

*A Solutions Manual for Network Professionals*



# Switching to VoIP

O'REILLY®

*Theodore Wallingford*

# Traditional Apps on the Converged Network

When first designed, landline phone service was intended to carry sound signals, and its uses as a carrier of data were years away from realization. It's ironic that the technology that predated the telephone was itself a data transport technology: the telegraph. This device carried encoded messages from terminal to terminal across the 19<sup>th</sup>-century equivalent of a peer-to-peer network.

A lifetime later, in the 1960s, sound-encoding devices emerged, and, very soon, computers were able to send data, represented as sound, across the telephone network. Those devices were modems, and later fax machines—the descendants of the telegraph. Modems, fax machines, voice mail systems, emergency 911 service, and a slew of other messaging tools evolved around the international telephone network.

Today voice and data networks converge and VoIP begins to replace Bell's brainchild. IP telephony has the same fundamental goal as legacy telephony: facilitate human interaction at a distance. But, since IP telephony goes about this goal differently, not all of the specialized devices that evolved around the old system work with the new one. Fax machines, modems, and voice mail systems aren't necessarily compatible with VoIP, because they grew into a mold that was shaped by the *old* network.

In this chapter, we'll cover some of the great legacy technologies we've come to rely on and discuss ways of migrating their functionality to the converged network.

## Fax and Modems

Fax machines and modems encode digital data into analog sound signals for transport across the PSTN. At the local CO, these analog sound signals are digitized into PCM digital signals at 64 kbps. After being transported over the network, the CO at the opposite end of the link decodes the PCM signal and plays it back for the analog receiver, whose job it is to reassemble it into a facsimile of the original digital data. Since the maximum rate of transmission on a POTS line is 64 kbps, faxes and

modems can't transmit data very quickly (one twenty-fourth the speed of a T1). Even with compression techniques, which introduce signaling overhead, most modems will never transmit data faster than about 54 kbps. They are limited by the digital resolution of the sound pathway they use.



Sending a fax through an encoded audio channel such as those provided by VoIP and the PSTN is called in-band faxing.

The challenge posed by VoIP is this: codecs like G.729A distort the sound signal by compressing it using lossy vocoder algorithms such as CELP. When a modem or fax's analog sound signal is encoded and decoded using vocoders, it becomes distorted, such that the device on the receiving end of the transmission is receiving a different analog signal than the one that was sent. The side effect of compressed VoIP codecs is that faxes and modems simply don't work. So devices that rely on modems, such as some burglary alarm systems, TiVo consoles, and maybe that old-fashioned Amiga 1000 computer, have a hard time getting along in a VoIP network. (Series 2 TiVo devices can work using the Internet instead of a modem—check out <http://www.tivo.com/adapters>.)

## Terminology Recap

### *POTS*

Plain old telephone service, the basic analog phone line most folks have in their homes.

### *Fax*

Facsimile, a method of sending a copy of a paper document over an analog telephone line.

### *Modem*

Modulator/demodulator, a device that connects computer systems or fax machines by transmitting signals using analog signals.

### *CO*

Central office, the phone company's switching center that provides POTS service to the surrounding area.

### *PCM*

Pulse code modulation, a way of storing an audio signal as a bit stream.

### *CELP*

Code-excited linear prediction, a compression codec used in advanced VoIP codecs. CELP reduces the number of bits necessary for successful transmission of the sound, but it does so by degrading the audio signal somewhat.

There are a few ways to tackle this problem. The first and most obvious is to avoid lossy codecs and use only non-lossy codecs like G.711 and G.722. It won't distort the analog modem signals as long as jitter is under control. But avoiding lossy codecs may not be possible, especially on bandwidth-starved WANs.

Fortunately, the ITU has two recommendations to aid in migration of fax to VoIP. Both are discussed in this chapter.

There is no real answer to the matter of modems, however. The ITU's V.150.1 standard describes how to relay modem signals over a converged network, but no open reference implementations exist yet. One could argue that it's a bit arcane to use a modem in a converged network, since any data sent by the modem could be sent hundreds of times faster using Ethernet. In other words, if you have an Ethernet connection, use that instead of the modem.



Thanks to the insistence of the U.S. Department of Defense, Version 4.1 of CallManager, Cisco's venerable softPBX, does indeed support modem pass-through using V.150.1.

## T.30, T.37, and T.38

The ITU's T.30 recommendation describes how fax devices should work. Just about all legacy fax machines (and fax software) are T.30-compliant.

The T.38 protocol is the ITU's recommendation for sending faxes over data networks. T.38 software runs on a server that can perform T.30 fax signaling in order to send and receive fax transmissions over analog lines to fax machines. The server encodes the fax signals into packets and sends them to a T.38 server that can decode the fax signals on behalf of a receiving fax machine. This circumvents reliance on in-band faxing and increases the efficiency of fax transmission over the network. T.38 has a few nicknames, including Group 3 faxing and FoIP (Fax over IP).

The T.37 protocol is another ITU recommendation. It also allows faxes to be sent over the IP network, but T.37 doesn't package fax signals into its own packets. Instead, it creates SMTP mail messages using MIME-encoded TIFF (Tagged Image File Format) files. These messages can be sent directly to an SMTP mail server for delivery to email users, or they can be sent to another T.37 server for delivery to a fax machine. T.37 has a few advantages over T.38, among them its ease of integration with standards-based email. T.37 is a store-and-forward protocol, meaning it can hold onto messages if they aren't deliverable at the moment. T.38 has no such capability.

### ATAs and fax/modem support

To support a fax machine on the VoIP network, you might think all you have to do is plug it into an analog telephone adapter (ATA) and configure that adapter to only use G.711. But this won't always work. Other VoIP devices on the network may alter

the sound encoding somewhere along the fax machine's call path, possibly crippling it.

There are three ways to ensure faxes can traverse your VoIP network:

- Guarantee that G.711, and only G.711, will be used for all possible fax call paths, and eliminate all jitter.
- Implement T.38 servers at locations with fax machines.
- Get a T.37-compatible fax machine or connect fax machines to T.37 servers.

There's only one way to positively ensure modem traffic can traverse your VoIP network:

- Guarantee that G.711, and only G.711, will be used for all possible modem call paths, and eliminate all jitter.



To make fax and modem communications over VoIP more reliable, decrease the baud rate of the fax or modem connection to 9600 or lower. While this will slow the connection down, it will allow you to overcome minor jitters, which will at least keep the connection intact.

## Project 14.1. Turn Your Linux Box into a Fax Machine

### What you need for this project:

- Asterisk running on Linux
- A POTS line connected to a Zaptel-compatible interface card

Asterisk offers a built-in fax-detection mechanism. This allows you to handle faxes that are sent to your Asterisk PBX on a POTS line. Asterisk's `Answer()` command triggers the fax detection. If an incoming fax is detected, Asterisk automatically transfers the call to the special extension called `fax`, if it exists.

In order to use this special extension, you'll need to compile and install the SpanDSP package. Download the latest version from [ftp.opencall.org](http://ftp.opencall.org), and unzip the file into `/usr/src/spandsp`. To compile it, issue these commands:

```
# ./configure --prefix=/usr
# make; make install
# cp app_rxfax.c /usr/src/asterisk/apps
# cp app_txfax.c /usr/src/asterisk/apps
# cp Makefile.patch /usr/src/asterisk/apps
# cd /usr/src/asterisk/apps
# patch <Makefile.patch
```



If you're worried about the security concerns associated with compiling as root, you can use a non-root account to compile SpanDSP.

