to operate the PBX.

9.14 SUMMARY

In this chapter, you have learned how to receive calls using an IVR or an auto-attendant. You have understood the concept of context inclusion, and implemented a few examples. Macros were used to avoid repetitive typing and the Asterisk database based on Berkley Db engine was used for functions that require data storage like call forward, do not disturb, or black lists. Finally, you have learned how to implement after hours behavior and implemented a complete dial plan using these concepts.

9.15 QUESTIONS

1. To include a time-dependent context, you can use:

include=> context|<times>|<weekdays>|<mdays>|<months>

The statement below:

include=>normalhours|08:00-18:00|mon-fri|*|*

a) Execute extensions from Monday to Friday between 08:00 to 18:00
b) Execute options everyday in all months
c) Its format is invalid.

2. When an user dials "0" to get an external line, Asterisk automatically cuts the audio. This can be bad because the user is familiarized to hear the external dialing tone before dialing the other numbers. You can simulate the old dialing behavior with the _____ statement.

3. The statements below (mark all that apply):

```
exten => 8590/482518888,1,Congestion
exten => 8590,2,Dial(Zap/1,20,j)
exten => 8590,3,Voicemail(u8590)
exten => 8590,103,Voicemail(b8590)
```

Makes the user who called to the 8590 extension:

a) Receive a busy tone if the CallerID=482518888
b) Receive a busy tone, independent from the number dialed.
c) Dial the ZAP/1 channel
d) Goto to the voicemail if ZAP/1 is busy or did not answered, except when CallerID=482518888.

4. To concatenate several extensions you can separate them using the _____ character.

5. A voice menu is usually created using the_____ application.

6. You can include files inside the configuration files using the _____ statement.

7. The Asterisk database is based in _____.

a) Oracle
b) MySQL
c) Berkley DB
d) PostgreSQL

8. When you use Dial(type1/identifier1&type2/identifier2), the Asterisk dials to each one in sequence and wait 20 seconds between one to another. The affirmative is:

a) False
b) True

9. Using the Background application, you need to wait until the message is played before you can choose an option sending a dtmf digit.

a) False
b) True

10. The valid formats for the goto application are:

a) Goto (context,extension)
b) Goto(context,extension,priority)
c) Goto(extension,priority)
d) Goto(priority)

11. Switches are used to direct the dial plan processing to another server. The affirmative is:

a) False
b) True

12. A macro can be used to automate the processing of an extension. The first macro argument is:

a) ${ARG1}
b) ${ENV1}
c) ${V1}
d) ${X}

page intentionally left empty

Chapter 10

Using PBX features

In this chapter, we will look at the main PBX functionality. We will explore features like call transfer, call parking and others.

10.1 OBJECTIVES

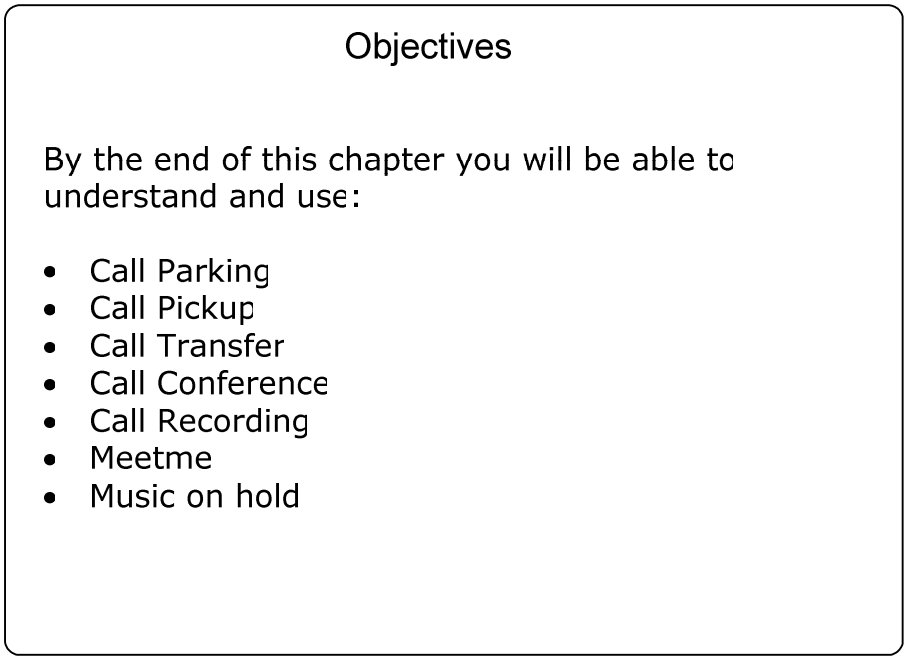


Figure 10.1 Objectives

10.2 PBX FEATURES SUPPORT

In the SIP kingdom, telephone rules. However, Asterisk supports several kinds of phones. Some resources can be found in ATA or the phone itself while others are found in the Asterisk PBX. It is important to standardize the phones in your installation or you will have to explain to each user how to use several of the PBX features.

First and foremost it is important to understand when the PBX features are being executed by or, alternatively, when the phone is doing all the work. As an example, you may transfer a call by using the TRANSFER button on the phone, or by dialing # (unconditional transfer executedby the PBX itself).

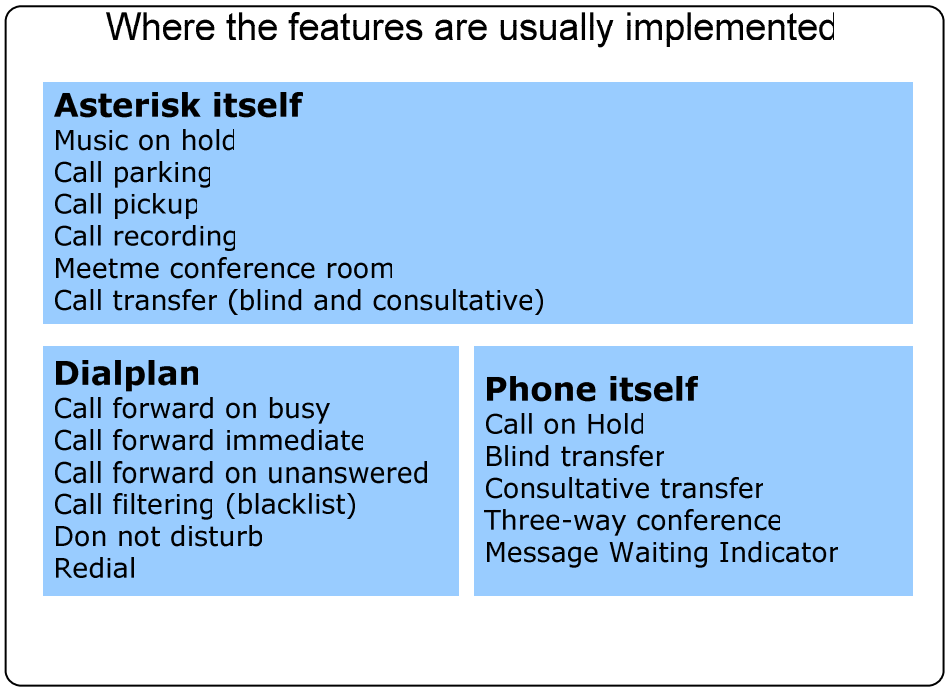


Figure 10.2 where the features are implemented

10.2.1 Features implemented by Asterisk

- Music on hold
- Call parking
- Call pickup
- Call recording
- Meetme conference room
- Call transfer (blind and consultative)

10.2.2 Features usually implemented by the Dial Plan

- Call forward on busy
- Call forward immediate
- Call forward on unanswered
- Call filtering (blacklist)
- Do not disturb
- Redial

10.2.3 Features usually implemented by the Phone

- Call on hold
- Blind transfer
- Consultative transfer
- Three-way conference
- Message waiting indicator

10.2.4 Features.conf configuration file

Some of the features presented in this chapter are configured in features.conf configuration file. It is possible to change the behavior of some features by modifying this file. We presented the whole sample file below. In the next sections of this chapter we will describe each feature.

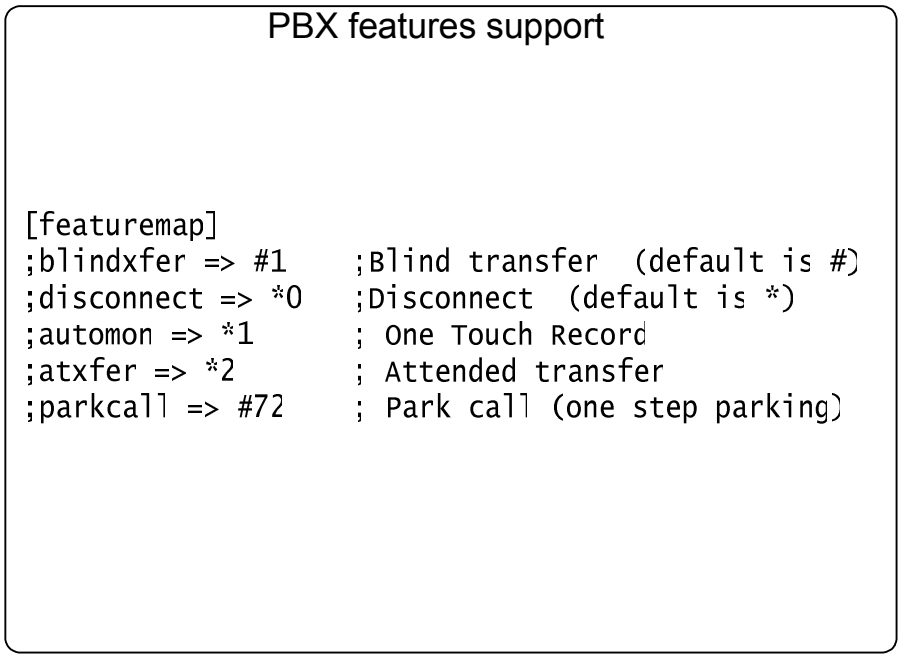


Figure 10.3 PBX feature support

```
;
; Sample Call Features (parking, transfer, etc) configuration
;
[general]
parkext => 700                  ; What extension to dial to park
parkpos => 701-720              ; What extensions to park calls on. These needs to be
                                ; numeric, as Asterisk starts from the start position
                                ; and increments with one for the next parked call.
context => parkedcalls          ; Which context parked calls are in
;parkingtime => 45              ; Number of seconds a call can be parked for
                                ; (default is 45 seconds)
;courtesytone = beep            ; Sound file to play to the parked caller
                                ; when someone dials a parked call
                                ; or the Touch Monitor is activated/deactivated.
;parkedplay = caller            ; Who to play the courtesy tone to when picking up a parked call
                                ; one of: parked, caller, both  (default is caller)
;adsipark = yes                 ; if you want ADSI parking announcements
;findslot => next               ; Continue to the 'next' free parking space.
                                ; Defaults to 'first' available
;parkedmusicclass=default       ; This is the MOH class to use for the parked channel
                                ; as long as the class is not set on the channel directly
                                ; using Set(CHANNEL(musicclass)=whatever) in the dialplan
;transferdigittimeout => 3      ; Number of seconds to wait between digits when transferring a call
                                ; (default is 3 seconds)
;xfersound = beep               ; to indicate an attended transfer is complete
;xferfailsound = beeperr        ; to indicate a failed transfer
;pickupexten = *8               ; Configure the pickup extension. (default is *8)
;featuredigittimeout = 500      ; Max time (ms) between digits for
                                ; feature activation  (default is 500 ms)
;atxfernoanswertimeout = 15     ; Timeout for answer on attended transfer default is 15 seconds.

[featuremap]
;blindxfer => #1                ; Blind transfer  (default is #)
;disconnect => *0               ; Disconnect  (default is *)
```

```
;automon => *1                  ; One Touch Record a.k.a. Touch Monitor
;atxfer => *2                   ; Attended transfer
;parkcall => #72                ; Park call (one step parking)

[applicationmap]
; Note that the DYNAMIC_FEATURES channel variable must be set to use the features
; defined here.  The value of DYNAMIC_FEATURES should be the names of the features
; to allow the channel to use separated by '#'.  For example:
;
;    Set(DYNAMIC_FEATURES=myfeature1#myfeature2#myfeature3)
;
;
; The syntax for declaring a dynamic feature is the following:
;
;<FeatureName> => <DTMF_sequence>,<ActivateOn>[/<ActivatedBy>],<Application>[,<AppArguments>[,MOH_Class]]
;
;  FeatureName   -> This is the name of the feature used in when setting the
;                   DYNAMIC_FEATURES variable to enable usage of this feature.
;  DTMF_sequence -> This is the key sequence used to activate this feature.
;  ActivateOn    -> This is the channel of the call that the application will be executed
;                   on. Valid values are "self" and "peer". "self" means run the
;                   application on the same channel that activated the feature. "peer"
;                   means run the application on the opposite channel from the one that
;                   has activated the feature.
;  ActivatedBy   -> This is which channel is allowed to activated this feature. Valid
;                   values are "caller", "callee", and "both". "both" is the default.
;                   The "caller" is the channel that executed the Dial application, while
;                   the "callee" is the channel called by the Dial application.
;  Application   -> This is the application to execute.
;  AppArguments  -> These are the arguments to be passed into the application.
;  MOH_Class     -> This is the music on hold class to play while the idle
;                   channel waits for the feature to complete. If left blank,
;                   no music will be played.
;
;
; IMPORTANT NOTE: The applicationmap is not intended to be used for all Asterisk
;   applications. When applications are used in extensions.conf, they are executed
;   by the PBX core. In this case, these applications are executed outside of the
;   PBX core, so it does *not* make sense to use any application which has any
;   concept of dialplan flow. Examples of this would be things like Macro, Goto,
;   Background, WaitExten, and many more.
;
;
; Example Usage:
;
;testfeature => #9,peer,Playback,tt-monkeys  ;Allow both the caller and callee to play
;                                            ;tt-monkeys to the opposite channel
;
;pauseMonitor   => #1,self/callee,Pausemonitor     ;Allow the callee to pause monitoring
;                                                  ;on their channel
;unpauseMonitor => #3,self/callee,UnPauseMonitor   ;Allow the callee to unpause monitoring
;                                                  ;on their channel;
; IMPORTANT NOTE: The applicationmap is not intended to be used for all Asterisk
;   applications. When applications are used in extensions.conf, they are executed
;   by the PBX core. In this case, these applications are executed outside of the
;   PBX core, so it does *not* make sense to use any application which has any
;   concept of dialplan flow. Examples of this would be things like Macro, Goto,
;   Background, WaitExten, and many more.
;
; Example Usage:
;
;testfeature => #9,peer,Playback,tt-monkeys  ;Allow both the caller and callee to play
;                                            ;tt-monkeys to the opposite channel
;
;pauseMonitor   => #1,self/callee,Pausemonitor     ;Allow the callee to pause monitoring
;                                                  ;on their channel
;unpauseMonitor => #3,self/callee,UnPauseMonitor   ;Allow the callee to unpause monitoring
;                                                  ;on their channel
```

10.3 CALL TRANSFER

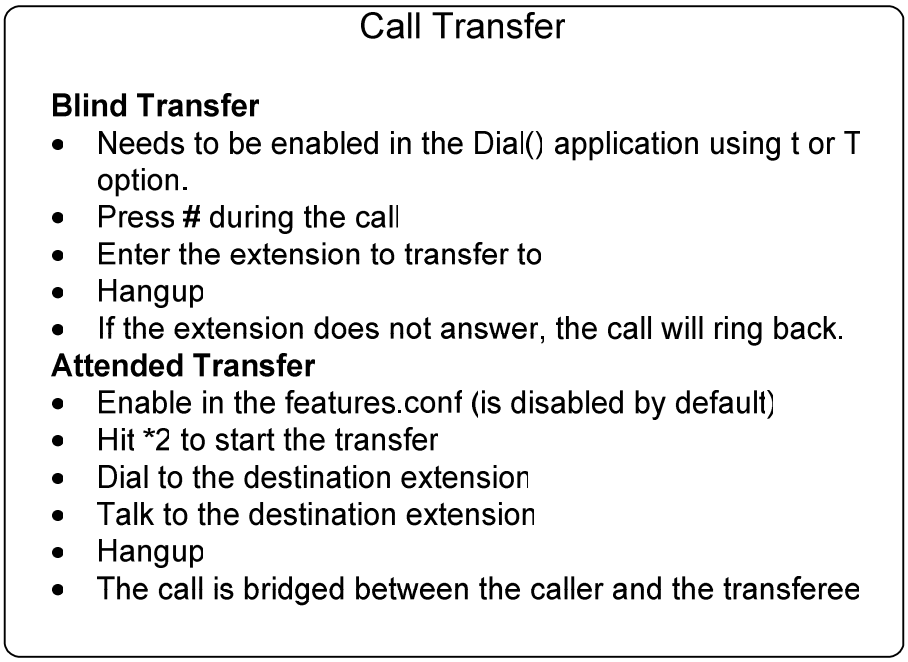


Figure 10.4 Call Transfer

Call transfer can be implemented by the phone, by ATA or by Asterisk itself. See your phone manual to understand how calls are transferred. If your phone does not support call transfer, you can use Asterisk to accomplish this task.

Call transfer can be done in two ways. The first way is to use the blind transfer feature. You simply dial # followed by the number to be transferred. Some times you will use the transfer feature of your IP phone or IP soft phone. You can change the transfer character editing the **blindxfer** parameter in the features.conf file.

You can enable assisted transfer in Asterisk by removing the ´;´ before the **atxfer** parameter in features.conf file. During a conversation, you would press *2. Asterisk will say transfer and will give you a dial tone. The caller is sent to music on hold. You talk to the destination person and hang-up the phone. The system then bridges the caller to the destination.

10.3.1 Configuration task list

1. If the phone is SIP based, make sure that the option canreinvite is equal to "no" or use a ´t´ or ´T´ option in the Dial() application

10.4 CALL PARKING

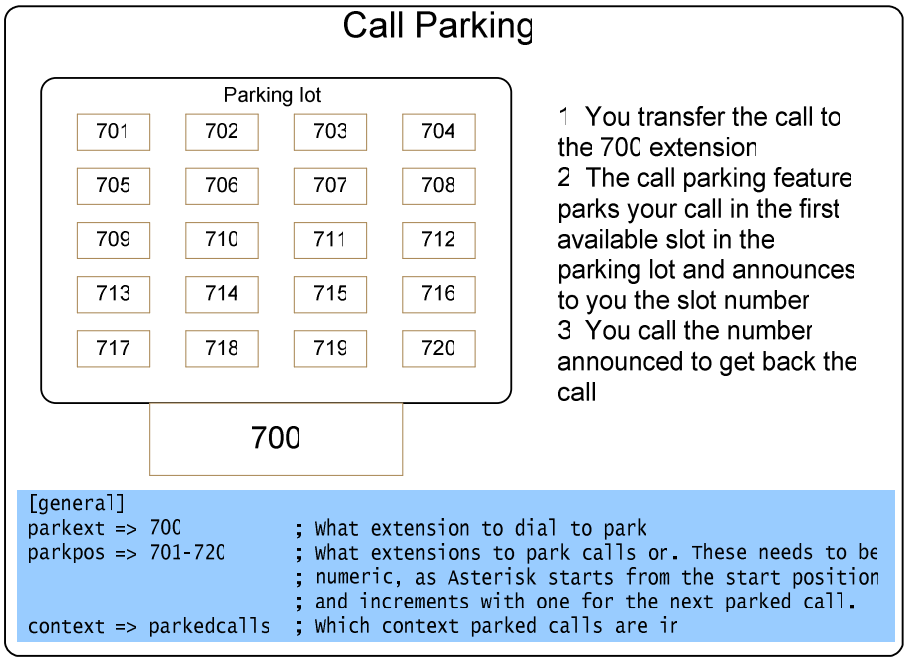


Figure 10.5 Call Parking

This feature is used to park a call. This helps, for example, when you are answering a phone call outside of your room and you want to transfer the call back to your desk. You may accomplish this by parking the call in an extension. Then, when you reach your desk, simply dial the number of the parking extension to recover the call.

By default the 700 extension is used to park a call. In the middle of a conversation press # to transfer the call to the 700 extension. Now the Asterisk will announce your parking extension, probably 701 or 702. Hang-up the phone and the caller will be placed on hold. Go to your desk phone and now dial the announced parking extension to recover the call. If the caller is parked for too much time, the timeout feature will trigger and the original dialed extension will ring again.

10.4.1 Configuration task list

10.4.2 Enable call parking: (required)

In the extensions.conf file type the statement below.

```
include=>parkedcalls
```

10.4.3 Test the call parking feature by dialing #700.

Notes:

- The parking extension won't be shown in the "dialplan show" CLI command.

- It is necessary to restart Asterisk after changing the features.conf file. A simple reload won't work.

- To park a call you need to transfer to #700. Verify the options t and T in the Dial application.

10.5 CALL PICKUP

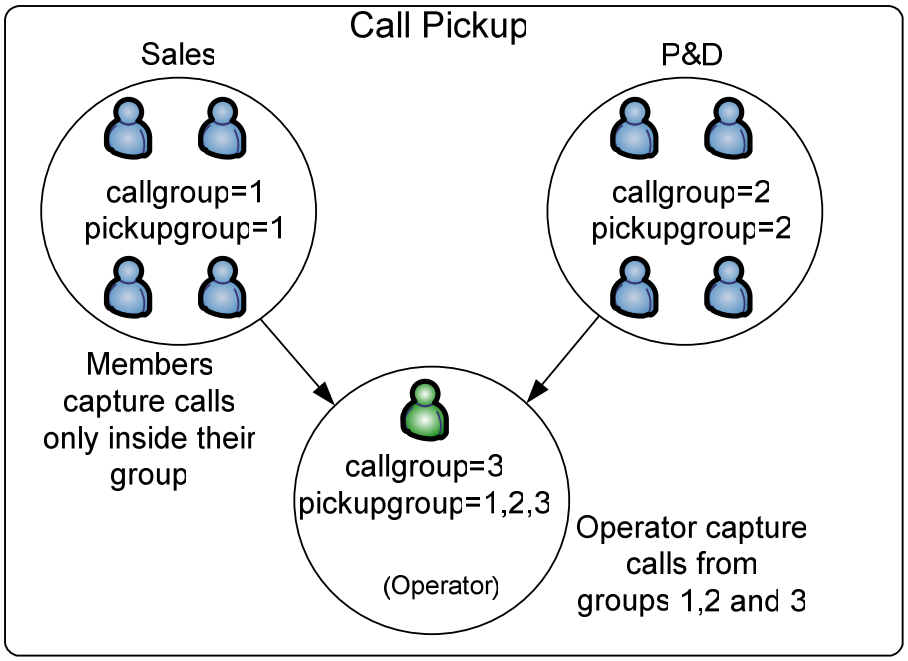


Figure 10.6 Call Pickup

Call pickup allows you to capture a call from a colleague in the same call group. This would help to avoid, for example, that you have to wake up to take a call which is ringing to another person in your room, but who is not present.

