Getting Started with Asterisk™ Free eBook

A step-by-step guide to building a simple IP PBX

Flavio E. Gonçalves

Published by V.Office Networks

Getting Started with Asterisk PBX

Printing History

First Edition: November 2006,

File Date: Thursday, July 01, 2010

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Adaptation"** means a work based upon the Work, or upon the Work and other pre-existing works, such as a translation, adaptation, derivative work, arrangement of music or other alterations of a literary or artistic work, or phonogram or performance and includes cinematographic adaptations or any other form in which the Work may be recast, transformed, or adapted including in any form recognizably derived from the original, except that a work that constitutes a Collection will not be considered an Adaptation for the purpose of this License. For the avoidance of doubt, where the Work is a musical work, performance or phonogram, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered an Adaptation for the purpose of this License.

b. **"Collection"** means a collection of literary or artistic works, such as encyclopedias and anthologies, or performances, phonograms or broadcasts, or other works or subject matter other than works listed in Section 1(f) below, which, by reason of the selection and arrangement of their contents, constitute intellectual creations, in which the Work is included in its entirety in unmodified form along with one or more other contributions, each constituting separate and independent works in themselves, which together are assembled into a collective whole. A work that constitutes a Collection will not be considered an Adaptation (as defined above) for the purposes of this License.

c. **"Distribute"** means to make available to the public the original and copies of the Work or Adaptation, as appropriate, through sale or other transfer of ownership.

d. **"Licensor"** means the individual, individuals, entity or entities that offer(s) the Work under the terms of this License.

e. **"Original Author"** means, in the case of a literary or artistic work, the individual, individuals, entity or entities who created the Work or if no individual or entity can be identified, the publisher; and in addition (i) in the case of a performance the actors, singers, musicians, dancers, and other persons who act, sing, deliver, declaim, play in, interpret or otherwise perform literary or artistic works or expressions of folklore; (ii) in the case of a phonogram the producer being the person or legal entity who first fixes the sounds of a performance or other sounds; and, (iii) in the case of broadcasts, the organization that transmits the broadcast.

f. **"Work"** means the literary and/or artistic work offered under the terms of this License including without limitation any production in the literary, scientific and artistic domain, whatever may be the mode or form of its expression including digital form, such as a book, pamphlet and other writing; a lecture, address,

sermon or other work of the same nature; a dramatic or dramatico-musical work; a choreographic work or entertainment in dumb show; a musical composition with or without words; a cinematographic work to which are assimilated works expressed by a process analogous to cinematography; a work of drawing, painting, architecture, sculpture, engraving or lithography; a photographic work to which are assimilated works expressed by a process analogous to photography; a work of applied art; an illustration, map, plan, sketch or three-dimensional work relative to geography, topography, architecture or science; a performance; a broadcast; a phonogram; a compilation of data to the extent it is protected as a copyrightable work; or a work performed by a variety or circus performer to the extent it is not otherwise considered a literary or artistic work.

g. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

h. **"Publicly Perform"** means to perform public recitations of the Work and to communicate to the public those public recitations, by any means or process, including by wire or wireless means or public digital performances; to make available to the public Works in such a way that members of the public may access these Works from a place and at a place individually chosen by them; to perform the Work to the public by any means or process and the communication to the public of the performances of the Work, including by public digital performance; to broadcast and rebroadcast the Work by any means including signs, sounds or images.

i. **"Reproduce"** means to make copies of the Work by any means including without limitation by sound or visual recordings and the right of fixation and reproducing fixations of the Work, including storage of a protected performance or phonogram in digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to reduce, limit, or restrict any uses free from copyright or rights arising from limitations or exceptions that are provided for in connection with the copyright protection under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to Reproduce the Work, to incorporate the Work into one or more Collections, and to Reproduce the Work as incorporated in the Collections;

b. to create and Reproduce Adaptations provided that any such Adaptation, including any translation in any medium, takes reasonable steps to clearly label, demarcate or otherwise identify that changes were made to the original Work. For example, a translation could be marked "The original work was translated from English to Spanish," or a modification could indicate "The original work has been modified.";

c. to Distribute and Publicly Perform the Work including as incorporated in Collections; and,

d. to Distribute and Publicly Perform Adaptations.

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. Subject to Section 8(f), all rights not expressly granted by Licensor are hereby reserved, including but not limited to the rights set forth in Section 4(d).

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may Distribute or Publicly Perform the Work only under the terms of this License. You must include a copy of, or the Uniform Resource Identifier (URI) for, this License with every copy of the Work You Distribute or Publicly Perform. You may not offer or impose any terms on the Work that restrict the terms of this License or the ability of the recipient of the Work to exercise the rights granted to that recipient under the terms of the License. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties with every copy of the Work You Distribute or Publicly Perform. When You Distribute or Publicly Perform the Work, You may not impose any effective technological measures on the Work that restrict the ability of a recipient of the Work from You to exercise the rights granted to that recipient under the terms of the License. This Section 4(a) applies to the Work as incorporated in a Collection, but this does not require the Collection apart from the Work itself to be made subject to the terms of this License. If You create a Collection, upon notice from any Licensor You must, to the extent practicable, remove from the Collection any credit as required by Section 4(c), as requested. If You create an Adaptation, upon notice from any Licensor You must, to the extent practicable, remove from the Adaptation any credit as required by Section 4(c), as requested.

b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

c. If You Distribute, or Publicly Perform the Work or any Adaptations or Collections, You must, unless a request has been made pursuant to Section 4(a), keep intact all copyright notices for the Work and provide, reasonable to the medium or means You are utilizing: (i) the name of the Original Author (or pseudonym, if applicable) if supplied, and/or if the Original Author and/or Licensor designate another party or parties (e.g., a sponsor institute, publishing entity, journal) for attribution ("Attribution Parties") in Licensor's copyright notice, terms of service or by other reasonable means, the name of such party or parties; (ii) the title of the Work if supplied; (iii) to the extent reasonably practicable, the URI, if any, that Licensor specifies to be associated with the Work, unless such URI does not refer to the copyright notice or licensing information for the Work; and, (iv) consistent with Section 3(b), in the case of an Adaptation, a credit identifying the use of the Work in the Adaptation (e.g., "French translation of the Work by Original Author," or "Screenplay based on original Work by Original Author"). The credit required by this Section 4(c) may be implemented in any reasonable manner; provided, however, that in the case of a Adaptation or Collection, at a minimum such credit will appear, if a credit for all contributing authors of the Adaptation or Collection appears, then as part of these credits and in a manner at least as prominent as the credits for the other contributing authors. For the avoidance of doubt, You may only use the credit required by this Section for the purpose of attribution in the manner

set out above and, by exercising Your rights under this License, You may not implicitly or explicitly assert or imply any connection with, sponsorship or endorsement by the Original Author, Licensor and/or Attribution Parties, as appropriate, of You or Your use of the Work, without the separate, express prior written permission of the Original Author, Licensor and/or Attribution Parties.

d. For the avoidance of doubt:
    i. **Non-waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme cannot be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License;

    ii. **Waivable Compulsory License Schemes**. In those jurisdictions in which the right to collect royalties through any statutory or compulsory licensing scheme can be waived, the Licensor reserves the exclusive right to collect such royalties for any exercise by You of the rights granted under this License if Your exercise of such rights is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(b) and otherwise waives the right to collect royalties through any statutory or compulsory licensing scheme; and,

    iii. **Voluntary License Schemes**. The Licensor reserves the right to collect royalties, whether individually or, in the event that the Licensor is a member of a collecting society that administers voluntary licensing schemes, via that society, from any exercise by You of the rights granted under this License that is for a purpose or use which is otherwise than noncommercial as permitted under Section 4(c).

e. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL,

CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.

c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights

granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.

Preface

This book is for anyone who wants to learn how to install and configure a simple PBX (Private Branch eXchange) based on Asterisk 1.6. Asterisk is an open source telephony platform capable to use VoIP and TDM channels.

This is an excerpt from the book Learning Guide for Asterisk PBX a full 15 chapters book. I have organized the book in two parts. The first three chapters works like a Getting Started. You have everything necessary to learn how to build a simple PBX and it is free. The full book is available from amazon.com.

1. Introduction to Asterisk PBX
2. How to download and install Asterisk
3. Building a simple PBX
4. Analog channels
5. Digital channels
6. Designing a VoIP network
7. The IAX Protocol
8. The SIP Protocol
9. Dial Plan advanced features
10. Using PBX features
11. Call Queues
12. Asterisk Call Detail Records
13. Extending Asterisk with AMI and AGI
14. Asterisk Real-Time
15. Building a simple PBX using AsteriskNOW

The Asterisk Open Source PBX concept is revolutionary. 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 such as 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 book. However, you will be able to build and configure a PBX with advanced features such as 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.

Notes about this edition

In this edition we had changed all the chapters to reflect the changes for the Asterisk version 1.6. A new chapter about Asterisk Now was included and all the formatting of the book has changed. For ecological reasons, we tried to reduce the number of pages as much as possible reducing the unnecessary white spaces. So even increasing the amount of content in the book we still got an approximate reduction of 20% in the number of pages compared to the last formatting.

Flavio 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.

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. E-mail address for feedback: flavio@asteriskguide.com

Use as a training material

We use this book for Asterisk training. If you are interested to use it in your training center, please send an e-mail to flavio@asteriskguide.com. We have additional materials such as presentations and Lab Guides.

Credits

Cover Work:

Karla Braga

Reviewers:

Luis F. Goncalves

Guilherme Goes dCAP

Edit Avenue, professional proofreaders

About the Author

Flavio E. Goncalves was born in 1966 in Brazil. Having always had a strong interest in computers, he got his first personal computer in 1983 and since then it has been almost an addiction. He received his degree in Engineering in 1989 with focus in the computer aided design and computer aided manufacturing.

He is also, CEO of V.Office Networks in Brazil, a consulting company dedicated to the areas of Networks, Security and Telecomunications and a training center since its foundation in 1996. Since 1993, he has participated in a series of certifications programs having being certificated as Novell MCNE/MCNI, Microsoft MCSE/MCT, Cisco CCSP/CCNP/CCDP, Asterisk dCAP and some others.

He started writing about open source software, because, he thinks the way certification programs were organized in the past, were very good to help learners. Some books today are written by strictly technical people, who, sometimes, do not have a clear idea on how people learn. He tried to use his 15 year experience as instructor to help people learn open source telephony software. His experience with networks, protocol analyzers and IP telephony, combined with the teaching experience, give him and edge to write this book. This is the second book he writes; the first one was the Configuration Guide for Asterisk PBX.

As the CEO of V.Office, Flavio E. Goncalves, balance his time between family, work and fun. He is a father of two children and lives in Florianopolis, Brazil, one of the most beautiful places in the world. He dedicates his free time in water sports such as surfing and sailing.

Writing this book has been a process that involved many people. I would like to thank the staff at V.Office Networks in all the process of reviewing and editing the book. I would like to thank Guilherme Goes by the countless tips on Asterisk and the book itself. I would also like to thank several students, who took courses of Asterisk for their feedback, more than a thousand users have already taken classes using this material in the last five years. Finally, I would like to thank my family, for all the support they gave me during all these years.

You can contact him at flavio@asteriskguide.com, or visit his website www.asteriskguide.com.

Summary

Preface **9**
Notes about this edition **10**
Audience **11**
Mistakes and errors in the e-Book **11**
Use as a training material **11**
Credits **11**
About the Author **12**
Summary **13**
Objectives **1**
What is Asterisk? **1**
What is AsteriskNOW? **2**
Role of Digium™ **2**
The Zapata project and its relationship with Asterisk 3
Why Asterisk? **3**
Extreme cost reduction 4
Telephony system control and independence 4
Easy and rapid development environment 4
Feature rich 4
Dynamic content on the phone 4
Flexible and powerful dial plan 5
Open-source running on top of Linux 5
Asterisk architecture limitations 5
Main objections to Asterisk PBX **5**
Asterisk’s market share is too small 5
If it is free, how does the manufacturer survive? 6
It is hard to find technical support! 6
Does Asterisk support more than 200 extensions? 6
Only “geeks” are able to install Asterisk 6
What if the server fails? 6
Our company does not use open-source software 6
Using the PC's CPU to process signalling and media is not recommended 7
Asterisk Architecture 7
Channels 7
Codec and codec translation 8
Protocols 9
Applications 10

Overview of an Asterisk system **10**
Comparing the old and the new world **11**
Telephony using Asterisk 11
Building a test system **12**
One FXO, one FXS 13
VoIP Service Provider: ATA 13
Inexpensive FXO card or ATA 13
Asterisk scenarios **14**
IP PBX 14
IP-enabling legacy PBXs 14
Toll Bypass 15
Application Server (IVR, Conference, Voicemail) 16
Media Gateway 17
Contact Center Platform 18
Finding information and help **19**
Additional references: Non-official websites 19
Mailing lists 20
Summary **20**
Objectives **21**
Minimum Hardware Required **21**
Hardware configuration 22
IRQ sharing 23
Choosing a Linux distribution **23**
Required dependencies 23
Installing Linux for Asterisk **24**
Preparing Linux for Asterisk 24
Which version to choose 25
Obtaining and compiling Asterisk 25
Starting and stopping Asterisk 29
Installation directories 31
Log files and log rotation 32
Starting Asterisk with a non-root user 34
Uninstalling Asterisk 35
Asterisk installation notes 35
Summary 36
Quiz 36
Objectives **37**
Understanding the configuration files **37**

Grammars **38**
Simple Group 38
Object options inheritance grammar 38
Complex entity object 39
Options to build a LAB for Asterisk **39**
Option 1: Complete LAB 39
Option 2: Economy LAB 40
Option 3: Super economy lab 40
Installation Sequence **40**
Configuration of the extensions **41**
SIP extensions 42
Using Templates 43
IAX Extensions 44
Configuring the SIP devices **46**
Configuring the IAX devices **47**
Configuring a PSTN interface **48**
Analog lines using DAHDI 49
Connecting to the PSTN using a VoIP provider 50
Dial plan introduction **51**
The structure of the file extensions.conf 51
The section `[general]` 51
The section [globals] 52
Contexts **52**
Extensions **53**
Patterns **55**
Special extensions **55**
Variables **56**
Global variables 57
Channel-specific variables 57
Environment-specific variables 58
Application-specific variables 58
Expressions **58**
Operators 59
LAB. Evaluate the following expressions: **60**
Functions **61**
String concatenation 61
Applications **62**

Answer() 62
Dial() 62
Hangup() 65
Goto() 65
Building a dial plan **65**
Dialing between extensions 65
Dialing to an external destination 66
Dialing 9 to get a PSTN line 66
Receiving a call in the operator extension 66
Receiving a call using direct inward dialing (DID) 67
Playing several extensions simultaneously 67
Routing by Caller ID 67
Using variables in the dial plan 67
Recording an announcement 68
Receiving the calls in an digital receptionist 68
Summary **70**
Quiz **70**

1

Introduction to Asterisk PBX

The popularity of ready-to-run distributions such as TrixBox and AsteriskNOW has recently grown. In this book, we will cover the classic Asterisk, which is the foundation for understanding these distributions. Asterisk PBX is open-source software capable of transforming an ordinary PC into a powerful multiprotocol PBX. In this chapter, we will learn about the possibilities of this new technology and its basic architecture. As it is much simpler to install Asterisk from a ready-to-run distribution, the last chapter will cover AsteriskNOW and its graphical interface called FreePBX.

