There's no place like a wireless home!
Create and install automated home services

Linux® Smart Homes FOR DUMMIES®

CD-ROM features MythTV, MisterHouse, and more

A Reference for the Rest of Us!®

FREE eTips at dummies.com®

Neil Cherry
Linux home automation guru

Linux® Smart Homes FOR DUMMIES®

by Neil Cherry

Wiley Publishing, Inc.

Linux® Smart Homes For Dummies®
Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774
www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

Library of Congress Control Number: 2006923951

ISBN-13: 978-0-7645-9823-4

ISBN-10: 0-7645-9823-6

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

1O/RZ/QX/QW/IN

About the Author

Neil Cherry has been working with computers, computer electronics, and software since 1978. He has been playing with X10 since 1982. He began automating his home in 1992 when a friend gave him an X10 computer interface, and he started the Linux Home Automation Web site (`www.linuxha.com`) in 1996. When he's not riding his bicycle or playing with home automation, he works for AT&T Research Lab South, Middletown, NJ, as a Test Engineer. You can reach him by e-mail at `linuxha@linuxha.com`.

About the Contributors

Terry Collings is the owner of TAC Technology, located in eastern Pennsylvania. He provides Linux consulting and training services to a variety of clients. Terry has been an adjunct faculty member at several colleges in his area where he taught A + and Network + certification courses. He also taught courses on UNIX, Linux, TCP/IP, and Novell Netware. Terry is the author of *Red Hat Enterprise Linux 4 For Dummies,* has co-authored three editions of Red Hat Networking and System Administration and contributed to several other Linux books. He was the technical editor for the following books: *KDE Bible, The Samba Book, UNIX Weekend Crash Course*, *Red Hat Linux 9 For Dummies, Solaris 9 For Dummies, Fedora Linux 2 For Dummies,* and *Linux Timesaving Techniques For Dummies.*

Gurdy Leete is a co-author of *OpenOffice.org For Dummies,* a technical editor for *Free Software For Dummies,* and the co-author of five other popular computer books. He's also an award-winning software engineer and a co-author of the Multitile plug-in for the GNU Image Manipulation Program (GIMP). Gurdy teaches digital imaging, graphic design, Web design, video, and animation at Maharishi University of Management in Fairfield, Iowa, where he has been a pioneer in using GNU/Linux applications in undergraduate art and design classes. His blog, titled *Free Software for Art, Music and Personal Creativity,* is at `www.peaceloveandhappiness.org`.

Mary Leete wrote *Free Software For Dummies* and co-wrote *OpenOffice.org For Dummies.* She has a B.S. in Computer Science and a Masters in Professional Writing, and she lives to write code as well as write about it. Mary has extensive experience as a systems analyst and programmer with a multitude of software on way too many platforms. She is also a freelance Web designer, a video producer, and an award-winning screenwriter, and she has written under contract for the producer of *The Buddy Holly Story,* among others.

Author's Acknowledgments

I wish to thank my wife, Diane, for putting up with my years of experiments and for not allowing me to kludge together anything. She's kept me honest and helped make our home automation work better. Honey, I love you and I'll take you out to dinner but first just one more compile.

Thanks to Terry Collings and Gurdy and Mary Leete who helped by writing various chapters that I was unable to. They really helped to make this book possible.

Thanks to Nicole Sholly and Virginia Sanders, the editors who worked with me on this book. I doubt most people know the amount of work a book takes to get written and how much help the editors give to make a book successful. I really appreciate all the help — thank you very much and I hope I get *it* now. I'd also like to thank the rest of the folks at Wiley who are too numerous to mention. They do a lot of the work to help get a book put together and to the stores but seldom get mentioned.

Thanks to Deepak Dube for his kind words of encouragement, without which I wouldn't have thought I could write a book.

Thanks also to Donald Brookman and Vincent Miller, my friends who always ask the most pertinent questions. (Are we there yet?!)

Publisher's Acknowledgments

We're proud of this book; please send us your comments through our online registration form located at www.dummies.com/register/.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial, and Media Development

Project Editor: Nicole Sholly

Acquisitions Editors: Kyle Looper, Tiffany Ma

Copy Editor: Virginia Sanders

Technical Editor: Dan DiNicolo

Editorial Manager: Kevin Kirschner

Media Development Specialists: Angela Denny, Kate Jenkins, Steven Kudirka, Kit Malone

Media Development Coordinator: Laura Atkinson

Media Project Supervisor: Laura Moss

Media Development Manager: Laura VanWinkle

Editorial Assistant: Amanda Foxworth

Sr. Editorial Assistant: Cherie Case

Cartoons: Rich Tennant (www.the5thwave.com)

Composition Services

Associate Project Coordinator: Tera Knapp

Layout and Graphics: Carl Byers, Andrea Dahl, Denny Hager, Joyce Haughey, Stephanie D. Jumper, Barbara Moore, Heather Ryan, Alicia B. South

Proofreaders: Leeann Harney, Heidi Unger

Indexer: Techbooks

Special Help: Andy Hollandbeck, Pat O'Brien

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Diane Graves Steele, Vice President and Publisher

Joyce Pepple, Acquisitions Director

Composition Services

Gerry Fahey, Vice President of Production Services

Debbie Stailey, Director of Composition Services

Contents at a Glance

Table of Contents

Chapter 9

Setting Up a Smart Phone System

In This Chapter

- Introducing a smart phone system
- Getting started
- Installing the hardware and software
- Making a smart call

Welcome to Asterisk, your open source toolkit for telephony applications and a full-featured, call-processing server. Or in plain English, it's like some of the fancy telephone equipment businesses use, but it comes with more features and is easier to set up so people like you and me can use it. You can use Asterisk as a stand-alone system, which is how I show you to use it in this chapter, or as an adjunct to a previously existing PBX or Voice Over IP (VoIP) implementation (something you'll probably begin using within the next few years). You can add telephone applications by using the AGI (Asterisk Gateway Interface). Telephone applications allow you to do all sorts of neat things, such as getting the status of or controlling MisterHouse (software I introduce in Part VI) via your phone or getting weather or other information from the Internet and listening to it over the phone.

Basically, Asterisk is your smart phone system. It has lots of features and supports a lot of hardware. Many of the features you probably wouldn't need in your home unless you have a *really* large family. In fact, there is so much to Asterisk that I can cover only a small fraction of what it's capable of. I'm sorry if that sounds like a cop-out, but it's true. To do Asterisk justice, I would have to devote an entire book to it; instead, I give you the basics in this chapter.

Note: In this chapter, I cover enough to get you started with a single extension, voice mail, and the ability to send and receive calls from your phone company. I make most of the configuration decisions because this makes it easier to go through the material. It's one of those unfortunate instances where you need to have experience with the material before you can properly understand and use it. Of course, to get the experience, you have to learn the material. By using my experience, you can get started using Asterisk quicker because I give you a base to expand on.

Asterisk 101

The telephone industry has lots of strange names, terminology, and TLAs (three-letter acronyms). Unfortunately, this causes a lot of confusion, but I need to use those names and TLAs (sorry).

This list explains the names and acronyms I use the most:

- **ATA (analog telephone adapter):** This device takes a telephone and/or the cable coming from the telephone company and allows you to connect it to an IP network and VoIP server (like Asterisk).
- **FXO (Foreign eXchange Office):** The interface that connects to the telephone company's switches. (You plug the cable from the telephone company into the FXO port. Also known as the *line port.*)
- **FXS (Foreign eXchange Station):** The interface that connects to the telephone. (You plug the cable from the telephone into the FSX port. Also known as the phone port.)
- **PBX (Private Branch eXchange):** A telephone switch located in a business or home.
- **POTS (plain old telephone service):** Anything to do with non-VoIP home telephone service, such as a *POTS line,* which is the telephone service and cable you get from your local telephone company.
- **PSTN (Public Switched Telephone Network):** The telephone company's telephone network.
- **SIP (Session Initiation Protocol):** One of the protocols that is used in VoIP.
- **Telephony:** The technology used in the telephone industry. *Telephony* is a blanket name for all the stuff that goes into sending a call from your phone to someone else's phone.
- **VoIP (Voice Over IP):** A way of making telephone calls over IP networks such as the Internet.