By dialing *8, you can capture a call within your call group. This number can be modified in the features.conf file.

10.5.1 Configuration task list

1. Configure a call group for your extensions. This is done in the channel configuration file (sip.conf, iax.conf, zapata.conf). This task is required.

```
[4x00]
callgroup=1
pickupgroup=1,2
```

2. Change the call-pickup feature number (optional).

```
pickupexten=*8; Configures the call pickup extension
```

10.6 CALL CONFERENCE (MEETME)

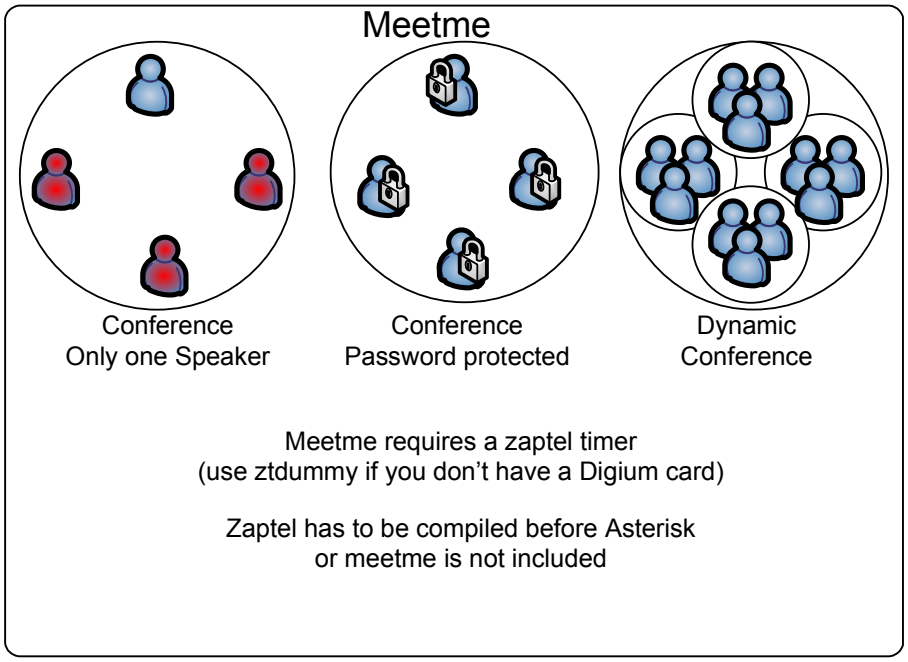


Figure 10.7 meetme application

Some SIP phones implement a three-way conferencing in the phone itself. If this is the case, you may choose to use the phone feature. Search the phone manual for instructions for conference calling.

Alternatively, you may choose to use the Meetme application. Meetme is a conference bridge, which is very simple to use. It works with any type of channel and it is the standard method for conference in the Asterisk platform. Let's look at this feature in more depth.

10.6.1 Meetme Application

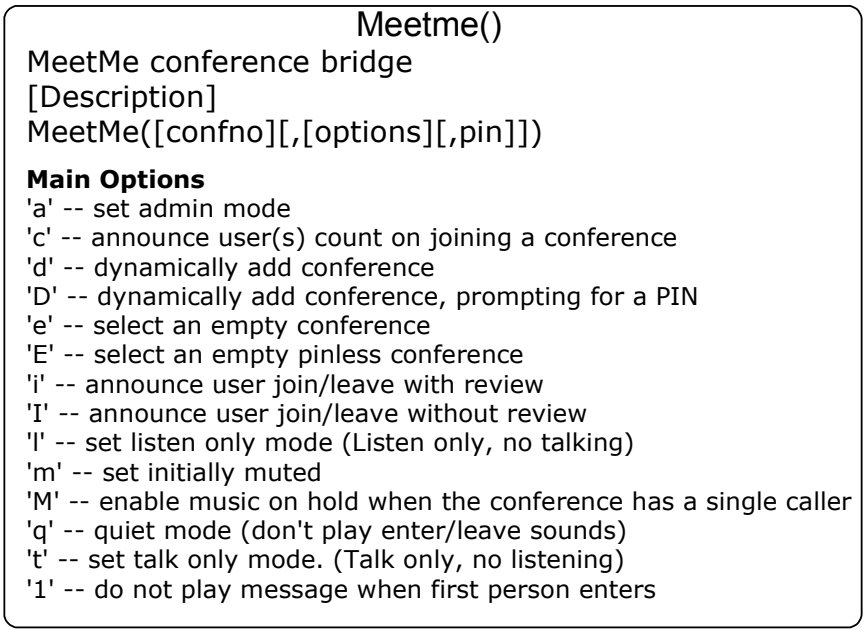


Figure 10.8 Meetme Application

Using the 'meetme show' CLI command you can obtain the description above. To use meetme, you need to compile the zaptel drivers and have at least one zaptel kernel module loaded. If you don't have at least one zaptel card installed, load the ztdummy kernel module to provide a timing source.

Description

The MeetMe application gets the user into a specified MeetMe conference. If the conference number is omitted, the user will be prompted to enter one. The user can leave the conference by hanging-up or, if the 'p' option is specified, by pressing '#'.

Please note: The Zaptel kernel modules and at least one hardware driver (or ztdummy) must be present for conferencing to operate properly. In addition, the chan_zap channel driver must be loaded for the 'i' and 'r' options to operate at all.

The option string may contain zero or one or more of the following characters:

- 'a' -- sets admin mode
- 'A' -- sets marked mode
- 'b' – runs the AGI script specified in ${MEETME_AGI_BACKGROUND}Default: conf-background.agi (Note: This does not work with non-Zap channels in the same conference)
- 'c' -- announces user(s) count on joining a conference

- 'd' -- dynamically adds conference
- 'D' -- dynamically adds conference, prompting for a PIN
- 'e' -- selects an empty conference
- 'E' -- selects an empty pinless conference
- 'i' -- announces user join/leave with review
- 'I' -- announces user join/leave without review
- 'l' -- sets listen only mode (Listen only, no talking)
- 'm' -- sets initially muted
- 'M' -- enables music on hold when the conference has a single caller
- 'o' -- sets talker optimization - treats talkers who aren't speaking as being muted, meaning (a) no encode is done on transmission and (b) received audio that is not registered as talking is omitted causing no buildup in background noise
- 'p' -- allows users to exit the conference by pressing '#'
- 'P' -- always prompts for the pin even if it is specified
- 'q' -- quiet mode (don't play enter/leave sounds)
- 'r' -- Records conference (records as ${MEETME_RECORDINGFILE}using format ${MEETME_RECORDINGFORMAT}). Default filename is meetme-conf-rec-${CONFNO}-${UNIQUEID} and the default format is wav.
- 's' -- Presents menu (user or admin) when '*' is received ('send' to menu)
- 't' -- sets talk only mode. (Talk only, no listening)
- 'T' -- sets talker detection (sent to manager interface and meetme list)
- 'w[(<secs>)]' -- waits until the marked user enters the conference
- 'x' -- closes the conference when last marked user exits
- 'X' -- allows user to exit the conference by entering a valid single digit extension ${MEETME_EXIT_CONTEXT} or the current context if that variable is not defined.
- '1' -- does not play message when first person enters

10.6.2 Meetme configuration file

This file is used to configure the application meetme.

Example:

```
;
; Configuration file for MeetMe simple conference rooms for Asterisk of course.
;
; This configuration file is read every time you call app meetme()

[general]
;audiobuffers=32        ; The number of 20ms audio buffers to be used
                        ; when feeding audio frames from non-Zap channels
                        ; into the conference; larger numbers will allow
                        ; for the conference to 'de-jitter' audio that arrives
```

```
                        ; at different timing than the conference's timing
                        ; source, but can also allow for latency in hearing
                        ; the audio from the speaker. Minimum value is 2,
                        ; maximum value is 32.
;
[rooms]
;
; Usage is conf => confno[,pin][,adminpin]
;
conf=>9000
conf=>9001,123456
```

It is not necessary to use either reload or restart to make Asterisk see the changes in the meetme.conf file.

10.6.3 Meetme related applications

The meetme application has two other support applications.

```
MeetMeCount(confno[|var])
```

It plays the number of users in the conference. If a variable is specified, it does not play the message but sets the number of users to it.

```
MeetMeAdmin(confno,command,[user]):
```

Run admin command for conference

- 'e' -- Ejects last user that joined
- 'k' -- Kicks one user out of conference
- 'K' -- Kicks all users out of conference
- 'l' -- Unlocks conference
- 'L' -- Locks conference
- 'm' -- Unmutes one user
- 'M' -- Mutes one user
- 'n' -- Unmutes all users in the conference
- 'N' -- Mutes all non-admin users in the conference
- 'r' -- Resets one user's volume settings
- 'R' -- Resets all users volume settings
- 's' -- Lowers entire conference speaking volume
- 'S' -- Raises entire conference speaking volume
- 't' -- Lowers one user's talk volume
- 'T' -- Lowers all users talk volume
- 'u' -- Lowers one user's listen volume
- 'U' -- Lowers all users listen volume
- 'v' -- Lowers entire conference listening volume
- 'V' -- Raises entire conference listening volume

10.6.4 Meetme configuration task list

1. Choose the extension for the Meetme room (required)

2. Edit the meetme file to configure the passwords (optional)

10.6.3 Examples

Example #1: Simple meetme room

1. In the extensions.conf file create the conference room 101

```
exten=>500,1,MeetMe(101,,123456)
```

2. In the meetme.conf file establish the password for the room 101.

Important Note:
The meetme application needs a timer to work. If you don't have digium hardware installed and configured, use ztdummy as a timing source.

10.7 CALL RECORDING

There are several ways to record a call in Asterisk. You can use the mixmonitor() application to easily record calls.

10.7.1 Using the mixmonitor application

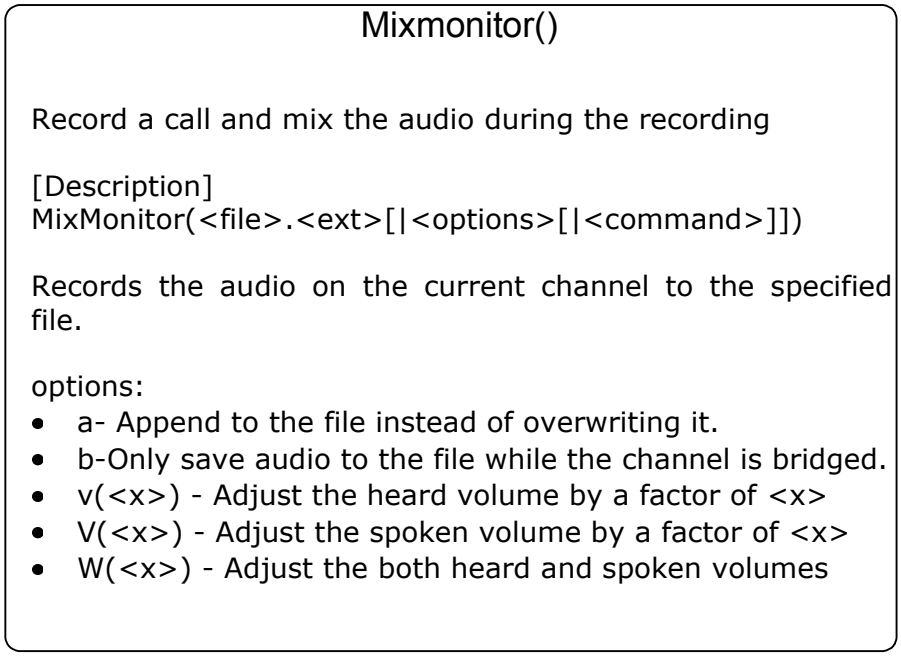


Figure 10.9 Mixmonitor application

Mixmonitor records the audio in the current channel to the specified file.
If the filename is an absolute path, ituses that path, otherwise it creates the file in the configured monitoring directory from asterisk.conf.

Valid options:

- a - Appends to the file instead of overwriting it.
- b - Only saves audio to the file while the channel is bridged.
- Note: does not include conferences.
- v(<x>) - Adjusts the audible volume by a factor of <x> (range -4 to 4)
- V(<x>) - Adjusts the spoken volume by a factor of <x> (range -4 to 4)
- W(<x>) - Adjusts both audible and spoken volumes by a factor of <x> (range -4 to 4)

<command> will be executed when the recording is over. Any strings matching ^{X} will be unescaped to ${X} and all variables will be evaluated at that time. The variable MIXMONITOR_FILENAME will contain the filename used to record.

An interesting resource is Automon. It allows you to simply dialing *1 to immediately start recording.

Example:

```
exten=>_4XXX,1,Set(DYNAMIC_FEATURES=automon)
exten=>_4XXX,2,Dial(SIP/${EXTEN},20,jtTwW);wW enables the recording.
```

The audio channels are incoming (IN) and outgoing (OUT) and are separated into two distinct files in the following directory.

/var/spool/asterisk/monitor

Both files can be mixed using the sox application.

```
debian#soxmix *in.wav *out.wav output.wav
```

If you don't want to use Set() before the Dial() application, you can set at the globals section:

```
[globals]
DYNAMIC_FEATURES=>automon
```

10.8 MUSIC ON HOLD

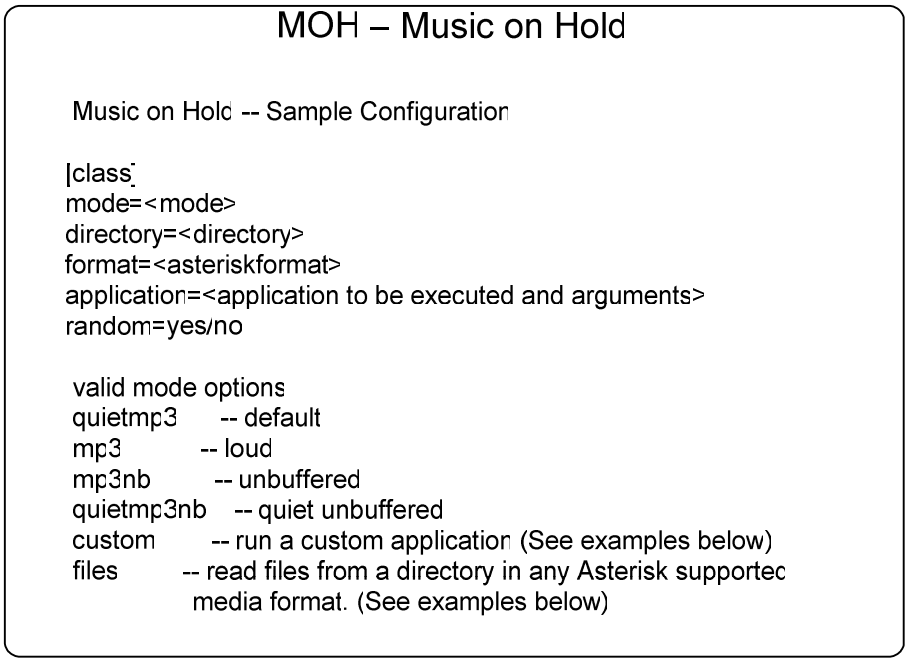


Figure 10.10 Music on Hold

MOH (Music on hold) has been changed several times between versions 1.0, 1.2, and 1.4. In the latest version, MOH defaults to "FILE-BASED". In other

words, Asterisk will supply the MOH files in formats like G.729, alaw, ulaw, gsm, and so on. Thus, it is not necessary to transcode the music before sending it to the channel. This saves processor time, which is a welcomed modification for those working with productions systems. In older versions, MOH was usually provided by MP3 (it still can be configured that way). Providing MOH using MP3 obligates Asterisk to transcode and spend valuable CPU power in the process.

The new configuration file is shown below. Note that the default class now uses the native file format "mode=files". All other modes are commented.

Each section is a class. The only uncommented class right now is default. If you want to have different classes for different files, you will need to create new sections (classes)

```
; Music on Hold -- Sample Configuration

;[samplemp3]
;mode=quietmp3
;directory=/var/lib/asterisk/mohmp3
;
; valid mode options:
; quietmp3      -- default
; mp3           -- loud
; mp3nb         -- unbuffered
; quietmp3nb    -- quiet unbuffered
; custom        -- run a custom application (See examples below)
; files         -- read files from a directory in any Asterisk supported
;                  media format. (See examples below)

;[manual]
;mode=custom
; Note that with mode=custom, a directory is not required, such as when
reading
; from a stream.
;directory=/var/lib/asterisk/mohmp3
;application=/usr/bin/mpg123 -q -r 8000 -f 8192 -b 2048 --mono -s

;[ulawstream]
;mode=custom
;application=/usr/bin/streamplayer 192.168.100.52 888
;format=ulaw

; mpg123 on Solaris does not always exit properly; madplay may be a better
; choice
;[solaris]
;mode=custom
;directory=/var/lib/asterisk/mohmp3
;application=/site/sw/bin/madplay -Q -o raw:- --mono -R 8000 -a -12
;

;
; File-based (native) music on hold
;
```

```
; This plays files directly from the specified directory, no external
; processes are required. Files are played in normal sorting order
; (same as a sorted directory listing), and no volume or other
; sound adjustments are available. If the file is available in
; the same format as the channel's codec, then it will be played
; without transcoding (same as Playback would do in the dialplan).
; Files can be present in as many formats as you wish, and the
; 'best' format will be chosen at playback time.
;
; NOTE:
; If you are not using "autoload" in modules.conf, then you
; must ensure that the format modules for any formats you wish
; to use are loaded _before_ res_musiconhold. If you do not do
; this, res_musiconhold will skip the files it is not able to
; understand when it loads.
;

[default]
mode=files
directory=/var/lib/asterisk/moh
;
;[native-random]
;mode=files
;directory=/var/lib/asterisk/moh
;random=yes     ; Play the files in a random order
```

10.8.1 MOH configuration tasks

Now to use music on hold, set the MOH class in the channel configuration files (zapata.conf, sip.conf, iax.conf and so on). The freeplay tunes installed are now in wav format. At the time of installation you can select (using make menuselect) the MOH file formats to be available. If you want to add new MOH files, you will have to supply them in the required formats.

Example:

In /etc/asterisk/zapata.conf, add the line:

```
[channels]
musiconhold=default
```

```
Edit the file /etc/asterisk/musiconhold.conf
```

```
[default]
mode=files
directory=/var/lib/asterisk/moh
```

In the dialplan, you can hear the MOH using the example below:

```
Exten=>100,1,SetMusicOnHold(default)
```

```
Exten=>100,2,Dial(Zap/2)
```

Example:

Configuring the extensions.conf to test MusicOnHold.

```
[local]
exten => 6601,1,WaitMusicOnHold(30)
```

10.9 APPLICATION MAPS

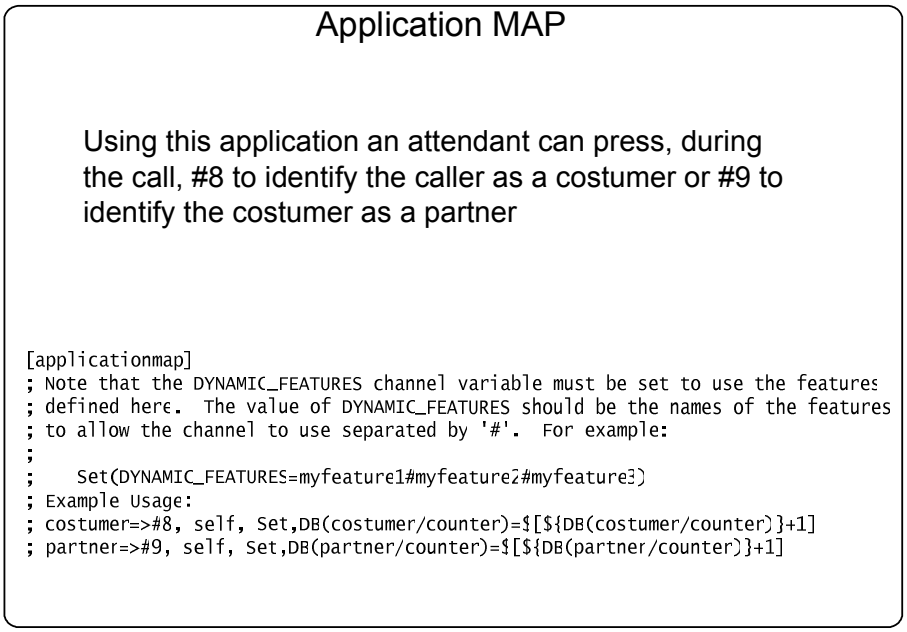


Figure 10.11 Application maps

In version1.2, it became possible to add new features by using the application maps section of the features.conf file. Suppose you need to identify to a Call Center the type of costumer you are answering. You could create an application map to each costumer type. This application could count the number of answered costumers per type.