Objectives

By the end of this chapter you should be able to:

- Explain what Asterisk is and what it does;
- Describe the role of Digium™;
- Recognize the basic architecture of Asterisk and its components;
- Point out several usage scenarios; and
- Identify sources of information and help.

What is Asterisk?

Asterisk is an open-source PBX software once installed in a PC's hardware along with the correct interfaces—can be used as a full-featured PBX for home users, enterprises, VoIP service providers, and phone companies. 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 not only have an exceptional upgrade to your existing PBX, but you can also 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 a voicemail integrated with the web and e-mail;
- 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 previously only found in high-end systems, such as:

- Music for customers on hold waiting in call queues, supporting media streaming and MP3 files;
- Call queues, whereby 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; and
- PSTN connectivity through both digital and analog lines.

What is AsteriskNOW?

Asterisk in its purest form, also known as “classic asterisk” (Debian package denomination) is considered more of a development tool than a finished product by itself. AsteriskNOW is an initiative to transform Asterisk in a soft-appliance. The distribution includes CentOS as the operating system and the FreePBX, which is the most used graphical interface. This distribution is licensed according to the GPL and can be freely downloaded. In 2007, Digium acquired a product called Switchvox targeted to commercial users in the SMB market, which it has been promoting vigorously. You can check out this good piece of software at www.digium.com.

Role of Digium™

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

Digium offers Asterisk under three types of license agreements:

- 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 more recent version of Asterisk. Some companies use the business edition because they do not want or cannot use the GPL license—usually because they don't want to release their source code together with Asterisk. The GPL license requires that any further code development of a GPL-licensed code be released to the source code.
- Asterisk OEM. This version is mostly used by PBX manufacturers who do not want to reveal to the public that their software is based on Asterisk.

The Zapata project and its relationship with Asterisk

The Zapata project was developed by Jim Dixon, who was also responsible for the development of this revolutionary hardware for use with Asterisk. Note that the hardware is open-source too; as such, it can be used by any company. Today, several companies produce cards compatible with this architecture. More details about the project can be seen at:

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

The Zapata project produced an architecture called Zaptel (recently renamed Digium Asterisk Hardware Drivers Interface [DAHDI]). One of the main benefits of this architecture is the ability to use the PC CPU to process media streaming, echo cancellation, and transcoding. In contrast, most existing cards use digital signal processors (DSP) to perform these tasks. The use of the PC CPU instead of dedicated DSPs reduces the board's price dramatically. Thus, these cards are significantly cheaper than previously available interfaces from other manufacturers. On the other hand, these cards require a lot of CPU; a misuse of the PC CPU can significantly impact voice quality. Recently, Digium launched a coprocessor card that uses DSPs to encode and decode G.729 and G.723, allowing better scalability for a large number of channels.

Why Asterisk?

I remember my first contact with Asterisk. Usually, the first reaction to something new—especially something that competes with what you already know—is to reject it! This is exactly what happened in 2003. Asterisk was competing with a solution that I was selling to a customer (4 E1 VoIP Gateway), and it was ten times less expensive than what I was charging for the solution I already knew. This disproportionate price led me to start studying Asterisk in order to identify potential pitfalls and drawbacks. For example, I found that the PC CPU at that time would not support 120 g.729 simultaneous sections, 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 customer base. We were in trouble with expensive quotes for IVR, unified messaging, call recording, and dialers; with appropriate dimensioning, the CPU problems could be worked around. Indeed, in just three years Asterisk became the flagship product of my company (I actually decided to open another company just for the Asterisk business). In my opinion, Asterisk is a revolution in telecommunication that represents to IP telephony what Apache represents to web services.

Extreme cost reduction

If you compare a traditional PBX with Asterisk in regard to digital interfaces and phones, Asterisk is slightly cheaper than those PBXs. However, Asterisk really pays off when you add advanced features such as voicemail, ACD, IVR and CTI. With these advanced features, Asterisk becomes significantly less expensive than traditional PBXs. In fact, comparing Asterisk PBXs with low-end analog PBXs is unfair because Asterisk offers so many features not available in low-end analog systems.

Telephony system control and independence

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

Easy and rapid development environment

Asterisk can be extended using script 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.

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 exceed the price of the platform itself.

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 practically limitless.

Flexible and powerful dial plan

Another Asterisk breakthrough is its powerful dial plan. In traditional PBXs, even simple features like least cost routing (LCR) are either not feasible or optional. With Asterisk, choosing the best route is easy and clean.

Open-source running on top of Linux

One of the greatest features of Asterisk is its community. Several resources are available, including the Asterisk wiki (www.voip-info.org <http://www.voip-info.org>), e-mail distribution lists, and forums. As Asterisk becomes increasingly adopted, any bugs found and fixed quickly. Asterisk is probably the most tested PBX software in the world. From versions 1.0 to 1.2, more than 3,000 changes and bugs in the source code were corrected, thereby ensuring a code that is both stable and almost error free.

Asterisk architecture limitations

Some limitations in Asterisk stem from the use of the Zapata telephony design. In this design, Asterisk uses the PC CPU to process voice channels instead of dedicated digital signal processors (DSPs), 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 be conservative about hardware dimensioning. You can also 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, which will make Asterisk even better.

Main objections to Asterisk PBX

It is common to hear objections to adopting Asterisk, which we will address here.

Asterisk's market share is too small

The market share is usually measured by the number of PBXs sold. These statistics are generally acquired from the biggest distributors. Asterisk is free software that does not appear in sales statistics. However, independent numbers prove that Asterisk "rocks the world". According to VoIP-Supply, more than 300,000 systems run Asterisk, and Digium has sold more than 4 million voice interfaces. Last year, the Eastern Management Group concluded that open-source PBXs account for 18% of the market share, with the vast majority of them being Asterisk. In fact, 85% of the open-source PBX market is based on Asterisk, which now ranks second in terms of lines connected to an IP PBX.

If it is free, how does the manufacturer survive?

Actually, there is no such thing as open-source software manufacturer. Digium is a software development company, as well as a community, and has been developing Asterisk since 1999. With more than a hundred employees, it has revenues attached to the sales of telephony interface cards, PBX systems such as Switchvox, and related software. The company has made a profit in the last 24 quarters.

It is hard to find technical support!

Digium provides technical support for those who buy the Asterisk Business Edition. Recently, technical support for open-source Asterisk has become available as well. Hundreds of professionals have already been certified as Digium Certified Asterisk Professional (dCAP) and serve as the first line of support and professional services, much like any IT company.

Does Asterisk support more than 200 extensions?

Yes, absolutely. Asterisk has been used in installations with more than 10,000 users. It is largely scalable using load balancing and failover systems. It is not uncommon to see more than a thousand users on a single server.

Only “geeks” are able to install Asterisk

With AsteriskNOW and freePBX, even professionals with limited knowledge about Linux are able to install and configure a PBX of medium complexity. With the help of a GUI, it is possible to configure an entire PBX in just a few hours.

What if the server fails?

One of the main advantages of Asterisk is its capability to run in fault-tolerant systems. It is relatively simple and inexpensive to have two servers running in parallel. I dare you to try this with a conventional PBX!

Our company does not use open-source software

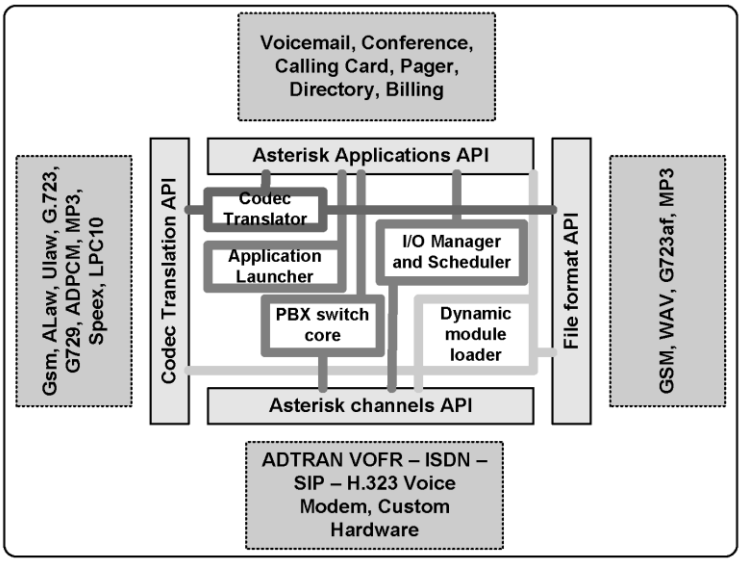
Your company probably uses open-source software without even realizing it. Several appliances use Linux as their operating system. Moreover, you can still license Asterisk commercially using the Asterisk Business Edition.

Using the PC's CPU to process signalling and media is not recommended

Asterisk uses the server's CPU to process signaling and media for voice channels instead of having dedicated DSPs. Although this allows a cost reduction of up to five times, it makes the system dependent on the performance of the main CPU. With the correct dimensioning, Asterisk is capable of handling large volumes. If you still want to release the main CPU from these tasks, you can also use hardware echo cancellation and even transcoder cards, such as the Digium's TC400B based on DSPs.

Asterisk Architecture

This section will explain how Asterisk's architecture works. The figure below shows the basic Asterisk architecture. Next, we will explain architecture-related concepts, including channels, codecs, and applications.



Channels

A channel is the equivalent of a telephone line, but in a digital format. It usually consists of an analog or digital (TDM) signaling system or a combination of codec and signaling protocol (e.g., SIP-GSM, IAX-uLaw). Initially, all telephony connections were analog and susceptible to echo and noise. Later, most systems were converted to digital systems, with the analogical sound converted into a digital format using pulse code modulation

(PCM) in most cases. This format allows voice transmission in 64 kilobits/second without compression.

Channels interfacing with the Public Switch Telephony Service (PSTN)

- **chan_dahdi:** Supports cards from Sangoma, Digium, Xorcom, and others
- **chan_mISDN:** Supports ISDN cards based in the Linux ISDN drivers

Channels interfacing with Voice-over IP

- **chan_sip:** Supports voice-over IP using SIP protocol. Dial string: sip/channel
- **chan_iax**: Supports voice-over IP using IAX2 protocol. Dial string: iax2/channel
- **chan_h323:** H.323 is one of the oldest and most implemented voice-over IP protocols. It's useful for connecting to existing H.323 networks. There are different flavors of H.323 in Asterisk, including `chan_h323`, `chan_oh323`, and `chan_ooH323`. The channel chan_h323 can be used in Asterisk as a gateway. Asterisk can point to a gatekeeper, but cannot work as one. Dial string h323/hostname if using a gatekeeper or `h323/extension@hostname` if going directly to the gateway.
- **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`
- **chan_skinny:** Supports Cisco™ voice-over IP skinny protocol. Dial String: `skinny/channel`.

Miscellaneous channels

- **chan_agent:** Used for automatic call distribution (ACD). It is not related to 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.
- **chan_local:** Is a pseudo channel that simply loops back into the dial plan in a different context. This is useful for recursive routing. Dial string: `Local/extension@context`

Codec 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, including compression, which is one of the most important features as it allows compression rates larger than 8 to 1. Other features include voice activity detection, packet loss concealment, and comfort noise generation. Several codecs are available for Asterisk and 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; these codecs cannot be translated. To verify which codecs are installed in your system, you can use the console command:

```
CLI>core show translation
```

The following codecs are supported:

- G.711 ulaw (USA) - (64 Kbps).
- G.711 alaw (Europe) - (64 Kbps).
- G.722 (High Definition) – (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)

Protocols

Sending data from one phone to another should be easy provided that the data find a path to the other phone on their own. 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. It has recently become extremely common to use SIP as a signaling protocol. IAX is another option becoming popular because it works well with NAT traversal and some bandwidth can be saved in trunk mode. Asterisk supports the following protocols.

- SIP
- H323
- IAX2
- MGCP
- SCCP (Cisco Skinny)
- Nortel unistim

Applications

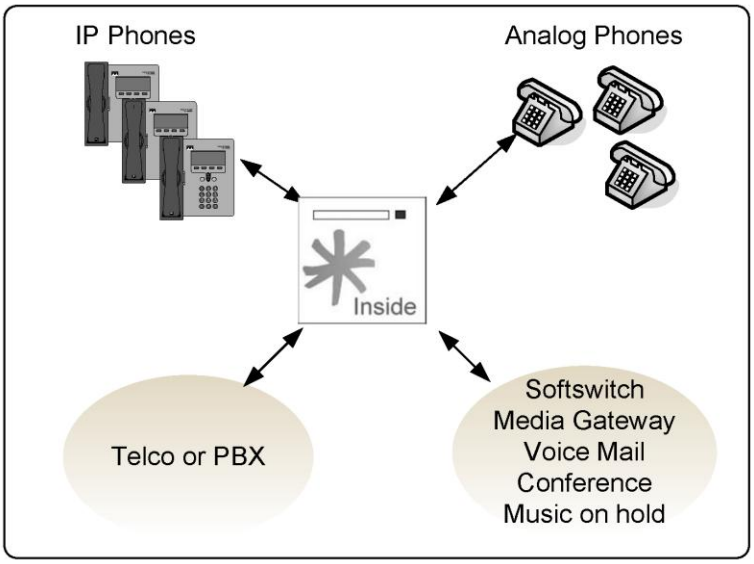
To bridge calls from one phone to another, the application `dial()` is used. Most Asterisk features (e.g., voicemail and conferencing) are implemented as applications. You can see available Asterisk applications by using the `core show applications` console command.

```
CLI>core show applications
```

You can add applications from Asterisk add-ons, third-party providers, or even those you develop yourself.