These commands compile the SpanDSP package, which provides a source code patch for Asterisk. Thus, you'll need to recompile Asterisk now:

```
# cd ../asterisk
# make clean ; make install
```

The next step is to enable fax detection on the Zaptel channel you want to use for faxing. This doesn't stop the channel from being used for normal voice calls; it just enables the channel to discern fax calls from normal calls. To enable this function, be sure that the channel's section in `/etc/asterisk/zapata.conf` has this entry:

```
faxdetect=both
```

The valid parameters for the `faxdetect` option are `incoming`, `outgoing`, `both`, and `no`. By default, fax detection is disabled.

## Receiving faxes

Now, consider the following snippet from a dial-plan:

```
[incoming-local]
exten => s,1,Answer
exten => s,2,Dial(SIP/202,45,rm)
exten => s,3,voicemail(202)

exten => fax,1,SetVar(TIFFFILE=/var/spool/faxes/thisfax.tif)
exten => fax,2,rxfax(${TIFFFILE})
```

In this context, the `Answer()` command triggers fax detection. If there's no fax, the dial-plan calls for a call to SIP peer 202. If there is a fax, the fax extension takes over, saving the fax image into a TIFF file. Another script could then process the file however you see fit, perhaps printing it immediately, like this:

```
exten => fax,1,SetVar(TIFFFILE=/var/spool/faxes/thisfax.tif)
exten => fax,2,rxfax(${TIFFFILE})
; dump the fax file to the default printer and remove the fax file
exten => fax,3,System('tiff2ps ${TIFFFILE} | lpr')
exten => fax,4,System('rm ${TIFFFILE}')
```



Tiff2PS is a utility provided in the *libtiff* distribution, a library for dealing with TIFF files. It's a standard part of many Linux distributions, Red Hat included.

## Sending faxes

Receiving faxes with Asterisk is quite a bit easier than sending them. This is because, when receiving them, the work of scanning them into a digital form is already done. This is the part neither Asterisk nor SpanDSP addresses. These packages *can* very easily fax a TIFF file. So it's up to you to get that TIFF file in a path where SpanDSP can grab it.

This can happen any number of ways. You could create a simple web interface that allows you to upload TIFF files to the server, or if you've got the right software, you could just scan them directly using the Linux machine. I don't recommend either of these approaches, however, because neither of them provides a straightforward way of telling Asterisk where to send the fax. Fortunately, the Hylafax package provides a solution.

## Project 14.2. Build an Inbound Fax-to-Email Gateway

### What you need for this project:

- Asterisk running on Linux
- SpanDSP
- Sven Slezak's LDAPGet module
- A T1 interface with an active PRI attached or multiple X100P cards with POTS lines attached

In Project 14.1, we built a configuration to direct all incoming faxes from ZapTel channels to a file, which, in turn, we could have automatically printed. But if the server were working on behalf of many possible fax recipients, we would have to rely on the incoming fax's cover sheet to know which recipient it's destined for. Worse still, we'd have to go to the printer, pick up the fax, and hand-deliver it to the correct person.

There's a better way, of course: email. It's just as easy to email that TIFF file to somebody's inbox as it is to print it:

```
exten => fax,1,SetVar(TIFFFILE=/var/spool/faxes/thisfax=${CALLERIDNUM}.tif)
exten => fax,2,rxfax(${TIFFFILE})
; email the fax file to the receptionist and then delete it
exten => fax,3,SetVar(EMAIL=receptionist@oreilly.com)
exten => fax,3,System('mewencode -e ${TIFFFILE} | mail -s fax ${EMAIL}')
exten => fax,4,System('rm ${TIFFFILE}')
```

This configuration receives the fax, MIME-encodes it using the `mewencode` command (a standard part of most Linux distributions), and emails it to the email address stored in the `EMAIL` variable. This is a catchall solution—it sends every fax that's received to the same recipient, who can then forward (and screen if necessary) to the appropriate person based on the content of each fax message.

### Automatic fax routing

In order to have the Linux server automatically route each fax to the right recipient (instead of a certain email user doing it), we must have a way of associating each fax with the correct recipient. We'll have to associate a certain line (or a certain DID) with each user, so that, whenever a fax is received on that line (or DID), we'll know

where to route it. Each phone line's number, or each DID number if we're using a PRI, will become a single user's fax number.

## Fax routing with DIDs

In order to associate a DID with a particular user's email address, we can use an LDAP inquiry. An LDAP client library for Linux, *openldap*, provides applications the ability to access LDAP servers and perform such inquiries. Your LDAP server may be an Exchange or Lotus Domino server where directory information is stored. The Red Hat distribution includes *openldap* and its developer packages.

But you'll need more than just the LDAP client library. You'll also need an actual LDAP client for Asterisk, like Sven Slezak's LDAPGet package. Download it from <http://www.mezzo.net/asterisk> and unzip it into the Asterisk source directory, */usr/src/asterisk*.

Next, as root, copy the *app\_ldap.so* file into */usr/src/asterisk/apps*. Then, use a text editor to add *app\_ldap.so* to the list of applications that begin with APPS= in */usr/src/asterisk/apps/Makefile*. While you've got the make file open, also add the following rule just below above the *app\_voicemail.so* line:

```
app_ldap.so : app_ldap.o
              $(CC) $(SOLINK) -o $@ $< -llber -lldap
```

Now, LDAPGet is ready to be compiled. Save your changes in the text editor and exit back to the shell, where you'll issue these commands to compile the package:

```
# cd /usr/src/asterisk
# make; make install
```

If Asterisk isn't currently running, start it. Then, go to the Asterisk command line load the LDAPGet module (alternatively, you could just restart Asterisk):

```
pbx*CLI> load app_ldap.so
pbx*CLI> show application LDAPget
```

The show application command will confirm the module is installed and loaded by showing you a brief description of the LDAPGet dial-plan command. Now, you can set up the LDAP inquiry your dial-plan will use to get email addresses based on the DID provided by `${EXTEN}`. To set this query up, open */etc/asterisk/ldap.conf*. It may not exist yet, since you've only just compiled the LDAP module. Create an entry like this in *ldap.conf*:

```
[mailfromdid]
host = ldap.oreilly.com
user = cn=root,ou=People,o=oreilly.com
pass = jarsflood
base = ou=Addressbook,o=oreilly.com
filter = (&(objectClass=person)(|(fax=%s)))
attribute = email
convert = UTF-8,ISO-8859-1
```

This configuration will cause the LDAP inquiry to `ldap.oreilly.com` asking for an object of the person class with the attribute `fax` equal to the value of the `%s` token (which will be replaced with the DID at runtime). The attribute setting tells the inquiry which attribute from the object to return as a value to the dial-plan's variable. This may seem confusing right now, but it should be clearer once you see how the `LDAPGet` command is used in the dial-plan.