10.10 QUESTIONS

1. The following statements are true about Call Parking:

a) By default, extension 800 is used for Call Parking
b) When you are out of your desk and receive a call, you can park a call. The system will announce to you the parking extension. Then, you go to your desk and dial the announced extension to retrieve the call.
c) By default, extension 700 is used for Call Parking. Calls are parked in extensions 701 to 720.
d) You need to dial 700 to retrieve a parked call.

2. To use the Call Pickup feature, all extensions are required to be in the same _____. For ZAP channels this is configured in the _____ file.

3. When transferring a call, you may choose between _____, where the destination extension is not consulted before the transfer and _____ where you talk first to the destination extension before the transfer.

4. To make a consultative transfer you use the ___ character, while for blind transfer you use ___.

a) #1, *2
b) *2, #1
c) #2, #1
d) #1, #2

5. To enable conference calls in the Asterisk server, it is necessary to use the _____ application.

6. If you have to supervise a conference, you can use the _____ application.

a) MeetMe()
b) MeetMeConsole()
c) MeetMeAdministrator()
d) MeetmeAdmin()

7. The best format for music on hold is MP3 because it uses very little processing power from the Asterisk server.

a) True
b) False

8. To capture a call from a specific call group you need to be in their respective _____ group.

9. You can record a call by using the utility mixmonitor() or using the automon feature. By default the automon feature uses the ___ character sequence.

a) *1
b) *2
c) #3
d) #1

10. In the meetme application, if you want to have users in the listening only mode you should:

a) Merge different conference rooms with different options
b) It is not possible using Asterisk.
c) Enable an extension that calls the meetme application with the 'l' option and instruct the listening users to call that extension.
d) Enable an extension that calls the meetme application with the 't' option and instruct the listening users to call that extension.

Page intentionally left empty

Chapter 11

ACD Automatic Call Distribution

11.1 OBJECTIVES

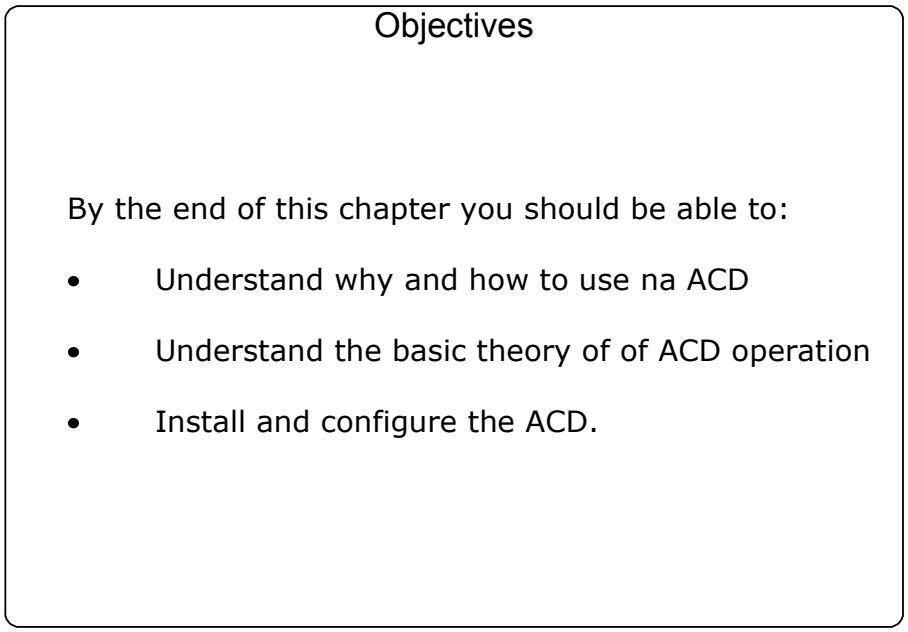


Figure 11.1 Objectives

11.2 INTRODUCTION

Call queues allow you to answer calls in a more efficient way. An automatic call distributor can help reduce costs, increase service or improve sales. This is because call distributors affect how your business functions, not for a few days, but for many years. With an ACD you can correctly dimension the number of agents, control who is a good and bad attendant and analyze call flow.

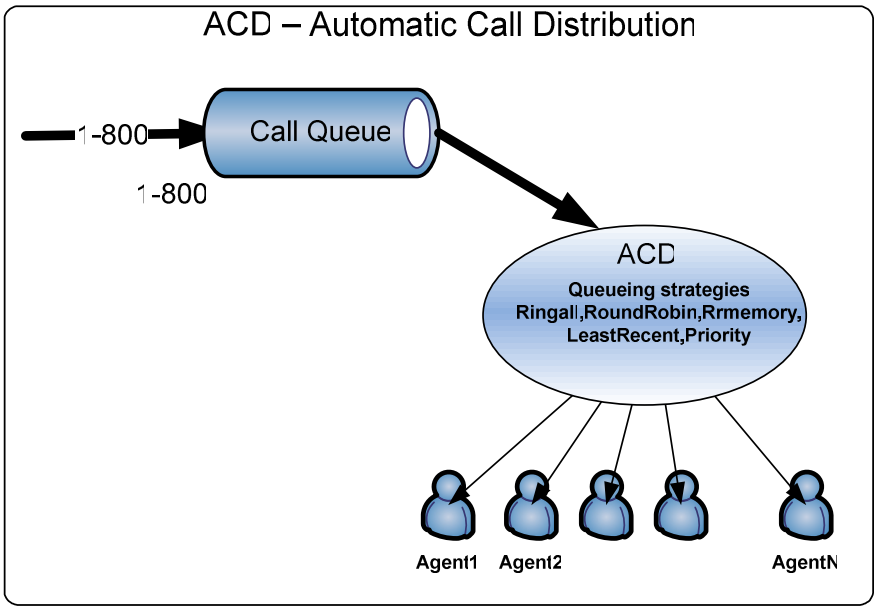


Figure 11.2 Automatic Call Distribution

Usually, a call queue works like this:

1. Calls are queued
2. Agents answer the queue (logged in agents)
3. A queuing strategy to distribute the calls is used.
4. Music on hold is played while the caller waits.
5. Announcements can de made to callers, warning about waiting time.

The main application for queues is costumer service or call centers. Using queues, you avoid to lost calls when your agents are busy. You can add new agents to the queue if you find that the number of callers in the queue is growing.

Another thing is, with queues you can now have statistics like Call abandon rate, average call duration and call answering target. These statistics will help you to dimension the number of agents and provide a better service to your costumer.

11.3 ACD ARCHITECTURE

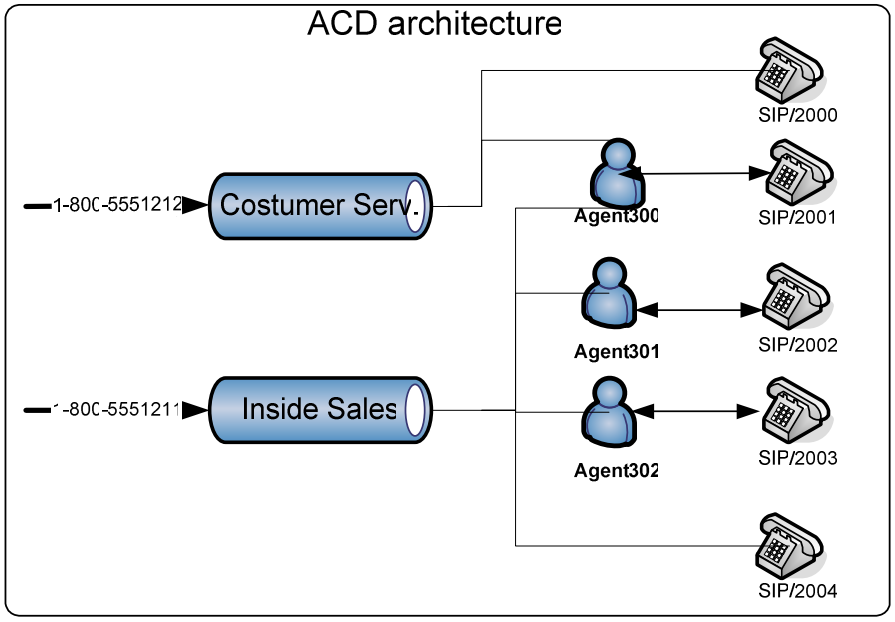


Figure 11.3 ACD architecture

The ACD architecture is formed by queues and agents. One agent can be in two queues at the same time. A queue can have agents, channels and agent groups.

11.4 QUEUES

Queues are defined in the queues.conf configuration file. Agents are attendants who log in and are members of queues. Agents are defined in the agents.conf file.

In version 1.4, the queue system has evolved largely and now the configuration file is huge. We will explain some of the major parameters.

General parameters

```
autofill=yes
```

The old behavior for the queue was serial type. The queue waited for a call to be dispatched before sending the succeeding call to the next agent. If an agent takes 15 seconds to get a call, the other calls in the queue had to wait until that call was answered. For high volume queues, this behavior was poor.

The new behavior (autofill=yes) does not wait until a call is answered. It works in a parallel fashion.

11.4.1 Queues.conf example:

queues.conf

```
[general]
persistentmembers = yes
autofill = yes
monitor-type = MixMonitor
[costumerservice]
musicclass = default
announce = queue-costumerservice
strategy = rrmemory
servicelevel = 60
context = costumerservice
timeout = 15
retry = 5
wrapuptime=15
announce-frequency = 90
periodic-announce-frequency=60
announce-holdtime = yes
monitor-format = wav
member => Agent/301,301
Member => Agent/300,300
```

Figure 11.4 queues.conf example

11.4.2 Members

Members are active channels responding to the queue. Members can be direct channels (sip, zap, mgcp) or agents who log in before receiving calls.

11.4.3 Strategies

Calls are distributed between members according to one of these strategies:

- **ringall:** Plays all channels available until someone answers.
- **roundrobin:** Distributes equally between members.
- **leastrecent**: Distributes to the least recent member.
- **fewestcalls:** Distributes to the member with fewest calls.
- **random:** Distributes randomly between members
- **rrmemory**: It is a roundrobin with memory. It remembers where it let off the call in the last pass.

11.5 AGENTS

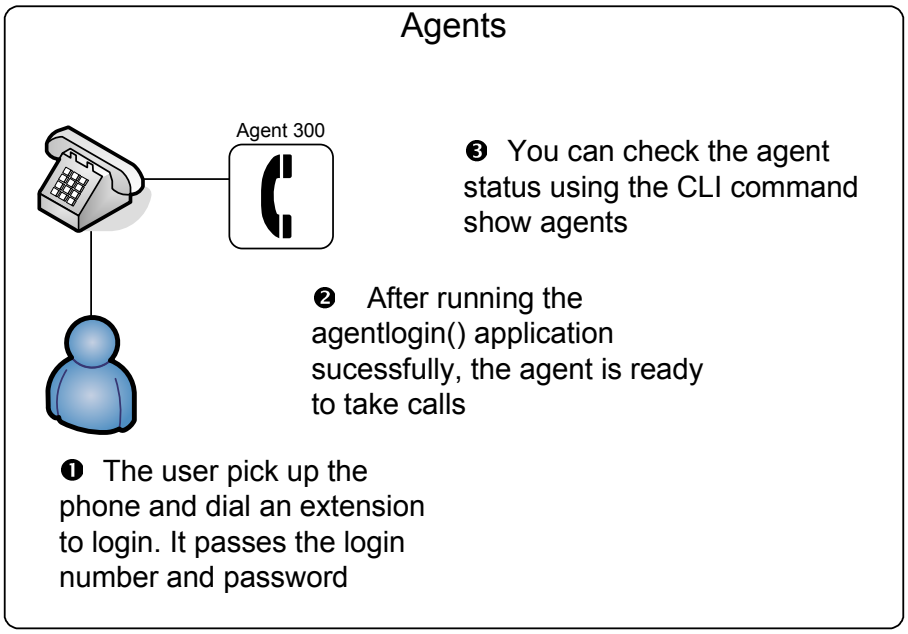


Figure 11.5 Agent login process

Agent is a proxy channel. It can be used with queues. Another use for Agents is extension mobility. The user can login within any phone and receive its calls. This allows a user to go to any room and make it their office. You can dial an agent in the dial plan using Dial(Agent/<name>). You define agents in the agents.conf file.

11.5.1 Agent Groups

You may choose to use agent groups. This function does not take consideration of ACD strategies. Probably you will prefer to list all agents individually. If you want to transfer to an agent group you can use in queues.conf:

```
member=>agent/@1  ;any agent in group 1
member=>agent/:1,1;any agent in group 1, wait for first available, ;do not use
agent groups.
```

11.5.2 Agents.conf example:

Agents.conf

```
; Agent configuration

[general]
persistentagents=yes

[agents]
autologoff=15
autologoffunavail=yes
ackcall=no
endcall=yes
wrapuptime=5000
musiconhold => default
;
;This section contains the agent definitions, in the form:
;
; agent => agentid,agentpassword,name
;
agent => 300,300
agent => 301,301
```

Figure 11.6 agents.conf file

11.6 ACD RELATED APPLICATIONS

11.6.1 queue()

The queue() application

Queue a call for a call queue

[Description]

Queue(queuename[|options[|URL][|announceoverride][|timeout][|AGI]):

Options:

- 'd' -- data-quality (modem) call (minimum delay).
- 'h' -- allow callee to hang up by hitting *.
- 'H' -- allow caller to hang up by hitting *.
- 'n' -- no retries on the timeout; will exit this application.
- 'i' -- ignore call forward requests from queue members and do nothing
- 'r' -- ring instead of playing MOH
- 't' -- allow the called user transfer the calling user
- 'T' -- to allow the calling user to transfer the call.
- 'w' -- allow the called user to write the conversation to disk via Monitor
- 'W' -- allow the calling user to write the conversation to disk via Monitor

Figure 11.7 The queue() application

This function queues incoming calls into a particular call queue as defined in queues.conf. The option string may contain zero or more of the following characters:

Options:

- 'd' -- data-quality (modem) call (minimum delay).
- 'h' -- allows callee to hang-up by hitting *.
- 'H' -- allows caller to hang-up by hitting *.
- 'n' -- no retries on the timeout; will exit this application and go to the next step.
- 'i' -- ignores call forward requests from queue members and does nothing when they are requested.
- 'r' -- rings instead of playing MOH
- 't' -- allows the called user to transfer the calling user
- 'T' -- allows the calling user to transfer the call
- 'w' -- allows the called user to write the conversation to disk via Monitor
- 'W' -- allows the calling user to write the conversation to disk via Monitor

In addition to transferring the call, a call may be parked and then picked up by another user.

The optional URL will be sent to the called party if the channel supports it.

The optional AGI parameter will setup an AGI script to be executed on the calling party's channel once they are connected to a queue member.

The timeout will cause the queue to fail out after a specified number of seconds, checked between each queues.conf 'timeout' and 'retry' cycle.

This application sets the QUEUE status variable upon completion:

- TIMEOUT
- FULL
- JOINEMPTY
- LEAVEEMPTY
- JOINUNAVAIL
- LEAVEUNAVAIL

11.6.2 Agentlogin()

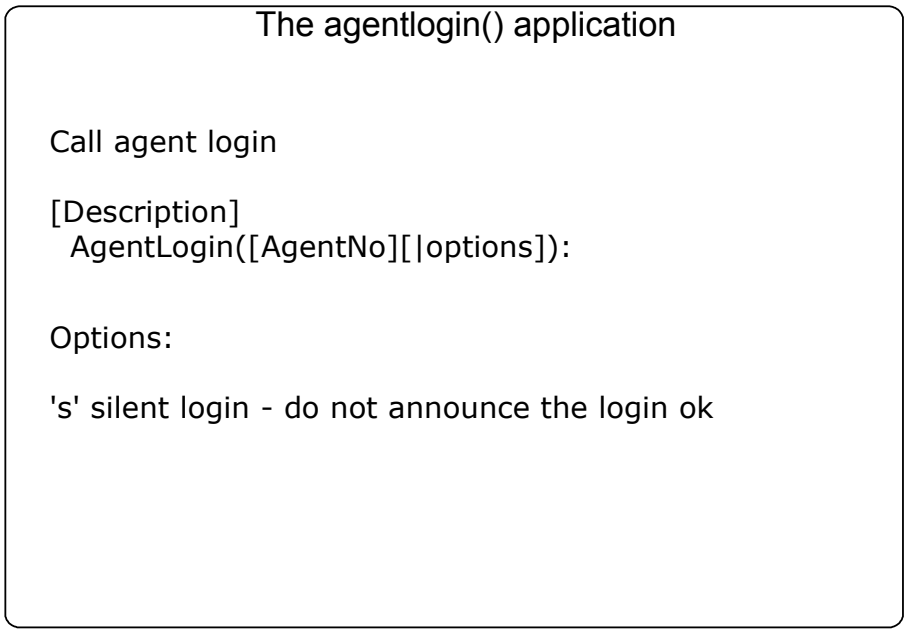


Figure 11.8 The agentlogin() application

This applications asks the agent to login into the system. It always returns -1. While logged in, the agent can receive calls and will hear a 'beep' when a new call comes in. The agent can dump the call by pressing the star key.

Options:

- 's' silent login - do not announce the login ok

11.6.3 Agencallbacklogin()

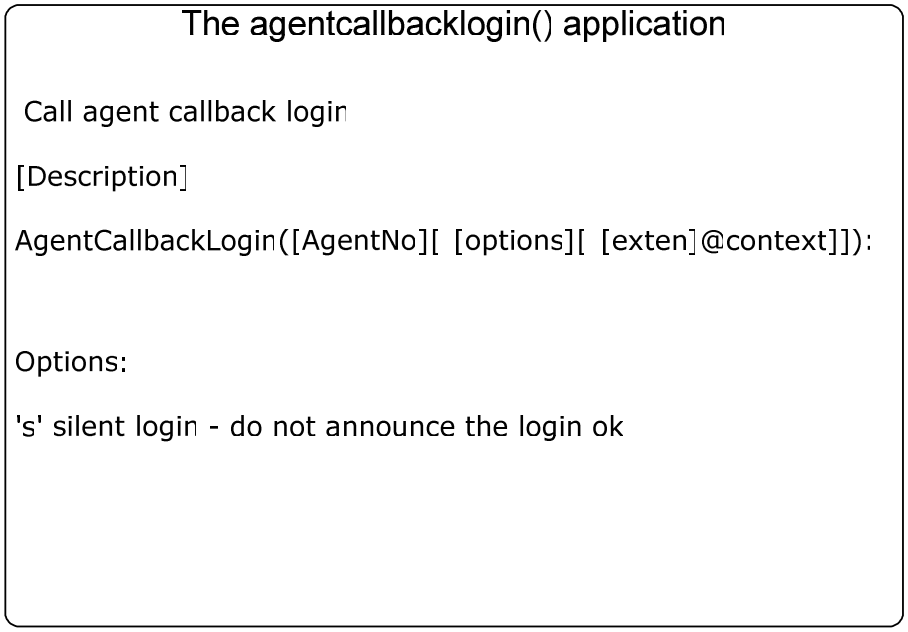


Figure 11.9 Agentcallbacklogin()

This applications asks the agent to login to the system and hang-up. The agent's extension will ring when someone is in the queue. The agent's callback extension is called (optionally with the specified context).

