



Published on Free Software Magazine (<http://www.freesoftwaremagazine.com>)

Asterisk, the easy way

Understanding the basics of the Asterisk (the free software phone system)

By Mitul Limbani

Did you know that it's possible to build an entire telephony system centered around computers? One which is free of licensing costs too? Asterisk is a free software application written to do just that, and much more. Why? For the uninitiated, here's why...

Wherever you go, your free software PBX follows!

The telephony industry has been facing a real crisis since the last decade. While the computer industry has undergone a host of transformations since its inception a couple of decades ago, the century old telephony industry has had problems coping with new ideas and advances. The reason for this is the lack of innovative thinking among makers of proprietary technologies that constitute the chaotic mess that we call the telephone system. These systems suffer from all the drawbacks that inherently bug all closed, proprietary systems. The problems surface as these systems progress deeper into their life cycle. The problems caused are typical of closed proprietary systems. From interoperability and scalability problems to problems caused due to lack of foresight during designing, these problems usually warrant expensive remedies. The technology used in these proprietary systems is usually closed and usually remains unpublished. Add to it the expensive licensing fees for this "sophisticated" proprietary technology and it becomes a no-no for users such as home users and small business users who also want a sophisticated, flexible telephony solution but cannot afford the expensive price tags. The success of revolutions like GNU/Linux is attributed to combined efforts of the likes of Linus Torvalds and a pool of geniuses, academicians, hobbyists, and geeks worldwide. Nobody in the telephony industry wanted to build an open, clean, scalable and seamlessly inter-operable platform for telephony like the internet. Fortunately for us, someone from computer science did! Meet Mark Spencer and his brainchild, aptly christened "Asterisk". Asterisk is the "" character that appears on a standard touch-tone telephone pad. This "" character represents a wild card character in the world of computer systems, meaning that it can represent anything from a single character to multiple characters. And this is precisely what Mark Spencer intended Asterisk to represent in the world of telephony—anything and everything. Asterisk not only encompasses what we can presently do with telephony but also whatever we may think of doing with it in future. It is a revolution in the making and is all set to transform telephony.

Get your facts right...

So what exactly is Asterisk? Asterisk is a telephony platform that exists entirely in software. It is distributed under the GNU GPL (General Public License). This makes it possible for everybody to build and deploy their own customized professional telephony solution, from large corporations in search of a reliable ITSP platform to individuals who want to have their own PBX system.

Asterisk was created by Mark Spencer, the founder of Digium (formerly Linux Support Services).

Asterisk, the easy way

Digium, a company founded by Mark Spencer is actively backing Asterisk. In fact, the purpose of creating Digium was largely for providing support services for the free software PBX. Today, Digium designs and supplies hardware for interfacing Asterisk with a host of networks including PSTN and ISDN PRI.

In compliance with free software tradition, community support for Asterisk is available through a host of portals including Digium's support portal and a number of other websites like voipinfo.org which are powered by hordes of Asterisk enthusiasts and professionals. Community support is also available through FreeNode IRC channel where Asterisk developers and serious Asterisk geeks meet. Commercial support for Asterisk is available through Digium and many other commercial establishments, some of whom are dedicated only to Asterisk installation and application development like Asterisk@Office.

How is Asterisk better than traditional telephony systems?

Asterisk is built as a vast flexible software platform composed of a bunch of programs that can perform whatever functions you program them to do instead of a bunch of wires and circuits which perform fixed functions in their range of operation. This is the reason why Asterisk achieves a degree of flexibility which traditional telephony platforms cannot do, no matter how ambitiously designed they are. Due to its being a software program, Asterisk's extensibility is almost infinite, limited only by the goals of its developers.

What can be done using Asterisk?

