Asterisk: The Open Source PBX Solution

Adam Olson

Systems and network administrators typically deal with data and functionality such as email communications, Web and database applications, network infrastructure, and network performance. Analog and voice may be part of the picture, but these tasks are typically handled by concentrated DSP cards that, in the end, hand the call off to internal data networks in the form of dial-up users or the like.

Today, more systems and network administrators are managing voice communications traffic in addition to their other duties. These tasks include managing private branch exchange (PBX) extensions, Session Initiation Protocol (SIP) phones, Interactive Voice Response (IVR) prompts, voicemail-to-email functionality, call forwarding, etc. Many of us have been waiting for a production grade solution to handle voice traffic inside the corporate network that offers an alternative to hardwired jacks, fixed desk locations, and remote staff members.

This is where Asterisk comes in, and it really does the job well. Asterisk is an open source software package that will turn almost any Linux, BSD, or Mac OSX box into a full-featured PBX. Public Switched Telephone Network (PSTN) calls are terminated via quad RJ-11 or PRI PCI cards, allowing administrators to choose between Voice over IP (VoIP) providers or maintaining their existing PSTN provider contracts.

## Asterisk

Mark Spencer created Asterisk, and he is also known for creating a few other packages, such as the popular chat client GAIM. Asterisk is capable of using only Internet Telephony Service Providers (ITSPs), also known as VoIP providers, in addition to the more traditional PSTN for handling incoming and outgoing calls.

Because Asterisk is an open source solution, a number of additional packages and feature add-ons are available online. These include a complete set of high-quality voice recordings for IVR prompts, a Web-based management interface, backend processing engines, etc.

## Requirements

You don't need any additional hardware if you are looking solely for a solution with SIP phones and an ITSP provider for inbound and outbound calling and you have at least one SIP phone and a server. If you want to utilize your existing PRIs or analog phone circuits, visit:

```
http://www.digium.com
```

Check out the PCI PRI or 4 Port FXO/FXS cards to meet your exact needs.

The examples in this article are based on Red Hat Fedora Core and Asterisk version 1.2.7.1 built from source. Make sure to download the latest version from:

```
http://www.asterisk.org/download
```

when you attempt the build process. Please consult the Asterisk Web site for more details regarding platform support if you are using a different operating system. I will be using an ITSP for outbound phone service and will be terminating DIDs over a local PRI, using a PCI card supplied by Digium.

Regarding phones, X-Lite makes a nice, free, soft SIP phone that can be downloaded from their site. This is a good way to get started without purchasing a hard SIP phone right away. You can also order network adapters online that will allow your existing phones to function. However, using hard SIP phones is nice when possible, because they are typically Web managed and provide administrators much appreciated flexibility when supporting them remotely.

## The Build Process

Asterisk compiles quickly and easily. You should also know that there is a version of Asterisk, called Asterisk@Home, which is basically a preconfigured version of Asterisk with the Asterisk Management Portal (AMP) included. After putting the ISO image on a CD and booting it, your hardware will be rebuilt as an Asterisk@Home PBX. Again, this article focuses on building Asterisk from source, which allows you to keep preexisting services running.

Download the most updated version of the packages listed below from:

```
http://www.asterisk.org/download
```

Include the following:

```
asterisk-1.2.7.1.tar.gz
asterisk-perl-0.08.tar.gz
asterisk-addons-1.2.2.tar.gz
asterisk-sounds-1.2.1.tar.gz
```

Uncompress all the files with:

```
# gzip -d *
```

To compile Asterisk, type:

```
# tar xf asterisk-1.2.7.1.tar
# cd asterisk-1.2.7.1
# make ; make install ; make samples
```

To compile Asterisk add-ons, type:

```
# tar xf asterisk-addons-1.2.2.tar
# cd asterisk-addons-1.2.2
# make ; make install
```

To compile Asterisk Perl, type:

```
# tar xf asterisk-perl-0.08.tar
```

```
# cd asterisk-perl-0.08
# perl Makefile.PL ; make install
```

To add the additional audio files, type:

```
# tar xf asterisk-sounds-1.2.1.tar
# cd asterisk-sounds-1.2.1
# cd sounds
# cp -pr * /var/lib/asterisk/sounds/
```

If you run into any errors, please consult the Asterisk documentation for assistance. I haven't seen Asterisk fail to compile on any supported platform that I've tested so far.

In addition to the Asterisk package and a few add-ons, which should now be installed, you will want to make sure you have mpg123 installed. This program is used to play mp3s, for example, when callers are placed on hold.

If you do not have mpg123 installed, download it, uncompress it, and type:

```
# tar xf mpg123-version.tar
# cd mpg123-version
# make linux ; make install
```

## Configuration

By default, the primary configuration directory is /etc/asterisk. Asterisk looks here for a number of configuration files like sip.conf, extensions.conf, asterisk.conf, etc. Let's take a look at each of the more important configuration files now. I will provide trimmed down examples that you can use to get started. Please note that Asterisk frequently deals within a "context", and contexts start when a name is read between brackets.