Options:

- 's' -- silent login - do not announce the login ok segment agent logged in/off

11.6.4 Support applications and CLI commands

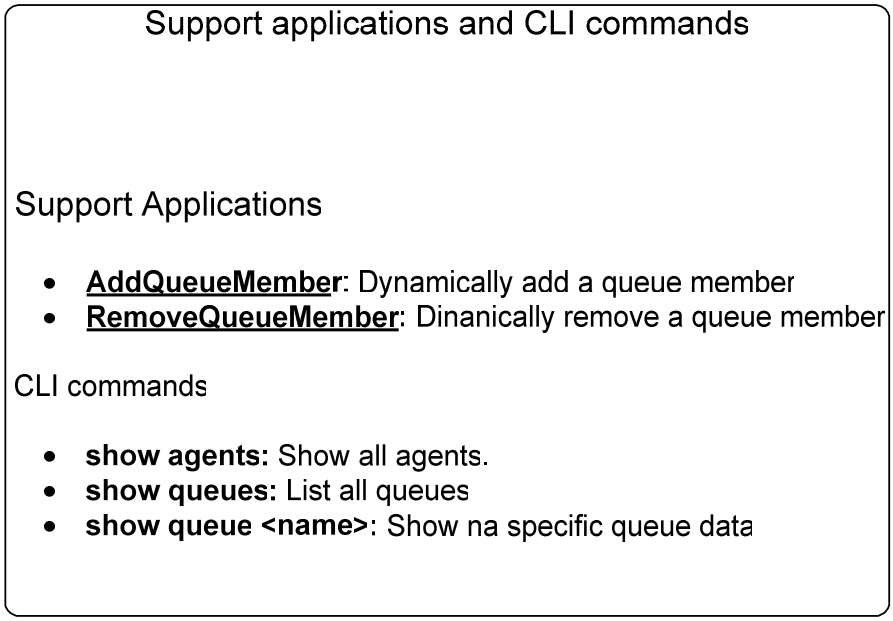


Figure 11.10 Support applications and CLI commands

11.7 CONFIGURATION TASKS

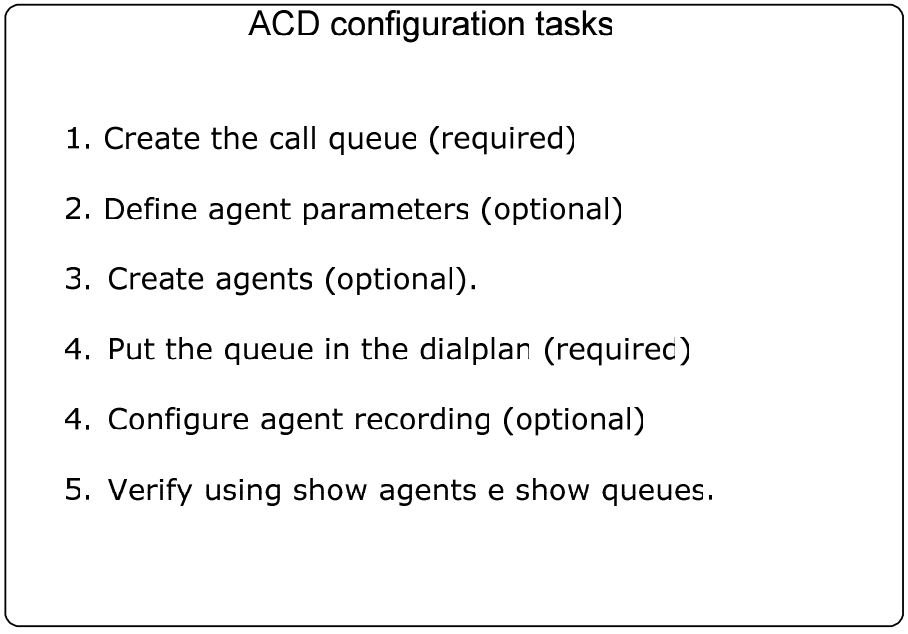


Figure 11.11 ACD Configuration Tasks

11.7.2. Create the call queue

queues.conf

```
[telemarketing]
music = default
;announce = queue-telemarketing
;context = qoutcon
timeout = 2
retry = 2
maxlen = 0
member => Agent/300
member => Agent/301
[auditing]
music = default
;announce = queue-auditing
;context = qoutcon
timeout = 15
retry = 5
maxlen = 0
member => Agent/600
member => Agent/601
```

11.7.3 Define agent parameters

```
agents.conf

debian:/etc/asterisk# cat agents.conf
;
; Agent configuration
;

[agents]
; Define maxlogintries to allow agent to try max logins before
; failed.
; default to 3
maxlogintries=5
; Define autologoff times if appropriate.  This is how long
; the phone has to ring with no answer before the agent is
; automatically logged off (in seconds)
autologoff=15
; Define autologoffunavail to have agents automatically logged
; out when the extension that they are at returns a CHANUNAVAIL
; status when a call is attempted to be sent there.
; Default is "no".
;autologoffunavail=yes
; Define ackcall to require an acknowledgement by '#' when
; an agent logs in using agentcallbacklogin.  Default is "no".
;ackcall=no
; Define endcall to allow an agent to hangup a call by '*'.
; Default is "yes". Set this to "no" to ignore '*'.
;endcall=yes
; Define wrapuptime.  This is the minimum amount of time when
; after disconnecting before the caller can receive a new call
; note this is in milliseconds.
;wrapuptime=5000
; Define the default musiconhold for agents
; musiconhold => music_class
;musiconhold => default
;
; Define the default good bye sound file for agents
```

```
; default to vm-goodbye
;agentgoodbye => goodbye_file
; Define updatecdr. This is whether or not to change the source
; channel in the CDR record for this call to agent/agent_id so
; that we know which agent generates the call
;updatecdr=no
;
; Group memberships for agents (may change in mid-file)
;
;group=3
;group=1,2
;group=
```

11.7.4 Create the agents

agents.conf

```
;agent => agentid,agentpassword,name
[agents]
agent => 300,300,Test Rep - 300
agent => 301,301,Test Rep . 301
agent => 600,600,Test Ver - 600
agent => 601,601,Test Ver . 601
```

11.7.5 Put the queue in the dialplan

extensions.conf

```
; Telemarketing queue.
exten=>_0800XXXXXXX,1,Answer
exten=>_0800XXXXXXX,2,SetMusicOnHold(default)
exten=>_0800XXXXXXX,3,Set(TIMEOUT(digit)=5)
exten=>_0800XXXXXXX,4,Set(TIMEOUT(response)=10)
exten=>_0800XXXXXXX,5,Background(welcome)
exten=>_0800XXXXXXX,6,Queue(telemarketing)

; Transfer to the queue auditing
exten => 8000,1,Queue,(auditing)
exten => 8000,2,Playback(demo-echotest); No auditor available
exten => 8000,3,Goto(8000,1) ; Verify auditor again

; Agent login for the telemarketing and auditing queues

exten => 9000,1,Wait(1)
exten => 9000,2,AgentLogin()
```

11.7.6 Configure queue recording

Calls may be recorded using Asterisk's monitor/MixMonitor application. This may be enabled from within the Queue application, beginning the recording

when the call is actually picked up. Only successful calls are recorded and no recordings are performed while people are listening to MOH.

To enable monitoring, simply specify "monitor-format"; this feature is otherwise disabled.

You can set the filename for the recording using “Set (MONITOR_FILENAME=<filename>)”; otherwise it will use MONITOR_FILENAME=${UNIQUEID}.

queues.conf

```
;
monitor-format = wav
;
monitor-type = MixMonitor
;
monitor-join = yes
```

11.8 QUEUE OPERATION

Working example:

Part 1: Agent login

Example: An agent in the telemarketing queue picks up the phone and dials #9000. The agent hears an invalid login message and is asked for his/her name and password. The auditing queue follows the same procedure.

Part 2: Queue

Once in the queue, the agent will hear MOH if defined. When a call comes in the telemarketing queue, the agent will hear a “beep” and will be connected to that call.

Part 3: Call ending

When the agent finishes the call, he/she can:

- Press ‘*’ to disconnect and stay in the queue.
- Disconnect the phone, disconnecting to the queue.
- Press #8000 to transfer the call for auditing.

11.9 ADVANCED RESOURCES

11.9.1 User menu

You can define a menu for a user while waiting in the queue. You can use one-digit extensions. To enable this option, define a context in the queue configuration (queues.conf).

11.9.2 Penalty

Agents can be configured with a penalty. A queue will send the calls first for users with lower penalty values. Let me exemplify. Since we know that our costumers love Susan with her soft voice, we may choose to assign priority 0 to her). Alternatively, an agent named Uber with less experience is less preferred for costumer service, and therefore we may assign a priority 10 for this agent.

Queues.conf

```
[costumerservice]
member=300,0,Susan the excelent agent
member=300,10,Uber the new guy
```

11.9.3 Priority

Queues operate in the FIFO (first in first out) mode. If you want to give priority for special costumers (platinum, gold) you can set up differentiated priorities.

Platinum or Gold costumers

```
exten=>111,1,Playback(welcome)
exten=>111,2,Set(QUEUE_PRIO=10)
exten=>111,3,Queue(costumerservice)
```

Blue Costumers

```

exten=>112,1,Playback(welcome)
exten=>112,2,Set(QUEUE_PRIO=5)
exten=>112,3,Queue(costumerservice)
```

11.10 QUESTIONS

1. Cite four strategies for routing call in a queue.

2. It is possible to record a conversation between an agent and a costumer using the statement _____ in the queues.conf file.

3. To login an agent you will use the application agentlogin([agentnumber]). When the agent finishes the call, he can press:

a) * to disconnect and stay in the queue
b) hang-up the phone and disconnect from the queue
c) Press #8000 to transfer to call audit
d) Press # to hang-up

4. The required tasks to configure a call queue are:

a) Create the queue
b) Create the agents
c) Configure the agents
d) Configure the recording
e) Put the queue in the dial plan

5. What's the difference between the applications AgentLogin() and AgentCallBackLogin().

6. When in a call queue, you can define a certain number of options that the user can dial. This is done including a _____ in the file queues.conf

a) Agent
b) Menu
c) Context
d) Application

7. The support applications AddQueueMember(), AgentLogin(), AgentCallBackLogin e RemoveQueueMember() should be included in the: _____

a) Dial plan
b) Command line interface
c) queues.conf
d) agents.conf

8. It is possible to record the agents, but it is necessary an external recorder.

a) True
b) False

9. "Wrapuptime" is the time the user needs after ending the call to complete business process related to that call.

a) True
b) False

10. A call can be prioritized depending on the CallerID inside the same queue: The affirmative is:

a) True
b) False

Chapter 12

Voicemail

12.1 OBJECTIVES

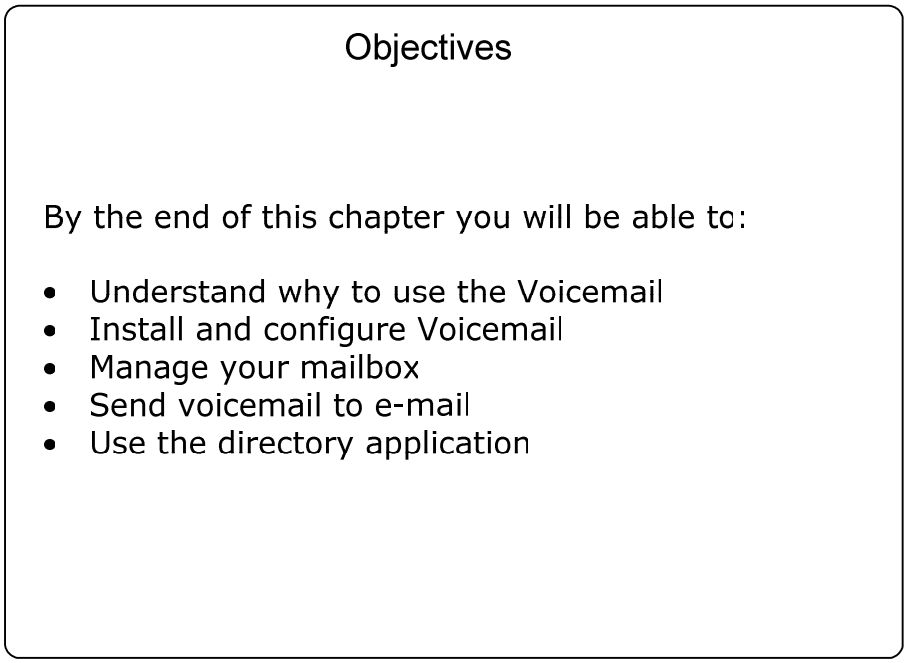


Figure 12.1 Objectives

12.2 INTRODUCTION

Voicemail is a computerized telephone answering system that records incoming voice messages, and saving them on disk or sending them via e-mail. It can be used to page a user; sometimes it has a directory where you can lookup voicemails by name. In the past, voicemail systems were very expensive. Now with IP telephony, voicemail systems became a standard feature.

12.3 CONFIGURATION TASK LIST

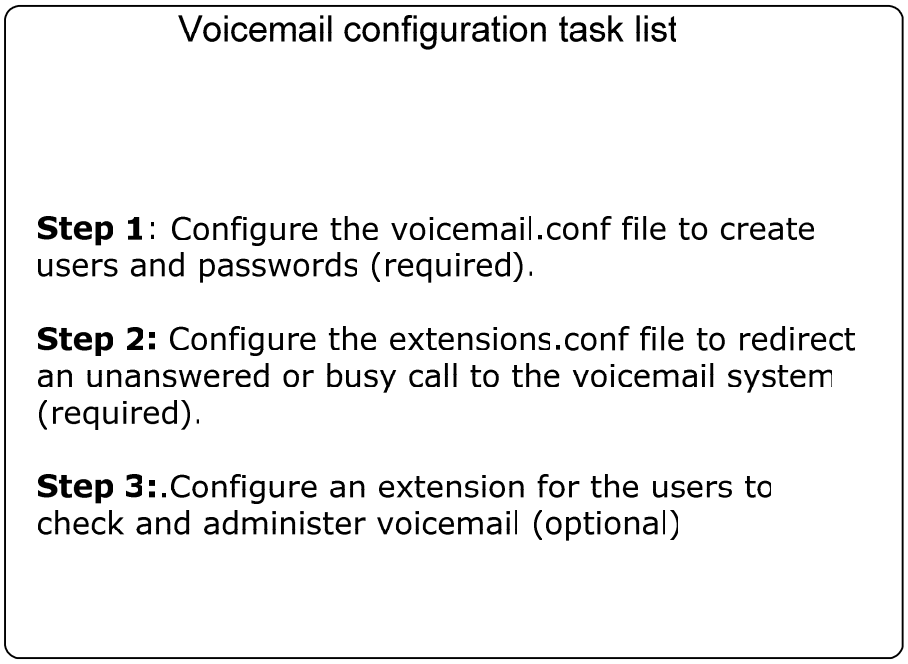


Figure 12.2 Configuration task list

To configure voicemail you should go through the following steps:

12.3.1 Configuring voicemail.conf

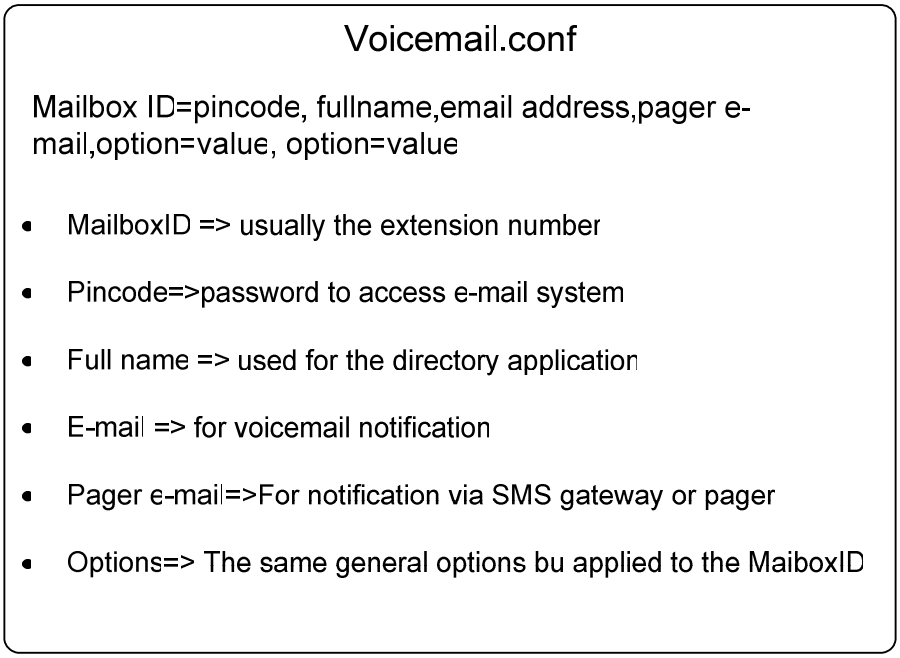


Figure 12.3 mailbox configuration

Voicemail has several options that control voicemail behavior. For now, we will stick to the default options and concentrate in mailbox definition. After the [general] section in the file you will start configuring the mailbox id in its own context.

Example:

```
[general]

[default]
1234=>1234,SomeUser,email@address.com,pager@address.com,saycid=yes|dialout=fro
mvm|callback=fromvm|review=yes|operator=yes
```

More options later.

12.3.2 Configuring the extensions.conf file

Voicemail Macro

```
;usally we will use a macro to process the voicemail.

[macro-stdexter]
exten=>s,1,Dial(${ARG1},20,t)
exten=>s,n,Goto(${DIALSTATUS})
exten=>s,n,hangup()
exten=>s,n(BUSY),voicemail(b${MACRO_EXTEN})
exten=>s,n,hangup()
exten=>s,n(NOANSWER),voicemail(u${MACRO_EXTEN})
exten=>s,n,hangup()
exten=>s,n(CANCEL),hangup
exten=>s,n(CHANUNAVAIL),hangup
exten=>s,n(CONGESTION),hangup

[local]
exten=>6601,1,Macro(stdexter,SIP/6601)
exten=>6602,1,Macro(stdexter,SIP/6602)
```

Figure 12.4 Voicemail Macro

12.3.3 Using the VoiceMailMain() application

```
exten=>9000,1,VoiceMailMain()
```

Dialing 9000 gives you access to the voicemail menu where you can search, retrieve and administer your voicemail mailbox.

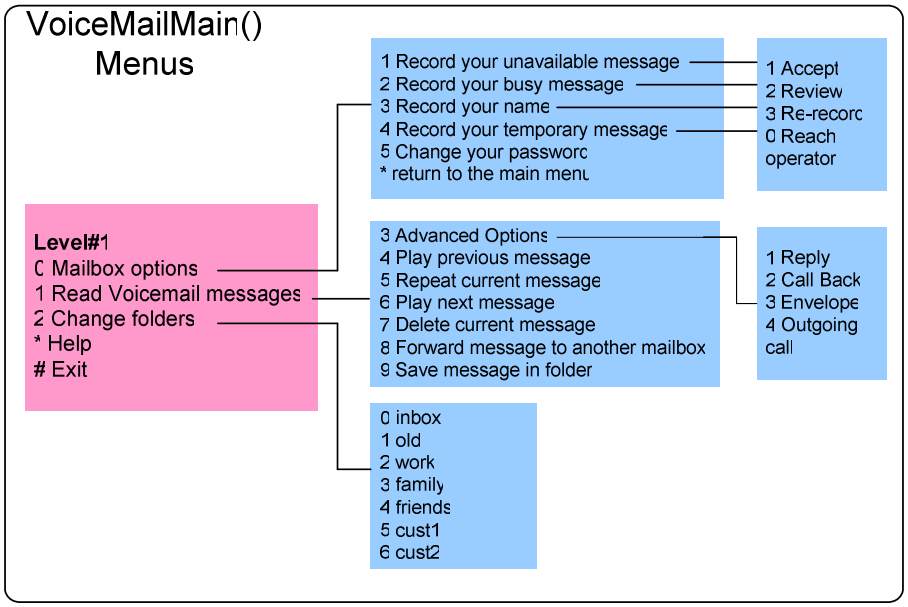


Figure 12.5 Voicemailmain menu

12.3.4 Voicemail application syntax

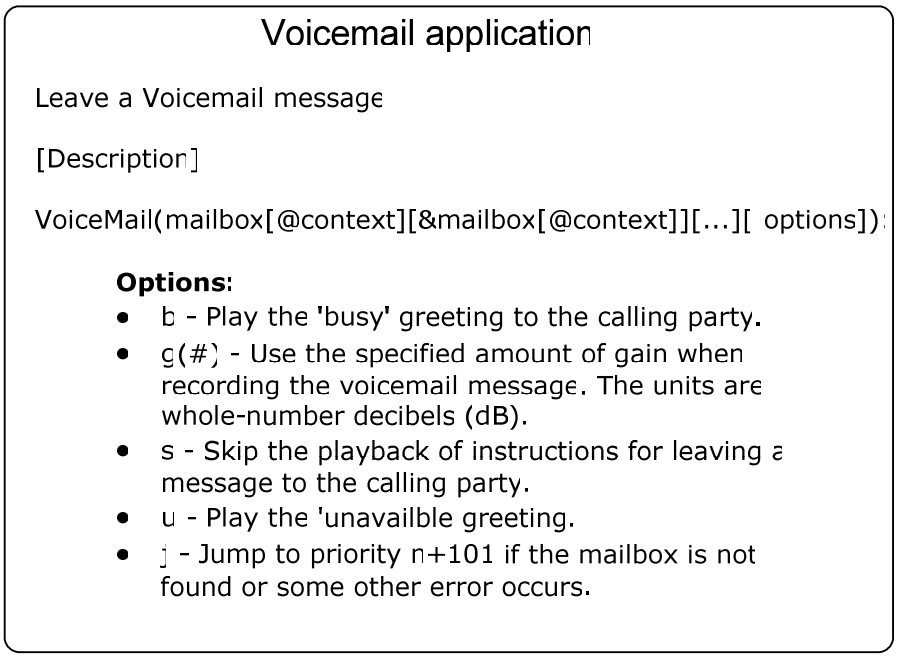


Figure 12.6 Voicemail application

This application allows the calling party to leave a message for a specified list of mailboxes. When multiple mailboxes are specified, the greeting will be taken from the first specified mailbox. Dialplan execution will stop if the specified mailbox does not exist. Voicemail application will exit if any of the following DTMF digits are received:

- 0 - Jumps to the 'o' extension in the current dialplan context.
- * - Jumps to the 'a' extension in the current dialplan context.

This application will set the following channel variable upon completion:

- VMSTATUS - This indicates the execution status of the VoiceMail application. The possible values are:
  - SUCCESS
  - USEREXIT
  - FAILED

Options:

- b - Plays the 'busy' greeting to the calling party.
- g(#) - Uses the specified amount of gain when recording voicemail messages. The units are whole-number decibels (dB).
- s - Skips the playback of instructions for leaving a message to the calling party.
- u - Plays the 'unavailble greeting.
- j - Jumps to priority n+101 if the mailbox is not found or some other error occurs.

In all cases, the beep.gsm file will be played before the recording begins. Voicemail messages will be stored in the inbox directory.

```
/var/spool/asterisk/voicemail/context/boxnumber/INBOX/
```

If a caller presses 0 (zero) during the announcement, it will be moved to the ‘o’ (out) extension in the voicemail current context. This can be used to exit to the operator.

If during the recording the caller presses # or the silence limit times out, recording is stopped and the call goes to the next priority. Make sure that you handle the call after the voicemail is played, as shown below.

```
exten=>somewhere,5,Playback(Goodbye)
exten=>somewhere,6,Hangup
```

12.4 SENDING VOICEMAIL TO E-MAIL

In some cases (like mine), we simply do not use the voicemailmain() application to read e-mail. It is simpler and practical to send all messages to e-mail with the audio attached. Using the parameters 'attach' and 'delete' you can send all mails to e-mail and delete them from the mailbox.

```
attach=yes
delete=yes
```

To send voicemail to e-mail, the voicemail application uses the MTA (Message Transfer Agent) a component of your operating system. Debian uses Exim as the MTA. The application that sends the e-mail is defined in the 'mailcmd' parameter.

```
mailcmd =/usr/sbin/sendmail -t
```

In the Debian distribution of Linux, the MTA is exim. To configure exim in Debian, use:

```
dpkg-reconfigure exim4-config
```

You can choose to make your MTA send e-mail directly through SMTP or a smarthost (probably your company mail server). Verify with your e-mail administrator the best way to send e-mail from the Asterisk server to your e-mail server.