The setup in this chapter consists of the Asterisk version 1.2 software and the Sipura SPA-3000 hardware. The SPA-3000 is an ATA that allows you to hook up your telephone and your hookup to the PSTN to your local VoIP network (that's what you're building). It really acts like two devices in one box. One of the nice features of the SPA-3000 is that when Asterisk isn't working or the power goes out you can still make and receive calls. This is an important feature that will keep you from getting in trouble with your spouse who might not have your affinity towards modern technology (toys!). I've set up a dial plan that should work for any North American telephony setup (the United States and Canada).

Dial plans

A *dial plan* is a set of rules that takes a pattern and instructs the device to do something with it. Normally, the phone company does this work, so your phone simply sends everything you dial directly to the telephone company's switch to be processed. With no PBX, you can't do much with the numbers you dial from your home.

To start with, you need to plan out what number pattern you want to dial. I substitute X for a single digit in my dial plan description. For most of North America, you need something like this:

- 7-digit local dialing (XXX-XXXX)
- 10-digit local dialing (XXX-XXX-XXXX) to handle overlay plans
- 11-digit long-distance dialing (1-XXX-XXX-XXXX)
- Call feature dialing (*XX)
- Emergency (911 or 311) or information dialing (411)
- Operator assistance (0)
- Overseas dialing (01XXX . . .)
- Long distance dial around (1010XXX . . .)
- Internal extensions (XXXX)

Number please?

Allow me to explain how a dial plan works with Asterisk. Normally, you dial a number and the phone company deals with it. It has the dial plan and knows how to properly route your call. If you dial 555-1212, you get your local operator; if you're in an area with 10-digit dialing, you dial something like 732-555-1212. When you add Asterisk, you now need Asterisk to deal with the dial plan. This allows you to have multiple telecom providers and local extensions. You can then set up a dial plan so that a number that starts with 9 (such as 9555-1212) is sent to AT&T local services, a number that starts with 8 (such as 8555-1212) is sent to AT&T Call Vantage, and any number in the 2000 range is a local extension. What Asterisk does is intercept each number you press, and it compares each digit to the dial plan. When it finds a pattern that matches the numbers dialed, it follows the instructions provided in the dial plan. If it finds an exact match, it dials that number immediately. If you dial 911 and you have an exact 911 pattern in your dial plan, it immediately processes the call according to your instructions in the dial plan. If you dial 9112 and use the same patterns, it matches the 911 part and just passes on the 2 to wherever you send the call. This is why you have to select your dial plan carefully.

Some businesses add 4- or 5-digit dialing for calling an extension local to the building. Even though you'll have only one extension, I show you how to use 4-digit dialing for your home setup. Adding new extensions to the `extensions.conf` file will be easy, and they don't have to be phones. Instead, they can be extensions to AGI applications (programs such as weather reports) that I mention earlier. (The `extensions.conf` file is included on this book's CD.)

After deciding what number patterns to dial, you can create a dial plan for the SPA-3000 and Asterisk. The SPA-3000 and Asterisk each have their own format for their dial plan, but the basics of the dial plan are the same. In fact, the SPA-3000 has three (short) dial plans:

- **A dial plan for your telephone on Line 1:** This dial plan decides whether the number is to be sent directly to the PSTN or to Asterisk. The reason for this is that 911 calls should not be handled by Asterisk when the SPA-3000 can send them directly to the PSTN without delay. The rest of the calls are sent to Asterisk.
- **A dial plan for calls being sent to the PSTN:** This one is relatively simple because the decision has already been made to send the call to the PSTN. Either Asterisk has made the decision or the Line 1 dial plan has made the decision (for 911 calling). This dial plan is set up to accept the number and forward it on.
- **A dial plan for a call coming from the PSTN:** All calls from the PSTN are sent directly to Asterisk. The SPA-3000 has many features that you won't be taking advantage of. The dial plan is just one of them, and you'll use only a portion of the power of the dial plan. The reason for this is that Asterisk provides you with much more flexibility, so it's better to allow Asterisk to handle the hard work.

Next, you need to know the rules for creating the dial plan. The SPA-3000 uses a subset of what Asterisk uses, and the two use different layouts but otherwise are quite similar. Here are the general rules that both use:

- Any number dialed not matched by a pattern is ignored. (You get a fast busy warble sound on your phone.)

  A *pattern* is a number, symbol (see the rules below for symbols), group of numbers, and/or a group of symbols that the number you're dialing needs to match.
- You can combine the following rules to make a complex pattern or rule:
  - `0`: Match the exact digit (zero in this case, but it can be any digit 0 through 9, *, or #).
  - `N`: Match a single digit, any digit between 1 through 9.

- `Z`: Match a single digit, any digit between 2 through 9.
- `X`: Match a single digit, any digit between 0 though 9.
- `XX`: Match any 2 digits (but no more than 2 digits).
- `XX.`: Match any 3 or more digits, any digit 0 through 9, *, or #.
- `_`: Means match the following pattern as a number, not as a literal string.
- `[]`: Match anything in the list between the brackets (single-digit match).
- `[2-6]`: Match a single digit in the range of 2 through 6.
- `[2-69]`: Match a single digit in the range of 2 through 6 or 9.

✓ A dash may be used only inside the brackets rule.

✓ Do not use spaces in the rules.

The SPA-3000 doesn't use patterns N and Z. Instead, it uses the list (`[]`) rule. Asterisk keeps its dial plan in a file called `extensions.conf` in the `/etc/asterisk` directory. I list a sample later in this chapter. (See Listing 9-6.) The SPA-3000 uses a Web interface to squeeze in its dial plans. Its dial plans are located under the Line 1 tab (shown later in Figure 9-4) and the PSTN tab (also shown later, in Figure 9-6). The _ (underscore) isn't used by the SPA-3000, and its use is rather confusing in Asterisk. VoIP introduces the concept of IP dialing, where the phone number is not a number but rather the name or IP address of the phone to call. The _ tells Asterisk to treat the pattern that proceeds as a number. With IP dialing, the pattern to be matched would be a string, and it needs to be matched exactly. Because you won't be using IP dialing right now, just make sure that your Asterisk extensions patterns all start with an _, except the `s` extension. I go into further detail about the `s` extension later in the chapter.

Context

Asterisk adds the concept of a context in the dialing plan. A *context* is a group of extensions with a name attached to it to make it easy to identify. This allows you to break down the dial plan into sections. Different contexts can be included in (or pulled into) a context by using the `include =>` command. All the extensions of the included context are now part of the context that pulled them in. This allows you to create commonly used extensions and use them in many places, as shown in Listing 9-1.

Listing 9-1: The from-pstn Context from /etc/asterisk/extensions.conf

```
; -[ Calls from the PSTN ]---------------------------------
          --
[from-pstn]
; Timing list for includes is
;   <context>|<time range>|<days of week>|
;            <days of month>|<months>
  include => daytime|8:00-22:59|*|*|*
  include => nighttime|23:00-7:59|*|*|*
```

The `from-pstn` context handles all the incoming calls from the telephone company. It includes two other contexts, `daytime` and `nighttime`. These include statements that also have the conditional arguments (`|8:00-22:59|*|*|*` and `|23:00-7:59|*|*|*`), which are optional. This tells Asterisk to include this context during the time prescribed. Without the conditional arguments, the context is always included. With conditional arguments, you can create contexts that can handle holiday announcements and other neat features.