Overview of an Asterisk system

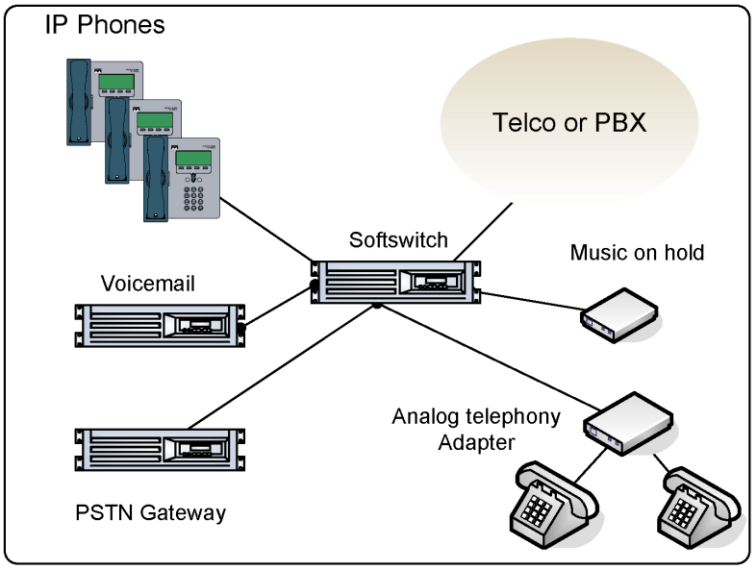
Asterisk is an open-source PBX that acts like a hybrid PBX, integrating technologies such as TDM and IP telephony. Asterisk is ready to implement functionality such as interactive voice response (IVR) and automatic call distribution (ACD); moreover, as previously mentioned, it is open to the development of new applications.



This figure shows how Asterisk connects to the PSTN and existing PBXs using analog and digital interfaces as well as supports analog and IP phones. It can act as a soft-switch, media gateway, voicemail, and audio conference and also has built-in music on hold.

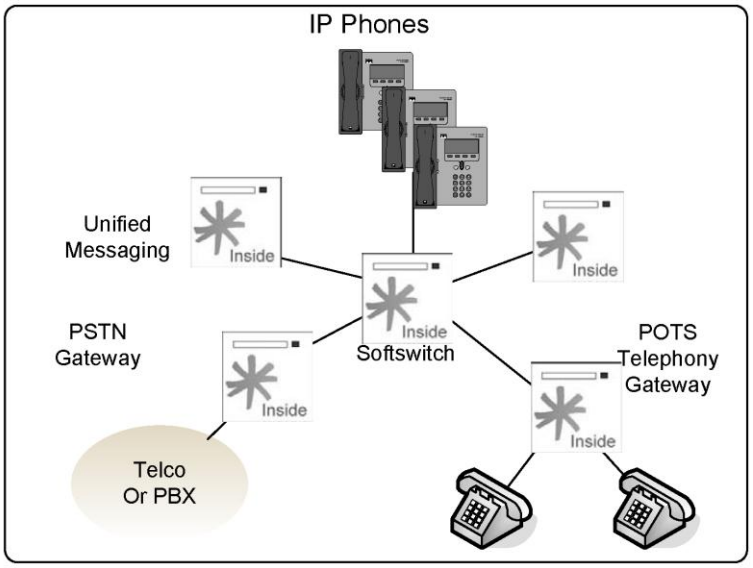
Comparing the old and the new world

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



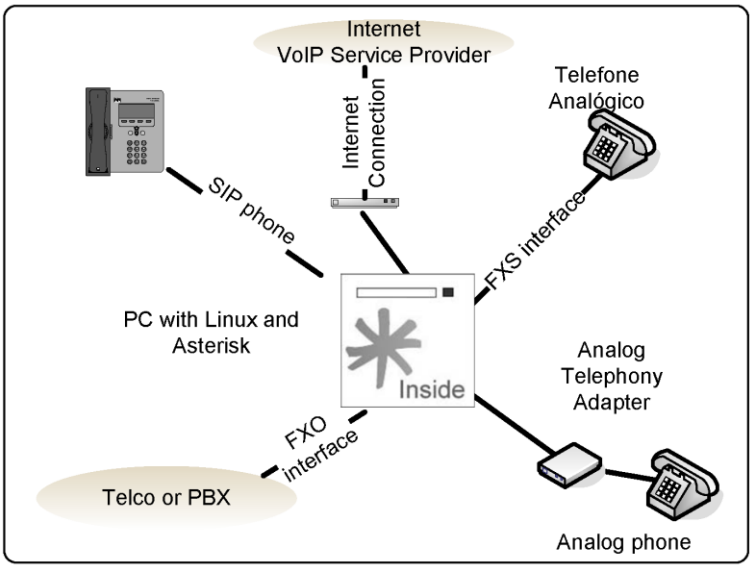
Telephony using Asterisk

All functions are integrated in the Asterisk platform in the same or in different boxes according to the dimensioning, and all are GPL licensed. Sometimes it is easier to install Asterisk than license some of the mainstream IP-PBXs



Building a test system

When implementing an Asterisk solution, our first step is generally to build a test machine. The easiest test machine is the 1x1 PBX, including at least one phone and one line. There are several ways to do this.



One FXO, one FXS

The first and simplest way to build a test machine is to purchase a card with one FXO and one FXS interface. Connect the FXO port to an existing line and connect one FXS to an analog phone. Thus, you have a 1x1 PBX.

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 a hundred dollars if you already have the PC.

Inexpensive FXO card or ATA

I started with an inexpensive FXO card. Some inexpensive V.90 fax/modems work with Asterisk as an FXO card. Some of the first Digium cards were created using these cards (e.g., X100P and X101P), which are old modems based on Motorola and Intel chipsets (Motorola 68202-51, Intel 537PU, Intel 537PG, and Intel Ambient MD3200 are known to work). These modems are often incompatible with new motherboards. Recently some manufacturers started to sell these cards as X100P clones. Some of the incompatibilities can be solved using a patch, more information can be found at:

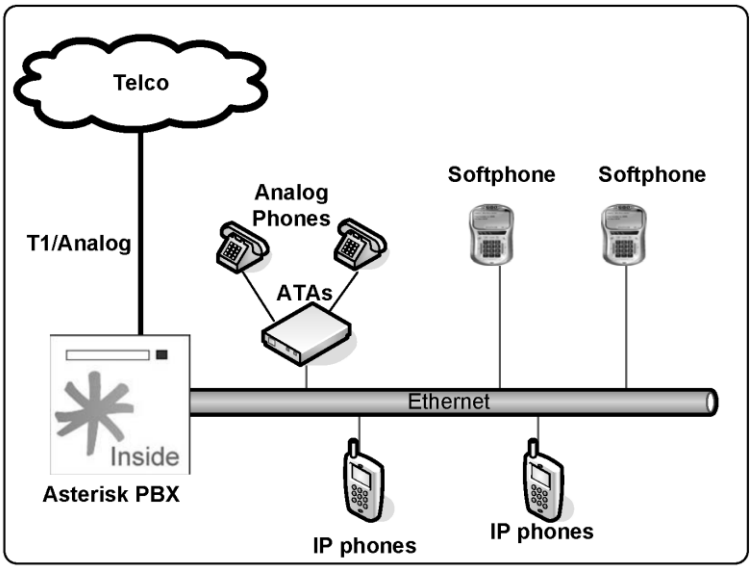
- http://www.asteriskguide.com/mediawiki/index.php/Asterisk_patch_for_the_X100P_card

Asterisk scenarios

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

IP PBX

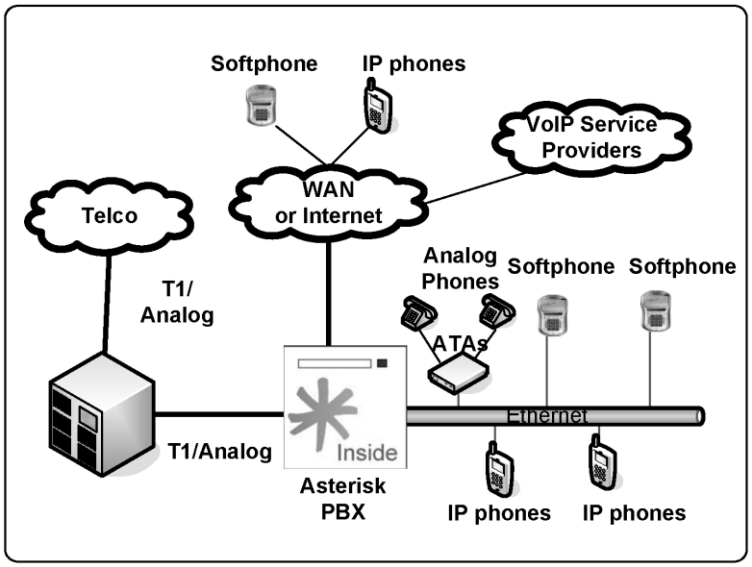
The most common scenario is the installation of a new or the replacement of an existing PBX. If you compare Asterisk with some other alternatives, you will find it to be cheaper and richer in features than most PBXs currently available on the market. Several companies are now changing their specifications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



IP-enabling legacy PBXs

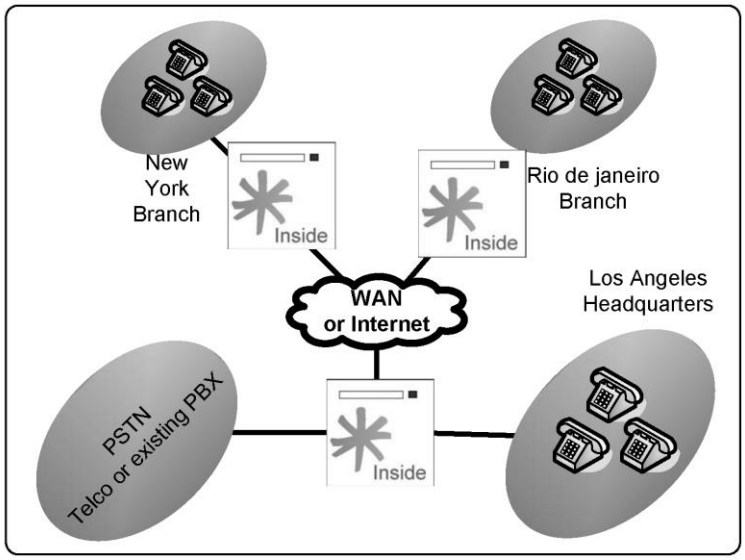
The following image illustrates one of the most commonly used setups. Large companies generally do not want to take significant risk when investing in new technologies and simultaneously wish to preserve their investments in legacy equipment. IP-enabling legacy PBX can be very expensive; thus, connecting an Asterisk PBX using T1/E1 lines

can be a good alternative for cost-conscious customers. Another benefit is the possibility of connecting to a VoIP service provider with better telephony rates.



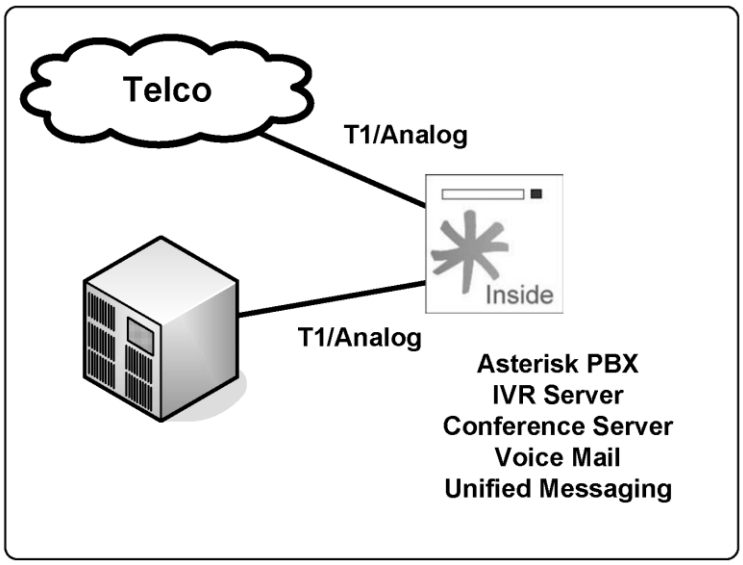
Toll Bypass

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.



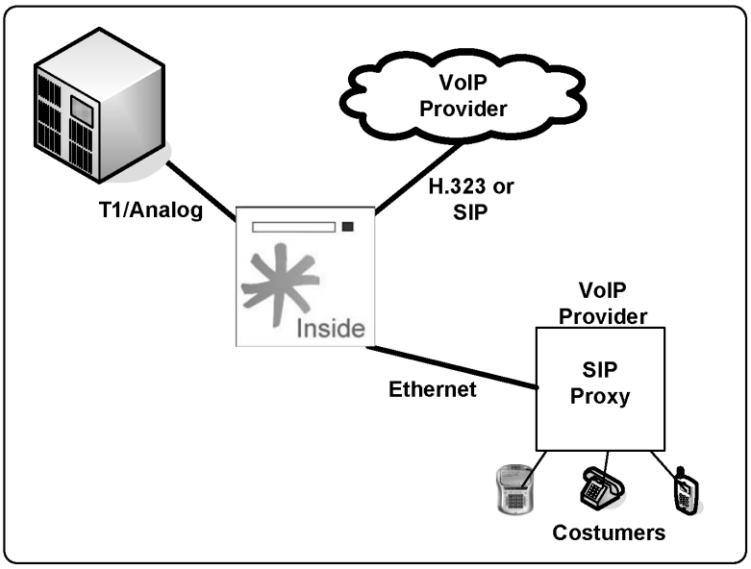
Application Server (IVR, Conference, Voicemail)

Asterisk can be used as an application server for the existing PBX or be directly connected to PSTN. Asterisk offers services such as voicemail, fax reception, call recording, IVR connected to a database, and an audio conferencing server. If you integrate voicemail and fax into an existing e-mail server, you will have a unified messaging system, which is usually an expensive solution. Using Asterisk as an application server provides extreme cost reduction compared to other solutions.