12.4.1 Customizing the e-mail message

You can control how messages are sent by setting-up the following variables:

Variables for e-mail subject and e-mail body:

- VM_NAME
- VM_DUR
- VM_MSGNUM
- VM_MAILBOX
- VM_CIDNUM
- VM_CIDNAME
- VM_CALLERID
- VM_DATE

The e-mail body and e-mail subject will be created as described below. You can modify the e-mail body and subject, but the size limit of the message is 512 bytes.

emailsubject=[PBX]: New message ${VM_MSGNUM} in mailbox ${VM_MAILBOX}. The following definition is very close to the default, but the default shows just the CIDNAME, if it is not null, otherwise just the CIDNUM, or "an unknown caller", if they are both null.

emailbody=Dear ${VM_NAME}:\n\n\tjust wanted to let you know you were just left a ${VM_DUR} long message (number ${VM_MSGNUM})\nin mailbox ${VM_MAILBOX} from ${VM_CALLERID}, on ${VM_DATE}, so you might\nwant to check it when you get a chance. Thanks!\n\n\t\t\t\t--Asterisk\n

12.5 VOICEMAIL WEB INTERFACE

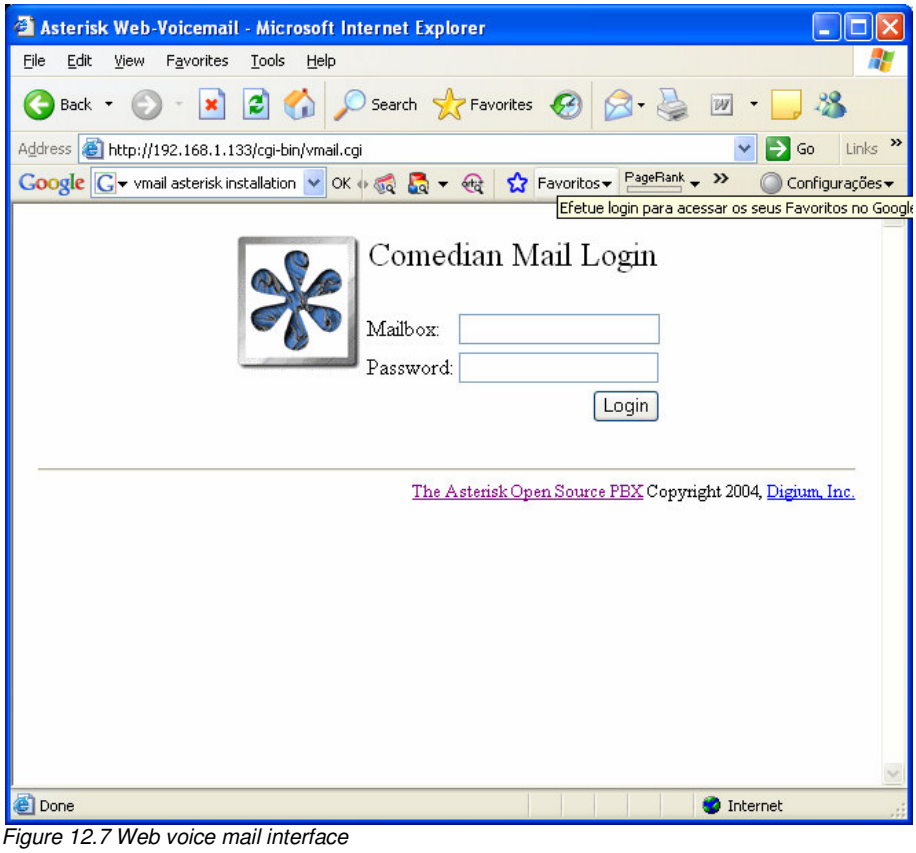


Figure 12.7 Web voice mail interface

There is a Perl script in the source distribution called vmail.cgi located in /usr/src/asterisk/vmail.cgi.

The command "make install" does not install this interface unless you run make webvmail. This script requires the Perl command interpreter and Apache installed in the server.

For the Debian distribution, detailed instructions can be found at

http://www.voip-info.org/wiki/index.php?page=Asterisk+gui+vmail.cgi

```
make webvmail
```

Maybe you will need to edit this script before installing. Copy the source files to the html directory.

```
cp /usr/asterisk/images/*.gif  /var/html/asterisk
```

Use the command below to make the cgi executable.

```
chmod +x vmail.cgi
```

12.6 VOICEMAIL NOTIFICATION

You can configure voicemail to send a notify message to your phone when you have new voicemail. Voicemail notification works with SIP phones, ZAP phones, and also with some IAX2 phones. To indicate an unheard voicemail, an indicator light may blink or the phone may play a shutter tone.
You need to configure the mailbox in the corresponding channel configuration file.

```
sip.conf
mailbox=8590
```

```
zapata.conf
mailbox=8580
```

12.6.1 Lab. Message Notification in the Phone

This lab was tested using xlite for windows 3.0.

1. Edit the sip.conf and add 'mailbox=4401' in the sip channel named 4401.

2. Edit the extensions.conf and create an extension to record a voicemail to 4401 extensions.

```
exten=9008,n,voicemail(b4401)
```

3. Go to the CLI> console and reload.

4. Go to the X-Lite > Mouse Right Button >SIP Account Settings >Properties >Voicemail and check the box 'check voicemail'.

5. Dial 9008 and leave a message.

6. Observe the message icon on the phone.

12.7 USING THE DIRECTORY APPLICATION

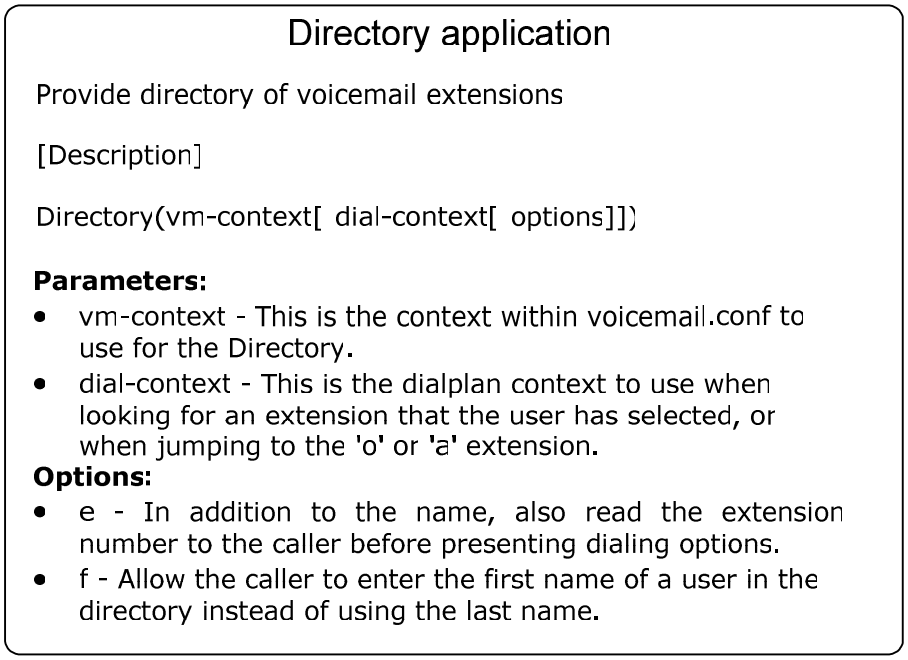


Figure 12.8 The directory application

This application will present the calling channel with a directory of extensions from which they can search by name. The list of names and corresponding extensions is retrieved from the voicemail configuration file, voicemail.conf. This application will immediately exit if one of the following DTMF digits are received and the extension to jump to exists:

- 0 - Jumps to the 'o' extension, if it exists.
- * - Jumps to the 'a' extension, if it exists.

Parameters:

- vm-context - This is the context within voicemail.conf to use for the directory.
- dial-context - This is the dialplan context to use when looking for an extension that the user has selected, or when jumping to the 'o' or 'a' extension.

Options:

- e - In addition to the name, also reads the extension number to the caller before presenting dialing options.
- f - Allows the caller to enter the first name of a user in the directory instead of using the last name.

12.7.1 Lab. using the directory application

1. Edit the voicemail.conf file to add two extensions in the dial plan

```
[default]
; Define maximum number of messages per folder for a particular context.
;maxmsg=50
4400=>4400,Clint Eastwood,ceastwood@asteriskguide.com
4401=>4401,John Wayne,jwayne@asteriskguide.com
```

2. Create these extensions in your dial plan

```
exten=9006,1,VoiceMailMain()
exten=9006,n,Hangup()
exten=9007,1,Directory(default|default)
exten=9007,n,Hangup()
```

3. Go to the console and reload

4. Dial 9006 and record a name for each extension (4400, 4401)

5. Dial 9007 and select the three letters of the last name for one extension (Eas=327). If this is the correct option, press '1' to transfer to the name.

12.8 SUMMARY

In this chapter we learned how to setup a voicemail system and how to use support applications like Voicemailmain(). You learned, for example that you need an MTA configured to send voicemail to e-mail. Finally, you have configured the directory application to easily find names in your directory.

12.9 QUESTIONS

1. The files involved in the voicemail configuration are:

a) sip.conf
b) iax.conf
c) asterisk.conf
d) voicemail.conf
e) vmail.conf
f) extensions.conf

2. In the voicemail application, the parameters "u" and "b" are _____ and _____ respectively. They are used to determine what message will be played.

a) BUSY, FREE
b) BUSY, UNASWERED
c) UNANSWERED, BUSY
d) FREE, ARRESTED

3. The VoiceMailMain() application is used for the caller to leave a message in the voicemail. The affirmative is:

a) True
b) False

4. To exit VoiceMailMain you should press:

a) 0
b) *
c) #
d) 9999

5. Write below the voicemail() application syntax.

_____

_____

_____

6. In the [general] section of the voicemail.conf file the parameter "attach=yes" makes Asterisk to send a notification by e-mail to the user with the audio file attached. The affirmative is:

a) False
b) True

7 The option "delete" makes that every message after being sent to the e-mail be erased from the mailbox.

a) False
b) True

8. The best format for voicemail audio is "WAV". It better support in Windows workstations.

a. False
b. True

9. It is possible to customize e-mail messages by modifying the e-mail subject and body. What variable can be used to indicate a CallerID in the message?

_____

10. The cgi name to install the web voicemail interface is _____.

Chapter 13

Asterisk Call Detail Records

13.1 Introduction

Asterisk, like other telephony platforms, allows billing of phone calls. Several programs in the market can import the records generated by PBXs. Those records are used to verify the correct amount of the bill, statistics, among other things.

13.2 Objectives

By the end of this chapter, the reader should be able to:

- Describe where and in what format the records are generated
- Generate records in MySQL database
- Implement an authentication scheme integrated with billing

13.3 Asterisk CDR Format

Asterisk generates a CDR (Call Detail Record) for each call. These records are stored, by default, in a text file in a CSV (comma separated value) in the /var/log/asterisk/cdr-csv. The file is organized in the following fields:

CDR Fields

CDR	Description	Type	Size
Accountcode	Account Number to use	String	20
Src	Caller ID Number	String	80
Dst	Destination Extension	String	80
Dcontext	Destination Context	String	80
Clid	Caller ID with Text	String	80
Channel	Channel Used	String	80
Dstchannel	Destination channel	String	80
Lastapp	Last application	String	80
Lastdata	Last application data	String	80
Start	Start of call	Date/Time	
Answer	Answer of call	Date/Time	
End	End of Call	Date/Time	
Duration	Time, from dial to hang-up	Integer	

	(seconds)		
Billsec	Time, from answer to hang-up (seconds)	Integer	
Disposition	What Happened to the call (ANSWERED, NO ANSWER, BUSY, FAILED)	String	20
Amaflags	Flags (DOCUMENTATION, BILLING, IGNORE)	String	20
User field	User defined field	String	255

Sample of csv file imported into a table.

AccountCode	CallerID No.	Extension	Context	CallerID text	Src	Dst
1234	4830258576	*72*1234*8584	admin	"Alexandre Keller" <4830258576>	SIP/8576-5f30	SIP/8584-9153
1234	4830258576	*72*1234*8584	admin	"Alexandre Keller" <4830258576>	SIP/8576-96f5	SIP/8584-3312
1234	4830258576	*72*1234*8584	admin	"Alexandre Keller" <4830258576>	SIP/8576-74ac	SIP/8584-297b
1234	4830258576	2012348584	admin	"Alexandre Keller" <4830258576>	SIP/8576-2c5d	SIP/8584-9870
1234	4830258584	2012348576	default	"Luiz Eduardo Dagios" <4830258584>	SIP/8584-03fd	SIP/8576-645c

Application	Appdata	Start	Answer	End	Dur	Bil	Disposition	Amaflags
Dial	SIP/8584|30|tT	27/3/2006 16:05	27/3/2006 16:05	27/3/2006 16:05	5	3	ANSWERED	DOCUMENTATION
Dial	SIP/8584|30|tT	27/3/2006 16:16	27/3/2006 16:16	27/3/2006 16:16	6	4	ANSWERED	BILLING
Dial	SIP/8584|30|tT	27/3/2006 16:22	27/3/2006 16:22	27/3/2006 16:22	9	5	ANSWERED	BILLING
Dial	SIP/8584|30|tT	27/3/2006 16:37	27/3/2006 16:37	27/3/2006 16:37	5	2	ANSWERED	BILLING
Dial	SIP/8576|30|tT	27/3/2006 16:37	27/3/2006 16:37	27/3/2006 16:37	9	5	ANSWERED	BILLING

13.4 ACCOUNT CODES AND AUTOMATED MESSAGE ACCOUNTING

You can specify account codes and "ama" flags on each channel. Usually this is done on the channel configuration file like zapata.conf, sip.conf, iax.conf and others.

Amaflags define what to do with the cdr record. Amaflag values are:

- Default:
- Omit:
- Billing:
- Documentation:

Similarly to the way a record can be flagged for billing or documentation, an account code can be set to each record. The account is a 20 characters string. Usually, it is used to assign a record to a department or business unit.

Example: (sip.conf)

```
[8576]
amaflags=default
accountcode=Support
type=friend
username=8576
```

13.5 CHANGING THE CDR FORMAT.

You can change the csv format changing the cdr_custom.conf file.

```
;
; Mappings for custom config file
;
[mappings]
Master.csv =>
"${CDR(clid)}","${CDR(src)}","${CDR(dst)}","${CDR(dcontext)}","${CDR(channel)}
","${CDR(dstchannel)}","${CDR(lastapp)}","${CDR(lastdata)}","${CDR(start)}","$
{CDR(answer)}","${CDR(end)}","${CDR(duration)}","${CDR(billsec)}","${CDR(dispo
sition)}","${CDR(amaflags)}","${CDR(accountcode)}","${CDR(uniqueid)}","${CDR(u
serfield)}"
```

You can change the CDR format in the cdr_custom.conf file.

13.6 CDR STORAGE

The storage of cdr can be done in several ways. The most important are CSV text files that can be imported easily into spreadsheets. For small businesses, this is usually ok. Some billing software accepts, by default, csv files. However, storing in a database is a lot better and safer, and Asterisk support several database flavors. There are some graphical interfaces for billing in the market. At V'Office we have tested the open source versions of A2billing and AsteriskStats from Areski (www.areski.net). In my opinion, they are very good.

13.6.1 Storage drivers available

cdr_csv – Comma Separated Value text files
cdr_SQLite – SQLite databases
cdr_pgsql – Postgres databases
cdr_odbc – unixODBC supported databases
cdr_mysql – MySQL databases
cdr_FreeTDS – Sybase and MSSQL databases
cdr_yada – yada databases
cdr_manager – CDR to Manager Interface
cdr_radius – CDR radius interface

CDR recording is done to all active modules loaded in /etc/asterisk/modules.conf. If the "autoload=yes" option is set, all modules are loaded.

13.6.2 CSV Storage

As we said before, by default, Asterisk sends all CDR to a csv text file. This is done by the cdr_csv.so module. If you can't see the files in the /var/log/asterisk/cdr-csv, check to see if the module is being loaded using the CLI command "module show". If it's not loaded, check modules.conf.

13.6.3 Storing in MySQL database

Due to licensing restrictions of MySQL, Digium cannot bundle the database with Asterisk. That's why MySQL support for CDR is in the Asterisk add-ons. You will have to download, uncompress and compile the module separately. Follow the instructions below to install MySQL Support.

Step 1: Install MySQL and MYSQL-devel packages.

```
#apt-get install mysql-server-4.1
#apt-get install libmysqlclient12-dev
#cd /usr/src
#wget http://ftp.digium.com/pub/asterisk/releases/asterisk-addons-1.4.1.tar.gz
#tar -xzvf asterisk-addons-1.4.1
#cd asterisk-addons-1.4.1
#make clean
#make
#make install
```

Step 2: Make the necessary adjustments on cdr_mysql.conf file. This configuration has to point to where the database will be located.

```
[global]
hostname=localhost
dbname=asteriskdb
password=asterisk
user=asterisk
port=3306
sock=/var/run/mysqld/mysqld.sock
;userfield=1
```

Use vi (or your preferred editor) to edit modules.conf to include the cdr_addon_mysql.so loading. In most cases you don't need to do it as the option autoload=yes is default.

Step 3: Create a database for cdr_addon_mysql

```
mysql -p
```

Just press <ENTER> when asked for the password.

```
CREATE DATABASE asteriskdb;

GRANT INSERT
  ON asterisk.*
  TO asterisk@localhost
  IDENTIFIED BY 'asterisk';

USE asteriskdb;

CREATE TABLE `cdr` (
`calldate` datetime NOT NULL default '0000-00-00 00:00:00',
`clid` varchar(80) NOT NULL default '',
`src` varchar(80) NOT NULL default '',
`dst` varchar(80) NOT NULL default '',
`dcontext` varchar(80) NOT NULL default '',
`channel` varchar(80) NOT NULL default '',
`dstchannel` varchar(80) NOT NULL default '',
`lastapp` varchar(80) NOT NULL default '',
`lastdata` varchar(80) NOT NULL default '',
`duration` int(11) NOT NULL default '0',
`billsec` int(11) NOT NULL default '0',
`disposition` varchar(45) NOT NULL default '',
`amaflags` int(11) NOT NULL default '0',
`accountcode` varchar(20) NOT NULL default '',
`userfield` varchar(255) NOT NULL default ''
);

ALTER TABLE `cdr` ADD INDEX ( `calldate` );
ALTER TABLE `cdr` ADD INDEX ( `dst` );
ALTER TABLE `cdr` ADD INDEX ( `accountcode` );
```

One tip is to copy and paste these commands to a text file named “cdr.sql” and execute the following command:

```
mysql --user=username --password=password asteriskdb <cdr.sql
```

13.7 APPLICATIONS AND FUNCTIONS

Several applications are related to billing.

13.7.1 CDR(accountcode)

Sets an account code before calling other application as Dial() for example.

Format:

```
Set(CDR(accountcode)=account)
```

The account code can be verified using the channel variable ${CDR(accountcode)}

13.7.2 CDR(amaflags)

```
Set(CDR(amaflags=amaflags)
```

Set a flag for billing purposes. Options are default, omit, documentation, billing.

13.7.3 NoCDR()

No CDR recorded to the file or database.

13.7.4 ResetCDR()

Resets to zero the CDR. If ´w´ option is set it saves the original CDR record.

13.7.5 Set(CDR(userfield)=Value)

This command sets a user field in the CDR. If you were using cdr_addon_mysql, check to see if you have the option userfield=1 in the cdr_mysql.conf. For csv text files, you have to edit the source code (cdr_csv.c) and recompile Asterisk if you want to use userfields. This command is useless if userfield is not enabled in the source code or in the myssql configuration file (cdr_mysql.conf).

13.7.6 AppendCDRUserField(Value)

Append data to the user field on the CDR.

13.8 USER AUTHENTICATION

Some companies use to bill the calls to their employees. In Asterisk you can set an authentication scheme that allows to bill the authenticated user on the CDR. This authentication can be done by a password passed as a parameter to the Authenticate application, a password file, indicated by a ´/´ (slash) before the parameter or a Asterisk database (dbput/dbget).

Format:

```
Authenticate(password[|options])
Authenticate(/passwdfile|[|options])
Authenticate(</db-keyfamily|d>options)
```

Options:

- a – Set the account code as the password.
- d – Interpret the parameter as a Asterisk DB key
- r – Removes the key after successful authentication (only with ´d´ option)
- j – Jump to priority n+101 for invalid authentication

Example: (International Calls)

```
exten=_9011.,1,Authenticate(/password|daj)
exten=_9011.,2,Dial(Zap/g1/${EXTEN:1},20,tT)
exten=_9011.,3,Hangup()
exten=_9011.,102,Playback(unauthorized)
exten=_9011.,103,Hangup()
```

To insert the password in a DB key from the console.

```
CLI> database put senha 123456 1
```

13.9 USING PASSWORDS FROM VOICEMAIL.

This application does the same as authenticate, but uses the voicemail configuration file for the password.

```
VMAuthenticate([mailbox][@context][|options])
```

If a mailbox is specified, only the mailbox password will be considered valid. If the mailbox is not specified, a channel variable AUTH_MAILBOX will be set with the authenticated mailbox. If the option ´s´ (silent) is set no prompt will be executed.

Example: (International Calls)

```
exten=_9011.,1,VMAuthenticate(${CALLERID(num)}@local|ajs)
exten=_9011.,2,Dial(Zap/g1/${EXTEN:1},20,tT)
exten=_9011.,3,Hangup()
exten=_9011.,102,Playback(unauthorized)
exten=_9011.,103,Hangup()
```

13.10 SUMMARY

In this chapter we have learned how to implement CDR recording in text files and in a mysql database, how to set amaflags and account codes. At the end of the chapter we have learned how to use an authentication scheme integrated with cdr and billing.

13.11 QUESTIONS

1. By default, Asterisk records the CDR in /var/log/asterisk/cdr-csv directory.

a. False
b. True

2. Asterisk allows using only these databases:

a. MySQL
b. Native Oracle
c. Microsoft SQL
d. CSV Text files
e. unix_ODBC supported databases

3. Asterisk generates a CDR only to single kind of storage.

a. False
b. True

4. Which are Asterisk amaflags available?

a. Default
b. Omit
c. Tax
d. Rate
e. Billing
f. Documentation

5. Fill the spaces left.

If you intend to associate a department to a CDR, you should use the command _____. The account code can be verified using the channel variable _____.