In the context where your incoming PSTN calls begin (specified in `zapata.conf`), you can capture the DID from the `${EXTEN}` variable, and use it to supply an argument to an LDAP inquiry. If the inquiry is successful, Asterisk's LDAP client will return the email address to which the fax number (i.e., DID) is associated, as in this snippet of `extensions.conf`:

```
[incoming-pstn]
exten => s,1,SetVar(DID=${EXTEN})
exten => s,2,Answer
exten => s,3,Ringing
; allow 4 seconds for the fax detection
exten => s,4,Wait(4)
; if no fax, send this call to be handled elsewhere
exten => s,3,GoTo(incoming-voice)

; here's the fax handling extension, which sends the call to the
; 'inc-fax' context
exten => fax,1,GoTo(inc-fax,1,1)

[inc-fax]
exten => s,1,SetVar(TIFFILE=/var/spool/faxes/${DID}.tif)
; The 'mailfromdid' LDAP inquiry is defined in Asterisk's ldap.conf file.
exten => s,2,LDAPGet(EMAIL=mailfromdid/${DID})
; If the LDAP inquiry succeeds, priority will be 2+1.
exten => s,3,rxfax(${TIFFILE})
exten => s,4,GoTo(105)
; If the LDAP lookup fails, priority will be 2+101.
exten => s,103,SetVar(EMAIL=receptionist@oreilly.com)
exten => s,104,rxfax(${TIFFILE})
; Now, email the fax file to whichever email address was decided upon.
exten => s,105,System('uuencode ${TIFFILE} uuenc | mail -s fax ${EMAIL}')
exten => s,106,System('rm ${TIFFILE}')

[incoming-voice]
; non-fax calls are handled here
```

The sum of all this compiling and config-tuning is that different email recipients now have assigned fax numbers on the PRI (or assigned POTS lines for their exclusive use as inbound fax lines). When you send a fax to Todd's fax number, Todd receives the email. When you send it to Susie's, Susie receives the email, and so on. Of course, it's up to you to populate your LDAP server with the right information and to make sure the inquiry config in `ldap.conf` matches your LDAP server's schema.



Don't have an LDAP server? You could use Asterisk's built-in database commands to resolve DID's to email addresses, instead. Chapter 17's dial-plan command reference covers these commands.

### For those who prefer PDF files

In the last project, we used the `tiff2ps` command to create a printable version of the fax, but with a few extra steps, we can turn it into a PDF file, too. PDF may be preferable to TIFF when using email as we are in this project. Consider the following dial-plan changes to the `[inc-fax]` context:

```
exten => s,105,System('tiff2ps -2eaz -w 8.5 -h 11 ${TIFFILE}| ps2pdf \  
>${TIFFILE}.ps')  
exten => s,106,System('uuencode ${TIFFILE}.ps uuenc | mail -s fax ${EMAIL}')  
exten => s,107,System('rm -f ${TIFFILE}*')
```

Now, instead of just encoding the TIFF file and emailing it, the file is converted to a PostScript file, then to a PDF file, before being uuencoded and emailed to the appropriate recipient.

### Simulating T.37

This project can be taken one step further, to crudely simulate T.37—that is, using SMTP to trunk faxes back and forth between email systems that have analog fax endpoints connected to them. In a scenario with three offices and multiple fax machines at each office, you could build a fax bank that routes the faxes to the right machines, anywhere within those three offices. Each participating server would be configured to package incoming faxes into mail messages, as before, but they'd also need to be able to automatically decode incoming email, save the TIFF file attachment to a temporary directory, and fax it to the correct fax machine using the `SpanDSP txfax()` dial-plan command.

## FoIP Solutions

### Cisco AVVID

Cisco media gateway routers can host analog telephone ports. Cisco implements T.37 and T.38 protocols on their analog telephone interface cards, so that, once you provision T.37 or T.38, you can connect a fax machine and have reliable faxing over IP. IOS commands are provided for administration of fax functionality on firmware loads for the Cisco 2600XM, 2800, 3700, and 3800 series routers. More information is available by searching Cisco's web site at <http://www.cisco.com>.

## Avaya Media Servers

The Communication Manager software platform for Avaya's Media Server chassis provides support for fax machines connected to analog ports. Avaya supports both T.37 and T.38, though its T.38 support is older and more refined. Only Media Servers with analog ports can host fax machines. More information is available by searching Avaya's web site at <http://www.avaya.com>.

## Other solutions

Nortel, Alcatel, and other, smaller vendors offer T.38 support. Chapter 16 contains a listing of web sites for commercial PBX vendors so you can investigate commercial solutions further.

## Commercial soft fax solutions

Since fax is a largely legacy technology that has been replaced, for better or worse, by email, you may want to consider the option of dropping the use of fax machines and switching to a 100% soft environment for faxing. Commercial software products like Interstar's Lightingfax and ACCPAC's faxServe can turn a Windows server into a fax concentrator that can route incoming and outgoing faxes to and from users' desktops. For a Macintosh-compatible commercial solution, try 4-sight fax from Soft Solutions.

## Hylafax

Without a lot of hacking, Asterisk just doesn't make a good day-to-day, occasional-use fax server for outbound fax transmittals. There are better solutions to this need already. One of them is Hylafax, a freely available fax server for Linux and BSD operating systems. Hylafax can use standard fax/modems, which also makes it cheaper to implement than Asterisk with (comparatively expensive) Digium voice cards. Hylafax can be obtained from <http://www.hylafax.org>.

## Hosted Fax

If all that dial-plan configuration seems a bit over the top, consider using a hosted fax service provider instead. Some service providers host fax servers with lots of PRI circuits, allowing subscribers to send and receive faxes from a desktop computer using the Internet. This is a stress-free alternative to implementing FoIP. Here is a partial list of hosted fax service providers:

- BroadFax (<http://www.broadfax.com>)
- DataOnCall UniFAX (<http://www.dataoncall.com>)
- eFax (<http://www.efax.com>)
- Extreme FAX (<http://www.extremefax.com>)

- InnoPort (<http://www.innoport.com>)
- MyFax (<http://www.myfax.com>)

## Fire and Burglary Systems

Many fire detection, sprinkler, and burglary alarm systems require the use of one or more POTS lines. This line is used to notify the monitoring company in the event of an alarm trip or a sprinkler head break. A modem inside the alarm system's CPU dials a computer at the monitoring station, and the alarm dispatcher's computer tells her information about the alarm trip.

These systems use an analog data modem that won't work reliably in a pure VoIP environment (see the earlier section, "Fax and Modems"). There are a few ways to overcome this problem. You could replace the system with a newer, Internet-aware system. Some alarm installers offer CPUs with Ethernet interfaces that can replace the modem for notification. But, if necessary, you may have to keep a POTS line (or two) around until you can upgrade.

## Surveillance Systems and Videoconferencing

As you move to a converged network, you may want to factor video into your plans. Just as the converged network replaces the legacy voice network, it can also replace the legacy surveillance and videoconferencing networks.

### Camera Surveillance

Most traditional surveillance networks consist of meters and meters of expensive, inflexible coaxial cable reaching from a low-resolution monochrome monitoring station out to a handful of analog surveillance cameras. The videocassettes used to record surveillance videos are bulky and prone to failure, and the quality of most surveillance video is poor.

With digital, IP-enabled surveillance cameras, a converged network, and good centralized surveillance software, you can eliminate all those shortcomings. Indeed, as you prepare your network for VoIP by implementing high-speed switching and quality of service, you're also preparing your network for video. Contrary to popular belief, it's surveillance, not videoconferencing, that ranks as the number one application for video in business.