The possibilities with Asterisk are unlimited. It can be used to build almost anything from a small home PBX to an ITSP (Internet Telephony Service Provider). With features that include voice mail, teleconferencing and call parking, Asterisk easily rivals the features of most modern PBX systems. Asterisk supports inter communications using protocols like the popular SIP or the newer IAX (Inter-Asterisk Exchange), H.323, MGCP, Skinny/SCCP and UNISTIM. To connect your PBX to the traditional PSTN (your telephone system) you can choose from a variety of options like traditional telephone lines, ISDN BRI lines, ISDN PRI lines and VoIP trunks from VoIP providers. Asterisk supports many popular codecs such as GSM, Ilibc, Speex, MP3, G.711, G.726, G.723 and G.729 for compression. Small businesses can set up Asterisk to have all features of a modern PBX at a fraction of the cost. Big corporations and businesses can build a PBX which can seamlessly connect their offices around the globe irrespective of their geographic locations.

What features of interest does Asterisk have for developers?

Asterisk is a developer's dream come true. It has in-built database AstDB, a Berkeley DB Version 1 database. You can also have Asterisk connect to traditional relational databases using AGI, a standard interface with which external programs can control Asterisk dialplan. AGI can be written using many programming languages, though the language of choice is Perl due to its prowess in text processing. Using Asterisk in conjunction with AGI, you can integrate Asterisk into almost any computer application you like. Such is the power of Asterisk.

Selecting hardware and OS for Asterisk

Asterisk, the easy way

Asterisk runs on Linux (preferably kernel 2.6 and above) with mostly any garden variety Intel X86 based CPU (I tested it with an old Intel Pentium II CPU running at 333 MHz with 128 MB of RAM and it seemed to work). But a powerful CPU like a Pentium 4 (preferably the multithreaded HT version or Dual Core) is recommended. The reason for this is that Asterisk may have to perform transcoding on incompatibly encoded streams (say GSM and MP3) in order to interlink them. This guzzles down a lot of CPU juice. Furthermore, the reason that Intel based CPUs are preferred is that not only do these CPUs have higher clock speeds (GigaHertz) but they have deeper pipelines. Such CPUs excel in operations such as encoding and decoding which have hardly any branching instructions that might require flushing of the CPU pipeline and hence make full use of the higher CPU speed.

Asterisk has been known to work with Windows and MacOS X but with severely crippled functionality. We won't discuss other platforms here.

Installing Asterisk

Note that you require root privileges to install and run Asterisk. To install Asterisk, you have to have some distribution of Linux installed on your machine. So far, Asterisk is known to work with Debian, Fedora, Gentoo, Mandrake, Red Hat, Red Hat Enterprise Clones, CentOS, Pie Box, Tao Linux, Whitebox, Slackware, SUSE and Yellow Dog. So here are simplified the Asterisk installation steps... Make a directory for asterisk and related sources, change into it using the `cd` command and proceed.

```
# mkdir /usr/src/asterisk-src
# cd /usr/src/asterisk-src
```

1. Download and install `mpg123`—note that Asterisk is choosy about using the 123 version given below. This package is necessary to play MP3 files for features such as custom music on hold. This is an optional package but installing it is highly recommended.

```
# wget http://www.mpg123.de/mpg123/mpg123-0.59r.tar.gz
# tar -zxvf mpg123-0.59r.tar.gz # cd mpg123-0.59r
# make linux
# make install
```

Make sure that `mpg123` is in the `/usr/bin` directory. If not, create a symbolic link to it from `/usr/bin/mpg123`.

1. Download and install `zaptel`—This step is necessary if you want to connect your PBX with TDM hardware, which is necessary to connect your PBX with the telephone system. This package is optional if you do not want to connect your Asterisk PBX to the PSTN (the telephone network).

```
# wget http://ftp.digium.com/pub/zaptel/zaptel-1.2.9.1.tar.gz
# tar -zxvf zaptel-1.2.9.1.tar.gz
# cd zaptel-1.2.9.1
# make clean ; make install
```

1. Download and install `libpri`—`libpri` is necessary if you want to connect your PBX system to ISDN PRI using the necessary hardware. This package is optional if you do not want to connect your Asterisk PBX with an ISDN PRI line.