Asterisk requires a reload when configuration changes are made. There are ways to selectively restart portions of Asterisk from the command line, but in this article, we will be restarting Asterisk completely. To fire up Asterisk with verbose output for testing, run `asterisk -cvvvv` from the command line. When you want to restart Asterisk after a change, hit Control-C and run `asterisk -cvvvv` again. You will encounter configuration problems, and when these occur, the console output is extremely helpful. You can run Asterisk in the background when you are ready by simply typing `asterisk` on the command line.

### sip.conf

The sip.conf file is where you define your SIP phones. For example, below you will see a general, default context and an entry for extension 1000:

```
[general]
context=default          ; Default context for incoming calls
allowguest=no            ; Allow or reject guest calls (default
                         ; is yes, this can also be set to 'osp'
realm=yourdomain.com     ; Realm for digest authentication
                         ; Realms MUST be globally unique
```

```
                        ; according to RFC 3261
                        ; Set this to your host name or domain
                        ; name
bindport=5060           ; UDP Port to bind to (SIP standard
                        ; port is 5060)
bindaddr=0.0.0.0        ; IP address to bind to (0.0.0.0 binds
                        ; to all)
disallow=all            ; Disallow all codecs
allow=ulaw              ; Use ulaw codec
callerid = Unknown


[1000]
type=friend             ; either "friend" (peer+user), "peer" or
                        ; "user"
context=from-sip
callerid=Joe Schmoe <1000>
secret=petname
host=dynamic            ; we have a static but private IP address
nat=yes                 ; there is not NAT between phone and
                        ; Asterisk
dtmfmode=inband         ; either RFC2833 or INFO for the
                        ; BudgeTone
mailbox=1000@default    ; mailbox 1234 in voicemail context
                        ; "default"
disallow=all            ; need to disallow=all before we can use
                        ; allow=
allow=ulaw              ; Note: In user sections the order of
                        ; codecs
                        ; listed with allow= does NOT matter!
```

For a full list of available values, please see:

```
http://www.digium.com/en/docs/asterisk_handbook/sip.conf.html
```

The following excerpt from the reference documentation is helpful:

> The sip.conf file is read from the top down. The first section is for general server options, such as the IP address and port number to bind to. The following sections define client parameters such as the username, password, and default IP address for unregistered clients. Sections are delineated by a name in brackets. The first section is called general (which cannot be used as a client name.) The following sections begin with the client name in brackets, followed by the client options.

We'll focus more on the context value later when we begin defining the dial plan in extensions.conf. The most important parameters are:

*type=friend* — Allows each SIP phone to receive and initiate phone calls.
*dtmfmode=inband* — Relates to how tones are registered for each SIP phone.

*mailbox=1000@default* — States where to send calls to voicemail.
*disallow=all* — Reject all codec types.
*allow=ulaw* — Force the ulaw codec; this can be set to a number of codecs.

The other options should be fairly self-explanatory, and the reference documentation listed above addresses each option as well.

### iax.conf

The iax.conf file is where you define ITSP accounts for inbound and outbound voice service, among other things. I will focus on outbound only, as a PRI is going to handle incoming calls. If you are interested in using an ITSP for inbound, check out Junction Networks as an option. They provide easy-to-implement documentation about using their service with Asterisk.

We don't need to add much to iax.conf for outbound service; we will add more to extensions.conf that really makes it work. For this example, set up an account with VoipJet and follow their instructions. VoipJet provides a chunk of free minutes, which is helpful when in test mode. In the end, you will add something like this to iax.conf:

```
[voipjet]
type=peer
host=64.34.45.100
secret=nkd38sne89d78fc
auth=md5
context=default
```

For a full list of available values, please reference:

```
http://www.digium.com/en/docs/asterisk_handbook/iax.conf.html
```

The following excerpts from the reference documentation and my own additions describe the example values above:

- peer — A peer receives calls from the Asterisk server, but does not place them. (This is what we want as we are only making outbound calls via VoipJet.)
- host — The IP address of your ITSP.
- secret — Your md5 value.
- auth — The type of secret being used.
- context — How the context is referenced in extensions.conf, if desired.

### extensions.conf

Over time, you will likely become most familiar with the extensions.conf file. Most of the logic behind handling call flow is managed here. Because of this, let's look a little bit more at how the configuration is parsed. I'm not going to cover all the syntax as there are quite a few options.

Recalling sip.conf, you'll remember we added a SIP phone at extension 1000. Until it is referenced in extensions.conf, the phone will be nonfunctional, as Asterisk doesn't know when to send a call to it. Extension to extension dialing will also not work. Add the following to extensions.conf:

```
[extensions]
  exten => 1000,1,Dial(Sip/${EXTEN}|15|t)
  exten => 1000,2,Voicemail(u1000@default)
  exten => 1000,3,Hangup
  exten => 1000,hint,Sip/1000
```

We've now defined extension 1000 and told Asterisk how to dial it. Pay special attention to the order value, the field after the first comma. This field tells Asterisk in which order to process the configuration block for 1000. So, the first line tells Asterisk to dial the SIP phone registered at extension 1000 for 15 seconds and to play back a traditional ring tone while dialing (Asterisk can be configured to play music while dialing extensions as well). If there is no answer after 15 seconds, the second line sends the call to the voicemail box of extension 1000. The third line then simply hangs up the line.