6. The difference between the applications NOCDR() and Reset CDR() is that NoCDR() does not generate any record and ResetCDR() zeroes the current record.

a. False
b. True

7. To use a user defined field with the cdr_csv.so module, is necessary to edit the source code and recompile the Asterisk.

a. False
b. True

8. The three authentication methods available to the Authenticate() application are:

a. Password
b. Password file
c. Asterisk DB (dbput e dbget)
d. Voicemail

9. Voicemail password are specified in a different section of the voicemail.conf file and are not the same as the voicemail users.

a. False
b. True

10. This option of authenticate command put the password used to authenticate in the CDR.

a. a
b. j
c. d
d. r

Chapter 14

Extending Asterisk with AMI and AGI

14.1 Introduction

In several situations it may be necessary to extend the features of Asterisk by using external applications. With conventional PBXs, this was normally done using a CTI (Computer Telephony Integration) interface. Asterisk is built in a computer and not based on a circuit switch; therefore, there are many different ways by which it may be extended. In this chapter we will cover Asterisk's CTI interface named AMI (Asterisk Manager Interface).

Since there are other ways to integrate Asterisk with other programs, we will also look at the command "asterisk –rx" and the System() application. At the end of this chapter, we will look at the powerful AGI (Asterisk Gateway Interface), which enables Asterisk to call external applications made with any languages that support Linux standard I/O, a much used resource to build IVRs (Interactive Voice Response). The only drawback in Asterisk integration is that it does not have a standard CSTA interface, which could make it easier for other programs to port applications like dialers, report generators and others. The abbreviations AMI (Asterisk Manager Interface) and AGI (Asterisk Gateway Interface) will be used throughout this chapter.

14.2 Objectives

By the end of this chapter the reader should be able to:

- Describe access options to external programs
- Use "asterisk –rx" command to execute a console command
- Use the System() app to call external programs in the dialplan
- Explain what is AMI and how it works
- Configure the manager.conf file and enable AMI
- Execute an Asterisk AMI command from a PHP program
- Explain what Asterisk Manager Proxy is and how it works
- Describe different AGI Flavors (DeadAGI, AGI, EAGI, FastAGI)
- Execute a simple AGI program created with PHP

14.3 Major ways to extend Asterisk

Asterisk has different ways to interface with external programs. In this chapter, we will cover:

- Linux command line and Asterisk Console
- System() Application
- Asterisk Manager Interface - AMI
- Asterisk Gateway Interface - AGI

14.4 EXTENDING ASTERISK WITH CONSOLE CLI

An application can easily call Asterisk from the Linux shell using the following command.

```
asterisk –rx <command>
```

Example:

```
#asterisk –rx “stop now”
```

Even a command with an output can be called:

```
asterisk:~# asterisk -rx "sip show peers"
Name/username              Host            Dyn Nat ACL Port     Status
4000/4000                  10.1.1.6         D          5060     Unmonitored
1 sip peers [1 online , 0 offline]
```

14.5 EXTENDING ASTERISK USING THE SYSTEM() APPLICATION

The System() command enables Asterisk to call an external application.

```
asterisk*CLI> show application system
asterisk*CLI>
  -= Info about application 'System' =-

[Synopsis]
Execute a system command

[Description]
  System(command): Executes a command  by  using  system(). If the command
fails, the console should report a fallthrough.
Result of execution is returned in the SYSTEMSTATUS channel variable:
   FAILURE      Could not execute the specified command
   SUCCESS      Specified command successfully executed
```

14.5.1 System() app example

This application does a screen-pop using netbios WindowsPopup.

```
exten => 9000,1,System(/bin/echo -e "'Incoming Call From -> ${CALLERID(num)}
\\r Received: ${DATETIME}'"|/usr/bin/smbclient -M target_netbiosname)
exten => 9000,2,Dial(SIP/9000,15,t)
exten => 9000,3,Hangup
```

14.6 WHAT IS AMI?

AMI enables a client program to connect to an Asterisk instance and issue commands or read events over a TCP connection. Systems integrators will find these resources useful to track channel states. AMI is built in a simple concept of a line protocol using key:value pairs over TCP. Asterisk by itself is not ready to handle too many connections over this interface. If you have lots of connections to AMI consider using Asterisk Manager Proxy.

14.6.1 What language to use for AMI?

Selecting a programming language can be hard these days. There are simply too many options like Java, php, Perl, C, C#, Python, and several others. It's possible to use AMI with any language that supports a socket or telnet interface. We have chosen php for this book because of the popularity.

14.6.2 AMI protocol behavior

- Before sending any commands to Asterisk, you need to establish an AMI session.
- The first line of a packet will have the key "Action" when sent from a client.
- The first line of a packet will have a key "Response" or "Event" when coming from Asterisk.
- Packages can be transmitted in any direction after the authentication.

14.6.3 Packet types

The type of the packet is determined by the existence of the following keys:

- **Action:** A packet sent from a client connected to AMI asking for a specific action. There is a finite set of actions available to clients. The loaded modules determine these actions. A packet contains the action name and its parameters.
- **Response:** The response sent from Asterisk to the last action sent from the client.
- **Event:** Data belonging to an event generated in the Asterisk core or by a module.

When a client sends packets of the Action type, a parameter named ActionID is included. Since the order in which the responses sent from Asterisk cannot be predicted, ActionID is used to correlate actions and responses.

Event packets are used in two different contexts. Firstly, events inform the client about changes in Asterisk (E.g. newly created channels, channels disconnected, or agents login in and out to a queue). Secondly, events are used to transport responses to a client action.

14.7 CONFIGURING USERS AND PERMISSIONS

To access AMI, it is necessary to establish a TCP connection listening to a TCP port (usually 5038). You will need to configure /etc/asterisk/manager.conf file to create a user account and permissions.

There is a finite set of permissions, "read", "write" or both. These permissions are defined in the manager.conf file.

```
[general]
enabled=yes
port=5038
bindaddr=127.0.0.1

[admin]
secret=senha
read=system,call,log,verbose,command,agent,user
write=system,call,log,verbose,command,agent,user
deny=0.0.0.0/0.0.0.0
permit=127.0.0.1/255.255.255.255
```

14.7.1 Logging into the AMI

To login and authenticate into AMI, you will need to send an Action packet of the login type with a username and account created in the manager.conf.

```
Action:login
Username:admin
Secret:password
```

14.7.2 Example:

Logging into AMI using php

```
<?php

$socket = fsockopen("127.0.0.1","5038", $errno, $errstr, $timeout);
fputs($socket, "Action: Login\r\n");
```

```
fputs($socket, "UserName: admin\r\n");
fputs($socket, "Secret: senha\r\n\r\n");

?>
```

If you don't need to receive the events you can use "Events Off".

```
<?php

$socket = fsockopen("127.0.0.1","5038", $errno, $errstr, $timeout);
fputs($socket, "Action: Login\r\n");
fputs($socket, "UserName: admin\r\n");
fputs($socket, "Secret: senha\r\n\r\n");
fputs($socket, "Events: off\r\n\r\n");

?>
```

14.7.3 Action packets

When you send an action packet to Asterisk, you can provide some extra keys (e.g. called number). You do this by passing key:value pairs after the Action. It is also possible too pass channel and global variables to the dialplan.

```
Action: <action type><CRLF>
<Key 1>: <Value 1><CRLF>
<Key 2>: <Value 2><CRLF>

Variable: <Variable 1>=<Value 1><CRLF>
Variable: <Variable 2>=<Value 2><CRLF>
...
<CRLF>
```

14.7.4 Action commands

You can use the CLI instruction "manager show commands" to list the available actions. In version 1.2.7 the commands were:

Action	Privilege	Synopsis
AbsoluteTimeout	call,all	Sets Absolute Timeout
AgentCallbackLogin	agent,all	Sets an agent as logged in by callback
AgentLogoff	agent,all	Sets an agent as no longer logged in
Agents	agent,all	Lists agents and their status
ChangeMonitor	call,all	Changes monitoring filename of a channel
Command	command,all	Executes Asterisk CLI Command

DBGet	system,all	Gets DB Entry
DBPut	system,all	Puts DB Entry
EventsControl	<none>	Event Flow
ExtensionState	call,all	Checks Extension Status
Getvar	call,all	Gets a Channel Variable
Hangup	call,all	Hang-up Channel
IAXnetstats	<none>	Shows IAX Netstats
IAXpeers	<none>	Lists IAX Peers
ListCommands	<none>	Lists available manager commands
Logoff	<none>	Logoff Manager
MailboxCount	call,all	Checks Mailbox Message Count
MailboxStatus	call,all	Checks Mailbox
Monitor	call,all	Monitors a channel
Originate	call,all	Originates Call
ParkedCalls	<none>	Lists parked calls
Ping	<none>	Keepalive command
QueueAdd	agent,all	Adds interface to queue.
QueuePause	agent,all	Makes a queue member temporarily unavailable
QueueRemove	agent,all	Removes interface from queue.
Queues	<none>	Queues
QueueStatus	<none>	Queue Status
Redirect	call,all	Redirects (transfers) a call
SetCDRUserField	call,all	Sets the CDR UserField
Setvar	call,all	Sets Channel Variable
SIPpeers	System,all	Lists SIP peers (text format)
SIPshowpeer	System,all	Shows SIP peer (text format)
Status	call,all	Lists channel status
StopMonitor	call,all	Stops monitoring a channel

If you need to know specific command parameters, use the "manager show command <command>".

Example:

```
asterisk*CLI> manager show command originate
Action: Originate
Synopsis: Originate Call
Privilege: call,all
Description: Generates an outgoing call to a Extension/Context/Priority or
  Application/Data
Variables: (Names marked with * are required)
        *Channel: Channel name to call
        Exten: Extension to use (requires 'Context' and 'Priority')
```

```
Context: Context to use (requires 'Exten' and 'Priority')
Priority: Priority to use (requires 'Exten' and 'Context')
Application: Application to use
Data: Data to use (requires 'Application')
Timeout: How long to wait for call to be answered (in ms)
CallerID: Caller ID to be set on the outgoing channel
Variable: Channel variable to set, multiple Variable: headers are allowed
Account: Account code
Async: Set to 'true' for fast origination
```

14.7.5 Event packets

Link Events:

Description:

Triggered when two channels are connected and voice transmission starts. More than on event can be triggered for a single call. Any call that needs transcoding will generate two events: the first one is a fail to establish a native bridge between the channels, the second is the call itself.

Example:

```
Event: Link
Channel1: SIP/4001-AAAA
Channel2: SIP/4000-BBBB
Uniqueid1: 1234567890.12
Uniqueid2: 1234567890.12
```

Unlink events:

Description:

Triggered when a link between two channels are disconnected a little before the call is completed.

Example:

```
Event: Link
Channel1: SIP/4001-AAAA
Channel2: SIP/4000-BBBB
Uniqueid1: 1234567890.12
Uniqueid2: 1234567890.12
```

14.7.8 Events available

AbstractAgentEvent	HoldEvent	PeerStatusEvent
AbstractParkedCallEvent	JoinEvent	QueueEntryEvent
AbstractQueueMemberEvent	LeaveEvent	QueueEvent
AgentCallbackLoginEvent	LinkageEvent	QueueMemberAddedEvent
AgentCallbackLogoffEvent	LinkEvent	QueueMemberEvent
AgentCalledEvent	LogChannelEvent	QueueMemberPausedEvent
AgentCompleteEvent	ManagerEvent	QueueMemberRemovedEvent
AgentConnectEvent	MeetMeEvent	QueueMemberStatusEvent
AgentDumpEvent	MeetMeJoinEvent	QueueParamsEvent
AgentLoginEvent	MeetMeLeaveEvent	QueueStatusCompleteEvent
AgentLogoffEvent	MeetMeStopTalkingEvent	RegistryEvent
AgentsCompleteEvent	MeetMeTalkingEvent	ReloadEvent
AgentsEvent	MessageWaitingEvent	RenameEvent
AlarmClearEvent	NewCallerIdEvent	ResponseEvent
AlarmEvent	NewChannelEvent	ShutdownEvent
CdrEvent	NewExtenEvent	StatusCompleteEvent
ChannelEvent	NewStateEvent	StatusEvent
ConnectEvent	OriginateEvent	UnholdEvent
DBGetResponseEvent	OriginateFailureEvent	UnlinkEvent
DialEvent	OriginateSuccessEvent	UnparkedCallEvent
DisconnectEvent	ParkedCallEvent	UserEvent
DNDStateEvent	ParkedCallGiveUpEvent	ZapShowChannelsCompleteEvent
ExtensionStatusEvent	ParkedCallsCompleteEvent	ZapShowChannelsEvent
FaxReceivedEvent	ParkedCallTimeOutEvent	
HangupEvent	PeerEntryEvent	
HoldedCallEvent	PeerlistCompleteEvent	

14.8 ASTERISK MANAGER PROXY

Asterisk was not meant to manage a large number of connections to the manager interface as in a large number of agents and supervisors in a contact center environment. If this happens, Asterisk may become instable. To solve this problem Asterisk manager Proxy was created. The main advantages of Astmanproxy are:

- Single and persistent connection to the Asterisk
- A safer (non-root) TCP interface
- I/O filtering
- Less load and connections to the Asterisk box
- Multiple ways to access (standard, http, xml, csv)
- SSL support
- Connection to multiple Asterisk servers
- I/O formats selectable per client

Astmanproxy supports web based applications using HTTP POST and HTTP GET and receiving the output in HTML. On the other hand, it is possible to use Astmanproxy as an XML feeder to a .NET program that keeps the Asterisk status.

14.8.1 Astmanproxy, Installation and configuration

To install astmanproxy you will need:

Step 1: Download astmanproxy

Use subversion system to download astmanproxy.

svn checkout http://svncommunity.digium.com/svn/astmanproxy/trunk

Step 2: Compile and install

```
make
make install
```

Step 3: Edit the configuration file

vi /etc/asterisk/astmanproxy.conf

```
; astmanproxy.conf
; Asterisk Manager Proxy Configuration Sample
; (C) 2005-2006 David C. Troy - dave@popvox.com

; List of asterisk host(s) you want to proxy
; host = ip_addr, port, user, secret, events, use_ssl
host = localhost, 5038, admin, senha, on, off

; Server reconnect interval (in seconds); how often to retry
; Connecting to an asterisk server whose connection was lost
retryinterval = 2

; Number of times to retry connecting to a given server
; use 0 for infinitely, or some finite number
maxretries = 10
```

Step 4: Edit the other files (optional)

vi /etc/asterisk/astmanproxy.users
vi /etc/asterisk/ssl.conf

Step 5: Start the program

```
#astmanproxy
```

Step 6: To see the output, start astmanproxy in debug mode.

```
#astmanproxy -dddddd
```

Step 7: To start astmanproxy at boot:

Put the following in /etc/rc.d/rc.local.

/usr/local/sbin/astmanproxy

You can find more information about the astmanproxy distribution in the README file.

14.9 ASTERISK GATEWAY INTERFACE

AGI is a gateway interface to Asterisk similar to CGI used by web servers. It allows the use of high level languages like Perl, php and Python to extend Asterisk's functionality. The main application to CGIs is IVRs building.

There are three AGI types.

- Normal AGI, which calls a program inside Asterisk's box.
- Fast AGI, which calls an AGI in another server using TCP sockets.
- EAGI, which enables sound channel access and control from the AGI.
- DEADAGI, which gives access to the channel even after hangup(). Usually called in the 'h' extension.

Application format:

```
asterisk*CLI> core show application agi
asterisk*CLI>
  -= Info about application 'AGI' =-

[Synopsis]
Executes an AGI compliant application

[Description]
  [E|Dead]AGI(command|args): Executes an Asterisk Gateway Interface compliant
program on a channel. AGI allows Asterisk to launch external programs
written in any language to control a telephony channel, play audio,
read DTMF digits, etc. by communicating with the AGI protocol on stdin
and stdout.
Returns -1 on hangup (except for DeadAGI) or if application requested
 hangup, or 0 on non-hangup exit.
Using 'EAGI' provides enhanced AGI, with incoming audio available out of band
on file descriptor 3

Use the CLI command 'agi show' to list available agi commands
```

You can show the available AGI commands using:

```
asterisk*CLI> agi show
              answer   Answer channel
      channel status   Returns status of the connected channel
        database del   Removes database key/value
    database deltree   Removes database keytree/value
        database get   Gets database value
        database put   Adds/updates database value
                exec   Executes a given Application
            get data   Prompts for DTMF on a channel
   get full variable   Evaluates a channel expression
          get option   Stream file, prompt for DTMF, with timeout
        get variable   Gets a channel variable
              hangup   Hangup the current channel
                noop   Does nothing
        receive char   Receives one character from channels supporting it
        receive text   Receives text from channels supporting it
         record file   Records to a given file
           say alpha   Says a given character string
          say digits   Says a given digit string
          say number   Says a given number
        say phonetic   Says a given character string with phonetics
            say date   Says a given date
            say time   Says a given time
        say datetime   Says a given time as specfied by the format given
          send image   Sends images to channels supporting it
           send text   Sends text to channels supporting it
      set autohangup   Autohangup channel in some time
        set callerid   Sets callerid for the current channel
         set context   Sets channel context
       set extension   Changes channel extension
           set music   Enable/Disable Music on hold generator
        set priority   Set channel dialplan priority
        set variable   Sets a channel variable
         stream file   Sends audio file on channel
 control stream file   Sends audio file on channel and allows the listner to
control the stream
            tdd mode   Toggles TDD mode (for the deaf)
             verbose   Logs a message to the asterisk verbose log
      wait for digit   Waits for a digit to be pressed
```

To debug use ‘agi debug’.

14.9.1 Using AGI

In this example, we will use php-cli, the php command line version. Install php-cli if it’s not already installed.

Follow these steps to use php AGI scripts.

Step 1: All AGI scripts are located in /var/lib/asterisk/agi-bin

Step 2: Change the permissions to allow execution.

```
chmod 755 *.php
```

Step 3: Shell interface (php specific)

The script first lines have to be:

```
#!/usr/bin/php -q
<?php
```

Step 4: Opening I/O channels

```
$stdin = fopen('php://stdin', 'r');
$stdout = fopen('php://stdout', 'w');
$stdlog = fopen('agi.log', 'w');
```

Step 5: Managing the Asterisk output

Asterisk sends the information set each time AGI is called.

```
agi_request:testephp
agi_channel: Zap/1-1
agi_language: en
agi_type: Zap
agi_callerid:
agi_dnid:
agi_context: default
agi_extension: 4000
agi_priority: 1
```

Saving the information sent.

```
while (!feof($stdin)) {
  $temp = fgets($stdin);
  $temp = str_replace("\n","",$temp);
  $s = explode(":",$temp);
  $agivar[$s[0]] = trim($s[1]);
  If (($temp == "") || ($temp == "\n")) {
    break;
  }
}
```

The above script will create an array named $agivar. Options available are:

- agi_request – AGI file name
- agi_channel – AGI Originating channel
- agi_language – Language set

- agi_type – Channel type (e.g. SIP, ZAP)
- agi_uniqueid – Unique identifier
- agi_callerid – CallerID (Ex. Flavio <8590>)
- agi_context – Originating context
- agi_extension – Called extensions
- agi_priority – Priority
- agi_accountcode – Originating account code

To call a variable named agi_extensions use $agivar[agi_extensions].

Step 6: Using channel AGI

At this point you can start talking to Asterisk. Use the fputs command to send commands to AGI. You can use the echo command too.

```
fputs($stdout,"SAY NUMBER 4000 '79#' \n");
fflush($stdout);
```

Notes about using quotes:

- AGI command options are not optional
- Some options need to be enclosed in quotes <escape digits>
- Some options should not be enclosed in quotes <digit string>
- Some options can use both formats
- You can use single quotes

Step 7 – Passing variables

Channel variables can be set in the AGI, but cannot be used inside the AGI. The example below does not work inside an AGI.

```
SET VARIABLE MY_DIALCOMMAND "SIP/${EXTEN}"
```

The example below does work:

```
SET VARIABLE MY_DIALCOMMAND "SIP/4000"
```

Step 8: Asterisk responses

The following is necessary to verify responses from Asterisk:

```
$msg  = fgets($stdin,1024);
fputs($stdlog,$msg . "\n");
```

Step 9: Kill the locked (zombie) processes

If your script fails for some reason, the process will hang. Use the killproc command to clean it before testing again.

```
#!/usr/bin/php4 -q
<?php
ob_implicit_flush(true);
set_time_limit(6);
$in = fopen("php://stdin","r");
$stdlog = fopen("/var/log/asterisk/agi.log", "w");

// Enable debug (more verbose)
$debug = false;

// Functions definition
function read() {
  global $in, $debug, $stdlog;
  $input = str_replace("\n", "", fgets($in, 4096));
  if ($debug) fputs($stdlog, "read: $input\n");
  return $input;
}

function errlog($line) {
  global $err;
  echo "VERBOSE \"$line\"\n";
}

function write($line) {
  global $debug, $stdlog;
  if ($debug) fputs($stdlog, "write: $line\n");
  echo $line."\n";
}

// Put agi headers in the array
while ($env=read()) {
  $s = split(": ",$env);
  $agi[str_replace("agi_","",$s[[0]])] = trim($s[[1]);
  if (($env == "") || ($env == "\n")) {
    break;
  }
}

// main program
echo "VERBOSE \"Start here!\" 2\n";
read();
errlog("Call from ".$agi['channel']." - Phone ringing ");
read();
write("SAY DIGITS 22 X"); // X is the escape digit. since X is not DTMF, no e
xit is possible
read();
write("SAY NUMBER 2233 X"); // X is the escape digit. since X is not DTMF, no
exit is possible
read();

// clean up file handlers etc.
fclose($in);
fclose($stdlog);
```

```
exit;
?>
```

14.9.2 DeadAGI

DeadAGI is used when you do not have a live channel. Usually you execute the DeadAGI in the ´h´ extension.