Asterisk, the easy way

```
# wget http://ftp.digium.com/pub/libpri/libpri-1.2.3.tar.gz
# tar -zxvf libpri-1.2.3.tar.gz
# cd libpri-1.2.3
# make clean ; make install
```

1. Download and install Asterisk—This will install the Asterisk PBX on your PC.

```
# wget http://ftp.digium.com/pub/asterisk/asterisk-1.2.12.1.tar.gz
# tar -zxvf asterisk-1.2.12.1.tar.gz
# cd asterisk-1.2.12.1
# make clean ; make install
;Do the next if you want sample configuration files to be created.
# make samples
```

That does it. In case you want to install the package Asterisk Addons, do the following.

1. Download and install `asterisk-addons`—This package provides MySQL support for Call Detail Records (CDR). Install it only if you want Asterisk to connect with MySQL.

```
# wget http://ftp.digium.com/pub/asterisk/asterisk-addons-1.2.4.tar.gz
# tar -zxvf asterisk-addons-1.2.4.tar.gz
# cd asterisk-addons-1.2.4
# make clean ; make install
```

1. Download and install `asterisk-sounds`—This package contains professionally recorded sounds to use with Asterisk in addition to what are already provided. Installing this package is optional.

```
# wget http://ftp.digium.com/pub/asterisk/asterisk-sounds-1.2.1.tar.gz
# tar -zxvf asterisk-sounds-1.2.1.tar.gz
# cd asterisk-sounds-1.2.1
# make clean ; make install
```

Now that installation is done, you can test it by typing...

```
# asterisk -vvvvc
```

This starts up Asterisk and provides you with the Asterisk Command Line Interface.

Structure of Asterisk

Asterisk exists on your filesystem in the following directories:

- `/etc/asterisk`—Asterisk's configuration files, including the dialplans.
- `/var/lib/asterisk/astdb`—AstDB file and following subdirectories...
- `/var/lib/asterisk/agi-bin`—AGI scripts.
- `/var/lib/asterisk/firmware`—Firmware for asterisk compatible devices.
- `/var/lib/asterisk/images`—Images to be used with applications supporting images.
- `/var/lib/asterisk/keys`—Authentication keys for public/private key authentication of peers.
- `/var/lib/asterisk/mohmp3`—Music on hold files.
- `/var/lib/asterisk/sounds`—Sounds such as voice prompts.

Asterisk, the easy way

- `/usr/lib/asterisk/modules`—Asterisk loadable kernel modules.
- `/var/spool/asterisk`—Asterisk uses this directory for certain things that can be done using spooling.
- `/var/run`—Process ID information on all active Asterisk processes.
- `/var/log/asterisk`—Asterisk writes its logs in this location.
- `/var/log/asterisk/cdr-csv`—Asterisk uses this directory to store call records.

Channels

Channels are required for the obvious purpose of creating pathways for communication within the PBX and with the external world. Channels use communication protocols to inter-communicate. The commonly used protocols with Asterisk are SIP and IAX. SIP is arguably the most widely used VoIP (Voice over IP) protocol in the world. It started out as a simple and elegant idea but became bloated and cumbersome over the years. The major drawback of SIP is that it uses separate streams for signaling and data, which presents problems with firewalled systems. NAT traversal is also an issue with SIP but workaround solutions are in place to manage this issue. IAX (deprecated in favor of IAX2) is a brand new protocol developed by the creators of Asterisk is built taking into account all the mistakes of the past and hence is competing with SIP as the VoIP protocol of choice. Channels are written in the file `sip.conf`.

Asterisk configuration

Now that you have installed Asterisk and got it to work, it's time to start with the real stuff. I'll now show you how to set up your PBX system as an intercom and interface it with the outside world in various simple but elegant configurations.

Here are a few requirements before you begin

Since you are developing a PBX, you will need terminals from which you can make and receive calls. You can use either hard phones (the ones that are normally used at home) or softphones (the ones that can be run on your PC). Hard phones are phones which are physically separate entities like your ordinary, humble analog phone. There are two different ways in which hard phones can be used with Asterisk...