Like VoIP endpoints, video surveillance endpoints (cameras) are fully self-contained, have a microprocessor, an Ethernet interface, and a variety of features. Some cameras have a web-based interface that gives you pan, tilt, and zoom control so you can

point a stationary camera, zoom in and out, and so forth. More sophisticated cameras add infrared night vision, multicast streaming, and other cool features.



To check out some surveillance cameras that are inadvertently available for your perusal on the Internet, trying Googling:

```
inurl:"ViewerFrame?Mode="
```

Video can eat up a lot of network bandwidth, 5 to 25 times that of a typical G.711 phone call, depending on the video codecs, resolution, and color depth employed. Some developers, like DIVR Systems (<http://www.divrsystems.com>) offer remote video surveillance solutions that ride on the same network as VoIP. In order to keep from having video swamp the network and break down phone call quality, your video surveillance apps need to play nice with QoS. This means making sure that surveillance traffic is treated with a lower priority than phone call traffic.

## Videoconferencing

Video meetings are closer to telephony than video surveillance, but their requirements are a bit different. First off, while video surveillance needs reliable delivery to ensure that every frame of video is recorded, videoconferencing is more like a phone call; if a frame gets dropped here and there, no problem. So reliable packet delivery, a la TCP, isn't necessary. Also, the need for multicasting (that is, having many viewers watch a single video source) is more prominent in video conferencing than in surveillance. Finally, videoconference transmissions tend to be bursty and fairly short—under a few hours usually. But surveillance video tends to be steady, and round the clock.

While not necessarily inclusive in VoIP, videoconferencing has implications for the VoIP network, just as surveillance video does—in QoS mainly. But, since videoconferencing is indeed a call-switching application, it is often implemented in, or with, phone systems. Some TSPs even offer videoconferencing service with special, camera-equipped screen phones. Packet8's VideoPhone plan is one such service. It allows Packet8 customers to place VoIP/video calls to other Packet8 customers.

Cisco's AVVID IP/VC solution allows video-equipped H.323 endpoints to participate in videoconferencing. SIPQuest Corp. offers a commercial SIP-based videoconferencing server that runs on Linux. Sony offers a SIP-based videoconferencing solution (Sony PCS) that works with any video-compatible SIP-switching server, such as NEC's Univerge SV7000. Technically, if SDP is used for call setup (as it is in just about every SIP implementation), adding videoconferencing to a call is merely as simple as supplying a SIP video phone on both ends.

As such, it's quite easy to create a videoconferencing solution with Asterisk. Start by enabling H.263 and H.261 video codecs in *sip.conf*. Video softphones you can try include LinPhone for Linux and FreeBSD, Windows (MSN) Messenger for Windows

desktops, EyeBeam for Mac OS X and Windows, and iFon for PocketPC. SDP decides the parameters of the video channel during call setup.

## Voice Mail and IVR

Voice mail and interactive voice response systems have always been soft-driven, because they embody business logic. They need to be customizable in ways that a traditional phone switch doesn't. Consequently, voice mail and IVR have been at home on the microcomputer server for a lot longer than call switching has been. Early IVR and voice mail software packages ran on Solaris, OS/2, and Windows NT.

The voice mail/IVR server would be connected to the PBX by dedicated trunks, usually a T1 interface or a handful of FXO/FXS links. The PBX would assign those trunks special extensions, and when a caller needed voice mail or IVR function, his station would be bridged with the trunk connected to the voice mail/IVR server. In some cases, especially on very high-end chassis, the voice mail/IVR and PBX would be housed in one unit.

Today, the IVR and PBX systems are both soft-driven and can therefore be maintained at the same place. Dedicated trunks are no longer needed to link PBX to IVR as long as you choose a platform that integrates them on the same server, unless you design it so they're housed separately for load-management or logistical reasons.

Asterisk offers built-in IVR functions, which are issued in *extensions.conf* or through the Asterisk Gateway Interface. Probably the most oft-used IVR application in Asterisk is its remarkably powerful voice mail system, Comedian Mail.

### Project 14.3. Build a Custom Voice Mail System

#### What you need for this project:

- Asterisk

One of Asterisk's most compelling features is its voice mail server. In fact, you could use Asterisk voice mail as a transition piece as you move your current PBX to VoIP. Chances are good that Asterisk offers more voice mail functionality than your current voice mail system, anyway. Once you transitioned your voice mail to Asterisk, you would be only a dial-plan away from having a full call manager online and ready to handle calls.

But since we've covered call management quite a bit already, let's concentrate solely on voice mail. In Asterisk, the voice mail system is configured by the */etc/asterisk/voicemail.conf* file. Like the dial-plan in *extensions.conf*, the voice mail configuration file establishes contexts. Each voice mail box exists within a certain context, allowing logical separation of subscriber groups, say, on a system that hosts voice mail for several different businesses.

Like *extensions.conf*, *voicemail.conf* can use include directives to encompass the content of other files, as long as they contain settings that are valid for the voice mail server—and not some other module. As is the convention elsewhere in Asterisk, the *voicemail.conf* file begins with a [general] section that defines settings that apply to the entire voice server.

In this project, we'll be customizing Asterisk's voice mail config in order to make it email newly recorded voice messages to different peoples' inboxes. We'll also be customizing the content of the voice mail announcement message that is sent when new voice mails are received.

To get started, let's establish a couple of voice mailboxes in *voicemail.conf*:

```
[default]
201 => 201,John Lennon,jl@oreilly.com,,attach=yes|tz=mountain
202 => 202,George Harrison,gh@oreilly.com,,attach=no|saycid=yes
```

Now, take a look at the format that defines the syntax of this config:

```
mailbox => pin,user_name,user_email[,user_pager_email][,option(s)]
```

The alphanumeric *mailbox* must be unique within the voice mail context but can be duplicated elsewhere in the voice mail configuration. The *pin* is the DTMF string that will be required in order for the subscriber to retrieve messages by phone. The *user\_name* and *user\_email* settings tell the server how to address email-forwarded messages. The *user\_pager\_email* setting gives the server a secondary email address that's intended for use with email-interfaced paging and cell phone text messaging networks. Finally, the *options* setting is a string of pipe-limited options that are outlined in Table 14-1.

Table 14-1. Asterisk voice mail box options

Option	Description	Possible values
tz	Defines the time zone for this subscriber so that timestamps can be localized	Any time zone configured in <i>/usr/share/zoneinfo</i>
attach	Specifies whether or not to attach the sound file of the voice mail message when notifying by email	yes; no
callback	Specifies the dial-plan context to which callers are returned after they've left a message for this subscriber	default
dialout	Specifies the dial-plan context in which subscribers exist if they dial an extension from within the voice mail application	default
operator	Specifies whether or not the user can return to the operator extension by dialing 0 (zero) within the voice mail application	yes; no
review	Specifies whether or not callers to this subscriber can review (listen to and approve or rerecord) messages	yes; no
saycid	Specifies whether the caller ID number of the person who left the message should be announced when the message is played by the subscriber	yes; no

## Turn on email notification

Enabling notification of new voice mail messages by email is as simple as three steps. First, edit the `servermail` and `fromstring` settings in the `[general]` section of `voicemail.conf` so that notification messages will come from an appropriate email address like `voicemail@yourcompany.com`. Second, make sure the local mail routing agent (`sendmail`) is operational on the voice mail server. Third, enter the email address of each subscriber into her mailbox declaration in `voicemail.conf`.