Both devices (the SPA-3000 is considered two devices) are registered with Asterisk. The `sip.conf` file in the directory `/etc/asterisk` contains the registration information for both Line 1 (where you have the telephone plugged in) and the PSTN (the cable to the telephone company). Under the PSTN registration information is the statement `context = from-pstn`. This is the `from-pstn` context that this device is assigned to. So when a call is received by Asterisk, the call follows the rules provided by the `from-pstn` context. Other devices can use the same context or a different one.

For your setup, there are two main contexts: `from-pstn` and `from-sip`. The context `from-pstn` handles calls from the PSTN, and the calls follow the rules in the context to determine what to do with them. The context `from-sip` handles calls to and from your extensions. Extensions configured in one context are unknown in another context unless they are included in that context by using the `include =>` statement. Listing 9-2 shows just a portion of `extensions.conf`. (I had to trim it to make it fit.) It's a good example of a dial plan (both contexts and extensions).

Listing 9-2: The stdexten Macro from /etc/asterisk/extensions.conf

```
; Some variables
PHONE1 = SIP/2201
VMAIL1 = 2201

[macro-stdexten]
;   ${ARG1} - Extension (could've used ${MACRO_EXTEN} here)
;   ${ARG2} - Device(s) to ring
;
  exten => s,1,Dial(${ARG2},20)
  exten => s,2,Goto(s-${DIALSTATUS},1)
  exten => s-NOANSWER,1,Voicemail(u${ARG1})
  exten => s-NOANSWER,2,Goto(default,s,1)
```

```
  exten => s-BUSY,1,Voicemail(b${ARG1})
  exten => s-BUSY,2,Goto(default,s,1)

  exten => _s-.,1,Goto(s-NOANSWER,1)
  exten => a,1,VoicemailMain(${ARG1})

; -[ Calls from the PSTN ]---------------------------------
[from-pstn]
  include => daytime|8:00-22:59|*|*|*
  include => nighttime|23:00-7:59|*|*|*

[daytime]
  exten => s,1,Macro(stdexten,${PHONE1},${RINGS})

[nighttime]
  exten => s,1,Voicemail(u${VMAIL1})
```

Assuming that the SPA-3000 is working, configured, and registered properly, here's what happens when a call arrives:

1. When the SPA-3000 receives a call from the PSTN, it forwards the call, according to its dial plan, to Asterisk.
2. Asterisk sees the call as a phone call from the registered device PSTN, looks at the PSTN SIP entry (in the file `sip.conf`), and sees that its context is `from-pstn`.
3. Asterisk then goes to the extensions and finds the context `from-pstn`.

   In that context, I've added conditional `include` statements, based on the time of day:

   a. If the time is between 11:00 p.m. and 7:59 a.m., it jumps to the context `nighttime`, where it then uses the Asterisk command `Voicemail` to send the call to a voice mail box.

   b. If the time is between 8:00 a.m. and 10:59 p.m., it jumps to the context `daytime` where it uses the macro `macro-stdexten`. In the macro, it then dials the extension `${PHONE1}`, a variable assigned to SIP/2201. (I explain macros more at the end of this list.)
4. If the dial command returns, the call was not completed; Asterisk then uses the `Goto` command to jump to the correct status (`s-NOANSWER` for no answer, `s-BUSY` for a busy line, or `s-.` for everything else that doesn't match).

   a. The `s-NOANSWER` and `s-BUSY` both send the call to voice mail, and if the user should return from voice mail, the next command instructs Asterisk to send the call to the default context, where more processing can occur.

   b. The `s-.` extension simply sends the call to the `s-NOANSWER` extension, where the call gets sent to voice mail. The `a` extension is used to catch a key press (you can use that to interrupt the voice mail message) and keep the call in voice mail.

The `macro-stdexten` looks like a context, but it isn't. It's an easy way to use the same series of commands in other extensions. This makes it easier to write (and read) extension rules. To call a macro, drop the `macro-` from the name (`macro-x`, just use the `x`). You can pass arguments to a macro, and they will be assigned the variable `${ARG`*n*`}`, where the *n* stands for the number in the order they were assigned (for example, `${ARG1}` will contain the first argument, `${ARG2}` will contain the second, and so on).

In the preceding example, there are several variable uses, such as `${ARG1}` and `${PHONE1}`. The `${ARG1}` variable is used inside macros, and the values are assigned by what is passed in the macro call. The `${PHONE1}` variable is a user-assigned variable. In the preceding example, I assign the variable `${PHONE1}` to `SIP/2201`. This is the POTS phone connected to the SPA-3000 Line 1.

The `s` extension is unique in that it matches extensions only when there are none. This ability is useful in macros and in calls from the PSTN where there are no extensions being called. It isn't a catch-all extension. A catch-all extension looks like this: `_.` or `_X.`. Both of these extensions are dangerous to use because when a context is read by Asterisk, it's sorted numerically and not by the order in the file. So the `_.` would show up before the `_2201`, and the `_X.` would show up last. Be careful of your use of the two catch-all extensions. Use of the `_X.` extension is preferred over the `_.` extension. Listing 9-3 shows an example of what I mean.

Listing 9-3: An Example of a Poorly Selected Dial Plan

```
[from-junk]
  exten => _2201,1,Macro(stdexten,${PHONE1},${RINGS})
  exten => _.,1,Voicemail(u${VMAIL})
  exten => _X.,1,Voicemail(u${VMAIL})
```

Although the preceding dial plan isn't really useful, it is a good example of what will happen when you try to use the catch-all extensions. If you jump into the Asterisk command line interface (see "Installing and compiling Asterisk," later in the chapter) and type **show dialplan from-junk**, you will see the following output:

```
mozart*CLI> show dialplan from-junk
[ Context 'from-junk' created by 'pbx_config' ]
  '_.' =>    1. Voicemail(u${VMAIL})         [pbx_config]
  '_2201' => 1. Macro(stdexten|${PHONE1})    [pbx_config]
  '_X.' =>   1. Voicemail(u${VMAIL})         [pbx_config]

-= 3 extensions (3 priorities) in 1 contexts. =-
mozart*CLI>
```

So what you have is the first extension matching everything with one or more digits, the second matching 2201 exactly, and the third matching two or more digits. As you can see, the order is different, and if you aren't careful, you could be sending everything to voice mail. (Yes, even the extension 2201 matches the first rule!)

Gathering the Ingredients

Alright, enough of the technical mumbo-jumbo — my head hurts. It's time to configure the hardware and then install the Asterisk software and configure it. The hardware requires a bit of searching on the Internet. I recommend that you use a search engine to find the best price. At the time of this writing, the SPA-3000 was less than $100 (U.S.). The good news is that you can start installing the software without having the hardware. You won't be able to make any calls until you get the hardware, but you will be able to try out the commands.

This list describes what you need, excluding PC requirements (which I discuss in the "How big a PC for Asterisk?" sidebar):

- **Software (found on the CD)**
  - My replacement configuration files
  - Asterisk software, version 1.2 (`asterisk-1.2.0.tar.gz`)
- **Hardware**
  - Sipura SPA-3000 ATA
  - A PC running Linux with an Ethernet interface card
  - A regular, push-button phone
  - PSTN service (the telephone line from your telephone company)
  - Two R-J11 telephone cables
  - An Ethernet cable
  - A home network
- **Optional**
  - A caller ID unit and an extra RJ-11 telephone cable
  - An answering machine and an extra RJ-11 telephone cable

Stop the presses!