- **IP Phones:** IP phones are digital phones which understand VoIP protocol and can make use of the Internet Protocol. IP phones are available in a wide range of pricing and feature sets that range from expensive brands to cheap Chinese alternatives. Branded IP phones from manufacturers like Snom are priced from US\$200 onwards while the cheap Chinese instruments start from US\$60 onwards. I have worked with Snom 200 and found to be packed with useful features, highly reliable and of excellent build quality. I have also tested the Chinese variants. Do get in touch with me to get some recommended ones.
- **Analog Telephone Adapters:** Asterisk is a software system running on a digital computer and it cannot directly make use of analog phones. However, you can use devices called ATAs or Analog Telephone Adapters. ATAs understand IP and some VoIP protocol, say, SIP. It provides one or more RJ 11 connector(s) into which you can plug in one or more ordinary analog phone machines, depending on how many lines the ATA provides. ATAs can be configured using your computer's browser, since ATAs usually have a web application running which lets you configure it.

Soft phones are entities which exist in software. You can hold conversations using soft phones with help of a headset (headphone and mic). These are the cheaper alternative if you have a computer available in places

Asterisk, the easy way

which you want to connect using your PBX. Out of all the soft phones I tested, I found CounterPath's X-lite to be the best free soft phone. You can download it from their website for your operating system. You can download X-lite for Linux, Windows and MacOS from the [Xten website](#) for free.

I will be using SIP as my protocol of choice for my extensions and IAX as the protocol of choice for Asterisk interconnection.

I assume that your Asterisk machine has the IP address 192.168.0.99 and that you have configured your firewall in order to unblock all traffic on SIP, RTP and IAX ports. Usually these port numbers are port 5060 to 5062 for SIP, 8000 to 20000 for RTP and port 4569 for IAX.

Configuring your hard/soft phones and ATAs

To configure your hard/soft phones and ATAs, you need to open their configuration interface (the IP address of the ATA if you are using an ATA). You need to put the following details wherever they are asked for:

1. Domain: Put your Asterisk machine's IP address here, for example 192.168.0.99.
2. SIP proxy: Put your Asterisk machine's IP address colon separated with your Asterisk's SIP port. For example, 192.168.0.99:5060.
3. Outbound SIP proxy: Put the SIP proxy for making outbound calls here. Usually this is same as the SIP proxy. For example 192.168.0.99:5060.
4. User name/Authorization user: Put the extension that you want to assign to the phone, provided that this matches the extension you will create while configuring Asterisk.
5. Password: The corresponding password of the user name that you entered.

It is important that you correctly configure your hard phones for them to register with your Asterisk machine.

Connecting to Asterisk Command Line Interface (CLI): Asterisk CLI helps you monitor Asterisk. To connect to it, go to your Linux shell and type...

```
# asterisk -r
```

If Asterisk is running you should get a prompt which looks like:

```
hostname*CLI>
```

So, in this case:

```
asterisk2*CLI>
```

You can stop Asterisk by typing at the Asterisk CLI:

```
asterisk2*CLI> stop gracefully
```

You can start Asterisk by typing at your Linux shell:

```
# asterisk -vvvvc&
```

Asterisk, the easy way

The above will start Asterisk with verbosity level 4 and connect you to the CLI. To re-read the configuration files or “reload” Asterisk, type at the command line...

```
# asterisk2*CLI> reload
```

After you start making changes to Asterisk’s configuration files, you may be required to refresh Asterisk for the changes to take effect. To do this, you can type “reload” at the Asterisk CLI. This should be enough for most changes to take effect. But at the time of major changes such as creating new extensions, I strongly recommend that you shut down Asterisk and start it again. In such cases, you type:

```
asterisk2*CLI> stop now
```

at the Asterisk CLI and type:

```
# asterisk -vvvvc
```

at the Linux shell prompt to start Asterisk again and reconnect to the CLI.

What is a dialplan?