14.9.3 FASTAGI

Fast AGI implements AGI using a TCP port (4573 by default) as the Input/Output channel. FastAGI format is (agi://).

Example:

```
exten => 0800400001, 1, Agi(agi://192.168.0.1)
```

When the TCP connection is lost or disconnected, the AGI ends and the TCP connection is closed followed by call disconnection. This resource is useful to ease the CPU load from your Asterisk server running scripts in an external server. You may obtain more details about FastAGI in the source code directory (please see the file “agi/fastagi-test”).

OrderlyCalls has a Java AGI server that implements Fast AGI for Java. For more information see http://www.orderlycalls.com.

14.10 CHANGING THE SOURCE CODE

Asterisk is developed in C language (not C++). To teach C programming is outside the scope of this document. Anyway, if you are interested, you will find related documentation in:

www.asterisk.org/developers

The above web site has good tips on how to apply and create patches to Asterisk as well API documentation, mostly generated by Doxygen software.

http://www.asterisk.org/doxygen/

For those familiar with C programming, changing the applications source code can be the most powerful (and dangerous) way to extend Asterisk.

14.11 SUMMARY

In this chapter, you have learned how to interface external programs to the Asterisk PBX. We have started with "asterisk –rx" passing commands from the Linux shell to the Asterisk console. Next, we have seen the System() application that allows calling an external program from the dial plan. AMI (Asterisk Manager Interface) is the closest to a CTI interface common in traditional PBXs. To call an application from the dial plan, we have used the AGI, with a taste for its different flavors: DeadAGI for dead channels, EAGI for handling the audio streaming, Fast AGI for using TCP sockets as the input/output interface, and normal AGI that calls and process the scripts inside the same Asterisk box.

14.12 QUESTIONS

1. Which of the following is not an interfacing method for Asterisk?

   a. AMI
   b. AGI
   c. Asterisk –rx
   d. System()
   e. External()

2. AMI (Asterisk Manager Interface) enables passing Asterisk commands via TCP sockets. This resource is enabled by default.

   a. True
   b. False

3. AMI is very safe, because its authentication is done using MD5 challenge/response.

   a. True
   b. False

4. To compensate the lack of security and scalability of AMI, we could use:

   a. AMI does not have any scalability or security problem
   b. Astmanproxy
   c. Sysproxy

5. FastAGI allows the calling of external scripts from the dial plan to an external machine using TCP sockets (usually 4573).

   a. True
   b. False

6. DeadAGI is used in active channels. It can be used in ZAP channels but not in SIP or IAX channels.

   a. True
   b. False

7. Only php can be used for AGI scripting.

   a. True
   b. False

8. The command _____ shows all available AGI commands.

9. The command _____ shows all available AMI commands.

10. To debug an AGI you should use the command _____.

Chapter 15

Asterisk Real-Time

15.1 Introduction

As you know, Asterisks configuration is accomplished with the use of several text files in the /etc/asterisk directory. In spite of the easiness of using text files, there are some known drawbacks:

- Necessity to reload Asterisk each time the files are changed
- Memory usage for large volume of users
- Hard to code a provisioning interface using text files
- No possibility of integration to existing databases

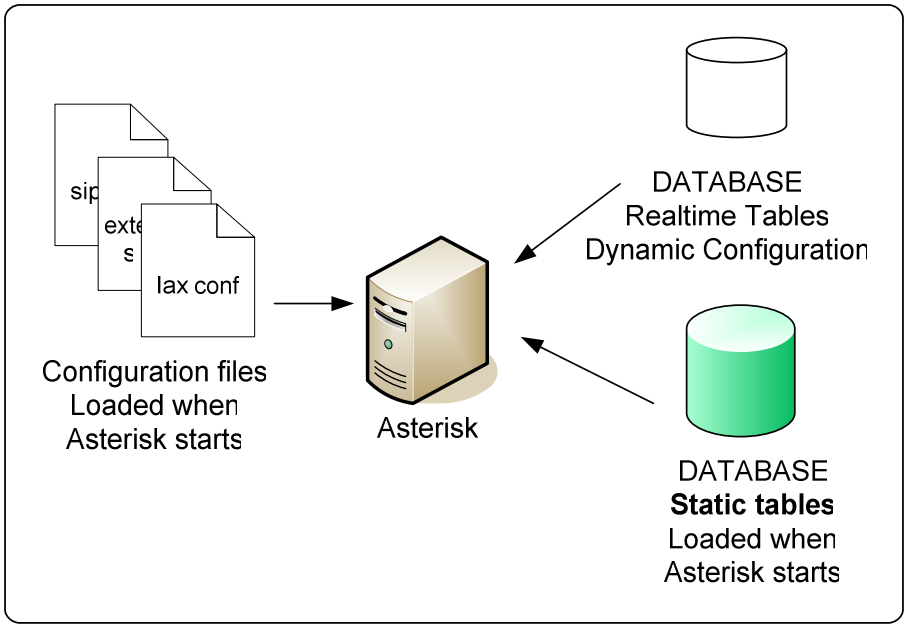
ARA or Asterisk Realtime, as it is known, was created by Anthony Minessale II, Mark Spencer, and Constantine Filin and was designed to allow transparent integration with SQL databases. A LDAP interface is available too. This system is also known as Asterisk External Configuration and is configured in /etc/asterisk/extconfig.conf. You can map configuration files to tables in a database (static configuration) and real time entries for dynamic creation of objects without the need to reload Asterisk.

15.2 Objectives

By the end of this chapter, the reader should be able to:

- Understand advantages and limitations of Asterisk Real Time.
- Install MySQL for use with ARA
- Compile and Install ARA using MySQL
- Test the system in a LAB environment

15.3 How does Asterisk Real Time work?



In the new Real Time architecture, all database specific code was moved to channel drivers. The channel only calls a generic routine that searches the database. The result is a much simpler and cleaner process from the source code point of view. The database is accessed by three functions:

- STATIC: Used to setup a static configuration when a module is loaded.
- REALTIME: Used to search objects during a call or another event.
- UPDATE: Used to update objects.

The channel database support was not changed. There are peers and users called static (normal) and peers and users called real time (database). For the static, it doesn't matter if it is loaded from a configuration file or from the database kept in the memory. The real time peers/users are loaded only when a call is made. After the call, the peer or user is deleted. Because of this, there is no support for NAT or MWI (Message Waiting Indicator). You can enable real time caching using the command "rtcachefriends=yes" in the sip.conf file or from the static database. By doing so, you will have NAT traversal and MWI, but, if you do any updates to this peer/user you will have to reload.

15.4 LAB 1 Installing Asterisk Real/Time

For this lab, we will assume that you still have the MySQL libraries installed in Chapter 13.

Step 1: Download the add-ons package

```
#wget http://ftp.digium.com/pub/asterisk/releases/asterisk-addons-1.4.x.tar.gz
```

Step 2: Uncompress the file

```
#tar -xzvf asterisk-addons-1.4.x.tar.gx
#cd asterisk-1.4.x
#make
#make install
#make samples
```

Confirm the module installation using 'module show'

15.5 Configuring Asterisk Real Time

res_mysql.conf

```
[settings]
; Static configuration files:
;
; file.conf => driver,database[,table]
;queues.conf => odbc,asterisk,ast_config

; Realtime configuration engine
;
iaxusers => mysql,asteriskdb,iax_buddies
iaxpeers => mysql,asteriskdb,iax_buddies
sipusers => mysql,asteriskdb,sip_buddies
sippeers => mysql,asteriskdb,sip_buddies
voicemail => mysql,asteriskdb,voicemail
extensions => mysql,asteriskdb,extensions_table
```

ARA is configures in the extconfig.conf text file, where two sections can be easily seen. The first one is the static configuration files section, where you can substitute the text configuration files for database tables. The second section is the realtime configuration engine where you configure database tables for dynamic objects (peers/users). It is not unusual to use text files for the static configuration and the database for dynamic entries. In this case, the first section is untouched.

extconfig.conf file format:

```
;
; Static and realtime external configuration
; engine configuration
;
; Please read doc/README.extconfig for basic table
; formatting information.
;
[settings]
;
; Static configuration files:
;
; file.conf => driver,database[,table]
;
; maps a particular configuration file to the given
; database driver, database and table (or uses the
; name of the file as the table if not specified)
;
;uncomment to load queues.conf via the odbc engine.
;
;queues.conf => odbc,asterisk,ast_config
;
; The following files CANNOT be loaded from Realtime storage:
;       asterisk.conf
;       extconfig.conf (this file)
;       logger.conf
;
; Additionally, the following files cannot be loaded from
; Realtime storage unless the storage driver is loaded
; early using 'preload' statements in modules.conf:
;       manager.conf
;       cdr.conf
;       rtp.conf
;
; Realtime configuration engine
;
; maps a particular family of realtime
; configuration to a given database driver,
; database and table (or uses the name of
; the family if the table is not specified
;
;example => odbc,asterisk,alttable
;iaxusers => odbc,asterisk
;iaxpeers => odbc,asterisk
;sipusers => odbc,asterisk
;sippeers => odbc,asterisk
;voicemail => odbc,asterisk
;extensions => odbc,asterisk
;queues => odbc,asterisk
;queue_members => odbc,asterisk
```

15.5.1 Static configuration section

The static configuration section is where you store the equivalent to configuration files in the database. These configurations are read during the Asterisk load. Some modules reread the database when you reload.

Static configuration example:

```
<conf filename> => <driver>,<databasename>[,table_name]
queues.conf => mysql,asteriskdb,queues_conf
sip.conf => odbc,asteriskdb,sip_conf
iax.conf => ldap,MyBaseDN,iax
```

Three examples are described above. In the first one, you bind queues.conf to a table queues in the asteriskdb database. In the second example, you bind sip.conf to the table sip_conf in the database asteriskdb defined in the odbc configuration. In the last example, you bind iax.conf to a LDAP directory. MyBaseDN is the base DN to be searched.

In the example above, the application app_queue.so is loaded while MySQL driver quries the database and gets the necessary information.

15.5.2 Real Time configuration section

The realtime configuration (second part of the extconfig.conf file) is where the configuration piece to be loaed is configured, updated, and unloaded in real time. With real time is not necessary to reload the configurations.

The realtime syntax follows:

```
<family name> => <driver>,<database name>[,table_name]
```

Example:

```
sippeers => mysql,asteriskdb,sip_peers
sipusers => mysql,asteriskdb,sip_users
queues => mysql,asteriskdb,queue_table
queue_members => mysql,asteriskdb,queue_member_table
voicemail => mysql,asteriskdb,test
```

Above, we have five configuration lines. In the first line, you bind the family sippeers to a table sip_peers in the asteriskdb MySQL database. In the last, you bind the voicemail family to the test table in the asteriskdb database. Note that sip_peers and sip_users could point to the same table.

15.6 DATABASE CONFIGURATION

Now that we have configured the extconfig.conf file, let's create the tables. Generally speaking, the database columns need to have the same fields as the configuration files.

Example: For a SIP or an IAX object, such as the one described below,

```
[4000]
host=dynamic
secret=senha
context=default
context=ramais
```

the database table should look like this:

name	host	secret	context	ipaddr	port	regseconds
4000	dynamic	senha	default;ramais	10.1.1.1	4569	1765432

To use with IAX, the tables need to have at least the following fields: "name", "port", and "regseconds". You may configure other columns to the desired fields. For example, if you want the parameter callerid, create a column named callerid (the same parameter as the text file).

A SIP table may look like the one below:

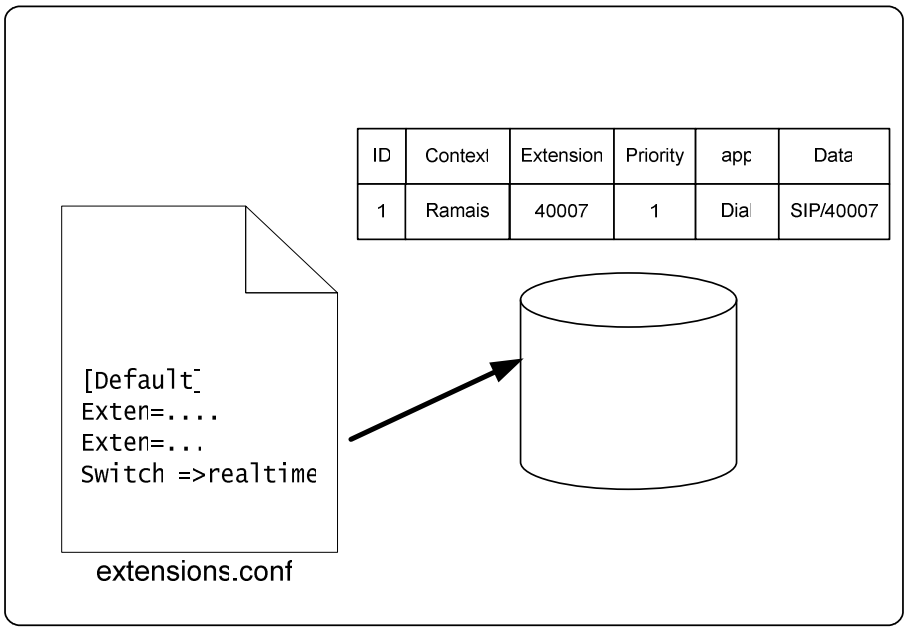
name	host	secret	context	ipaddr	port	regseconds	username
4000	dynamic	senha	default	10.1.1.1	5060	1765432	4000

A voicemail table should look like this:

Uniqueid	mailbox	context	password	email	fullname
1	4000	default	4000	joao@silva.com	Joao Silva

The uniqueid should be unique to each voicemail user and can be autoincrement. It need not have any relationship to the mailbox or context.

15.6.1 Building a dialplan using Asterisk Realtime



The extension table should look like the one below:

context	Exten	priority	app	appdata
Ramais	4000	1	dial	SIP/4000&IAX2/4000

In the dialplan, you have to use the switch command to use the real-time.

```
[local]
switch => realtime
```

or

```
[local]
Switch =>realtime/ramais@extensions
```

15.7 Lab 2 – Installing and creating the database tables

In this lab we will prepare the database to receive Asterisk parameters. We will prepare just the REALTIME databases. The static configuration will be left to the configuration text files (Cool isn't it ?).

15.7.1 Table creation in MySQL

Step 1: Get into to the MySQL database using root

```
#mysql –u root –p
```

Step 2: Create a database for Asterisk Realtime

```
mysql>create database asteriskdb;
```

Step 3: Create a user with access to the asteriskdb database

```
mysql>grant all privileges on ‘asteriskdb’.* to 'asterisk'@'localhost'
identified by 'asterisk';
```

Step 4: Exit MySQL and login again using the user created in step 3

```
#mysql –u asterisk –p asteriskdb
```

When asked for the password type “asterisk”.

Step 5: Creating the necessary tables

Download from blog.asteriskguide.com/sql the following file:

```
#wget blog.asteriskguide.com/realtime.sql
```

Execute the following commands:

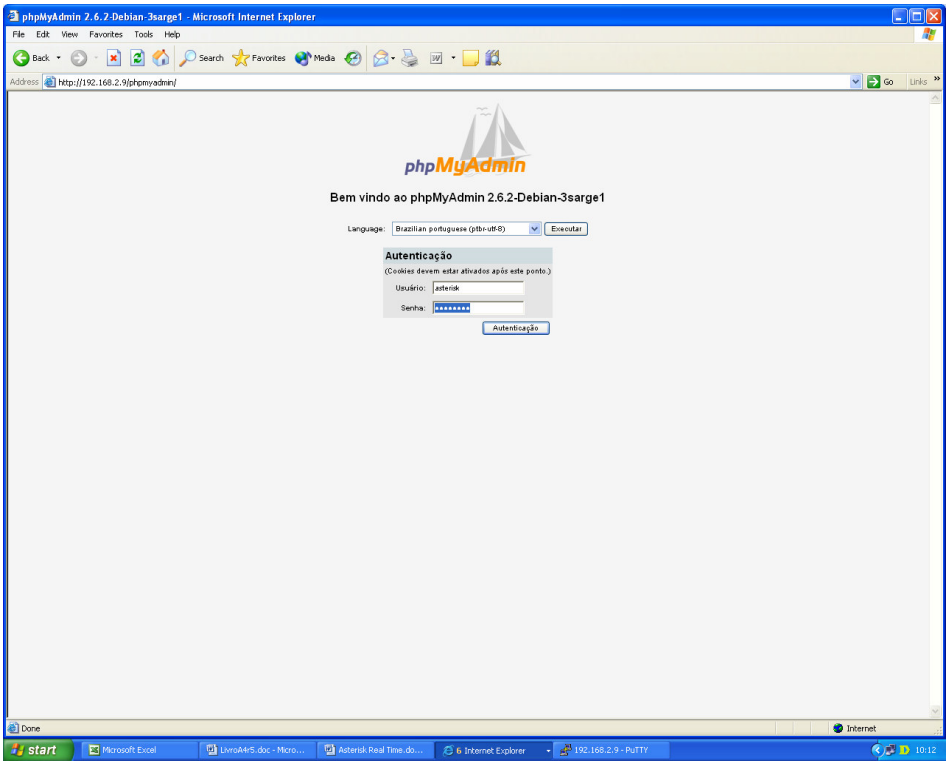
```
#mysql asteriskdb -u asterisk -p <realtime.sql
```

Use ‘asterisk’ as the password.

Step 6: Install phpmyadmin to handle database tasks

```
#apt-get install phpmyadmin
```

Phpmyadmin screenshot.



Login screenshot

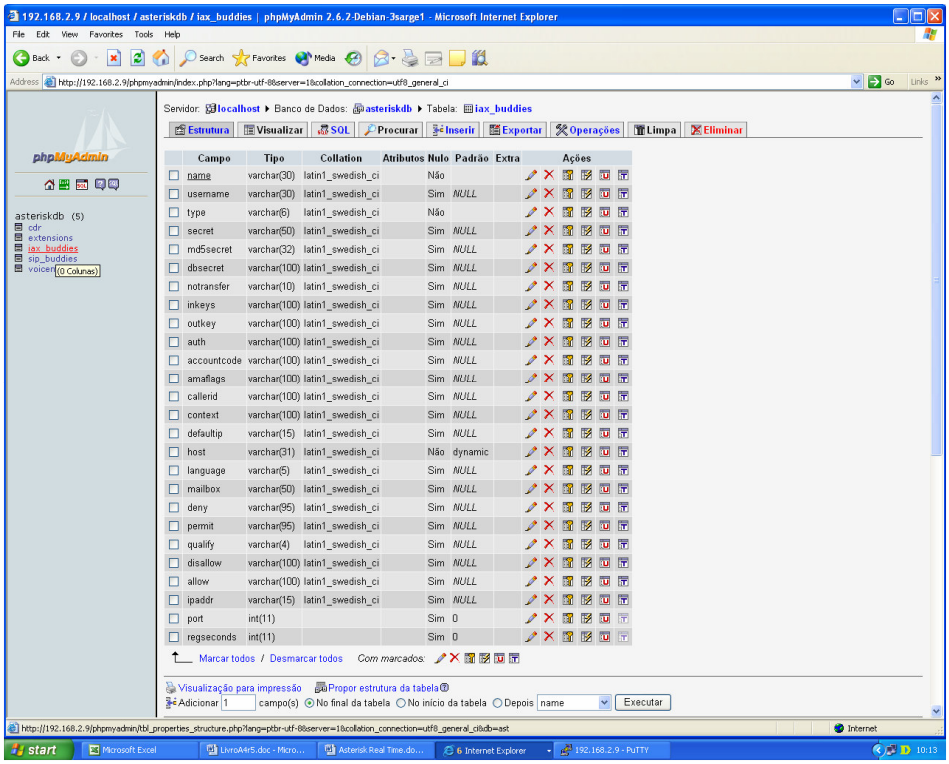


Table screenshot

Step 7: Configure Asterisk to access the database

In the res_mysql.conf

```
[general]
dbhost = 127.0.0.1
dbname = asteriskdb
dbuser = asterisk
dbpass = asterisk
dbport = 3306
```

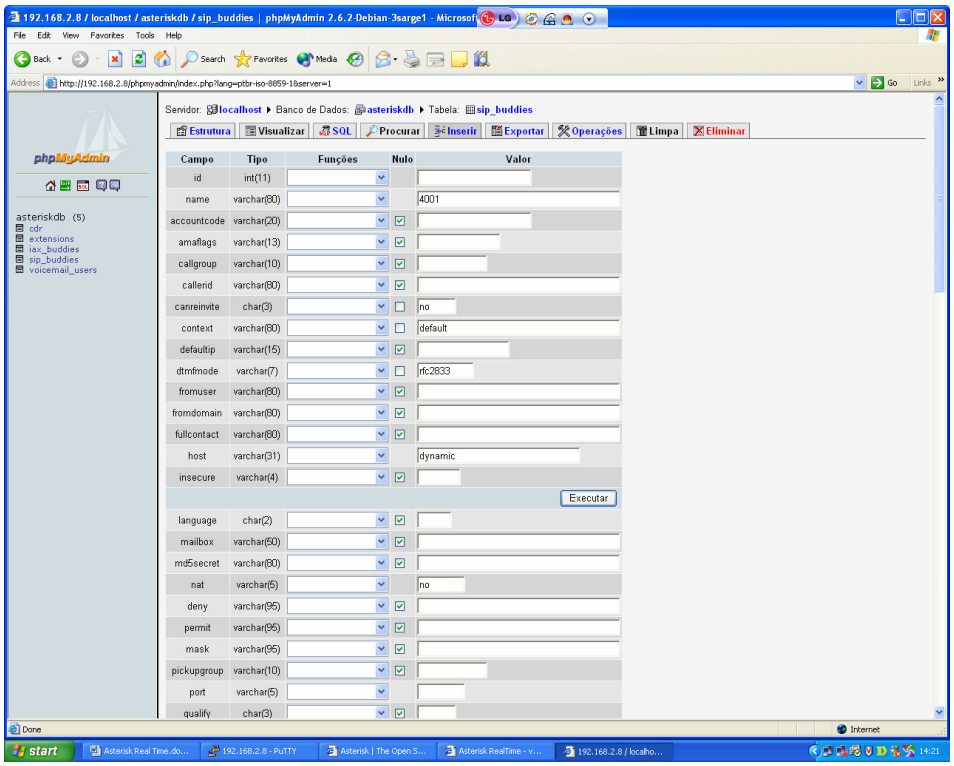
15.8 Lab 3 - Configuring and testing ARA

I this lab we will change the extconfig.conf configuration to reflect our database configuration and tables.