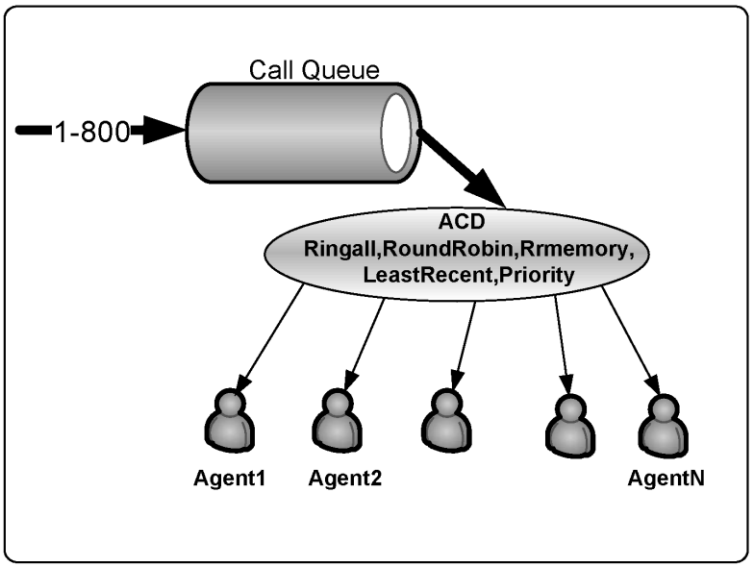
Media Gateway

Most voice-over IP service providers use an SIP proxy to host all registration, location, and authentication of SIP users. They still have to send calls to the PSTN directly or route it through a wholesale call termination provider using an SIP or H.323 voice-over IP connection. Asterisk can act as a back-to-back user agent (B2BUA) or media gateway, replacing 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 of translating signaling protocols (H.323, SIP, IAX…) and codecs (G.711, G.729…).



Contact Center Platform

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



Finding information and help

This section will provide some of the main sources of information related to Asterisk.

- Asterisk's official website: <http://www.asterisk.org> Here you can find information about:
  - Support-> <http://www.asterisk.org/support>
  - Knowledge base-> <http://kb.digium.com/>
  - Forum-><http://forums.digium.com/>
  - Bug tracking-><http://bugs.digium.com/>

Additional references: Non-official websites

These sites are not official, but they provide useful content.

- <http://www.voip-info.org>
- <http://www.asteriskguru.com>
- <http://svn.digium.com/svn> (check the doc directory on each branch)

Mailing lists

Mailing lists are quite handy when you have questions. Usually, you will receive answers for your questions. Try to gather as much information as possible before posting to the list. Nobody will help you if you haven't done your homework. In other words, try at least once to solve the problem by yourself.

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

Summary

The Asterisk is software licensed according to the GPL that enables an ordinary PC to act as a powerful IP PBX platform. Digium's Mark Spencer created Asterisk in the late 1990s. Digium survives by selling hardware related to Asterisk. Hardware is open-sourced as well and originated in the Zapata project developed by Jim Dixon. The Asterisk architecture has the following main components:

- CHANNELS: Analog, digital, or voice-over IP.
- PROTOCOLS: Communication protocols, which are responsible for signaling the calls, can be SIP, H323, MGCP, and IAX.
- 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 various scenarios, from a small IP PBX to a sophisticated contact center. You can easily find help at www.asterisk.org

2
How to download and install Asterisk

In the first chapter, we learned a bit about how Asterisk is useful in the telephony environment. In this chapter, we will cover how to download and install Asterisk. Before starting, it is essential to learn how to compile and install it. The compilation process may seem weird for traditional Microsoft™ Windows™ users, but it is fairly common in the Linux™ environment. One can get an optimized code for your hardware when compiling Asterisk, which is what we will do here. Asterisk runs in several operating systems, but we chose to keep things easy and start with only one of them: Linux. We chose Debian as the Linux™ distribution because the dependencies are easy to install and the distribution is stable, with a low footprint. If you want to use another distribution, please change the name of the dependencies accordingly.

Objectives

By the end of this chapter you should be able to:

- Determine the hardware requirements for Asterisk;
- Install Linux with the required dependencies;
- Download a stable version using FTP;
- Compile Asterisk; and
- Learn how to start Asterisk at boot time.

Minimum Hardware Required

Asterisk does not need a lot of hardware to run, however there are some tips to choose the best hardware for your requirements. You should take into consideration the following main factors when choosing your hardware:

- Total number of registered users. Define how many registrations per second you need to support

- Total number of simultaneous calls. Define how many network conversations you need to process in the network adapter and bridge on the Asterisk server
- Which codecs you need to support. High complexity codecs will require a lot of CPU/FPU power in your server, a single iLBC session can require as much as 18MIPS
- Echo cancellation. Echo cancellation may take a lot of CPU/FPU, in some cases you should choose hardware echo cancellation using DSPs in the telephony interface card
- Availability. Use RAID1 or 5 to increase availability. Remember, Asterisk is 24x7 application.
- Redundancy on the telephony interfaces. Xorcom (http://www.xorcom.com) and Red-fone(http://www.red-fone-com) have very good solutions for this.

The main component for an Asterisk Server is the network adapter. A good server network adapter is recommended. CPU is important when you need to support high complexity codecs such as g.729 and iLBC and echo cancellation. You may choose to use dedicated DSPs, Digium provides a DSP card named TC400B capable to support 120 g729 simultaneous calls.

The best practice is to choose a new, server class, computer from a known manufacturer. To know exactly how many simultaneous calls or how many registered users an specific machine can support, you should test this hardware with a stress test tool such as SIPP (http://sipp.sourceforge.net). Some hardware manufacturers such as Xorcom (http://www.xorcom.com) publish its results in the website.

Note: Some Asterisk applications, such as meetme and music on hold, requires a clock source. Usually, the clock source is an telephony interface card. If your system does not use a telephony interface card, you will have to load dahdi_dummy to provide a clock source.

Hardware configuration

The Asterisk hardware does not need to be sophisticated. You don't need an expensive video card or numerous peripherals. Some tips about hardware configuration;

- Disable unused USB, serial and parallel ports to avoid the consumption of unnecessary interrupts.
- A robust network interface card is essential.
- Take particular care if you are using telephony interface cards. Some cards use a 3.3 volts PCI bus, and it is not easy to find motherboards for them. In these days, PCI express is more easily found.
- Pay a close attention to the hard disk, PBX used to work in a 24x7 regime while desktops work 8x5. Do not use desktop hardware for a PBX, usually the hard disk fails before the first year if used intensively. My recommendation is to use a server machine or an appliance designed to run 24x7 applications.

IRQ sharing

Telephony interface cards (e.g., X100P) generate large quantities of interruptions. Serving these interruptions requires 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 dedicated hardware to run 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
```

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

Choosing a Linux distribution

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. Several Linux distributions were successfully tested with Asterisk (e.g., Fedora, Redhat, SuSe, Debian, and Gentoo); choose one for your system. You can download the Debian distribution from the address below:

`http://www.us.debian.org/CD/netinst/#netinst-stable.`

Required dependencies

The following dependencies are required to compile Asterisk.

- bison
- libssl-dev
- openssl
- libasound2-dev
- libc6-dev

- libnewt-dev
- zlib1g-dev
- gcc
- g++
- make
- libncurses5-dev
- doxigen
- libxml2-dev

Required by DAHDI

- kernel sources

> Caution: DAHDI packages are necessary to compile some Asterisk applications like `meetme().` If you have compiled Asterisk before DAHDI, you will have to recompile it again to include the application `meetme()` as well as certain others.

Required by Xorcom Astribank

- libusb-dev
- fxload

Installing Linux for Asterisk

Install your Linux as usual, without a graphical user interface. Install and configure the email server as well. We will need the email server (exim4) to send voicemail notifications later in this book.

> Caution: This installation will format your PC. All your disk data will be erased. Please make sure to back up all data before starting.

Step 1: Put the CD in the CD-ROM drive and boot your PC. Most questions are very simple to answer.

Preparing Linux for Asterisk

Immediately after installing Asterisk, we will install the packages required for the subsequent compilation of Asterisk and DAHDI drivers. First, we will indicate to Debian where the packages will be downloaded from. This is done by using the apt-setup utility.

Step 1: Login as root.

Step 2: Install the kernel headers.

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

Step 3: Install the required packages.

```
apt-get install bison openssl libssl-dev libusb-dev fxload libasound2-dev libc6-dev libnewt-dev libncurses5-dev zlib1g-dev gcc g++ make doxygen libxml2-dev
```

Which version to choose

As a rule of thumb, you should use the version with the required features. Versions 1.2 and 1.4 are more stable than the newest 1.6 while the newer versions include the new features, meaning 1.2 and 1.4 are feature frozen. The Asterisk team has changed the version system for 1.6. Now, instead of having major versions each year, they are releasing major and minor versions. The newest version is 1.6.2; it is undergoing just bug fixes, too. All new development is integrated in the trunk. With 1.6 you will have at least three versions maintained simultaneously, which allows you an extended period to upgrade from one version to another. Recently they announced the change back to the old versioning system.

> All examples in this book were created or converted to Asterisk 1.6.2, but most should work in 1.4.

Obtaining and compiling Asterisk

The next step is the installation of Asterisk. To obtain the sources, you should download them from www.asterisk.org. 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 version is the newest.

For Asterisk 1.4

Download the source files from the Asterisk repository. Please, check for a newer version.

```
cd /usr/src
wget http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-1.4.29.1.tar.gz
wget http://downloads.asterisk.org/pub/telephony/libpri/releases/libpri-1.4.10.1.tar.gz
wget http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-addons-1.4.10.tar.gz
```

For Asterisk 1.6

Download the source files from the Asterisk repository. Please, check for a newer version.

```
cd /usr/src
wget http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-1.6.2.5.tar.gz
wget http://downloads.asterisk.org/pub/telephony/libpri/releases/libpri-1.4.10.2.tar.gz
```

```
wget http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-addons-1.6.2.0.tar.gz
```

DAHDI

The same version of DAHDI is used for both versions.

```
wget http://downloads.asterisk.org/pub/telephony/dahdi-linux/releases/dahdi-linux-2.2.1.tar.gz
wget http://downloads.asterisk.org/pub/telephony/dahdi-tools/releases/dahdi-tools-2.2.1.tar.gz
```

Uncompress the files using:

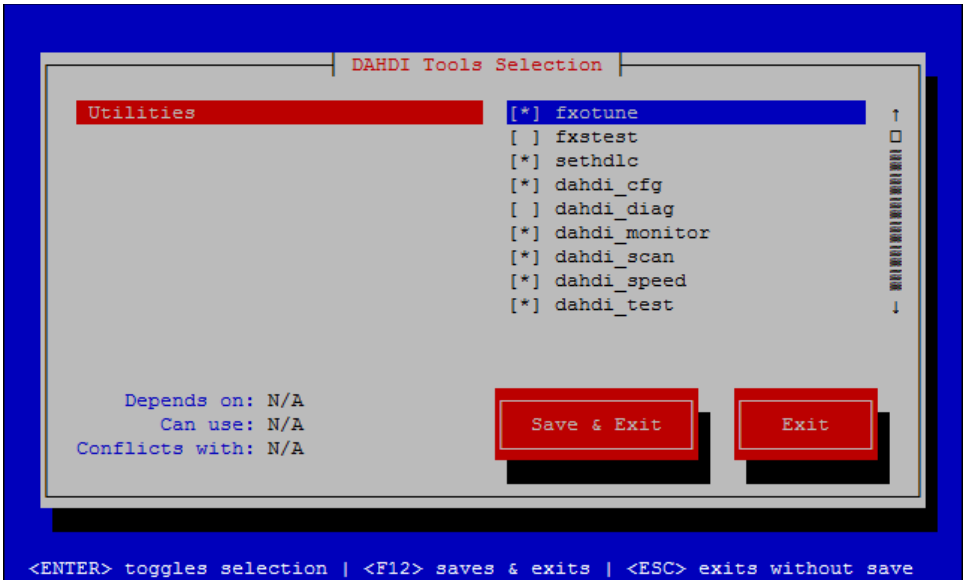
```
tar xzvf file.tar.gz
```

Compiling DAHDI drivers

You will need to compile the DAHDI modules. The commands `./configure` and `make menuselect` were added in version 1.4. The latter enables you to select which utilities and modules to build. The following commands will do this:

```
cd /usr/src/dahdi-linux-2.2.1
make
make install
cd /usr/src/dahdi-tools-2.2.1
./configure
make menusect #(optional, you may select some options)
make
make install
make config #(optional, it installs the init scripts)
```

Use `make menuselect` to install only the necessary modules. This is the `make menuselect` screenshot.



Just after executing `make config`, the init scripts will be installed, and the following screen will be shown.

```
install -D dahdi.init /etc/init.d/dahdi
/usr/bin/install -c -D -m 644 init.conf.sample /etc/dahdi/init.conf
/usr/bin/install -c -D -m 644 modules.sample /etc/dahdi/modules
/usr/bin/install -c -D -m 644 blacklist.sample /etc/modprobe.d/dahdi.blacklist
/usr/sbin/update-rc.d dahdi defaults 15 30
 Adding system startup for /etc/init.d/dahdi ...
   /etc/rc0.d/K30dahdi -> ../init.d/dahdi
   /etc/rc1.d/K30dahdi -> ../init.d/dahdi
   /etc/rc6.d/K30dahdi -> ../init.d/dahdi
   /etc/rc2.d/S15dahdi -> ../init.d/dahdi
   /etc/rc3.d/S15dahdi -> ../init.d/dahdi
   /etc/rc4.d/S15dahdi -> ../init.d/dahdi
   /etc/rc5.d/S15dahdi -> ../init.d/dahdi
DAHDI has been configured.
If you have any DAHDI hardware it is now recommended you
edit /etc/dahdi/modules in order to load support for only
the DAHDI hardware installed in this system.  By default
```

support for all DAHDI hardware is loaded at DAHDI start.

I think that the DAHDI hardware you have on your system is:

usb:004/002 xpp_usb- e4e4:1150 Astribank-multi no-firmware

This screen (above) asks you to change the file /etc/dahdi/modules to load only the required drivers for your specific configuration and show the detected hardware. Edit the file /etc/dahdi/modules and load only the required hardware. In my case, I was using a test machine with a Xorcom Astribank 6FXS and 2FXO. The file is shown below.

```
# Contains the list of modules to be loaded / unloaded by /etc/init.d/dahdi.
#
# NOTE:  Please add/edit /etc/modprobe.d/dahdi or /etc/modprobe.conf if you
#        would like to add any module parameters.
#
# Format of this file: list of modules, each in its own line.
# Anything after a '#' is ignore, likewise trailing and leading
# whitespaces and empty lines.