A dialplan is the logic that instructs Asterisk how to handle calls. The Asterisk dialplan exists purely in software and is predominantly written in the file `extensions.conf`.

For instance, in the file `extensions.conf`:

```
[incoming]
exten => 1001,1,Dial(SIP/1001,18)
exten => 1001,2,Congestion()
exten => 1001,102,Busy()
```

Contexts—the file `extensions.conf` consists of blocks called “contexts”, indicated by square brackets `[<context name>]` and statements that start with `exten=>`. In the above example, the context “incoming” has a match in the file `sip.conf`, in the sense that there has to be extension 1001 in `sip.conf` that has the context “incoming” designated to it. This will direct Asterisk to send all the calls coming to SIP/1001 (sip extension 1001) to the extension “incoming” in `extensions.conf`.

Example (in the file `sip.conf`):

```
[1001]
user = 1001
type = friend
secret = 1234
host = dynamic
callerid = 1001
context = from-sip-internal
```

Priorities—priorities are simply the ordering integers (or labels) that the `exten=>` statements have in order to decide the order of execution of the statements. In the first example, the ‘1’, ‘2’, and ‘102’ in the `exten=>`

statements are nothing but priorities.

Applications—the expressions of the form that you see in `Dial()` are calls to applications. When applications are called, they perform functions such as dial an extension, and connect it to the dialling channel, as in the above case. In a way, applications enable Asterisk to do different things after Asterisk has finished resolving the logic.

So this is how you instruct Asterisk to handle incoming calls.

But what about the calls going out? And what about the IVR system? For this, you need to build a simple customised dialplan that addresses all these questions. I will now look at this dialplan, and try to help you understand it in the simplest manner possible.

Our dialplan will:

- Create an intercom system to enable calls between internal callers.
- Greet external callers with a welcome message and direct the caller on the basis of the response, thus building a preliminary Interactive Voice Response (IVR) system.
- Provide outbound calling to the telephone system (assuming that you have a Digium OEM X100P or equivalent card installed and plugged with the telephone line).
- Provide voicemail facility.

The prerequisites

Here I'll make use of the following parts of the Asterisk structure on the file system:

- The SIP configuration file: `/etc/asterisk/sip.conf`
- The extensions file (dialplan): `/etc/asterisk/extensions.conf`
- The sounds directory: `/var/lib/asterisk/sounds`

If you are looking for an X100P card, you can find it [here](#), which is a thread dedicated to finding the hardware in India.

Creating SIP extensions

Let's dive into the configuration. Edit the file `/etc/asterisk/sip.conf`:

```
pico /etc/asterisk/sip.conf
```

Here is the content:

```
;cut from here
; sip.conf - SIP configuration file
[general]

port=5060           ; Port to bind to (SIP is 5060)
bindaddr=0.0.0.0   ; Address to bind to (all addresses on machine)
disallow=all ; First, disallow all codecs, then allow codecs one by one
allow=ulaw
```

Asterisk, the easy way

```
allow=alaw
context = from-sip-external ; Send unknown SIP callers to this context
callerid = Home
nat=yes ; Important if your Asterisk server and extensions are behind NAT
qualify=yes
canreinvite=no
dtmfmode=rfc2833

[101]
user=101
type=friend
secret=1234
host=dynamic
callerid = User-1
context=from-sip-internal

[102]
user=102
type=friend
secret=1234
host=dynamic
callerid = User-2
context=from-sip-internal

[103]
user=103
type=friend
secret=1234
host=dynamic
callerid = User-3
context=from-sip-internal

[104]
user=104
type=friend
secret=1234
host=dynamic
callerid = User-4
context=from-sip-internal

; upto here
```

I have created 4 SIP extensions above.

Make sure to restart Asterisk in order for these changes to take effect.

Here:

- The “user” identifier is assigned to the extension that you wish to use with your phone.
- The “type” identifier tells you whether the user is a “user” (takes incoming calls), “peer” (makes outgoing calls) or a “friend” (who does both).
- The “secret” identifier is assigned the password for authenticating the user—this has to be accurately entered in the phone’s configuration settings for it to log on to the Asterisk server.
- The “host” identifier tells Asterisk what kind of host you are dealing with and the value “dynamic” informs Asterisk that this host will register with our server.
- The “callerid” identifier is the caller identification presentation string that you want to be seen for this caller.