The new Linksys ATA — the SPA-3102 — became available after I wrote this chapter, and by the time you read this, the SPA-3000 may not be available. Normally that would be bad news, but I have the new SPA-3102, and it seems to behave pretty much the same as the SPA-3000 — at least for the needs of this chapter. To find out if there are any changes related to this, visit my Web site (`www.linuxha.com/`) where I'll post further updates as needed.

You must have a working network to start with. Usually, you set this up at install time. The caller ID and the answering machine are optional. My wife wanted the messages left on the answering machine, so I configured Asterisk to go to voice mail on four rings and my answering machine for three rings. Of course, call after 11 p.m. and you'll get Asterisk and not the answering machine. My wife is happy with that.

The various files on the CD contain the configuration changes I've made to copies of the original Asterisk files. Don't worry about the originals; they'll be renamed with a `.bak` extension during the install, so you can compare my changes to the original. The files are in the `/etc/asterisk` directory.

Fitting the hardware pieces together

Figure 9-1 shows the hardware put together properly (remember that the answering machine is optional). Installation instructions are also included with the SPA-3000; both instructions will work properly. The SPA-3000, you'll need to purchase. I suggest using your favorite search engine to find a good price. The telephone must be a push-button phone and not an old-fashioned rotary phone. The SPA-3000 won't recognize the pulses from a rotary phone. (It's strange how many people still say you're going to dial a phone number when you no longer have a dial on the phone.) The home network can be as simple as a Linux server with an Ethernet network interface card connected via an x-over (cross over) Ethernet cable to the Ethernet port of the SPA-3000. Figure 9-1 shows an Ethernet hub or switch being used, which is the preferred method.

The SPA-3000 is a gateway device that connects the PSTN, your phone, and VoIP services (in this case, Asterisk). The SPA-3000's job is to convert a call from your telephone or the PSTN to IP and back. This device does a lot of the

hard work for you. Technically, the SPA-3000 doesn't need Asterisk to work, but it's a great interface to use with Asterisk in a home environment. The SPA-3000 has four ports on it: an Ethernet port, an FXO port (telephone company line port), an FXS port (the phone port), and a power port.

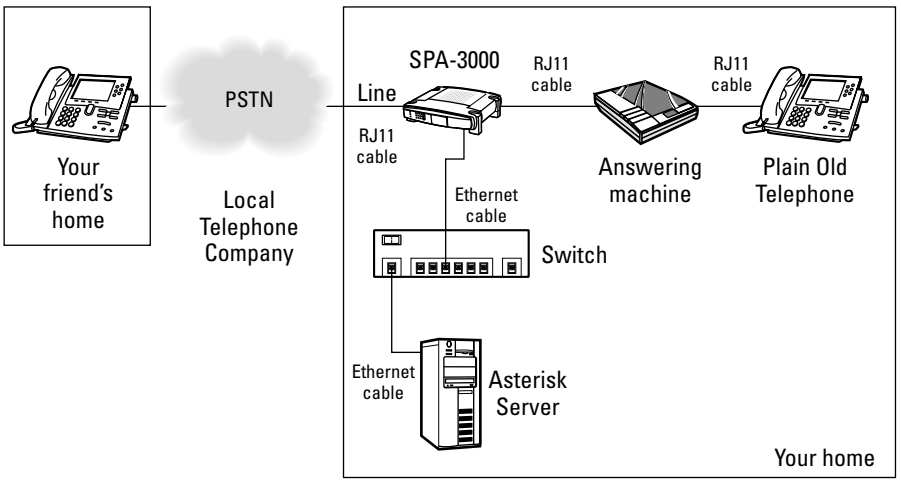

Figure 9-1: An assembled system.

The SPA-3000 is an interesting ATA. It has several useful features, the first being that on the loss of power the SPA-3000 will pass PSTN calls to Line 1, and Line 1 calls to the PSTN. This means that you can still make and receive outside calls even when the power is out and your Asterisk server is down. You won't be able to call any of your extensions, but you will be able to call the outside world. In addition to handling the loss of power (and/or the loss of connectivity to your Asterisk server) properly, the SPA-3000 can direct 911 calls directly to the PSTN without going through your Asterisk server. It is very important not to interfere with the 911 services because seconds count in an emergency.

Configuring the SPA-3000

Before you configure your SPA-3000, you need a couple pieces of information. The first is the IP address of your Linux PC (the one where Asterisk will be running). This is the address to enter as the SPA-3000's gateway address. Normally the gateway address is the address of the device that allows you access to the Internet. But that's not what needs to be configured here as the gateway. The next thing you need is the next available IP address on your

network. Remember, no two devices can have the same IP address. When you have this information, you can then begin configuring the SPA-3000 with the following steps:

1. **Pick up the phone and dial four asterisks (****).**

   Ignore the SIT tones. You should be greeted by someone saying, "Sipura configuration menu." If not, try again.

2. **Configure the SPA-3000 for a static IP address:**

   a. *Dial 111# and then dial the IP address by using the asterisk key (*) for periods followed by the pound key (#). For example, 192*168*1*10#.*

   b. *Dial 121# and then dial the IP mask. For example, 255*255*255*0#.*

3. **Check your work:**

   a. *Dial 110# to check your IP address.*

   b. *Dial 120# to check your IP mask.*

   If you configured the SPA-3000 incorrectly, go to Step 2 and do it again.

Now that you've got the IP address out of the way, you can use a browser to configure the rest of the settings. Fire up your favorite browser and make sure JavaScript is turned on. I've tried Opera, Konqueror, and Firefox, and they all work fine. Now go to this URL:

```
http://192.168.1.10/admin/advanced
```

How big a PC for Asterisk?

Many people ask, "How fast a processor, how much RAM, and how big a hard drive does my PC need to run Asterisk?" And the quick answer is: It depends. Some folks are running Asterisk on a Linksys WRT54GS router that has only 8MB of flash storage, 16MB of flash memory, and a 200-MHz processor. That's right — no hard drive. They're using external SIP devices like the SPA-3000 that I describe in the "Fitting the hardware pieces together" section. This setup is serving only a few IP phones and/or ATAs. Other setups require more processor speed and can require more hard drive space and RAM to run all the applications that the user wants. For the setup in this chapter, you need at least a 500 MHz processor, at least 64MB of RAM, and at least a 5GB hard drive. You might be able to run a lesser machine, but I don't go into those details (or how to run and install Asterisk on the WRT54GS). If I'd gone with another starter kit using an X100P PCI board and an IP phone, you would have needed a much more powerful processor. I've run into problems with the X100P board with a 2-GHz processor when the system became busy with other processes, such as MisterHouse. (See Chapter 15 for more on MisterHouse.)

You need to replace the *192.168.1.10* with the IP address you used to configure in the SPA-3000 in Step 2a. This brings up the main Web page with the Info tab selected. This page contains various information such as the IP address, mask, Line 1 status, and PSTN Line Status. You can move to the other tabs by clicking each tab name toward the top of the advanced admin page. The SIP, Provisioning, Regional, User 1, and PSTN User tabs' defaults are fine for your needs, so you don't need to make any changes there. You will need to make changes on the System, Line 1, and PSTN Line tabs; I show you those changes next.

The System tab

The first tab you need to visit is the System tab, shown in Figure 9-2. Just fill in the Admin Passwd and User Password fields. You can leave these blank if you'd like, but it isn't very secure — anyone who has access to a Web browser on your network will be able to make changes to your configuration. Almost all the other fields will be blank, and that's fine. To save the changes, click the Submit All Changes button at the very bottom of the Web page (although you might need to scroll down to see it).

Figure 9-2: The System tab.