Step 1: Configure extconfig.conf and reload Asterisk

```
; Realtime configuration engine
;
; maps a particular family of realtime
; configuration to a given database driver,
; database and table (or uses the name of
; the family if the table is not specified
;
;example => odbc,asterisk,alttable
iaxusers => mysql,asteriskdb,iax_buddies
iaxpeers => mysql,asteriskdb,iax_buddies
sipusers => mysql,asteriskdb,sip_buddies
sippeers => mysql,asteriskdb,sip_buddies
voicemail => mysql,asteriskdb,voicemail_users
extensions => mysql,asteriskdb,extensions_table
```

Step 2: Real Time extension test



Create a new 4001 SIP friend inserting a record on the sip_buddies table and try to authenticate on this peer/user with a softphone.

Step 3: Try a call from the 4000 extension created before (static) and the new 4001

Verify using SIP shows peers the SIP objects. You will note that only the static peer shows. This behavior is normal since the peer is just created when you call. If you need to have NAT traversal support or MWI, use rtcachefriends=yes in the sip.conf file.

Step 4: Put the command "rtcachefriends=yes" in the [general] section of the sip.conf file

Step 5: Try again a call from 4000 to 4001

Verify using show peers. Why it appears now?

Step 6: Create a new SIP peer in the database with the name 4007

Change the phone registration to 4007 without reloading Asterisk

Step 7 : Include the extensions in the database

```
mysql -u asterisk -p
Enter password:
```

--> Use “asterisk” when asked.

Use phpadmin to include na extension in the database. If you prefer, use the following comands instead in the mysql client interface.

```
use asteriskdb;
insert into extensions_table(id, context, exten, priority, app, appdata) VALUES ('1','teste',
'40007','1','Dial','SIP/40007');
```

Step 8: Include Asterisk realtime into the dialplan.

In the default context:

```
switch => realtime/teste@extensions
```

Reload the extensions to activate the change.

```
asterisk-server*CLI>extensions reload
```

Step 9 : Reconfigure one of the phones to 40007. if you have not already done.

Step 10: Dial from na existent phone to the 40007

15.9 SUMMARY

On this chapter, you have learned that Asterisk Real Time allows you to put you configurations into a database. Databases supported are MySQL and any other unix ODBC supported databases. The configuration is divided into static and real-time. Static configuration replaces the configuration files, while the realtime configuration creates dynamic objects that are loaded only when a call or other related event happens. We concluded with a practical lab teaching how to install and configure ARA.

15.10 QUESTIONS

1. Asterisk real-time is part of the standard Asterisk distribution.

    a. True
    b. False

2 – To compile ARA and use it with MySQL databases the following libraries have to be installed.

a. Libmysqlclient12-dev
b. Mysql-server-4.1
c. Perl
d. Php

3 – Configuration of database server's IP addresses and ports are done in the following file:

a. extensions.conf
b. sip.conf
c. res_mysql.conf
d. extconfig.conf

4 - The file extconfig.conf is used to configure the tables that are used by real time. This file has two distinct sections:

a. Static configuration
b. Realtime configuration
c. Outbound routes
d. IP addresses and database ports

5 – In the static configuration, once you load the objects from the database, they are loaded dynamically into Asterisk's memory whenever necessary.

a. True
b. False

6 – When a SIP channel is configured in real-time, it's not possible to use resources as "qualify" or "MWI" (message waiting indicator) because the channel does not exist until a call is made. This causes the following problems:

a. This channel can call but not receive calls
b. The SIP channel could not be used behind NAT because qualify is used to keep NAT translation open.
c. It's not possible to make Message Waiting indicator works in the phones that support it.
d. It's not possible to use the channel since SIP is always static.

7 – If you want to use realtime configuration with SIP channels, but need support to NAT and MWI you should use:

a. Realtime was not created for use with NAT
b. “rtcachefriends=yes” in sip.conf
c. Only MWI is possible
d. To use NAT, the configuration needs to be static

8 – You can still use text configuration files even after installing ARA.

a. True
b. False

9 – Phpadmin is mandatory when you use real-time.

a. True
b. False

10 – The database has to be created with all the existing fields of the configuration file.

a. True
b. False

Appendix A

Question's Responses

CHAPTER 1

1. Mark the correct answers. Asterisk has four basic architectural components.

- A, B, E, F -

2. If necessary, you can create an Asterisk PBX with four trunks and eight phones using three TDM400 cards. The first one with four FXO channels and the other two with four FXS channels each. This affirmative is:

- B -

3. A FXS channel generates a dialing tone, while a FXO channel receives a dialing tone from the PSTN or another PBX. The affirmative is:

- B -

4. Mark the correct answers. Asterisk allows the use of the following features:

- A, B, C, D -

5. To play music on hold, Asterisk needs an external player like a MP3 or CD player. The affirmative is:

- A -

6. This technology is responsible for automatic answering of costumers. Usually plays a "prompt" and wait an option dialed by the user. In some cases can be integrated with a database to provide information by the telephone using text-to-speech technology. It is named _____.

- B -

7 – An E1 trunk supports ___ voice channels while a T1 trunk supports ___ voice channels.

- B -

8 – In traditional PBXs, usually ACD, IVR and voicemail are included in the PBX together with their respective licensing. This affirmative is:

- A -

9 – It is possible to connect several branches using voice over IP, thereby reducing long distance toll rates imposed by telephony companies. Asterisk can act at a branch as:

- A, B -

10 – Asterisk can be used as a contact center platform. What are the three main types of contact centers?

- C, D, E -

CHAPTER 2

1. What's the minimal Asterisk hardware configuration?
Pentium 300 Mhz, 256 MB RAM, 100 MB free space in hard-disk

2. Telephony interface cards for Asterisk usually have some DSPs (Digital Signal Processors) built in and do not need a lot of CPU resources from the PC.

- B -

3. If you want perfect voice quality, you need to implement end-to-end QoS (Quality of Service).

- A -

4. It is possible to have good voice quality with 100 Mbps switches in a non congested local area network.

- A -

5. List the necessary packages for Asterisk and the Zaptel compilation.
Gcc, ncurses, ncurses-devel, bison, termcap, openssl, openssl-developer

6. If you don't have a TDM interface card, you will end up needing a clock source for synchronization. The ztdummy driver is responsible for this role by using USB as a clock source (Kernel 2.4). This is necessary because some applications like **Meetme()** and **MusicOnHold()** require a time reference.

7. When you install Asterisk, it's better to leave GUIs uninstalled because Asterisk is sensible to performance variations. GUIs stole a lots of CPU cycles.

- A -

8. Asterisk configuration files are located in the **/etc/asterisk** directory.

9. To install Asterisk sample files you need to type the following command:
Make samples

10. Why is it important to start Asterisk with a non-root user?

To avoid in a buffer-overflow attacks, that the attacker get root rights to the server

CHAPTER 3

1. These are channel configuration files

- B, C, D -

2. It is important to define a context in the channel configuration file, because this will define the incoming context for a call. In the extensions configuration file (extensions.conf) a call from this channel will be processed in the matching incoming context.

- A -

3. The main differences between the playback() and background() applications are (choose two):

- A, C -

4. When a call gets into Asterisk using a telephony interface card (FXO) this call is handled in the special extension:

- C -

5. Valid formats for the goto() application are (choose three):

- A, C, D -

6. An extension cannot be defined as (choose all correct answers):

- C -

7. A pattern _7[1-5]XX matches (choose all correct answers):

- A -

8. An incoming context for a zaptel compatible telephony interface is defined in the _____ configuration file:

- B -

9. In the Options Inheritance grammar used by zapata.conf you:

- B -

10. Priorities must be consecutive!

- B -

CHAPTER 4

1 – Supervision signaling includes:

- A, B, C -

2 – Information signaling includes:

- B, C, D, E, F

3 – There are two types of analog interfaces available for Asterisk, FXS and FXO. Mark the correct answers.

- B, C –

4 – About T1 and E1 signaling, mark the correct affirmations.

- C, D –

5 – MFC/R2 signaling is supported by a third-party driver developed by Steve Underwood available in www.soft-switch.org.

- B –

6 – To configure zaptel hardware you should first edit the _____ file:

- A –

7 – The zaptel hardware is independent of Asterisk. In the zapata.conf, you configure Asterisk channels and not the hardware itself.

- B –

8 – When using a TDM400, with ___ ports, is necessary to connect the PC power source to the card using a specific connector (similar to the used to power the hard disk).

- B -

9 – Echo, pops and noise in a zaptel card are often related to the:

- C -

10 – R2 signaling defined by ITU is standardized in the whole world and there are no variations to the standard county dependent.

- B -

CHAPTER 5

1. Please, list at least four benefits of voice over IP.
Cost reductions, mobility, IP integrated IVR, Remote Agents

2. Convergence is the integrations of voice, data and video in a single network and their main benefit is the cost reduction in the implementation and maintenance of separate networks.

- B –

3. Asterisk cannot use simultaneously resources from PSTN and VoIP because the codecs are not compatible.

- A –

4. Asterisk is a SIP proxy with integration to other protocols

- A –

5. Using the OSI reference model, SIP, H.323 and IAX2 are in the _____ layer.

- D –

6. SIP is the most adopted protocol for IP phones and is an open standard ratified by IETF.

- B –

7. H.323 is an inexpressive protocol with very few applications, abandoned by the market, which is moving to SIP.

- A –

8. IAX is a proprietary Digium protocol. In spite of its small adoption by phone vendors, IAX is excellent when you need:

- A, C –

9. “Users” can receive calls from Asterisk.

- A –

10. Mark the correct answers concerning codecs.

- A, C, D –

CHAPTER 6

1. Two of the main benefits of IAX are bandwidth savings and easier NAT traversal.

- B –

2. IAX protocols use different UDP ports for signaling and media.

- A –

3. The bandwidth used by the IAX protocol is the voice payload and the following headers (Mark all that apply):

- A, B, C –

4. It is important to match the codec payload (20 to 30 ms) with frame synchronization (20ms default) when using trunk mode.

- B –

5. When IAX is used in trunk mode, just one header is used for multiple calls.

- B –

6. IAX is the most used protocol to connect to service providers, because it is easier for Nat traversal.

- A –

7. In an IAX channel as shown below, the option <secret> can be a password or a digital key.

```
IAX/[<user>[:<secret>]@]<peer>[:<portno>][/<exten>[@<context>][/<options>]]
```

8. The IAX2 show registry shows information about:

- B -

9. Jitter buffer sacrifices latency to have a steady flow of voice.

- A –

10. RSA keys can be used for IAX authentication. You have to keep the _____ key secret and give to your costumers and partners the matching _____ key.

- B -

CHAPTER 7

1. SIP is a protocol similar to _____ and _____.

- B, D –

2. SIP can have sessions of type: (mark all that apply)

- A, C, D, E –

3. Are SIP components: (mark all that apply)

- A, B, D, E –

4. Before a phone can receive calls, it needs to **REGISTER**.

5. A SIP server can operate in the PROXY or REDIRECT mode. The difference between them is that in the Proxy mode, all signaling pass by the SIP proxy. In the redirect mode, after discovering the location, the clients signal between themselves.

- A, B –

6. In proxy mode, the media flow goes through the SIP Proxy.

- B –

7. Asterisk is a SIP Proxy.

- B –

8. The canreinvite=yes/no option is fundamental. It will define if the media pass inside Asterisk or goes directly from one client to another. It has a major impact in Asterisk scalability.

- A -

9. Asterisk supports silence suppression in the SIP channels.

- B –

10. The hardest NAT type to traverse is:

- D -

CHAPTER 8

1. In the [general] section the default value to the option "writeprotect" is 'no'. If you issue a command "save dialplan" in Asterisk's CLI (mark all that apply).

- A, B, D -

2. Usually, the global variables are written in uppercase and the channel variables with only the first letter in uppercase. This is not mandatory, but makes it easier to identify the variable's type

- A –

3. The 's' extension is used as the starting point in a context. Usually you use the 's" extension in the following cases:

- A, B, D –

4. Write four situations where contexts could be used

Security implementation
Routing
Multilayer menus
Privacy

5. To use a variable in the dial plan you should use the following format

- D –

6. The Asterisk variable type could be (mark three).

- C, D, F –

7. To obtain a string length you could use the function: **${LEN(string)}.**

8. To concatenate strings it is simply put them together:

${foo}${bar}
555${thenumber}

- A -

9. Suppose that you are configuring an analog PBX based on Asterisk. Write the necessary instructions to build a dial plan to receive calls in the operator (SIP/4000). If the operator extension is not answered before the timeout, it will have to ring channels SIP/4000 and SIP/4001 simultaneously.

exten=4000,1,Dial(SIP/4000,15)

exten=4000,2,Dial(SIP/4000&SIP/4001,15)

10. Suppose that you are configuring a digital PBX based on Asterisk. Write the necessary instructions to allow the external dialing for long distance numbers.

[extensions]

exten=>_9XXXXXXXXXX,1,Dial(ZAP/g1/${EXTEN:1},15)

CHAPTER 9

1. To include a time-dependent context, you can use:

include=> context|<times>|<weekdays>|<mdays>|<months>

The statement below:

include=>normalhours|08:00-18:00|mon-fri|*|*

- A –

2. When an user dials "0" to get an external line, Asterisk automatically cuts the audio. This can be bad because the user is familiarized to hear the external dialing tone before dialing the other numbers. You can simulate the old dialing behavior with the ignorepat=> statement.

3. The statements below (mark all that apply):

```
exten => 8590/482518888,1,Congestion
exten => 8590,2,Dial(Zap/1,20,j)
exten => 8590,3,Voicemail(u8590)
exten => 8590,103,Voicemail(b8590)
```

Makes the user who called to the 8590 extension:

- A, C, D –

4. To concatenate several extensions you can separate them using the _**&**_ character.

5. A voice menu is usually created using the **background()** application.

6. You can include files inside the configuration files using the **#include** statement.

7. The Asterisk database is based in _____.

- C –

8. When you use Dial(type1/identifier1&type2/identifier2), the Asterisk dials to each one in sequence and wait 20 seconds between one to another. The affirmative is:

- A –

9. Using the Background application, you need to wait until the message is played before you can choose an option sending a dtmf digit.

- A –

10. The valid formats for the goto application are:

- B, C, D –

11. Switches are used to direct the dial plan processing to another server. The affirmative is:

- B –

12. A macro can be used to automate the processing of an extension. The first macro argument is:

- A -

CHAPTER 10

1. The following statements are true about Call Parking:

- C, D –

2. To use the Call Pickup feature, all extensions are required to be in the same **group**. For ZAP channels this is configured in the **zapata.conf** file.

3. When transferring a call, you may choose between _____, where the destination extension is not consulted before the transfer and _____ where you talk first to the destination extension before the transfer.

4. To make a consultative transfer you use the ___ character, while for blind transfer you use ___.

- B –

5. To enable conference calls in the Asterisk server, it is necessary to use the **MEETME()** application.

6. If you have to supervise a conference, you can use the _____ application.

- D –

7. The best format for music on hold is MP3 because it uses very little processing power from the Asterisk server.

- B –

8. To capture a call from a specific call group you need to be in their respective **pickup** group.

9. You can record a call by using the utility mixmonitor() or using the automon feature. By default the automon feature uses the ___ character sequence.

- A –

10. In the meetme application, if you want to have users in the listening only mode you should:

- C –

CHAPTER 11

1. Cite four strategies for routing call in a queue.
Ringall, roundrobin, leastrecent, fewestcalls, random, rrmemory

2. It is possible to record a conversation between an agent and a costumer using the statement **record=yes** in the queues.conf file.

3. To login an agent you will use the application agentlogin([agentnumber]). When the agent finishes the call, he can press:

- A, B, D -

4. The required tasks to configure a call queue are:

- A, E –

5. What's the difference between the applications AgentLogin() and AgentCallBackLogin().

Using the Agentlogin() application keeps the phone open. The operator just press # to take the calls. When you use AgentCallBackLogin() you hang up the phone after the login. If call gets into the queue, the phone will ring in the respective agent.

6. When in a call queue, you can define a certain number of options that the user can dial. This is done including a _____ in the file queues.conf

- C –

7. The support applications AddQueueMember(), AgentLogin(), AgentCallBackLogin e RemoveQueueMember() should be included in the: _____

- A –

8. It is possible to record the agents, but it is necessary an external recorder.

- B -

9. “Wrapuptime” is the time the user needs after ending the call to complete business process related to that call.

- A –

10. A call can be prioritized depending on the CallerID inside the same queue: The affirmative is:

- B -

CHAPTER 12

1. The files involved in the voicemail configuration are:

- D, F –

2. In the voicemail application, the parameters "u" and "b" are _____ and _____ respectively. They are used to determine what message will be played.

- C –

3. The VoiceMailMain() application is used for the caller to leave a message in the voicemail. The affirmative is:

- B –

4. To exit VoiceMailMain you should press:

- B –

5. Write below the voicemail() application syntax.

VoiceMail(mailbox[@context][&mailbox[@context]][...][|options]):

6. In the [general] section of the voicemail.conf file the parameter "attach=yes" makes Asterisk to send a notification by e-mail to the user with the audio file attached. The affirmative is:

- B –

7 The option "delete" makes that every message after being sent to the e-mail be erased from the mailbox.

- B –

8. The best format for voicemail audio is "WAV". It has better support in Windows workstations.

- B -

9. It is possible to customize e-mail messages by modifying the e-mail subject and body. What variable can be used to indicate a CallerID in the message?
VM_CALLERID

10. The cgi name to install the web voicemail interface is vmail.cgi.

CHAPTER 13

1. By default, Asterisk records the CDR in /var/log/asterisk/cdr-csv directory.

- B –

2. Asterisk allows using only these databases:

- A, C, D, E –

3. Asterisk generates a CDR only to single kind of storage.

- A –

4. Which are Asterisk amaflags available?

- A, B, E, F –

5. Fill the spaces left.

If you intend to associate a department to a CDR, you should use the command **Set(CDR(Account)=).** The account code can be verified using the channel variable ${ACCOUNTCODE}.

6. The difference between the applications NOCDR() and Reset CDR() is that NoCDR() does not generate any record and ResetCDR() zeroes the current record.

- B –

7. To use a user defined field with the cdr_csv.so module, is necessary to edit the source code and recompile the Asterisk.

- B –

8. The three authentication methods available to the Authenticate() application are:

- A, B, C –

9. Voicemail passwords are specified in a different section of the voicemail.conf file and are not the same as the voicemail users.

- A –

10. This option of authenticate command put the password used to authenticate in the CDR.

- A -

CHAPTER 14

1. Which of the following is not an interfacing method for Asterisk?

- E –

2. AMI (Asterisk Manager Interface) enables passing Asterisk commands via TCP sockets. This resource is enabled by default.

- B –

3. AMI is very safe, because its authentication is done using MD5 challenge/response.

- B –

4. To compensate the lack of security and scalability of AMI, we could use:

- B –

5. FastAGI allows the calling of external scripts from the dial plan to an external machine using TCP sockets (usually 4573).

- A -

6. DeadAGI is used in active channels. It can be used in ZAP channels but not in SIP or IAX channels.

- B –

7. Only php can be used for AGI scripting.

- B –

8. The command **agi show** shows all available AGI commands.

9. The command **manager show commands** shows all available AMI commands.

10. To debug an AGI you should use the command **agi debug**.

CHAPTER 15

1. Asterisk real-time is part of the standard Asterisk distribution.

- B –

2 – To compile ARA and use it with MySQL databases the following libraries have to be installed.

- A, B –

3 – Configuration of database server's IP addresses and ports are done in the following file:

- C –

4 - The file extconfig.conf is used to configure the tables that are used by real time. This file has two distinct sections:

- B, D –

5 – In the static configuration, once you load the objects from the database, they are loaded dynamically into Asterisk's memory whenever necessary.

- B –

6 – When a SIP channel is configured in real-time, it's not possible to use resources as "qualify" or "MWI" (message waiting indicator) because the channel does not exist until a call is made. This causes the following problems:

- B, C –

7 – If you want to use realtime configuration with SIP channels, but need support to NAT and MWI you should use:

- B –

8 – You can still use text configuration files even after installing ARA.

- A –

9 – Phpadmin is mandatory when you use real-time.

- B –

10 – The database has to be created with all the existing fields of the configuration file.

- B -

AUTHOR'S BIO

Flávio E. Gonçalves

Flávio Eduardo de Andrade Gonçalves is degreed in Mechanical Engeneering in 1989.
He has always worked with the training business being owner of a NAEC (Novell Authorized Education Center 1993-2004), TEC (Certified Technical Education Center 1995-2000).
Early in 2000 he started a Cisco VAR working with WAN, VPN and VoIP implementations. He has started to work with Asterisk in 2003, implementing IP PBXs, Call Centers and Voice over IP providers. The training business is still strong and Asterisk training has been a flagship product with more than 250 people trained in Brazil.

Certifications earned:

- Digum - dCAP, Digium Certified Asterisk Professional (2006). He´s first brasilian with this certification!
- Cisco - CCSP/CCDP/CCNP (started in 1997)
- Novell - MCNE/MCNI (started in 1993)
- Microsoft - MCSE/MCT (started in 1994).

flavio.goncalves@voffice.com.br

kkmkt

3RD GENERATION

VOFFICE

http://www.voffice.com.br