## Attach voice mail sound files to email notifications

Adding `attach=yes` to the options string in a mailbox declaration will cause recorded messages to be forwarded (as attachments) to the email addresses associated with that mailbox. There's really only one thing you can customize here, and that's the format in which the recorded sounds are encoded. You have three choices—`wav49`, `wav`, and `gsm`. `WAV49` is highly compressed and compatible with most client desktops, so it's the recommended format for emailing. Unless you're doing some further processing with an SMTP agent after the message that requires it is sent, there's no reason to use `WAV` or `GSM`.

## Customize the email notification message

The template for the notification email is stored in the `emailbody` variable, defined in `voicemail.conf` or one of its includes. You just have to define it. Feel free to mix in any of these system variables while you're at it:

`VM_NAME`

The name of the subscriber for whom the message was left

`VM_DUR`

The duration, in seconds, of the message

`VM_MSGNUM`

The number of the message relative to the subscriber's mailbox

`VM_MAILBOX`

The mailbox number of the subscriber

`VM_CALLERID`

The caller ID string for the caller who left the message

`VM_DATE`

The date and time the message was left, relative to the time zone specified in `voicemail.conf` for this subscriber

Consider the following `emailbody` declaration:

```
emailbody=Hello ${VM_NAME},\n\nThere is a new voice mail message in mailbox number\n${VM_MAILBOX}. It was left on ${VM_DATE} from ${VM_CALLERID}. To listen to this\nmessage, dial 1000 from your desk phone or 216-524-0071 from an outside phone.\n\nThanks,\nThe voicemail System
```

This message would appear more or less as follows, when used to send a voice mail notification:

Hello John Lennon,

There is a new voicemail message in mailbox number 201. It was left on 7/3/05 12:42:42 from (440) 365-1964. To listen to this message, dial 1000 from your desk phone or 216-524-0071 from an outside phone.

Thanks,  
The Voicemail System

The `\n` token, common in Unix, starts a new line. If the recorded message were actually attached to the email, we would need to use a different message template, indicating that.

### Connect the dial-plan to voice mail

Once the voice mail server is set up the way you want it, connecting it to the dial-plan is a snap. First, establish an extension where subscribers can retrieve their messages and manage their mailboxes. This example attempts to plug in the extension number of the caller in order to forgo a prompt to enter their mailbox number, assuming the extension and mailbox numbers match:

```
; extensions.conf
[default]
...
exten => 1000,1,VoiceMailmain(${CALLERIDNUM})
```

If the caller ID doesn't match any mailbox, the subscriber will have a chance to enter his mailbox number. Next, make sure that some extensions trigger the voice mail application:

```
; extensions.conf
[default]
...
exten => 201,1,Dial(SIP/201,30)
exten => 201,2,VoiceMail(201)
```

In this case, extension 201 will attempt to ring SIP peer 201 for 30 seconds. If there's no answer, the voice mail greeting for mailbox 201 will be played and the caller will have an opportunity to record a message.

### Web access to voice mail

Asterisk includes a Perl CGI script (*vmail.cgi*) that you can use to give your voice mail subscribers access to their mailboxes via the Web. This is great for road warriors who may not have direct telephony access to the phone system while they're out and about. By placing the script in Apache's *cgi-bin* script path and making it executable by the web user, this script works fine for a plain-vanilla install of Asterisk. Figure 14-1 shows what it looks like from a web browser.

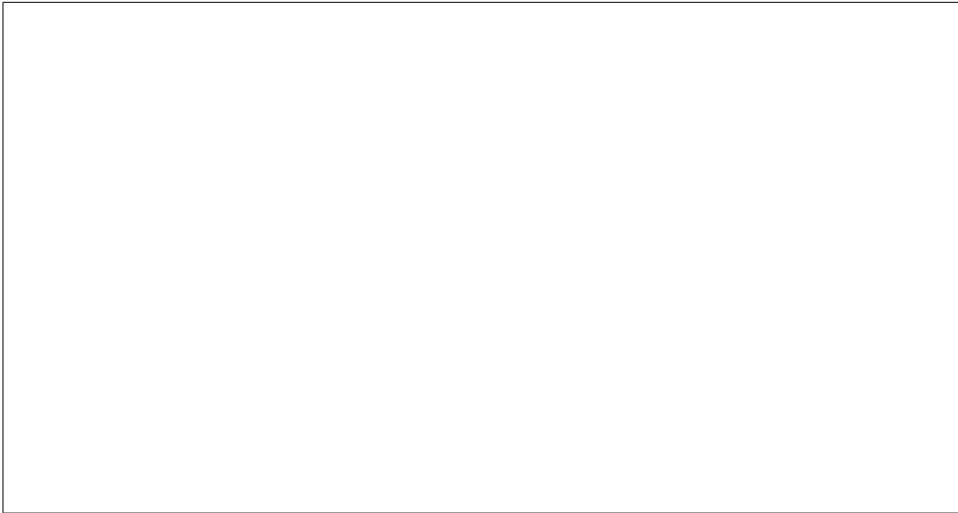


Figure 14-1. Asterisk's *vmail.cgi* allows web-based access to voice mail

## Project 14.4. Create Custom Sounds to Interact with Callers

### What you need for this project:

- Asterisk
- A sound recording application (optional)
- SoX (<http://sox.sourceforge.net>)

Interactive voice response is the backbone of voice mail, but there's so much more you can do with it. You could replace pen-and-paper time cards and punch clocks. You could use it to perform billing functions, such as collecting a credit card number and storing it in a database. But in order to make all these things cohesive to the caller (the user), your user interface will have to be customized. This means using, and making, sounds.

### Use a prerecorded sound for a simple paging routine

We'll start with a quick little dial-plan tool that allows us to send quick emails. This will allow us to use one of Asterisk's prerecorded sounds. In this example, we'll build a basic numeric pager. It will screen an incoming call, record the caller's phone number, and email it to an SMTP recipient with instructions to "please call" the number.

```
exten => *89,1,Read(PhoneNo|allison7/enter-phone-number10|10)
exten => *89,2,System('echo Please call ${PhoneNo}. | mail user@oreilly.com')
exten => *89,3,Playback(thank-you)
exten => *89,4,Hangup()
```

You may also notice output like this in Asterisk's verbose console log when the `Read()` command captures the dialed digits, in accordance with the first priority of extension \*89:

```
VERBOSE[1122993344] --User entered '5552345678'
```

The `Read()` command plays the *enter-phone-number.gsm* file and then stores the 10 digits entered in the `{PhoneNo}` variable, which is used at the next priority in a shell command (via the `System()` command) that sends a brief email message to `<user>@oreilly.com`.

## Use your own sounds in IVR

If you can record WAV or AIFF files, you can make custom IVR prompts. These formats must be converted into GSM-encoded files, though, to work with Asterisk. It's better to record the prompts directly into GSM encoding rather than recording in WAV and then encoding to GSM. But not all sound recording applications support GSM out of the box, so you might need to record in one step, and convert to GSM, using the Unix application SoX, in another step.