The Line 1 tab

On the Line 1 tab, you need to make changes so your telephone can be extension 2201 and talk to the Asterisk server. You must make changes to four sections. At the top of the Line 1 tab, make sure that Line Enable is set to Yes. Then scroll down to the Proxy and Registration section. (See Figure 9-3.) Here, you configure Line 1 to talk to your Asterisk server. Now take care of the following settings:

- **Proxy:** <*Asterisk IP Address*>

  Replace <*Asterisk IP Address*> with the IP address of your Asterisk server (your Linux IP address).
- **Use Outbound Proxy:** No
- **Use OB Proxy in Dialog:** No
- **Register:** Yes
- **Make Call Without Reg:** No
- **Register Expires:** 30
- **Ans Call without Registration:** Yes

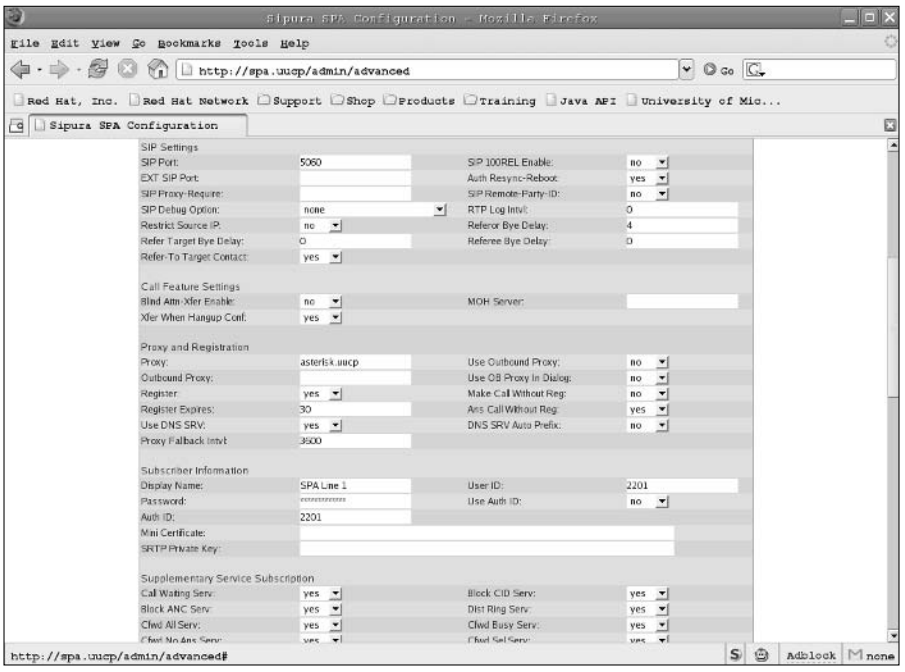

Figure 9-3: The Line 1 tab.

You also need to make changes to the Subscriber Information section:

- **Display Name:** SPA Line 1
- **User ID:** 2201
- **Password:** 2201

You can change the Display Name field to what ever you'd like. It's your internal caller ID. You can't change what the telephone company sends because that's in the company's system. Now proceed to the Audio Configuration section and make these changes:

- **Preferred Codec:** G.711u
- **Silence Supp Enable:** No
- **Use Pref Codec Only:** Yes

Now scroll down to the VoIP Fallback to PSTN section and set Auto PSTN Fallback to Yes. This allows you to make phone calls to the PSTN if your Asterisk server gets hosed. You can change the Display Name field (your internal caller ID) to what ever you'd like. And the last part for the Line 1 tab is the dial plan. (See Figure 9-4.)

```
([2-79]11S0<:@gw0>|*xx|0|00|2xxx|[2-9]xxxxxx|1xxxxxxxxxxS0|xxxxxxxxxxx.)
```

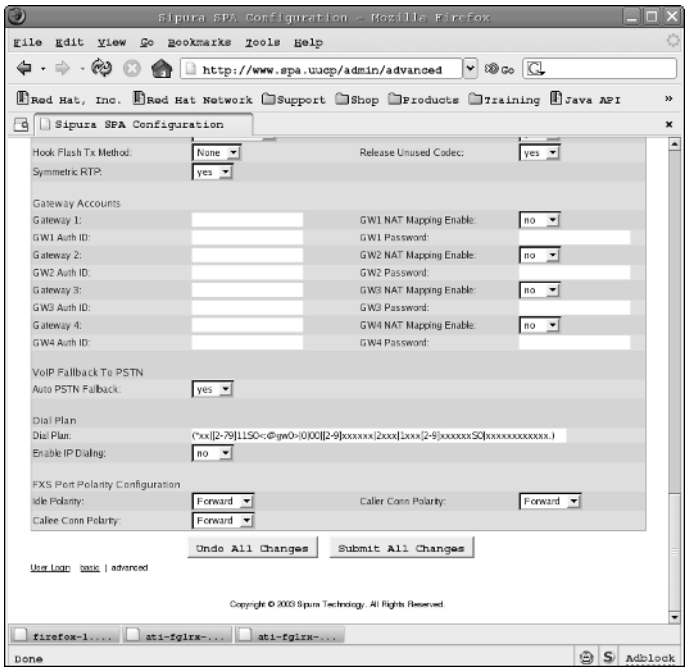

Figure 9-4: Line 1 tab, Dial Plan section.

To save the changes, click the Submit All Changes button at the very bottom of the Web page.

Although the rest of the changes are difficult to explain (or would take too long), I explain this dial plan. First, I need to tell you about a couple rules for the SPA-3000's dial plan. The dial plan (everything in parentheses) is broken into sections by the pipe symbol (|). Each section describes an expression that matches a specific number pattern (a telephone number) and describes what to do with it. Here's an explanation of the rules for the SPA-3000's dial plan:

- `()`: The entire dial plan.
- `|`: An expression separator.
- `1`: The SPA-3000 matches the exact keys. (Valid keys are 1 through 9, 0, *, and #.)
- `x`: The SPA-3000 matches any number (0 thru 9).
- `[]`: The SPA-3000 matches a single digit from a list of digits.
- `[2-9]`: The SPA-3000 matches any single digit in a range (always used inside brackets).
- `S0`: The SPA-3000 matches the previous expression immediately.
- `<a:b>`: The SPA-3000 substitutes the dialed expression *a* with expression *b*.
- `<:@gw0>`: The SPA-3000 sends the matching string to the gateway listed.
- `.`: The SPA-3000 matches one or more keys no matter what they are.

I've used the substitution command to insert the expression `@gw0` (which is the PSTN gateway). This tells the SPA-3000 to send the call to the PSTN port. Also, I've used the brackets, `[2-79]`, to include a list that means match any single number between 2 and 7 or 9. Here's exactly what the dial plan does:

- `[2-79]11S0<:@gw0>`: The SPA-3000 sends 211, 311, 411, 511, 611, 711, and 911 directly to the PSTN without delay.
- `*xx`: The SPA-3000 sends numbers that start with * and two digits — for example, *69 — to the Asterisk server.
- `0`: The SPA-3000 sends 0 directly to the Asterisk server.
- `00`: The SPA-3000 sends 00 directly to the Asterisk server.
- `2xxx`: The SPA-3000 sends any 4-digit number that starts with 2 directly to the Asterisk server.
- `[2-9]xxxxxx`: The SPA-3000 sends any 7-digit number, starting with any number between 2 and 9 inclusive — for example, 5551212 — directly to the Asterisk server.

- `1xxx[2-9]xxxxxxS0`: The SPA-3000 sends any 11-digit number, starting with 1 and whose 5th digit is between 2 and 9 inclusive — for example, 17325551212 — immediately to the Asterisk server.
- `xxxxxxxxxxxx.`: The SPA-3000 sends any 13-digit, or longer, number to the Asterisk server.

Okay, that was difficult! Dial plans aren't easy, and they take a lot of thought to make them work right. So why didn't I just use `x.`? Well, the SPA-3000 will send the number pretty soon after matching the 2 (or more) digit sequence. When the sequence reaches Asterisk, Asterisk will try to match a dial plan entry for the exact number of digits it gets sent. This might not be your exact intention.