# Digium TE205P/TE207P/TE210P/TE212P: PCI dual-port T1/E1/J1
# Digium TE405P/TE407P/TE410P/TE412P: PCI quad-port T1/E1/J1
# Digium TE220: PCI-Express dual-port T1/E1/J1
# Digium TE420: PCI-Express quad-port T1/E1/J1
#wct4xxp

# Digium TE120P: PCI single-port T1/E1/J1
# Digium TE121: PCI-Express single-port T1/E1/J1
# Digium TE122: PCI single-port T1/E1/J1
#wcte12xp

# Digium T100P: PCI single-port T1
# Digium E100P: PCI single-port E1
#wct1xxp

# Digium TE110P: PCI single-port T1/E1/J1
#wcte11xp

# Digium TDM2400P/AEX2400: up to 24 analog ports
# Digium TDM800P/AEX800: up to 8 analog ports
# Digium TDM410P/AEX410: up to 4 analog ports
#wctdm24xxp

# X100P - Single port FXO interface
# X101P - Single port FXO interface
#wcfxo

# Digium TDM400P: up to 4 analog ports
#wctdm
```

```
# Xorcom Astribank Devices
xpp_usb
```

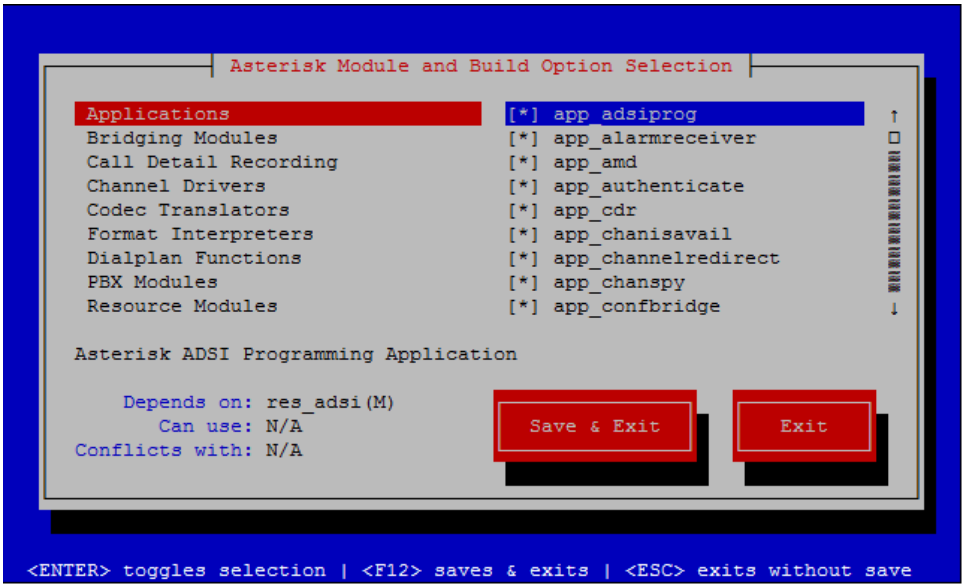
Re-initialize your computer and verify the correct loading of the drivers.

Compiling Asterisk

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

```
cd /usr/src/libpri-1.4.10.2
make
make install
cd /usr/src/asterisk-1.6.2.5
./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.



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
```

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

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

Available runtime options for Asterisk

You can show the available runtime options using `asterisk -h`

```
sipast:/usr/src/asterisk-1.6# asterisk -h
```

```
Asterisk 1.6.1.1, Copyright (C) 1999 - 2008, 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 DAHDI 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 debugging and console output on the console
  -n              Disable console colorization
  -p              Run as pseudo-realtime thread
  -q              Quiet mode (suppress output)
  -r              Connect to Asterisk on this machine
  -R              Same as -r, except 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)
```

```
-s <socket>    Connect to Asterisk via socket <socket> (only valid with -r)
```

Installation directories

Asterisk is installed on several directories, which can be modified in the `asterisk.conf` file.

asterisk.conf

```
[directories](!) ; remove the (!) to enable this
astetcdir => /etc/asterisk
astmoddir => /usr/lib/asterisk/modules
astvarlibdir => /var/lib/asterisk
astdbdir => /var/lib/asterisk
astkeydir => /var/lib/asterisk
astdatadir => /var/lib/asterisk
astagidir => /var/lib/asterisk/agi-bin
astspooldir => /var/spool/asterisk
astrundir => /var/run/asterisk
astlogdir => /var/log/asterisk

[options]
;verbose = 3
;debug = 3
;alwaysfork = yes ; same as -F at startup
;nofork = yes ; same as -f at startup
;quiet = yes ; same as -q at startup
;timestamp = yes ; same as -T at startup
;execincludes = yes ; support #exec in config files
;console = yes ; Run as console (same as -c at startup)
;highpriority = yes ; Run realtime priority (same as -p at startup)
;initcrypto = yes ; Initialize crypto keys (same as -i at startup)
;nocolor = yes ; Disable console colors
;dontwarn = yes ; Disable some warnings
;dumpcore = yes ; Dump core on crash (same as -g at startup)
;languageprefix = yes ; Use the new sound prefix path syntax
;internal_timing = yes
;systemname = my_system_name ; prefix uniqueid with a system name for global
uniqueness issues
;autosystemname = yes ; automatically set systemname to hostname - uses
'localhost' on failure, or systemname if set
;maxcalls = 10 ; Maximum amount of calls allowed
;maxload = 0.9 ; Asterisk stops accepting new calls if the load average exceed
this limit
;maxfiles = 1000 ; Maximum amount of openfiles
;minmemfree = 1 ; in MBs, Asterisk stops accepting new calls if the amount of
free memory falls below this watermark
;cache_record_files = yes ; Cache recorded sound files to another directory
during recording
;record_cache_dir = /tmp ; Specify cache directory (used in cnjunction with
cache_record_files)
```

```
;transmit_silence_during_record = yes ; Transmit SLINEAR silence while a
channel is being recorded
;transmit_silence = yes ; Transmit SLINEAR silence while a channel is being
recorded or DTMF is being generated
;transcode_via_sln = yes ; Build transcode paths via SLINEAR, instead of
directly
;runuser = asterisk ; The user to run as
;rungroup = asterisk ; The group to run as
;lightbackground = yes ; If your terminal is set for a light-colored background
documentation_language = en_US ; Set the Language you want Documentation
displayed in. Value is in the same format as locale names
;hideconnect = yes ; Hide messages displayed when a remote console connects and
disconnects

; Changing the following lines may compromise your security.
;[files]
;astctlpermissions = 0660
;astctlowner = root
;astctlgroup = apache
;astctl = asterisk.ctl

[compat]
pbx_realtime=1.6
res_agi=1.6
app_set=1.6
```

Log files and log rotation

Asterisk PBX logs its messages on the `/var/log/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
```

```
    ;
```

Some console commands are associated with 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.

```
    /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
```

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 fewer privileges to the user limits an intruder's possible actions.

To change Asterisk's running user:

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

Step 2: Uncomment the following lines:

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

Step 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/dahdi
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/dahdi
```

Step 4: Test changes using `/etc/init.d/asterisk`

Uninstalling Asterisk

To uninstall Asterisk, use:

```
make uninstall
```

To uninstall Asterisk and all configuration files, use:

```
make uninstall-all
```

Asterisk installation notes

This section will provide some advice about issues to address before installing Asterisk.

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 such as X-Windows. If you need to run CPU-intensive processes (e.g., a huge database), use a separate server. Generally speaking, Asterisk is susceptible to hardware performance variations. Thus, try using Asterisk in a hardware environment that does not require more than 40% of CPU utilization.

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 jitters; however, if you use a poorly configured local area network, voice quality will suffer. It is only possible to guarantee good voice quality using quality of service (QoS) in switches and routers. 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 a distorted or crappy voice. Follow these recommendations to ensure the best possible voice quality:

- Use end-to-end QoS if possible or economically feasible. With end-to-end QoS, the voice quality is perfect. No excuses!
- Avoid using 10/100 Mbps hubs for voice in a production environment. Collisions can impose jitters on the network. Full duplex 10/100 Mbps are preferred because no collisions occur.
- Use VLANs to separate unnecessary broadcasts of the voice network. You don't want a virus destroying your voice network with ARP broadcasts.
- Educate users about expectations in a voice network. Without QoS, don't state that the voice will be perfect as in most cases it won't be. A quality of voice similar to a mobile phone will most often be achieved. Use quality phones as problems with firmware and hardware design are common.

Summary

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

Quiz

1. What's the minimal Asterisk hardware configuration?

2. Telephony interface cards for Asterisk usually have some Digital Signal Processors (DSPs) 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 quality of service (QoS).

   A. True

   B. False

4. You should always choose the latest Asterisk version as it is the most stable version.

   A. True

   B. False

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

6. If you don't have a TDM interface card, you will end up needing a clock source for synchronization. The `dahdi_dummy` driver fills this role by using the 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 desktop interfaces such as GNOME or KDE out. Graphical user interfaces take up numerous 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?

3
Building a simple PBX

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

Objectives

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

- Understand and edit configuration files;
- Install soft-phones based on SIP;
- Install and configure a SIP trunk;
- Install and configure an analog connection;
- Dial between extensions;
- Dial between phones and external destinations; and
- Configure an auto attendant.

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 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. Use "=" when you want to declare a variable and "=>" to designate an object. The syntax is the same between all files, but three types of grammar are used, as discussed below.

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	chan_dahdi.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

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, and op3 while object 2 is created with options op1, op2, and op3.

Object options inheritance grammar

This format is used by the files `chan_dahdi.conf` and `agents.conf`, where numerous options are available, and most interfaces and objects share the same options. Typically, one or more sections have 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 the options op1 and op2 to “bas” and “adv”, respectively. When object 1 is instanced, it is created using option 1 as “bas” and option 2 as “adv”. After defining object 1, we change option 1 to “int”. Next, we create object 2 with option 1 as “int” and option 2 as “adv”.

Complex entity object

This format is used by `iax.conf`, `sip.conf`, and other configuration files in which numerous entities with many options exist. Typically, this format does not share a large volume of common configurations. Each entity receives a context. Sometimes reserved contexts exist, 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.

Options to build a LAB for Asterisk

To configure a PBX, you will need some basic hardware. It is not hard or expensive, but there are some options to be considered. All you will need are two phones and a connection to the public network. Some options and combinations are possible when creating your lab, which we will discuss below.

Option 1: Complete LAB

With the complete LAB, it is possible to test all the scenarios available and compare solutions such as ATA, IP-phones, and soft-phones. You can also learn about analog and SIP trunks.

Qty.	Description
1	SIP Analog Telephone Adapter
2	IP Phone
3	Dedicated Server for the Asterisk Server
4	Workstation with the soft-phone
5	Analog Interface Card with at least two interfaces: 1 FXO and 1 FXS
6	VoIP provider Account

Option 2: Economy LAB

With the economy LAB, we simplify it a bit. We use the ATA, which is usually less expensive than the IP-phone, and a single FXO card, which is really inexpensive. We won't be able to use analog phones connected directly to the server, but this does not commonly occur in practice.

Qty.	Description
1	SIP Analog Telephone Adapter
2	Dedicated Server for Asterisk
3	Workstation for the soft-phone
4	Analog Interface Card with 1 FXO
5	Account in a VoIP provider

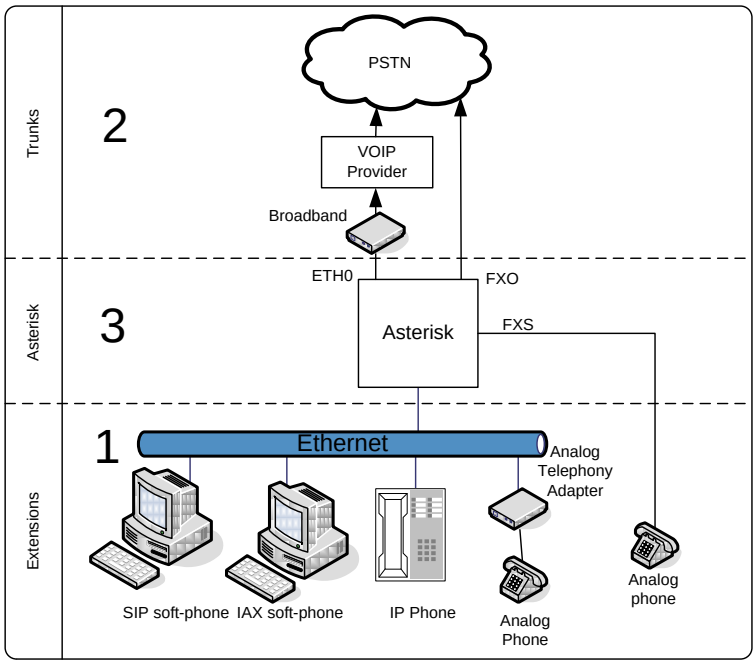
Option 3: Super economy lab

The third LAB uses a virtualized server in the student's own notebook. The problem with this model is the conflicts generated by the UDP port. Sometimes both the Asterisk server and the soft-phone try to access the same port, preventing Asterisk from binding the address port. Another issue is the quality of the calls; virtual environments are not indicated for real-time applications such as Asterisk. Use a free soft-phone for the server and workstation and a trunk connection to a SIP provider.

Qty.	Description
1	Laptop with 1 GB memory and a soft-phone
3	Virtual Machine (VMWare, Xen, or other) to install Asterisk and a soft-phone
4	Account in a VoIP provider

Installation Sequence

To help you understand the installation sequence, we outlined the sequence of steps necessary to install and configure Asterisk.



1. Extensions configuration
    a. SIP extensions (ATA, Soft-phone, IP Phone)
    b. IAX extensions
    c. FXS extensions
2. Trunk configuration
    a. Configuration of a SIP trunk
    b. Configuration of a FXO trunk
3. Building a basic dial plan
    a. Dialing between extensions
    b. Dialing external destinations
    c. Receiving a call from in the operator extension
    d. Receiving a call in an auto-attendant

Configuration of the extensions

The extensions are SIP, IAX, or analog phones connected to an FXS port. To configure an extension, you should edit the configuration file related to the channel (`sip.conf`, `iax.conf`, `chan_dahdi.conf`)

SIP extensions

Let's configure the SIP phones. The idea is to configure a simple PBX. (Subsequent chapters will provide 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.

The section `[general]` includes some parameters to be configured; it is the first section we will configure. The main options are:

- allow/disallow: Defines which codecs are going to be used.
- bindaddr: Address to be bound to the 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 it is changed in the client section. We used dummy for security reasons. Unauthenticated users get into this context when the option allowguest is set to yes.
- bindport: SIP UDP port to listen.
- maxexpirey: Maximum time to register (seconds).
- defaultexpirey: Default time to register (seconds).
- register: Registers Asterisk to another host.
- allowguest: Usually set to no to avoid non-authenticated users in the context of the [general] section.
- alwaysauthreject: When an incoming INVITE or REGISTER is received, always reject with an identical response (valid username, invalid password). This avoids username guessing.

Example:

```
[general]
bindport = 5060
bindaddr = 10.1.30.45
context = dummy
disallow = all
allow = ulaw
maxexpirey = 120
defaultexpirey = 80
allowguest=no
alwaysauthreject=yes
```

SIP clients

After completing the general sections, it is time to set up the SIP clients. I would once again like to remind the reader 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 occur at the same time.
- **host:** IP address or host name. The most common option is "dynamic", which is used when the host registers to Asterisk.
- **secret:** Password to authenticate peers and users.

> Warning: **Use strong passwords,** with at least 8 characters, alphanumeric and numeric characters, and at least one symbol. Reports of hacked servers have appeared in the mailing lists, and brute force password crackers for SIP are easily available for script kiddies. Toll fraud costs thousands of dollars for consumers and providers.

Example:

```
[6000]
type=friend
secret=#MySecret1#7
host=10.1.30.50
context=from-internal

[6001]
type=friend
secret=Mys3cr3t#
host=dynamic
context=from-internal
defaultip=10.1.30.17
```

Using Templates

One of the greatest benefits of the version 1.6 is the use of templates. You can now define a template for your SIP/IAX peers. The syntax for defining templates is as follows:

```
[section](options)
label = value
```

Example #1 - Defining a template

```
[default](!)
type=friend
host=dynamic
```

In the example above, the character (!) tells the parser that this section is only a template, it should not be parsed.

Example #2 – You can use a pre-defined template simply adding the template name after the object using parenthesis. Do not leave a space between the end bracket "]"end the parenthesis "(". Templates are transitive, so you can define templates inheriting from other templates.