In Windows, you can use the trusty Windows Sound Record application to create WAV files. On the Mac, you can use a sound recorder like Audicorder (<http://www.blackcatsystems.com>) to create AIFF files. Either way, once you record the sounds, you'll have to convert them using SoX, the Unix SOund eXchange application.

## Converting sounds using SoX

Using SoX for this task couldn't be any easier. This example converts an AIFF file called *announce.aif* into GSM encoding, merely by specifying the file extension (*.wav*) in the output filename:

```
# sox announce.aif announce.gsm
```

SoX can do plenty of cool things. It can change the sampling rates, equalization, length (of time), and encoding of the output, too. In fact, a full-rate, high-fidelity sample (say, 44,000 Hz) wouldn't be a very good idea in an IVR app, so you could trim it down to size by resampling it into 8 kHz:

```
# sox -r8000 44KHzfile.aif 8KHzfile.gsm
```

There also wouldn't be much sense in using stereo samples, so those can be made single-channel sounds:

```
# sox -c1 stereo.aif mono.aif
```

Finally, you could boost the volume of a quiet sample with the `-v` option, followed by a decimal value that represents an increase in volume if it's greater than 1, or a decrease in volume if it's lower than 1:

```
# sox -v1.5 quiet.gsm louder.gsm
```

## Use soxmix to put background music into an announcement

The soxmix application can mix two sound files into a single file. This could be useful for combining background music with a voice recording. The mix of sound levels between the two source files is determined using SoX's "average" algorithm. Before mixing the two files, you can adjust their volume levels, as shown earlier. Here's a simple example of soxmix:

```
# soxmix music.gsm holdmessage.gsm combined.gsm
```

### Sounds for Business and Pleasure

"One moment please..." (*one-moment-please.gsm*)

"That party is busy; please hold..." (*busy-please-hold.gsm*)

"Enter an extension..." (*enter-ext-of-person.gsm*)

Also included are a few that are suitable for, well, *nonbusiness* use:

"All your base are belong to us!" (*all-your-base.gsm*)

"This is not the extension you're looking for..." (*jedi-extension-trick.gsm*)

"We're off gambling..." (*gambling-drunk.gsm*)

## Record IVR announcements using Asterisk instead of a sound recording app

You can use Asterisk to record GSM files by configuring the dial-plan itself to record your channel into a sound file. This is accomplished using the `Monitor()` command. Consider the following dial-plan snippet:

```
exten => *50,1,SetVar(pathname=/var/lib/asterisk/sounds/)
exten => *50,2,Read(FileName|beep|4)
exten => *50,3,Monitor(wav,${pathname}${FileName}.wav)
exten => *50,4,Read(foo|beep|1)
exten => *50,5,StopMonitor()
exten => *50,6,System('sox ${pathname}${FileName}.wav-in.wav ${pathname}${FileN$
exten => *50,7,Playback(${FileName})
exten => *50,8,System('rm -f ${pathname}${FileName}.wav-*)
```

This novel little snippet is designed to record your voice when you dial \*50. It stores the recording in a WAV file, converts it immediately to GSM using SoX, plays it back for you, and then deletes the WAV file originally used to record the call. To use it, call \*50, listen for the beep, and enter a four-digit DTMF string. This will be used for the filename. As soon as you enter four digits, you'll hear a second beep. Now, begin speaking. When finished, dial any digit. The file is automatically saved and converted, and you should hear it play back immediately. Once finished, the unconverted files used to record the call are deleted.



You'll won't hear any DTMF tones in the recording as long as you use a phone with out-of-band signaling, such as SIP Info.

## Emergency Dispatch/911

E911, or Enhanced 911, is the common American PSTN's public safety protocol for emergency situations. PSTN subscribers dial 911 and are connected to the fire department or police dispatcher at the closest PSAP (public safety answer point). Between the subscriber's CO and the PSAP are dedicated trunks that carry ALI (automatic location identification) signals from the phone company to the PSAP, so that emergency personnel can know the geographic location of the emergency taking place. This is particularly useful for fires and emergency medical responses.

### E911 on the PSTN

The federal government mandates that all ILECs are required to provide 911 signaling service by way of dedicated trunks. In turn, the local public safety authorities collect a tax, usually through phone bills, to fund operation of the system. Public Utilities Commissions enforce the compliance of 911 operations, and the FCC approves standards that are used in the system.

Competitive LECs that don't have switching facilities can use those of the ILECs in order to route 911 calls.

### E911 on wireless providers' networks

In addition to LECs, wireless voice carriers are now required to provide Enhanced 911 services with ALI signaling. All the major carriers have chosen the Global Positioning System (GPS) as a method of providing accurate location information to the PSAP during emergencies.

### E911 on the converged network

As long as your PSTN trunks come from a LEC and the entire premises served by those trunks are at a single postal address, the default E911 service will work fine. Just make sure that your dial-plan permits 911 dialing. If it doesn't, it could literally have fatal consequences. There are essentially two ways to handle 911 calls from a VoIP network in accordance with FCC and public safety standards:

- Attach a LEC's POTS line or PRI circuit to each office location on the network for local routing of 911 calls, rather than handling them centrally in another location.
- Use a PBX-attached database server that can pinpoint the locations of users who dial 911, and pass that information to the PSTN.

A third option exists when direct PSTN connectivity just isn't possible, such as a VoIP network whose only outlet to the PSTN is a VoIP trunk that has no ALI service attached:

- Use local dial-plan configuration to route calls to “extension 911” out to the local authority via a standard 10-digit E.164 phone number.

### 911 on large VoIP networks

In situations in which the private premises is very large—say, a college campus, a high-rise building, or several floors of a skyscraper—E911 is more tricky to implement. If you had a VoIP network with two offices that were 50 miles apart and a single PSTN connect point served both offices, public safety workers could be dispatched to the wrong office. This is because the E911 ALI signals would reference the postal address of the connect point, even if the emergency occurred at the other office!



Routing all 911 calls in an Asterisk context to a POTS trunk is very easy. Just make an extension called 911 and use the `Dial()` command with an outbound trunk group:

```
exten => 911,1,Dial(Zap/g1/911)
```

Fortunately, there are a few ways around this problem. Probably the easiest (though not necessarily the most economical) way to deal with E911 is to put a LEC's POTS line at each remote location and reserve it for use only when somebody dials 911 at that location. That way, the PSAP would get the call via the local CO closest to that location. You could also use this POTS line as a failover line if the WAN link from the remote office failed; this way, people there could still make PSTN calls.

Another way of dealing with the E911 issue is to use database servers that store location data about each endpoint or TDM gateway. This way, when somebody dials 911, the PBX can pull the location of her endpoint from the database and supply it, via ALI signals, to the municipality. Such a solution would mean being able to send ALI signals through the LEC's network, rather than having them originate there. This is an expensive proposition that makes sense in only the largest environments. If you had 10 floors of a skyscraper, 3 floors of the skyscraper across the street from it, and 1,400 users, then you would benefit from this technology. Cisco makes such a solution. They call it the Cisco Emergency Responder.

### Private Switch ALI

If Duane works in Lexington half the time and Louisville the other half of the time, he may tote a WiFi SCCP phone back and forth to both offices. This SCCP phone may register with two different CallManager servers—so who knows who will be reached when Duane dials 911 on it? Aside from statically linking endpoint assets to

locations, Cisco Emergency Responder (CER) also provides a simple way for mobile users like Duane to use the same phone to reach the closest PSAP from many geographically disparate locales.