Sipura has to be given a lot of credit for their product. They've managed to squeeze a lot of power into a Web interface. You can use the SPA-3000 anywhere in the world by adjusting the various parameters in the Web interface. Unfortunately, all that power means complexity. Sipura has a wonderful 87-page manual on its site, and the manual really does explain everything. Unfortunately, you have to understand all the terms and read the entire manual.

The PSTN Line tab

The PSTN Line tab (see Figure 9-5) configuration is very similar to the Line 1 tab configuration. So you start at the very top and enable the line by setting the Line Enable option to Yes.

Figure 9-5: The PSTN Line tab.

Next, jump down to the SIP Settings section. You need to change the port because Line 1 is already using 5060. Set the SIP port as 5061.

Scroll down to the Proxy and Registration section and make the following changes:

- **Proxy:** <*Asterisk IP Address*>

  Replace <*Asterisk IP Address*> with the IP address of your Asterisk server.
- **Use Outbound Proxy:** No
- **Use OB proxy in Dialog:** Yes
- **Register:** Yes
- **Registration Expires:** 30

Next up, scroll down to the Subscriber Information section and make these changes:

- **Display Name:** PSTN
- **User ID:** pstn
- **Password:** pstn
- **User Auth ID:** No

Move down to the Audio Configuration section and enter **G711u** for the Preferred Codec.

Now the dial plan rollercoaster starts up again. (See Figure 9-6.) This time things are a little easier. So scroll down to the Dial Plans section and make the following changes:

- **Dial Plan 1:** (S0<:s@<*Asterisk IP Address*>>)

  Replace the <*Asterisk IP Address*> with the IP address of your Asterisk server.
- **Dial Plan 8:** ([2-79]11S0|*xx|0|00|2xxx|[2-9]xxxxxx|1xxxxxxxxxS0|xxxxxxxxxxx.)

I explain the dial plans, but not until after a few more changes. So scroll down to VoIP-To-PSTN Gateway Setup and make these changes:

- **VoIP-To-PSTN Gateway Enable:** Yes
- **One Stage Dialing:** Yes
- **Line 1 VoIP Caller DP:** 8
- **VoIP Caller Default DP:** 8

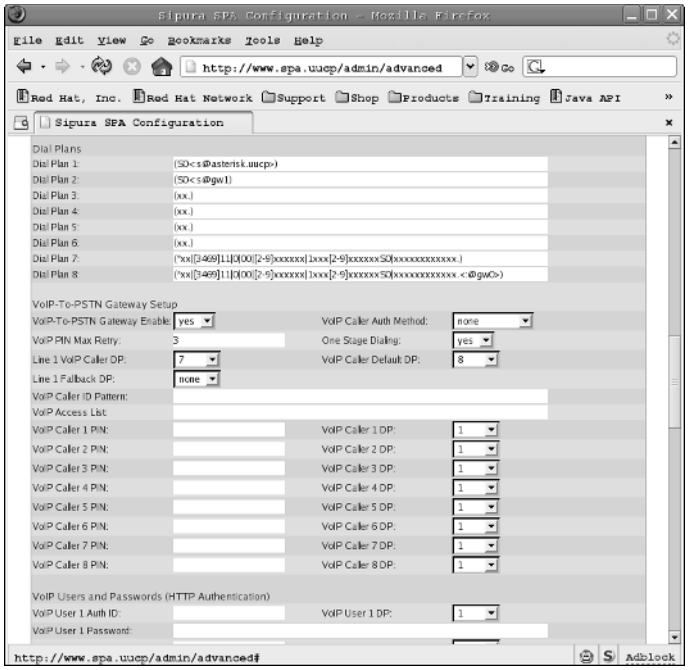

Figure 9-6: The PSTN Line Dial Plans.

In the PSTN-To-VoIP Gateway Setup section, make these changes:

- **PSTN-To-VoIP Gateway Enable:** Yes
- **PSTN Ring Thru Line 1:** No
- **PSTN CID For VoIP CID:** Yes
- **PSTN Caller Default DP:** 1

To save the changes, click the Submit All Changes button at the very bottom of the Web page.

Okay, now I can explain the dial plans for the PSTN Line. The complicated dial plan (#8) is the plan being used by the VoIP-To-PSTN plan. This one handles the numbers coming from Asterisk to the PSTN and from Line 1 to the PSTN. It's the same as the Line 1 dial plan, so see the preceding section for the explanation. The odd dial plan is #1, which is used for calls from the PSTN to VoIP (to Asterisk). What the dial plan says to the SPA-3000 is to immediately send any call directly to your Asterisk server (in this case, <*Asterisk IP Address*>). This dial plan uses IP addressing instead of an extension or a gateway (`gw1`, for example, which you can use for other VoIP services).

How the other half lives: The software

The Asterisk installation instructions are pretty scary-looking to someone who hasn't compiled a large application under Linux. It's intimidating to watch one line after another go scrolling off the screen, not knowing what they mean. (See Figure 9-7.) If you've done a full install of your Linux distribution, you should have no problems with loading the Asterisk package. If you haven't, you need to make sure that you have C compiler and development kit.

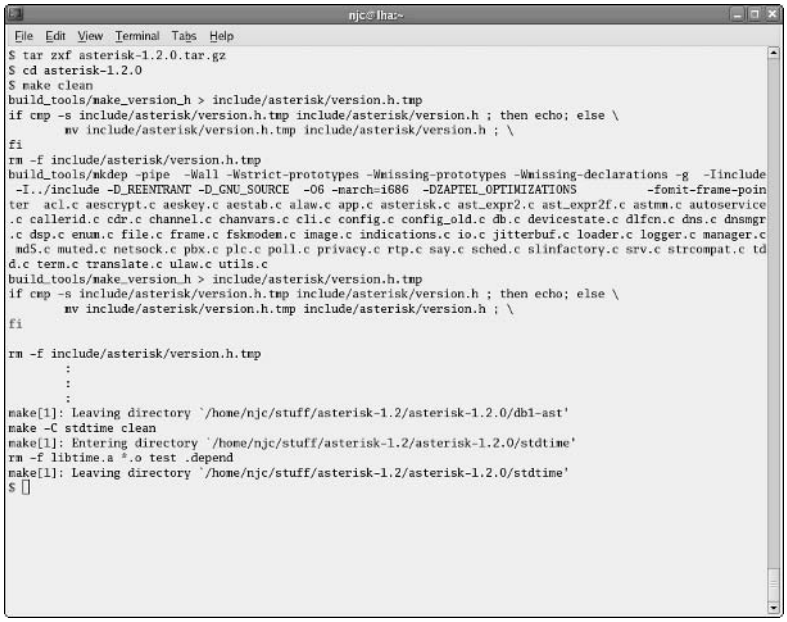

Figure 9-7: The Asterisk compile screen.

Installing and compiling Asterisk

To compile Asterisk, open a shell or terminal and follow these steps:

1. **Type** su - **and press Enter, after which you will be greeted by a password prompt. Enter the password for `root`.**
2. **Type** cd **and press Enter. You will be in `root`'s home directory.**
3. **Type** cp /media/disk/chapter09/asterisk.tar.gz . **and press Enter.**

   This copies the file to `root`'s home directory.
4. **Type** tar zxf asterisk.tar.gz **and press Enter to extract the source from the `tar` archive file.**
5. **Type** cd asterisk-1.2.0 **and press Enter to change to the directory where the Asterisk source code is.**

6. **Type** make clean **and press Enter to make sure you start with a clean slate and keep odd errors from cropping up.**

   You can ignore any warnings that appear on the screen.