Asterisk, the easy way

- The “context” identifier is the most important since this is what ties up the Asterisk SIP user with the dialplan in `extensions.conf` as mentioned above. Here, you have the value “from-sip-internal” assigned to “context”. This means that when Asterisk receives a call for 1001, it will look in the context “from-sip-internal” for the action to be performed. If it does not find the necessary context, it will simply exit with a “404 Not Found” code (unless you have configured it to handle such cases using a wildcard, which will be discussed at a later stage). So be careful. The SIP callers who are not registered with your server (that is, the external callers), are directed to the appropriate context based on the information provided in the general context in the `sip.conf` file—look at the sample given here and the rest is self-explanatory. Comments are denoted by a ‘;’ symbol—everything to the right of a semicolon is a comment.

You’ll need your hard/soft phones or ATAs, as mentioned earlier, properly configured and online in order to use the extensions that you have created.

Remember to reload Asterisk so that the changes made to `sip.conf` can take effect.

Go to the Asterisk command line interface (type `asterisk -r` from your Linux shell) and type:

```
asterisk2*CLI> sip reload
asterisk2*CLI> sip show peers
```

The extensions should get listed with their IP addresses if they have been configured properly.

Now that you have everything working, here is the dialplan (run `pico /etc/asterisk/extensions.conf`):

```
; extensions.conf#x2014;the extensions configuration file

[globals]
OUTBOUNDTRUNK=ZAP/1 ; outbound calling trunk
INTERCOM=SIP ; protocol for the intercom
IVR-OPERATOR=0 ; IVR key for the operator
OPERATOR=501 ; actual extension of the operator

[incoming]
include => internal

[from-pstn]
include => intercom
exten => s,1,Playback(silence/1)
exten => s,2,Background(welcome)
exten => s,3,Background(enter-ext-of-person)
exten => s,4, Background(or-press)
exten => s,5, SayDigits(${ IVR-OPERATOR})
exten => s,4, Background(to-reach-operator)

[intercom]
exten => _5XX,1,Macro(dial,${INTERCOM},${EXTEN})
exten => ${IVR-OPERATOR},1,Macro(dial,${INTERCOM},${OPERATOR})
exten => 500,1,VoiceMailMain() ; Enabling voicemail at extension 500
include => rules

[internal]
include => intercom
```

Asterisk, the easy way

```
include => outbound-local

[outbound-local]
exten => _9XXXXXXXX,1,Macro(dial,${OUTBOUNDTRUNK},${EXTEN}) ; Mobile numbers
exten => _2XXXXXXXX,1,Macro(dial,${OUTBOUNDTRUNK},${EXTEN}) ; MTNL
exten => _3XXXXXXXX,1,Macro(dial,${OUTBOUNDTRUNK},${EXTEN}) ; Reliance
exten => _5XXXXXXXX,1,Macro(dial,${OUTBOUNDTRUNK},${EXTEN}) ; Tata
exten => _0.,1,Macro(dial,${OUTBOUNDTRUNK},${EXTEN}) ; STD / ISD numbers

; Emergency and special numbers
exten => _1.,1,Macro(dial,${OUTBOUNDTRUNK},${EXTEN})

[macro-dial]
exten => s,1,Dial(${ARG1}/${ARG2})
exten => s,2,Goto(s-${DIALSTATUS},1)
exten => s,102,Goto(s-${DIALSTATUS},1)
exten => s-NOANSWER,1,Voicemail(u${ARG2}@default)
exten => s-BUSY,1,Voicemail(b${ARG2}@default)
exten => s-CHANUNAVAIL,1,Playback(pbx-invalid)

[rules]
exten => t,1,Playback(connection-timed-out)
exten => t,n,Playback(hangup-try-again)
exten => t,n,Hangup()

exten => i,1,Playback(pbx-invalid)

; cut upto here
```