CER relates geography and floor plans, not just to endpoint assets, but to network identifiers that aren't mobile, like MAC hardware addresses. The Cisco Discovery Protocol (CDP) is enlisted to find out where on the network each endpoint is at any given moment, then CER correlates that network location with a physical postal address. Finally, the call is connected to the PSAP using a PRI channel whose ALI address matches that which exists in the CER database. Cisco calls this technique Private Switch ALI, or PSALI.

## **Mapped ALI**

Several companies that hail from the computer-aided dispatching software business offer software tools for Mapped ALI, an ALI solution that integrates geographic information systems (GIS) technologies such as GPS. Telecontrol Corp., Telecommunications Systems Inc. (<http://www.telecomsys.com>), and Contact One (<http://www.contactone.com>) each offer Mapped ALI solutions for service providers and public safety organizations. The approach of Mapped ALI is different from that of PSALI, because it doesn't require any stationary infrastructure like Ethernet switches. This makes it ideal in wireless applications or in hosted PBX situations in which you, as the service provider, don't have control over the customer's network. Not all PSAPs are able to use Mapped ALI technology.

## **POTS pass-through**

Of course, all of these solutions work only when there's power. If the lights go out, so do the IP phones, the PBX, and the CER server, if you have one. Now, you wouldn't put in a phone system without a battery backup, but even a top-quality UPS can fail. In such a situation, you need to be able to reach an emergency dispatcher without the use of any of your VoIP gear—because none of it will be operational.

To solve this problem, connect an analog telephone to a pass-through port on the PBX server. A pass-through port is a connector on a POTS trunk interface that allows a standard phone to be used with the trunk directly, circumventing the PBX—even *when the PBX has no power*. Digium's X100P card has a pass-through port that you can use to make calls on the attached POTS trunks even when the host PC is turned off. So, the POTS pass-through phone, which draws its power from the CO through the POTS trunk, becomes your link to the PSAP during a total blackout power failure.

## **Dial-around**

In networks with no POTS lines or PRIs connected to a LEC, such as a VoIP network that gets its PSTN connectivity by way of VoIP trunks, there's no reliable way

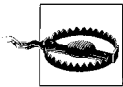
of sending ALI signals to the PSAP. Some VoIP TSPs are working on this problem; others aren't planning on offering a solution. This doesn't mean you can't reach an emergency dispatcher with VoIP. It just means that the dispatcher won't know your location.

Still, dialing 911 won't work with most TSPs. You'll have to reprogram your on-premises VoIP gateway or IP phone to "translate" 9-1-1 into the full E.164 telephone number of the PSAP. To make matters worse, it's not always easy to find out what this number is. With the ubiquity of 911, these numbers aren't generally publicized. You may be able to get only the local fire department's or police department's administrative phone number—and in an emergency, this *isn't* who you're going to want on the phone. Do the research and find out how to get through to the emergency dispatcher (and not some receptionist) via a 10-digit phone number. Then, you can program your VoIP devices to connect 911 calls to that number. An example of such a configuration is given in Project 14.5.

## Project 14.5. Use VoIP Dial-Around to Connect 911 Calls

### What you need for this project:

- Asterisk
- VoIP trunk service from an Asterisk-friendly TSP like VoicePulse



This is very much a hack! The FCC and your local public safety offices would much prefer you to use a solution that supports ALI, like those discussed earlier. But if you absolutely cannot support ALI by providing a POTS line or PRI, this project shows you how to route 911 calls using only a VoIP TSP and Asterisk.

If you use a service such as VoicePulse Open Access, then you are permitted to have a single SIP call ongoing at any time. If your VoIP trunk has this restriction, then dialing 911 from one phone while that trunk is in use won't result in a connection to the emergency dispatcher. Instead, you'll just get a busy signal. That wouldn't be good.

Fortunately, Asterisk allows you to "hijack" channels in priority situations such as a 911 call. The dial-plan command that does the job is `SoftHangup()`. You supply it with the channel name, and it hangs up the channel so a call can be originated on it. So, a very rough hack to hang up the line and grab it to make a 911 call would be as follows:

```
exten => 911,1,SetVar(LocalPSAP=440-361-9000)
exten => 911,2,SoftHangup(SIP/VOICEPULSE)
exten => 911,3,Dial(SIP/VOICEPULSE/${LocalPSAP})
```

The second priority in this example hangs up the channel, while the third priority places a call to the local public safety answering point, whose phone number has been stored in `LocalPSAP` with a `SetVar` command in priority 1.

This is fine when you have only one SIP trunk channel at your disposal for 911 calls, but if you have more than one, you wouldn't want to automatically hang up a channel based on some proprietary setting in the dial-plan. The `ChanIsAvail()` command will help us determine which of several SIP trunk channels can be used to place the call—kind of like an outbound hunt group. Consider this dial-plan snippet:

```
exten => 911,1,ChanIsAvail(SIP/VOICEPULSE1&SIP/VOICEPULSE2)
exten => 911,2,SetVar(LocalPSAP=440-361-9000)
exten => 911,3,Dial(${AVAILCHAN}/${LocalPSAP})
exten => 911,4,Hangup()
exten => 911,102,SoftHangup(SIP/VOICEPULSE1)
exten => 911,103,Dial(SIP/VOICEPULSE1/${LocalPSAP})
```

In this example, priority 1 checks to see if either of the two named SIP channels is available (these correspond to SIP peers in *sip.conf*). If one is available, its channel name is stored in `AVAILCHAN` and processing continues at the next priority, 2.

Priority 2 sets the phone number of the public safety dispatcher as in the previous example. Priority 3 dials that number to connect the 911 call on the available SIP channel.

If no channels are available in priority 1, the priority becomes 1+101, or 102. Since no channels are available, priority 102 hangs up a channel arbitrarily so priority 103 can place the emergency call.

## Administrator Tools

In legacy telephony, the tools of the system administrator ranged from a primitive serial terminal all the way up to a dedicated desktop software application that allowed administrative control over the phone system. Traditional PBX CPUs, such as those made by old-schoolers Isoetec, AT&T, and Executone, used a 2,400-baud serial connection to an RS-232 port for management of channels, trunks, and subscribers.

Later model phone systems offered dedicated Windows-based software for managing these things. Most recently, the administrator tools have become web-based. Administering phone systems has become easier and easier, even as PBX systems have grown increasingly feature rich. Commercial softPBX platforms extend telephony administration further than ever before by using desktop and web-based applications with CTI to give administrators point-and-click power.

### Asterisk's administrative interface

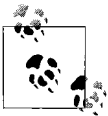
Asterisk doesn't come with any administrative applications other than the (rather limited) `Astman`. As a result, if you want to administer Asterisk, you're going to be

making a lot of visits to the command line and a host of text-based config files, just as we've done in many of the projects herein.

This makes Asterisk administration more akin to an old-school, fridge-sized PBX than to a next-generation VoIP telephony platform. Fortunately, some excellent third-party solutions exist to fill the void. Some are simple command-line tools, while others are XML applications that run on softphones. There are even full-blown, turn-key web-administration solutions for Asterisk.