7. **Type** make linux26 **and press Enter to compile the Asterisk program.**

   Lots of messages scroll off the screen. As long as there are no errors, you can safely ignore the messages. See Figure 9-8 for a sample of what shows up on the screen. (I trimmed it because it's several pages long.)

8. **Type** make install **and press Enter. If there were no errors on the previous command, it installs all the asterisk files in the correct places.**

9. **Type** make samples **and press Enter to create the sample configuration files.**

   These are stored in `/etc/asterisk`.

You must be `root` to properly run Asterisk.

After the compile and install has finished (warnings can be ignored), you can verify that Asterisk is really working:

1. **Type** asterisk -vvvc **and press Enter at a command prompt.**

   This is a way to test that Asterisk compiled and installed properly. You should see a bunch of messages fly by and then a prompt:

   ```
   hostname*CLI>
   ```

   Don't worry if you don't understand them yet.

2. **Type** help **and press Enter to get a help message (that will scroll off the screen).**

3. **To exit Asterisk's command line prompt, simply type** quit **and press Enter at the prompt.**

Here are a few useful commands:

- To properly start Asterisk, you can either let it start when you reboot (via an `init.d` script that I've set up for you) or by simply typing **asterisk** and pressing Enter. (You'll need to be `root`.) Asterisk starts up and runs in the background. If all went well, you see nothing but your prompt return.
- To get to the Asterisk command line, type **asterisk -r** and press Enter. (Remember this command; it's a popular one.)
- To stop Asterisk, type **stop now** and press Enter at the Asterisk command line. This immediately stops Asterisk and drops any calls.
- To simply exit the Asterisk command line interface and still allow Asterisk to handle calls, type **quit** and press Enter. For now, I recommend the `stop now` command. You can restart Asterisk later by typing **asterisk** and pressing Enter.

Making (necessary) changes to Asterisk

To simplify the installation of Asterisk, I've taken the liberty of providing the bulk of the changes in the file `spa-asterisk.rpm`. Simply type **sh /media/disk /chapter09/ast_install.sh** and press Enter to install my configuration files.

In the `/etc/asterisk` directory, you can find the original configuration files backed up (as `.bak`) and the new ones installed. This doesn't mean you're going to get away without editing files; it just means you won't have to type in hundreds of lines of configuration information. This is all well and good in that you should be able to start Asterisk and begin using it right away (yeah!), as long as you're in the United States or Canada. For the rest of the world, you need to do some editing to customize it to your needs. For further details about editing these files, visit `www.asteriskdocs.org`.

Speaking of editing, it's now time to change a few things to suit your telephony needs. So open up your favorite Linux editor and edit the file `/etc/asterisk/extensions.conf`. Don't use a Windows editor — it will mess up the line endings, causing you no end of mysterious problems. Search for the following line:

```
AREACODE = 732
```

Change the `732` to match your area code (the first three digits in your telephone number after the 1; for example, the 732 in 1-732-555-1212). Now, you might be fortunate and not have to dial all ten digits to call a local number. If you're one of those lucky few (I'm not), you need to comment out the line of code that deals with adding the area code and uncomment the line before it. First, search for `${AREACODE}`. The code should look like what's shown in Listing 9-4.

Listing 9-4: Part of the to-pstn Context from /etc/asterisk/extensions.conf

```
; Remove the comment from the next line and comment
; out the line after if you can still use 7 digit
; dialing.
; exten => _ZXXXXXX,1,Dial(SIP/pstn/${EXTEN})
  exten => _ZXXXXXX,1,Dial(SIP/pstn/1${AREACODE}${EXTEN})
  exten => _ZXXXXXX,2,Goto(s-${DIALSTATUS},1)
;
  exten => _ZXXXXXX-NOANSWER,1,Voicemail(u${VMAIL})
  exten => _ZXXXXXX-NOANSWER,2,Goto(default,s,1)
;
  exten => _ZXXXXXX-BUSY,1,Voicemail(b${VMAIL})
  exten => _ZXXXXXX-BUSY,2,Goto(default,s,1)
```

To comment out a line, simply insert a semicolon (`;`) at the beginning of the line. To uncomment a line, remove the semicolon from the start of a line. When you uncomment a line, I suggest replacing the semicolon with a space to keep the lines tidy. This makes the code easier to read.

Making (optional) changes to Asterisk

This section is optional. You may skip to the next section. You don't need to make any further changes. I describe how you add another SIP extension.

I've already added the 2201 extension to the extensions file, configured voice mail so 2201 has a mail box, and configured the SIP file so that the SPA-3000's Line 1 is properly recognized, but I explain it here so you will understand what is needed to add a device and extension to Asterisk. That way, you'll be able to add more devices in the future. Now that you have the SPA-3000 properly configured, you'll use that information to add it to the SIP configuration file. Open your editor, edit the file `/etc/asterisk/sip.conf`, and scroll down until you see the section shown in Listing 9-5. (It isn't a very large file.) You probably won't need to make any changes, but if you do, this is a good place to catch mistakes.

Listing 9-5: SIP Entry 2201 from /etc/asterisk/sip.conf

```
[2201]
  username      = 2201
  secret        = 2201
  type          = friend
  host          = dynamic
  port          = 5060
  context       = from-internal
  callerid      = "SPA3000" <2201>
  mailbox       = 2201
  nat           = never
  dtmfmode      = rfc2833
  canreinvite   = yes
  qualify       = yes
  insecure      = yes
  disallow      = all    ; need disallow before can allow
  allow         = ulaw
```

You might want to open a browser to the SPA-3000 and go to the Line 1 tab. (Refer to Figure 9-3.) I jump around a bit so that you can be done with the browser quickly, so pardon my choice of order:

1. **In your browser, scroll down to the Subscriber Information section where you can find the User ID and Password fields.**

   That subscriber information translates into the SIP configuration's username and secret password, respectively. In your editor, you should see a `type` line. This should be set to `friend` for devices such as the SPA-3000 (or any IP phone). This allows you to send calls to Line 1 and receive calls from Line 1 without more configuration in the `sip.conf` file.

2. **In your browser, under the SIP Settings section, (towards the top of the tab), make sure the SIP port is set to 5060.**

This should match the `port` setting under `2001` in the `sip.conf` file. This is the default for most devices, and you didn't change this for the Line 1 settings. (You did change it in the PSTN Line settings.)

You're now done with the browser. From here on, you need only look at the editing and the information in Listing 9-5.

3. **The next important line to check for the entry 2001 (in `sip.conf`) is the `context`.**

   The context `from-internal` is the context you will put your extension information under. It's in the `/etc/asterisk/extensions.conf` file (more on that in a moment). The `context` tells Asterisk where to look for this extensions dial plan.

4. **Next, change the `callerid`, which is used internally only by Asterisk.**

   You can set it up to your liking, but keep the format used above. The mailbox setting should match the username (2201 in this case).

5. **If you have IP phones (hence multiple entries in the `sip.conf` file), you may elect to have those devices use one voice mail box instead of many.**

   For example, each extension can have its mailbox setting like so: `mailbox = 2201`. This sets the mailbox for that extension to just 2201.

6. **Set the `nat` entry to `never` when you use an IP phone on your local network.**

   If you need to set it to `yes`, expect to have a lot of problems and to do a great deal of custom work trying to get it to work.

7. **The `dtmfmode` is a device-specific setting, so you need to match the device's settings (in the device's configuration).**

   For the SPA-3000, it's `rfc2833`.

8. **The `canreinvite` is important. First the device must support this option (which the SPA-3000 does).**

   Setting it to `yes` allows the IP phones to start up a conversation by first sending information to Asterisk. When both phones have been contacted, they can talk to each other without Asterisk playing the part of the middleman. This lowers the workload on the Asterisk server.