```
[2003](default)
secret=2003
context=default
```

In the example above the resulting peer will be parsed as:

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

There are more advanced scenarios using templates, you can check some more advanced examples in the sip.conf sample file.

IAX Extensions

You may also create IAX extensions. This protocol is native to the Asterisk, and we will have an entire section devoted to it later in this book. For now, let's create a few extensions using the protocol.

The file is very similar to `sip.conf`. As the first section to be configured, the section `[general]` has certain parameters to be configured. The main options are:

- allow/disallow: Defines which codecs are going to be used.
- bindaddr: Address to be bound 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. We used dummy for security reasons. Unauthenticated users get into this context when the option allowguest is set to yes.
- bindport: SIP UDP port to listen.
- delayrejects: When set to yes, delays sending the authentication rejects, which improves the security against brute force password attacks.
- bandwidth: When set to high, it allows the selection of high bandwidth codecs, such as the g711 in their variants `ulaw` and `alaw`.

The following is a sample of the `[general]` section of the file `iax.conf`.

```
[general]
bindport = 4569
bindaddr = 10.1.30.45 ;(use your IP)
context = dummy
delayreject=yes
bandwidth=high
disallow = all
```

```
allow = ulaw
```

IAX Clients

After finishing the general sections, it is time to set up the IAX clients.

- **[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 occur at the same time.
- **host:** IP address or host name. The most common option is `dynamic`, which is used when the host registers to Asterisk.
- **secret:** Password to authenticate peers and users.

> Warning: **Use strong passwords** with at least 8 characters, alphanumeric and numeric characters, and at least one symbol. Reports of hacked servers have appeared in the mailing lists, and brute force password crackers for SIP md5 hashes are available for script kiddies. Toll fraud costs thousands of dollars for consumers and providers.

Example:

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

[6003]
context=from-internal
type=friend
secret=#sup3rs3cr3t#
host=dynamic
context=from-internal

[6004]
context=from-internal
type=friend
secret=#s3cr3ts3cr3t#
host=dynamic
context=from-internal
```

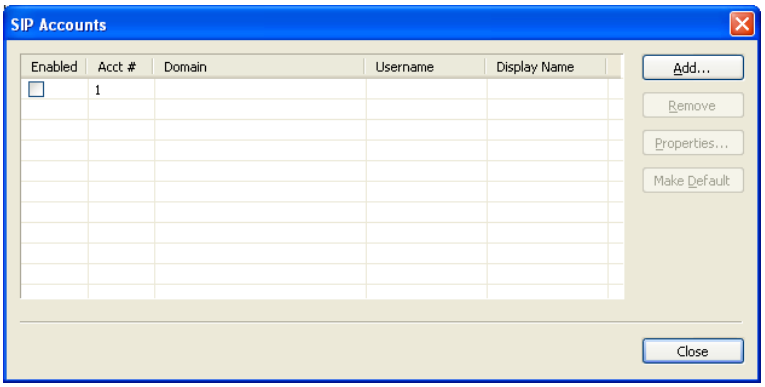
Configuring the SIP devices

After defining the phones in the Asterisk configuration file, it is time to configure the phone itself. In this example, we will show how to configure a free soft-phone—in this case, xlite from Counterpath (http://www.counterpath.com). Check your device's manual to understand the parameters of your phone.

Step 1: Configure the phone to use the extension 6000

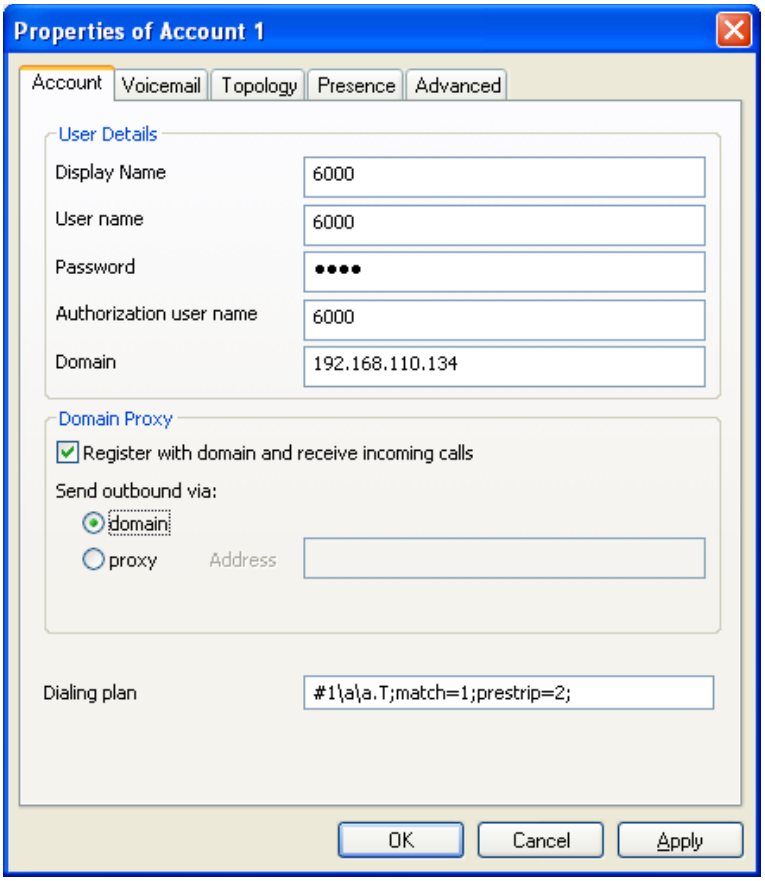
Execute the installation program.

After the execution, click the mouse's right button and choose SIP Account Settings.



Select the button Add...

Fill in the required information.



Display Name: 6000
User Name: 6000
Password: =#MySecret1#7
Authorization User Name: 6000
Domain: ip_of your_server

Confirm that your phone is registered using the console command `sip show peers`.

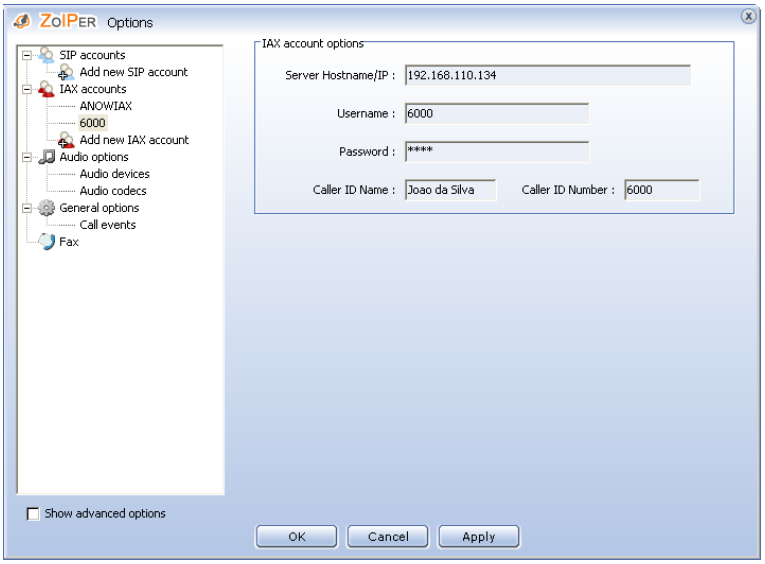
Repeat the configuration for the phone 6001.

Configuring the IAX devices

In this example, we are going to use the free soft-phone Zoiper, which you can download from www.zoiper.com.

1. Download and install the Zoiper Free.
2. Click with the right button to access options.

3. Select new IAX account.
4. Insert the related options for the 6003 phone and optionally for the 6004.

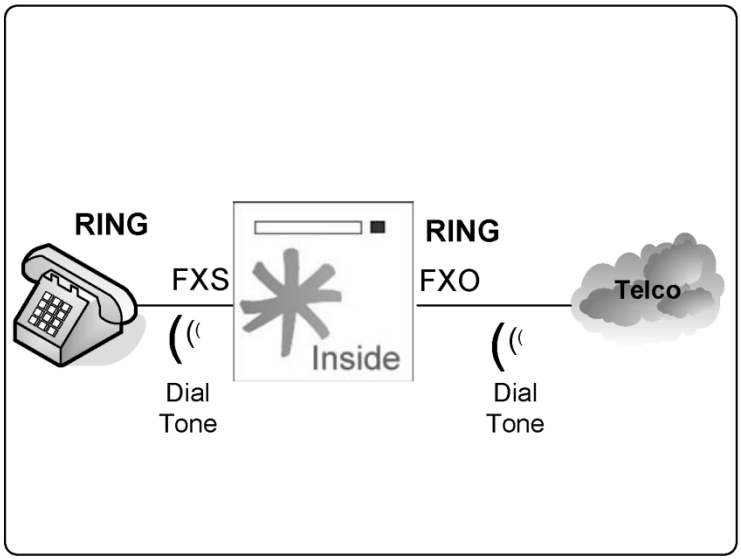


5. Save the configuration and check if the phone is registered using `iax2 show peers`.

> Important: Use one account for SIP and another one for IAX. If you want to configure the system to ring both IAX and SIP at the same time, we will show you how to do so in the dial plan section.

Configuring a PSTN interface

To connect to the PSTN, you will need an interface foreign exchange office (FXO) and a telephone line. You can use an existing PBX extension too. You can obtain a telephony interface card with an FXO interface from several manufacturers. In this example, we will show you how to install a DAHDI interface card.



Analog lines using DAHDI

You can buy an analog card compatible with the DAHDI from several manufacturers. X100P was one of the first Digium cards and had already been discontinued. Some manufacturers still produce similar clones. In addition to the price of the X100P, we have found several issues between these cards and new motherboards, so use it with care. X100P, in my opinion, is not a good choice for a production environment. Any card compatible with DAHDI should work.

Thanks to the team of DAHDI developers, we now have a tool for detecting and configuring the interface cards almost automatically. If you have just installed the DAHDI drivers, please don't forget to run `make config` and reboot the machine to load it automatically. You can use the commands below to detect and configure your card.

Step 1: To detect your hardware, use:

`dahdi_hardware`.

Step 2: To configure use:

`dahdi_genconf`.

The command above will generate two files `/etc/system/dahdi.conf` and `/etc/asterisk/dahdi-channels.conf`. The default parameters for `dahdi_genconf` are usually fine, but you can change them in the file `/etc/dahdi/genconf_parameters`. By default, it will insert the lines (FXO) in the context `from-pstn` and the phones (FXS) in the context `from-internal`.

Step 3: After running `dahdi_genconf`, in the last line of the file `/etc/asterisk/chan_dahdi.conf` insert the following line:

```
#include dahdi-channels.conf
```

Step 4: Edit the file `/etc/dahdi/modules` and comment for all the unused drivers. Reboot before proceeding and check if the channels are being recognized using:

```
CLI>dahdi show channels
```

Connecting to the PSTN using a VoIP provider

If your budget is really limited, you can configure a SIP trunk to connect to the PSTN. It is certainly the most affordable way to connect to the PSTN. Thousands of VoIP providers exist worldwide. To connect to one of them, you will need some parameters.

Parameters provided by the SIP provider.

- username: **login**
- password: **secret**
- Provider's domain: **domain**
- UDP port: **5060**
- Allowed codecs:**g729, ilbc, alaw**

Two parameters should be determined by you.

- Extension to receive calls—in this case: **9999**
- context: **from-sip**

Configure the file sip.conf using the following parameters:

```
 [general]
srvlookup=yes
register => login:secret@domain:port/9999

[siptrunk]
username=login
type=peer
secret=secret
port=5060
insecure=invite
host=dominio
fromuser=login
fromdomain=domain
dtmfmode=rfc2833
context=from-sip
disallow=all
allow=ilbc
allow=alaw
allow=g729
```

To access this trunk, we will use the channel name `SIP/siptrunk`

Dial plan introduction

Dial plan is like Asterisk's heart. It defines how Asterisk handles every single 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 in the `/etc/asterisk` directory. This file uses the simple group grammar and has four major concepts:

- Extensions
- Priorities
- Applications
- Contexts

Let's create a basic dial plan. In subsequent sections of this book, I will devote a chapter exclusively to the 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.

The structure of the file extensions.conf

The `extensions.conf` file is separated into 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.

The section `[general]`

The general section sits at the top of the file. Before starting to configure the dial plan, it is helpful to know the general options that control certain dial plan behaviors. These options are:

- **static and write protect:** If `static=yes` and `writeprotect=no`, you can use the CLI command `save dialplan`.

> Warning: If you issue a `save dialplan` command from the CLI, you will end up losing 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.

- **extenpatternmatchnew (**new in the 1.6 version): This uses a new algorithm to match the extension from 1.5 to 300 times faster than the existing one, particularly if you have a large number of extensions. It is a new feature and should be used with care; it defaults to `no`.
- **userscontext**: This is the context where the entries from the `users.conf` are registered.