## AMP

Probably the most ambitious one of these is the Asterisk Management Portal, or AMP. This package is a set of Perl and PHP scripts that not only automate administration of the Asterisk PBX via a web interface, but truly transform it into a turnkey, entry-level PBX product suitable for deployment to offices with a single inbound trunk group. When you load AMP onto an Asterisk PBX system, it uses MySQL to store the system configuration and dial-plan, and a group of custom Perl scripts to mirror that stored configuration with the active configuration of Asterisk.



Even though Asterisk no longer supports MySQL for CDR, because of license conflicts, AMP still uses MySQL to store its dial-plan

AMP turns the Asterisk PBX into a basic, small-office phone system that supports any number of SIP or IAX phones and any number of Zaptel channels. The Zaptel channels are all considered “inbound” and the SIP/IAX phones are all considered private extensions. Extensions can be grouped together in ring groups very easily using the web interface, as can simple IVR call flows and call-parking zones. You can download AMP from <http://sourceforge.net/projects/ampportal>.

## AM

Asterisk Manager (not Astman and not Asterisk Manager API, which are different pieces of software) provides a simple, Perl-driven web interface for configuring Asterisk's dial-plan and SIP peer configurations (see Figure 14-2). Intended to deal with adds, moves, and changes of private extensions, AM gives you more flexibility in programming the dial-plan than AMP. But AM is more of a convenience tool for an administrator who's already familiar with Asterisk than it is a total turnkey solution. You can get AM from <http://astman.sourceforge.net>.

## AstConsole and Asterisk setup assistants

These packages provide those running Asterisk on Mac OS X the ability to maintain SIP extensions and monitor the PBX in a friendly Aqua interface. AstConsole can also connect (via Asterisk Manager API) to remote Asterisk servers to manage and monitor.

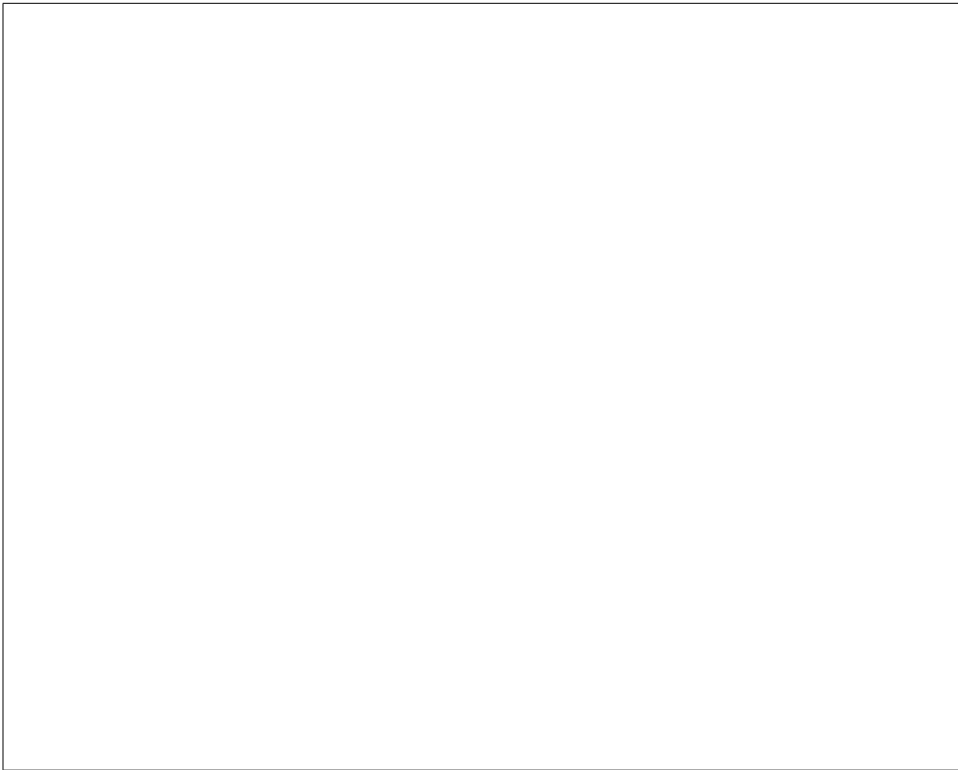


Figure 14-2. The Asterisk Management Portal web interface

The VoicePulse Connect Assistant provides Mac system operators with an easy GUI for configuring trunk connections to VoicePulse’s long-distance termination service. These software packages can be downloaded from <http://www.sunrise-tel.com/>.

### **Open H.323 administrative GUIs**

A number of GUI add-ons for Open H.323 exist. A simple Java-based gatekeeper status monitoring tool that allows you to disconnect registered users is available from <http://unix.freshmeat.net/projects/gkgui>. For the GnuGK, there’s a Windows-based console snap-in that allows remote observation and control of the gatekeeper. It’s called GnuGK Control Center, and you can download it from <http://www.gnugk-cc.com>.

### **Other administrator GUI tools**

For more sources of tasty, GUI admin goodies, take a look at the supplier references in Chapter 16.

## Key Issues: Traditional Apps on the Converged Network

- VoIP protocols and standards don't always make turnkey provisions for "old reliable" telephony applications like fax, modems, E911, voice mail, and administrator interface software.
- T.30 faxes and modems don't work over non-real-time voice pathways such as VoIP networks, unless a lossless codec is used and no jitter occurs.
- Lower modem baud rates are more likely to be successful if you inexplicably decide to run modems over VoIP anyway.
- Two ITU standards define your options for implementing Fax over IP, or FoIP. They are T.37 and T.38.
- T.37 is the store-and-forward, email-based Fax-over-IP solution.
- T.38 is the real-time, transport-inclusive FoIP solution.
- The SpanDSP module package for Asterisk gives you the ability to build fax servers that use Digium's Zaptel TDM hardware.
- Hylafax is a freely available (and more cost-effective than Asterisk with SpanDSP) package for building Linux-based fax servers. It uses cheap, abundant fax/modem cards instead of Digium cards.
- For small to medium businesses, hosted fax services can be a money saver compared to supporting an in-house fax server.
- Most fire and burglary systems have a modem that requires a POTS or Centrex line. For this reason, you're pretty much stuck supporting a legacy trunk until more security systems support IP networking for central office monitoring.
- IP-enabled video surveillance will eventually replace many analog camera systems. If you're thinking about moving to IP-based surveillance (or heavy use of videoconferencing), consider the impact of digital video on your converged network as you plan your transition to VoIP.
- Asterisk voice mail can perform email notification and forward sound attachments of the recorded messages to email recipients. Asterisk also supports web-based voice mail retrieval.
- It's very easy to create custom IVR greetings for Asterisk call flows using a basic sound recorder and the Unix-standard SoX sound conversion application.
- Alternatively, you could use the Asterisk Monitor( ) command to transform your phone into a voice recorder for IVR greetings.
- soxmix is used to merge two sounds together, such as background music and a foreground voice greeting.

- Emergency dispatch/911 services are not handled by VoIP protocols. You must build an adequate solution to handle emergency calls.
- POTS lines and PRI channels have built-in automatic location ID (ALI) signals that can alert the public safety dispatcher to a caller's location, but VoIP trunks usually don't.
- There are a number of open source administrative user-interface tools for Asterisk, including Asterisk Management Portal (AMP).