9. **(Optional) Set the next two parameters, `qualify` and `insecure`, to `yes`.**

   I do this step because it allows me to see more information when I'm at the Asterisk command line. I recommend setting them to `yes`.

10. **Note the last two lines of Listing 9-5; these lines deal with the *codecs* (the code used to translate your voice from sound to data) supported by the devices.**

The first line is recommended to always be first (`disallow = all`). Then list the allowed codecs. I chose to allow only `ulaw`, which the SPA-3000 supports as G711 (Line 1 tab, Audio section). Although the SPA-3000 can support many different codecs, there are limitations to each. By using the G711 on the SPA-3000, I've kept the Asterisk server from having to do translation from one codec to another (a real CPU killer).

Now that you have an SIP entry, you need an extension. Under the context `from-sip`, you would add the extension, and it would look like Listing 9-6.

Listing 9-6: Extension 2201 from /etc/asterisk/extensions.conf

```
; 2201
  exten => 2201,1,Dial(sip/2201,20,)
  exten => 2201,2,voicemail(u2201)
  exten => 2201,3,Hangup
  exten => 2201,102,voicemail(b2201)
  exten => 2201,103,hangup
```

The extension in Listing 9-6 is 2201. The dial plan works exactly the same way as the previous dial plan in the `from-pstn` context in Listing 9-4. The difference is that, instead of using the `s` extension, you actually have an actual extension associated with the dial plan. The interesting parts of this dial plan are the `voicemail` commands. The first `voicemail` is when no one answers the phone. The `u` in front of the extension tells the `voicemail` command to play the unavailable message. A `b` tells it to play the busy message.

Voice mail is the easiest part of Asterisk to configure. Simply open the file `/etc/asterisk/voicemail.conf` with your editor and add a line, at the end of the file, that looks like Listing 9-7.

Listing 9-7: A Voice Mail Entry in the /etc/asterisk/voicemail.conf File

```
; Line 1 on the SPA-3000
2201 => 1234,Neil Cherry,asterisk@linuxha.uucp,attach=yes
```

Adjust accordingly with your name, password (the *1234*), and correct e-mail address to send a sound file to. The first number is the extension that the voice mail box is for. The second number is the password that you need to enter from a phone to access this mail box. Next is the user ID (a name), and then an e-mail address to send a notification that a message has been received. The last part is an option to attach a copy the message to the e-mail. If you don't want to send an e-mail message when you receive a voice mail message, set `attach` to `no`. The ability to send e-mail and attachments is dependent on a working e-mail system.

Making a Smart Call

When everything is set up and ready to use, you can make some calls. Earlier in the chapter, at the end of the "Installing and compiling Asterisk" section, I had you stop Asterisk from running. You need to restart it because you've made changes to Asterisk. If you're already logged in as `root`, skip to Step 2; otherwise, begin with Step 1 as usual:

1. **Type** su - **and press Enter, and when you're greeted by a prompt, enter the correct password for `root`.**
2. **Type** asterisk **and press Enter.**

   You're greeted by your `root` prompt. It looks like nothing happened, which is okay. Asterisk is running in the background.
3. **Type** asterisk -r **and press Enter.**

   You're greeted by *`Hostname`*`*CLI>`, where *`Hostname`* will be the hostname of your machine.
4. **At the Asterisk prompt, type** sip show users **and press Enter.**

   You see output that looks similar to this:

```
mozart*CLI> sip show users
Username   Secret  Accountcode  Def.Context      ACL  NAT
2201       2201    2201         from-internal    No   No
pstn       pstn    pstn         from-sip-pstn    No   No
```

   This shows you that you have 2 users (extensions).
5. **Type** sip show peers **and press Enter.**

```
mozart*CLI> sip show peers
Name/username Host           Dyn Nat ACL Mask             Port  Status
2201/2201     192.168.24.192 D           255.255.255.255  5060  OK (4 ms)
pstn/pstn     192.168.24.197 D           255.255.255.255  5061  OK (16 ms)
```

   This shows you the status of your two SIP peers. In this case, Asterisk can talk to both and it shows as okay.
6. **Type** show dialplan from-sip **and press Enter.**

```
mozart*CLI> show dialplan from-sip
[ Context 'from-sip' created by 'pbx_config' ]
  '2201' =>         1. Macro(stdexten|SIP/2201)
                    [pbx_config]
  Include =>        'to-pstn'
                    [pbx_config]
mozart*CLI>
-= 1 extensions (1 priorities) in 1 contexts. =-
mozart*CLI>
```

This shows your dialplan for the context from-sip.

7. **Type** show dialplan to-pstn **and press Enter.**

```
mozart*CLI> show dialplan to-pstn
[ Context 'to-pstn' created by 'pbx_config' ]
  '_0' =>           1. Macro(dial-pstn|SIP/${EXTEN}@${PSTN})
            [pbx_config]
  '_00' =>          1. Macro(dial-pstn|SIP/${EXTEN}@${PSTN})
            [pbx_config]
  '_01.' =>         1. Macro(dial-pstn|SIP/${EXTEN}@${PSTN})
            [pbx_config]
  '_1N.' =>         1. Macro(dial-pstn|SIP/${EXTEN}@${PSTN})
            [pbx_config]
  '_2201' =>        1. Macro(dial-extension|${PHONE1})
            [pbx_config]
  '_[2-79]11' =>    1. Macro(dial-pstn|SIP/${EXTEN}@${PSTN})
            [pbx_config]
mozart*CLI>
-= 6 extensions (6 priorities) in 1 contexts. =-
mozart*CLI>
```

The results of Steps 6 and 7 show you what your dial plan looks like inside Asterisk.

Before you start dialing, you have one more thing to check:

1. **Open up your browser and go to `http://192.168.1.10/admin/advanced`.**

   Remember that you need to replace the *`192.168.1.10`* with the IP address you used to configure in the SPA-3000 earlier in this chapter.

   This brings up the main Web page with the Info tab selected. (Refer to Figure 9-2.) Check the status' of Line 1 and the PSTN line. If both are registered, you're ready to dial; if they aren't, you must check the SPA-3000's configuration and Asterisk's configuration files. But first things first — you have to check the status.

2. **Scroll down to the Line 1 Status section and make sure that the Registration State is Registered (which means that Line 1 is ready).**

   If Line 1 is having problems, check the SPA-3000's Line 1 tab. Also check the `sip.conf` file in `/etc/asterisk`. For a problem with Line 1, check the lines under the [2201] heading.

3. **Scroll down to the PSTN Line Status section, and again, make sure that the Registration State is Registered.**

   If the PSTN Line is having problems, check the SPA-3000's PSTN Line tab. Also check the `sip.conf` file in `/etc/asterisk`. For a problem with PSTN Line, check the lines under the [PSTN] heading. If both show Registered, you're ready to dial.

Now on to the dialing!

1. **Try accessing voice mail by dialing 81.**

   You might need to wait a second or two before it responds. To make it dial immediately, dial the pound sign (#), which might speed things up a bit.

   You should be greeted by a request for your password.

2. **Dial 1234, and you'll be greeted by the voice menu.**

3. **Follow the directions.**

   You should have mail from me.

If you're having problems with dialing, first determine what kind of number you're dialing (as I describe earlier in the chapter). Match that to one of the extensions in your dial plan, like that in Listing 9-6 (the file `extensions.conf` in the directory `/etc/asterisk`). You might need to see whether it correctly matches your dial plan. You should now be able to make a call to the outside world, and the outside world should be able to call you. If you have any questions or problems, check out my Web forums at `www.linuxha.com`; I should be able to help. If you're looking for suggestions, I can provide you with plenty of pointers. On the forums, you can also discuss what features can be added and what other neat things you can do with Asterisk.