The section [globals]

In the `[globals]` section you will define global variables and their initial values. You can access the variable in the dial plan 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:

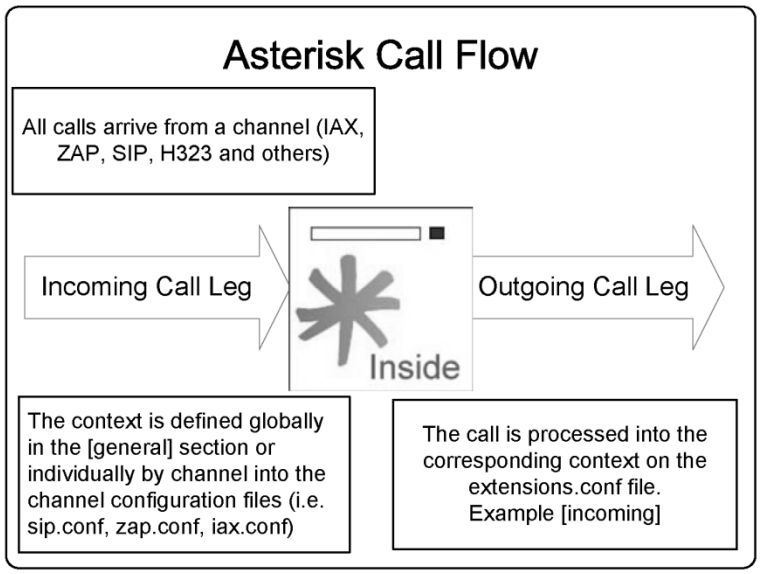
```
INCOMING>DAHDI/8&DAHDI/9
RINGTIME=>3
```

In the following example, you can set and test a global variable in the dial plan.

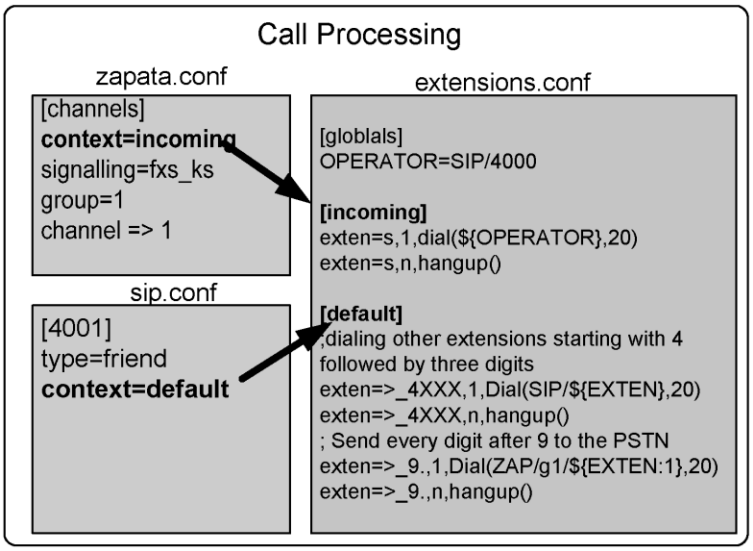
```
exten=9000,1,set(GLOBAL(RINGTIME)=4)
exten=9000,n,Noop(${GLOBAL(RINGTIME)})
exten=9000,n,hangup()
```

Contexts

Context is the named partition of the dial plan. After the `[general]` and `[globals]` sections, the dial plan is a set of contexts in which each context has several extensions, each extension has several priorities, and each priority calls an application with several arguments.

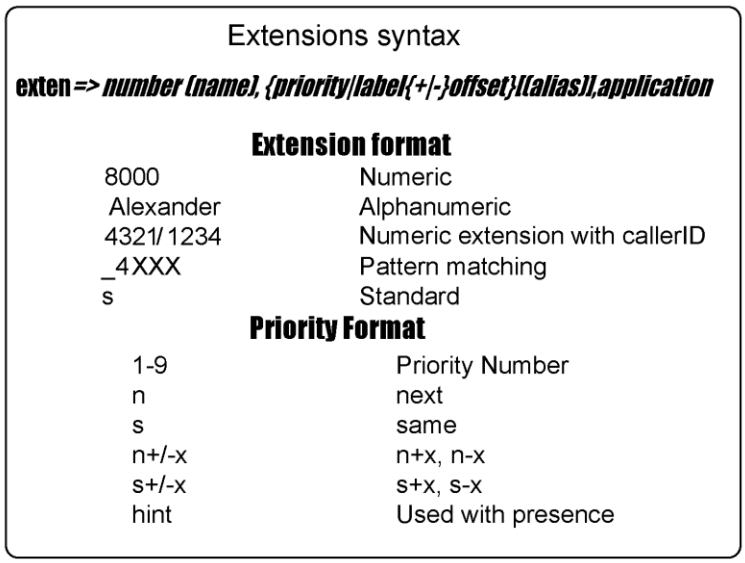


You can build a simple dial plan to reach other phones and the PSTN. However, Asterisk is much more powerful than that. Our objective is to teach you more details of what is possible in the dial plan.



Extensions

Unlike 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 as well. Extensions are case sensitive. However, you cannot create two extensions with the same name but 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,n,Playback(tt-weasels)
exten=>123,n,Hangup
```

Above you find the list of instructions to be processed when the 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 play back an audio file called `tt-weasels`. The third priority hangs up the channel.

Another option is to handle the call according to the caller ID. You can use the `/` character to specify the caller ID to be processed.

Examples:

```
exten=>123/100,1,Answer()
exten=>123/100,n,Playback(tt-weasels)
exten=>123/100,n,Hangup()
```

This example will trigger extension 123 and execute the following options only if the caller ID is 100. This can also be done by using the pattern described below:

```
exten=>1234/_256NXXXXXX,1,Answer()
```

hint: maps an extension to a channel. It is used to monitor the channel state. It is used in conjunction with presence. The phone has to support it.

Patterns

You can use patterns and literals in the dial plan. Patterns are very useful for reducing the dial plan size. All patterns start with the "_" character. The following characters may be used to define a pattern. The figure identifies the patterns available for use with Asterisk.

Pattern Matching

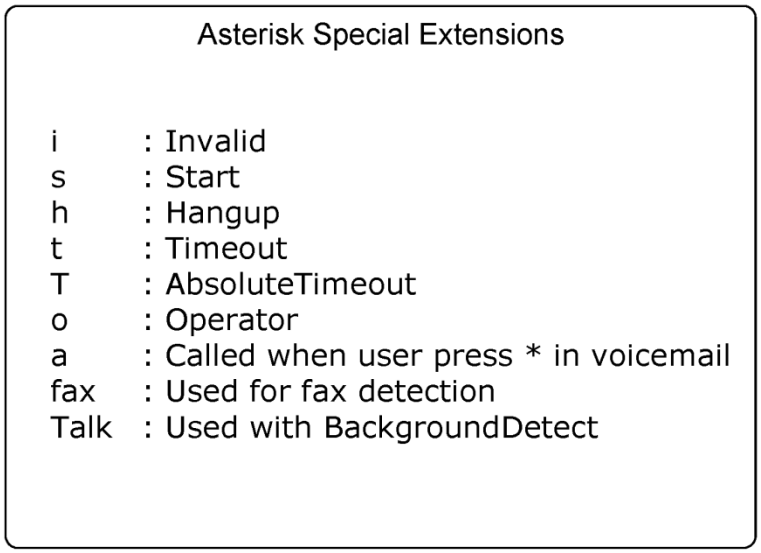
_ (Underscore)	Start of a match
. (dot)	Matches one or more characters
! (exclamation)	Matches zero or more characters
[123-7]	Matches any digit in the brackets (1,2,3to 7)
X	Any digit from 0-9
Z	Any digit from 1-9
N	Any digit from 2-9

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, 7400-7999)
_9.	Any number staring with 9
_9XXXXXXX	Any number with 8 digits starting with 9

Special extensions

Asterisk uses some extension names as 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 remain inactive after a prompt has been played. It is also used to hang up an inactive line.

T: AbsoluteTimeout. If you establish a call limit using the `absolutetimeout()` function, once the call exceeds the limit defined, it will be sent to the T extension.

h: Hangup. It is called after the user disconnects the call.

i: Invalid. It is triggered when you call an non-existent 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" during the voicemail.

The use of these extensions can change the content of the billing records (CDR)—in particular, the field `dst` will not have the number dialed. To work around this problem, you should use the option `g` in the `dial()` application and consider the functions `resetcdr(w)` and/or `nocdr()`

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 in the following example, 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. Thus, the variable `${EXTEN}` is different from `${exten}`.

Global variables

Global variables can be configured in the `[global]` section in the `extensions.conf` file or using the application:

```
set(Global(variable)=content)
```

Channel-specific variables

Channel-specific variables are configured using the application `set()`. 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 commonly used variables are:

- ${EXTEN} Extension dialed
- ${CONTEXT} Current context
- ${CALLERID(name)}
- ${CALLERID(num)}
- ${CALLERID(all)} Current caller ID
- ${PRIORITY} Current priority

Other channel-specific variables 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
```

Environment-specific variables

Environment-specific variables can be used to access variables defined in the operating system. You can set environment-specific variables using the function ENV(). For example:

```
${ENV(LANG)}
Set(ENV(LANG))=en_US
```

Application-specific variables

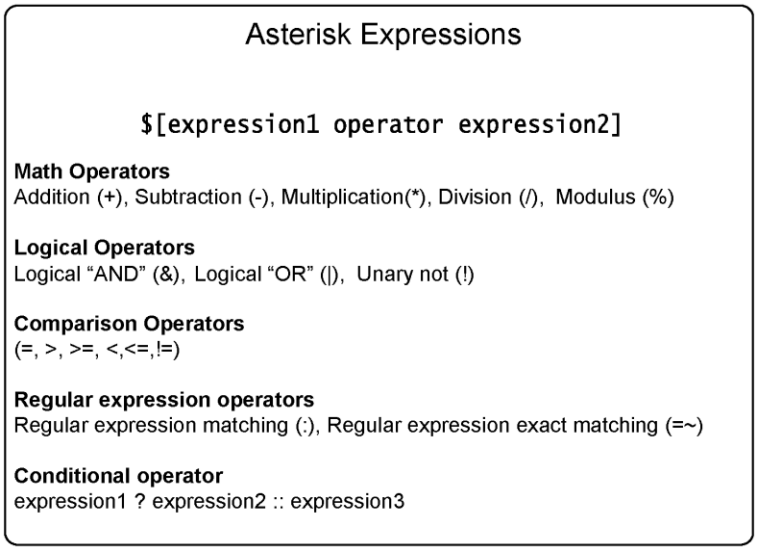
Some applications use variables for data input and output. You can set variables before calling the application or retrieve the variable after the application execution. For example:

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 the actual call.
- ${DIALSTATUS} This is the status of the call:
    - CHANUNAVAIL
    - CONGESTION
    - NOANSWER
    - BUSY
    - ANSWER
    - CANCEL
    - DONTCALL
    - TORTURE
- ${CAUSECODE} -> Error message for the call.

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 as follows:

```
$[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 entire expression by the resulting value.

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 used to describe a search pattern. You can think of regular expressions as wildcards. Regular expressions are used to match a string to a pattern to check the 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 it is false.

- = equal
- != not equal
- < less than
- > greater than
- <= less than or equal to
- >= greater than or equal to

LAB. Evaluate the following expressions:

Put these expressions in your dial plan and use the NoOP() application to evaluate the expressions. Dial 9002 and examine the 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
```

Functions

After version 1.2, some applications were replaced by functions to allow the processing of certain variables in a more advanced way than only expressions. You can see the full list of functions by issuing the following console command:

```
CLI>core show functions
```

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 shows 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").

Substrings: Returns the substring, starting from the positing defined by the "offset" parameter, with the string length defined in the "length" parameter. If the offset is negative, it starts from right to left, beginning at the end of the string. If the length is omitted or negative, it takes the whole string starting with the offset.

```
${string:offset:length }
```

Example #1: Several substrings

```
${123456789:1}-returns 23456789
${123456789:-4}-returns 6789
${123456789:0:3}-returns 123
${123456789:2:3}-returns 345
${123456789:-4:3}-returns 678
```

Example #2: Take the area code from the first three digits.

```
exten=>_NXX.,1,Set(areacode=${EXTEN:0:3})
```

Example #3: Takes all digits from the variable ${EXTEN}, except for the area code.

```
exten=>_516XXXXXXX,1,Dial(${EXTEN:3})
```

String concatenation

To concatenate two strings, simply write them together.

```
${foo}${bar}
555${number}
${longdistanceprefix}555${number}
```

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 the following example:

```
CLI>core show application dial
```

To build a simple dial plan, you need to know a few applications. We will discuss more advanced examples later in the book.

Simple applications to build a dialplan

- Answer – Answer a channel
- Dial – Dial other channel
- Hangup – Hang up a channel
- Playback – Play back an audio file
- Goto – Jump to a particular priority, extension or context

We will use these applications (above) to create a simple dial plan for two basic PBXs.

Answer()

[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 the number of milliseconds specified in 'delay' before answering the call.