This is nice so far, but how exactly do calls end up referencing the extensions context? Look at the sip.conf file again, and you'll see that we placed extension 1000 inside the context of "from-sip". Make sure the following exists in extensions.conf:

```
[from-sip]
  include => extensions
  include => eleven-digit
```

This context definition tells Asterisk that when a SIP phones dials a number, it should first check the extensions context and then check the 11-digit context. You can add as many contexts as you like; these are only examples. If the dialed number matches a rule within a context, that action is then taken. Our 11-digit context looks like this:

```
[eleven-digit]            ;11 digit dialing
  exten => _1NXXNXXXXXX,1,Set(CALLERID(num)=7078222005)
  exten => _1NXXNXXXXXX,n,Dial(IAX2/9142@voipjet/1${EXTEN}||T)
```

These rules tell Asterisk to match any 11-digit pattern starting with a 1. The caller ID is set to 7078222005 and VoipJet is then used to place the outgoing call.

You'll want to define another SIP phone and test extension-to-extension dialing after adding a configuration for both phones to sip.conf, extensions.conf, and restarting Asterisk. You can also try dialing 11-digit numbers now from a registered SIP phone.

Inbound calls over the PRI are also defined in extensions.conf. Regarding setting up the hardware and installing/configuring the card, please reference Digium's documentation. We will place the configuration information for calls coming in over the PRI under the context default in extensions.conf. The configuration will look like:

```
[default]  ; For the PRI

  exten => 8222005,1,Answer
  exten => 8222005,n,Wait(1)
  ;exten => 8222005,n,GotoIfTime(9:00-18:00|mon-fri|*|*?,,CLOSED)
  exten => 8222005,n,Goto(ivr,s,1)
  exten => 8222005,n(CLOSED),Macro(closed)
```

These rules tell Asterisk that when a call comes in on 8222005, which is the DID of the PRI, it should make sure it is within business hours and then send the call over to the IVR menu for further call routing.

## IVR Configuration

IVR menus can be complicated and spiffy, but I want to briefly cover some of the basics. The IVR configuration, like many other functions, is handled inside of extensions.conf. The context "ivr" was referenced above for calls coming in on the PRI, so we need the following in extensions.conf:

```
[ivr]
  exten => s,1,Set(TIMEOUT(digit)=5)
  exten => s,n,Set(TIMEOUT(response)=5)
  exten => s,n,Background(custom/aa)       ;thanks for calling
                                           ;press 1, 2, etc
  exten => s,n,WaitExten(5)
  exten => s,n,Background(silence/4)
  exten => s,n,Goto(ivr,0,1)

  exten => 0,1,Background(silence/1)
  exten => 0,n,Dial(SIP/1000|10|t)
  exten => 0,n,Dial(SIP/1001|10|t)
  exten => 0,n,VoiceMail(u1000)     ; Send the call to voicemail
                                    ; if nobody answers
  exten => 0,n,Hangup
```

These rules comprise a basic example of how to get started with IVR prompts. For more information on how to actually record the audio for playback on the menu, please check out the online documentation. In this example, we have defined a general operator extension that dials two SIP extensions consecutively. Extension 1000 is dialed for 10 seconds and then extension 1001, if 1000 does not answer the call. If both extensions ring for 10 seconds without an answer, the call is sent to voicemail. The voicemail box is defined within the parenthesis, and in this example the call is sent to the voicemail box of extension 1000.

## Voicemail Configuration

We've looked at how to set up a SIP phone, how to handle basic call management and IVR prompts, but we still haven't told Asterisk how to handle calls when we attempt to forward them to the voicemail system. Voicemail configuration is handled in voicemail.conf, and the file needs to include:

```
[default]
1000 => 1000,Joe Schmoe,joeschmoe@yourdomain.com
```

Asterisk will now be able to associate a name and email address to the voicemail box for extension 1000. If you have properly defined the mailer in voicemail.conf, all voicemail messages will be recorded and sent via email to the recipient's email address.

## Conclusion

Asterisk is a great open source PBX solution that is easy to install and straightforward to maintain once you get the hang of the configuration files. With an ever-increasing number of staff members working remotely, it also helps to be able to get inside the PBX and configure it to your own specifications, instead of relying on a vendor with a closed PBX system.

## Resources

Asterisk Web site — `http://www.asterisk.org`
Great VoIP Resource — `http://www.voip-info.org`

*Adam Olson lives in Northern California. He's been active in network design, systems administration, and systems programming for more than 10 years with various companies like MCI WorldCom, small Bay Area startups, and a California-based ISP. He recently co-founded a company, called Office Appliance (`http://officeappliance.com`), which serves the needs of small- and medium-sized businesses.*