Dial()

The following description can be obtained by issuing the show application dial in the dial plan. For easy searching, it is reproduced below. The syntax for the Dial application is also shown below:

```
;dial to a single 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 requested channels will then be hung up.

Unless a timeout is specified, the Dial application will wait indefinitely until one of the called channels answers, the user hangs up, or all of the called channels are busy or unavailable. The execution of the dial plan 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 an 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 included in that group (as in `Set(GROUP()=...)`).

The following table summarizes some of the most frequently used options for the application dial. For the complete list, use the console command `core show application dial`.

A(x)	Plays an announcement to the called party, using 'x' as the file.
C	Resets the CDR for this call.
D	Allows the calling user to dial a 1-digit extension while waiting for a call to be answered. Exits to that extension if it exists in the current context or to the context defined in the EXITCONTEXT variable, if it exists.
D([called][:calling])	Sends 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	Forces the caller ID of the calling channel to be set as the extension associated with the channel using a dial plan 'hint'. For example, some PSTNs do not allow caller ID to be set to anything other than the number assigned to the caller.
g	Proceeds with dial plan execution at the current extension if the destination channel hangs up.
G(context^exten^pri)	If the call is answered, transfers the calling party to the specified priority and the called party to the specified priority+1.Optionally, an extension—or extension and context—can be specified. Otherwise, the current extension is used.
h	Allows the called party to hang up by sending the '*' DTMF digit
H	Allows the calling party to hang up by hitting the '*' DTMF digit.
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) Plays sounds for the caller. LIMIT_PLAYAUDIO_CALLEE yes|no Plays sounds for the person called. LIMIT_TIMEOUT_FILE File to be played when time is up. LIMIT_CONNECT_FILE ->File to be played when the call begins. LIMIT_WARNING_FILE ->File to be played as a 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.
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 '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 within 20 seconds, the extension will go to the next priority.

Hangup()

Hangs up the calling channel

[Description]

`Hangup([causecode])`: This application will hang up the calling channel. If a cause code is given, the channel's hang-up cause will be set to the given value.

Goto()

Jump to a particular priority, extension, or context

[Description]

`Goto([[context|]extension|]priority):` This application will cause the calling channel to continue the dial plan execution at the specified priority. If no specific extension (or extension and context) are specified, this application will jump to the specified priority of the current extension. If the attempt to jump to another location in the dial plan is not successful, the channel will continue at the next priority of the current extension.

Building a dial plan

To build a simple dial plan, you need to treat all incoming and outgoing calls by creating contexts and extensions. In this section, we will show you how to build the most common extensions.

Dialing between extensions

To enable dialing between extension, we could use the channel variable ${EXTEN}, which refers to the dialed extension. For example, if the extension range is between 4000 and 4999 and all extensions use SIP, we could adopt the following command:

```
[from-internal]
exten=_4XXX,1,Dial(SIP/${EXTEN})
```

Dialing to an external destination

To dial an external destination you could precede the number dialed with a route. In North America, it is common 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 the following:

> If you want to use the SIP trunk instead of the DAHDI, use SIP/trunk as the channel

```
[from-internal]
exten=_9NXXXXXX,1,Dial(DAHDI/1/${EXTEN:1},20,tT)

or

exten=_9NXXXXXX,1,Dial(SIP/trunk/${EXTEN:1},20,tT)
```

The above line will permit you to dial 9 and the desired number. In the example given, you will use the first DAHDI channel (DAHDI/1). If you have several lines and this one is busy, the call will not be completed. However, you could use the following line to automatically choose the first available DAHDI channel. Optionally, you can use the SIP trunk instead of DAHDI.

```
[from-internal]
exten=_9NXXXXXX,1,Dial(DAHDI/g1/${EXTEN:1},20,tT)
```

The "g1" parameter will search for the first available channel in the group, allowing the use of all channels. Using the line below, you could dial a long distance number.

```
[from-internal]
exten=_91NXXNXXXXXX,1,Dial(DAHDI/g1/${EXTEN:1},20,tT)
```

Dialing 9 to get a PSTN line

If you do not have any restrictions to external dialing, you could simplify and use the following:

```
[from-internal]
exten=9,1,Dial(DAHDI/g1,20,tT)
```

Receiving a call in the operator extension

In the following example, the operator extension is 4000. The PSTN line is connected to an FXO interface. In the chan_dahdi.conf file, the context specified is `from-pstn`. Any call coming from the PSTN will be routed to the context `from-pstn` in the dial plan. This line does not have direct inward dialing (DID); as such, we will have to receive the call via the "s" extension. If receiving from the SIP trunk, use the context `[from-sip]`.

```
[globals]
OPERATOR=SIP/6000

[from-pstn]
exten = s,1,Dial(${OPERATOR},40,tT)
```

```
exten = s,n,Hangup()

[from-sip]
exten = s,1,Dial(${OPERATOR},40,tT)
exten = s,n,Hangup()
```

Receiving a call using direct inward dialing (DID)

If you have a digital line, you will receive the dialed extension. When this is the case, you don't need to forward the call to the operator; rather, 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 would look like the following example:

```
[from-pstn]
exten => _85[5-9]X,1,Answer()
exten => _85[5-9]X,n,Dial(SIP/${EXTEN},15,tT)
exten => _85[5-9]X,n,Hangup()
```

Playing several extensions simultaneously

You can set Asterisk to dial an extension and, if it is not answered, to dial several other extension simultaneously, as indicated in the following example:

```
exten => 0,1,Dial(DAHDI/1,15,tT)
exten => 0,n,Dial(DAHDI/1&DAHDI/2&DAHDI/3,15)
exten => 0,n,Hangup()
```

In this example, when someone dials the operator, the channel DAHDI/1 is initially tried. If nobody answers after 15 seconds (timeout), the channels DAHDI/1, DAHDI/2 and DAHDI/3 will ring simultaneously for another 15 seconds.

Routing by Caller ID

In this example, you could give different treatments based on the caller ID, which could be useful for call spammers. For example:

```
exten => 8590/4832518888,1,Playback(I-have-moved-to-china)
exten => 8590,1,Dial(DAHDI/1,20)
```

In this example, we have added a special rule that, if the caller ID is 4832518888, you play back a message from the previously recorded file "I-have-moved-to-china". Other calls are accepted as usual.

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 following examples:

```
[globals]
Flavio => DAHDI/1
Daniel => DAHDI/2&SIP/pingtel
```

```
Anna => DAHDI/3
Christian => DAHDI/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.

Recording an announcement

In some of the options discussed later in this section, we will use recorded prompts. Here we show you an easy way to record them. We will use the application `Record()` to save the announcement using one's own phone.

```
[from-internal]
exten => _record.,1,Record(${EXTEN:6}:gsm)
exten => _record.,n,wait(1)
exten => _record.,n,Playback(${EXTEN:6})
exten => _record.,n,Hangup()
```

These instructions allow you to record any message from a soft-phone.

Example: dialing `recordmenu` from the softphone

The instructions will call the recording with the variable ${EXTEN:6} without the first six letters. In other words, the instruction is equivalent to `record(menu:gsm)`. All you have to do is dial `record + name_of_the_file_to_be_recorded`, press `#` to finish the recording, and wait to hear the recording.

Receiving the calls in an digital receptionist

Now that we have some simple examples, let's expand our learning about the applications `background()` and `goto()`. The key for interactive systems in Asterisk is the application `background()`, which allows you to execute an audio file that, when the caller presses a key, is interrupted in order to send the call to the extension dialed.

Syntax of the `background()` application:

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

Another application very useful is `goto()`. As the name implies, it jumps to the context, extension, and priority indicated.

Syntax of the application `goto()`:

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

Valid formats for the `goto()` command:

```
goto(context,extension,priority)
goto(extension,priority)
goto(priority)
```

In the following example, we will create a digital receptionist. It is very simple to edit the file `extensions.conf` and configure the following extensions:

```
[globals]
OPERATOR=SIP/6000

[from-pstn]
include=aapstn

[from-sip]
include=aasip

[aapstn]
exten=>s,1,answer()
exten=>s,n,set(TIMEOUT(response)=10)
exten=>s,n,background(menu1)
exten=>s,n,WaitExten(30)
exten=>s,n,Dial(${OPERATOR})
exten=>6000,1,Dial(SIP/6000)
exten=>6001,1,Dial(SIP/6001)
exten=>6003,1,Dial(IAX2/6003)
exten=>6004,1,Dial(IAX2/6004)
[aasip]
exten=>9999,1,answer()
exten=>9999,n,set(TIMEOUT(response)=10)
exten=>9999,n,background(menu1)
exten=>s,n,WaitExten(30)
exten=>9999,n,Dial(${OPERATOR})
exten=>6000,1,Dial(SIP/6000)
exten=>6001,1,Dial(SIP/6001)
exten=>6003,1,Dial(IAX2/6003)
exten=>6004,1,Dial(IAX2/6004)
```

In the file `menu1.gsm`, record the message "press the extension or wait for the operator". When the user dials the number 6000, he will be sent to extension 6000.

At this point, you should have a clear understanding of the use of several applications, including `answer()`, `background()`, `goto()`, `hangup()`, and `playback()`. If you do not have a clear understanding, please read this chapter again until you feel comfortable with the content. You will use the background application very often.

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.

Summary

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

Quiz

1. The channel configuration files are:

   A. /etc/dahdi/system.conf
   B. /etc/asterisk/chan_dahdi.conf
   C. sip.conf
   D. iax.conf

2. It is important to define a context in the channel configuration file as 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. The pattern _7[1-5]XX matches (choose all correct answers):

A. 7100

B. 7600

C. 7630

D. 7230

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

A. /etc/dahdi/system.conf

B. /etc/asterisk/chan_dahdi.conf

C. /etc/asterisk/asterisk.conf

D. /etc/asterisk/modules.conf

9. In the Options Inheritance grammar used by `chan_dahdi.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.

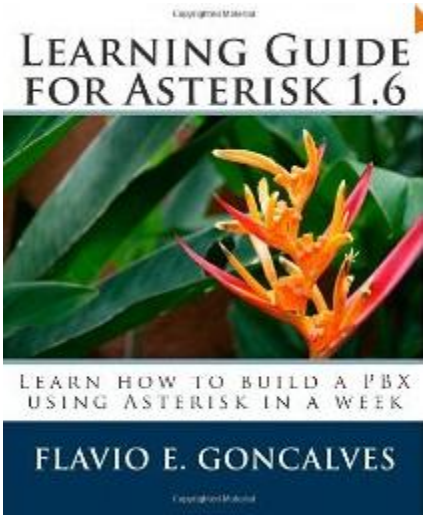
10. Priorities must be consecutive!

A. False

B. True

Thank you for downloading the Free Getting Started with Asterisk**.** Other books and trainings available from the same author:

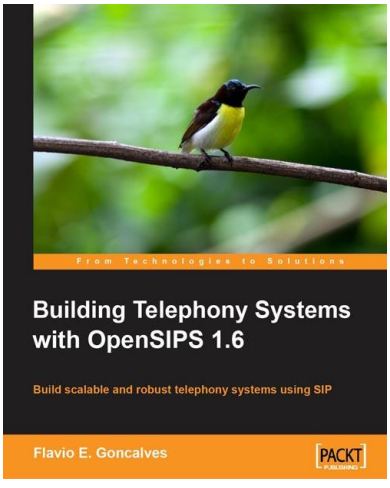
Learning Guide for Asterisk PBX



Language: English
Paperback: 324 pages
Release Date: June 2010
ISBN-10: 1452889368
ISBN-13: 978-1452889368
Author(s) : Flavio E. Goncalves

Learn how to build an IP PBX reading this book. It was prepared not as a reference but as a text book to teach you how to install, configure and manage the most advanced open source IP-PBX available in the market. This is the 10th edition of the Book, and this version was extensively reviewed. The book covers the installation, design and configuration of IP telephony networks based on the Asterisk PBX. Please, check the availability at http://www.amazon.com/Learning-Guide-Asterisk-1-6-Learn/dp/1452889368.

Building Telephony Systems with OpenSIPS 1.6



Language : English
Paperback : 284 pages [235mm x 191mm]
Release Date : January 2010
ISBN : 1849510741
ISBN 13 : 9781849510745
Author(s) : Flavio E. Goncalves

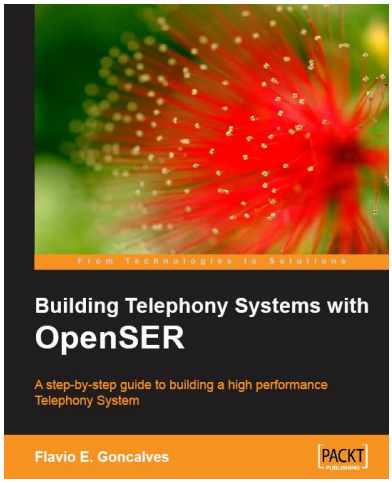
SIP is the most important VoIP protocol and OpenSIPS is clearly the open source leader in VoIP platforms based on pure SIP. The whole telecommunication industry is changing to an IP environment, and telephony in the way we know today will disappear in less than ten years. SIP is the protocol leading this disruptive revolution and it is one of the main protocols on next-generation networks. While a VoIP provider is not the only kind of SIP infrastructure created using OpenSIPS, it is certainly one of the most difficult to implement.

This book will give you a competitive edge by helping you to create a SIP infrastructure capable of handling tens of thousands of subscribers. You can extend the examples given in this book easily to other applications such as a SIP router, load balancing, IP PBX, and Hosted PBX as well. This book is an update of the title Building Telephony Systems with OpenSER.

The book starts with the simplest configuration and evolves chapter by chapter teaching you how to add new features and modules. It will first teach you the basic concepts of SIP and SIP routing. Then, you will start applying the theory by installing OpenSIPS and creating the configuration file. You will learn about features such as

authentication, PSTN connectivity, user portals, media server integration, billing, NAT traversal, and monitoring. The book uses a fictional VoIP provider to explain OpenSIPS. The idea is to have a simple but complete running VoIP provider by the end of the book.

Building Telephony Systems with OpenSER



Language : English
Paperback : 324 pages [235mm x 191mm]
Release Date : April 2008
ISBN : 1847193730
ISBN 13 : 978-1-847193-73-5
Author(s) : Flavio E. Goncalves

OpenSER is a flexible, free open-source VoIP server based on the Session Initiation Protocol (SIP), an application-layer control (or signaling) protocol for creating, modifying, and terminating sessions with one or more participants, including internet telephone calls, multimedia distribution, and multimedia conferences. Engineered to power IP telephony infrastructures up to large scale, OpenSER is written in pure C for Linux/Unix-like systems with architecture-specific optimizations to offer high performance; it is able to handle 4 million users on a single processor server. The server keeps track of users, sets up VoIP sessions, relays instant messages, and creates space for new plug-in applications. It can be used on systems with limited resources as well as on carrier-grade servers, scaling up to thousands of call setups per second. It is customizable, being able to feature as fast load balancer; SIP server flavors: registrar, location server, proxy server, redirect server; gateway to SMS/XMPP; or advanced VoIP application server. This book teaches how to develop a fast and flexible Session Initiation Protocol (SIP) server using OpenSER and shows how OpenSER can be used to implement features not available in Asterisk PBX.

OpenSIPS Bootcamp

The OpenSIPS 1.6 Bootcamp is a full 5 day (40 hours) intensive training providing in depth coverage of OpenSIPS Installation, Configuration and Administration. The students will learn how to download, compile and install OpenSIPS. After the installation, you will start to learn step by step how to configure OpenSIPS to authenticate users, install a GUI to help with daily administration, forward calls to the PSTN through Dialplan, integrate Asterisk and Voice Mail, Presence agent, Load Balancing, Traverse Nat for SIP and generate CDR records to a Radius Server. At the end, you will learn how to use troubleshooting tools to solve end user problems. All the knowledge that is transferred to you will be strongly backed-up by practice sessions where you will get hands-on experience in handling OpenSIPS SIP Server. The training is structured to be offer 50% - 50% between the theoretical and practical sessions.More information at **http://www.opensips.org/Training**

Consulting

If you are interested in consulting in innovative telephpny projects or in house training, please contact flavio@asteriskguide.com.