In the above dialplan:

- The “globals” context—this is where global variable declarations are done in Asterisk. The variable assignment in the “globals” section is pretty simple—just variable names, an equals symbol and the value.
- The “include” statement tells Asterisk to literally include the code in the context that follows the “include” keyword.
- The `Playback()` application plays the file whose name it takes as an argument. These files are in GSM format—note that you have to exclude the `.gsm` extension of the filename.
- `Background()` works similarly; `Background()` detects key presses and interrupts the playback to return the code of the key pressed (usually DTMF) to Asterisk; on the other hand, `Playback()` ignores any key presses and returns control to Asterisk only after it has finished playing a file. Note that DTMF means Dual Tone Multi Frequency—these are the tones that our modern telephone’s touch-tone dial pads produce.
- The `SayDigits()` application does precisely what its name suggests— it plays the digits of the number that it takes as an argument.
- The `Dial()` application is more difficult to explain than the previous applications mentioned. So, it deserves a section in its own rights.

The `Dial()` application

`Dial()` dials the channel that it takes as an argument and bridges it with the active channel (the calling channel in the case of our dialplan). The `Dial()` application takes two main parameters as arguments (I will not cover the optional ones here)— the channel to dial and the time for which it has to keep trying before exiting. `Dial()` sets a certain environment variable `DIALSTATUS` on exit, which you can make use of in your dialplan. `DIALSTATUS` is set to `<extension> -NOANSWER`, if the called line does not answer, and

Asterisk, the easy way

<extension> -BUSY, if the called line is busy. It sets DIALSTATUS to <extension> -CHANUNAVAIL in the event that the channel does not exist. Besides this method, another behaviour of Dial () can be used, in which it takes control to the next priority in case the called line does not answer and to the Dial's current priority + 101 if the called line is busy. There is one more property of Dial () that you should know about. It takes the control to extension 'i' (invalid) when invalid extensions are dialed and extension 't' (timeout) when the caller takes an abnormally long time to respond. You may notice the \${<variable name>} notation used in many places—this is how you can recall values stored in variables. Wherever such notations are encountered, Asterisk simply replaces the value of the variable between the curly brackets. You must have also noticed that some notations in the dialplan begin with an underscore, '_'. The underscore informs Asterisk that the notation following it uses wildcards. Wildcards are symbols that can replace one or more, other symbols. In Asterisk, wildcard symbols that are present are listed below:

- X—Matches any digit from 0 to 9.
- Z—Matches any digit from 1 to 9.
- N—Matches any digit from 2 to 9.
- [15-7]—Matches any digit or range of digits specified. In this case, matches a 1, 5, 6, or 7.
- . (period)—Wildcard match; matches one or more characters.

For instance, the expression `_X` will match all one-digit numbers from 0 to 9. `_2XXXXXXX` will match all numbers beginning with the digit 2 (like our MTNL Mumbai and Delhi landline numbers) and `_.` will match everything.

A word of caution here—make use of wildcards very judiciously and only after you are certain of the outcome.

I have also made use of forms of code called “macros”. An Asterisk macro is a set of statements that can be invoked or called from anywhere in the dialplan. Macros are written in sections just like contexts and their names have to be preceded by the prefix `macro-`. Macros can take arguments. The macro arguments are denoted by `ARG<num>` where `num` is replaced by the argument number that is serially assigned by Asterisk, in the order of passing. Macros are called using the `Macro()` application.

Priorities in the `exten=>` statements in the file `extensions.conf` are auto-incremented if the following priority is indicated by the symbol 'n' instead of a number. This saves the trouble of keeping track of priority numbers and manually changing all following priority numbering in case you change intermediate code. The extensions can also have labelled priorities—for instance, `exten=> <some extension>, n(<label>), <do something>`. Labelling extensions not only saves a lot of effort in case of changes, but also helps keep the code readable.

Voicemail

Voicemail users are defined in the file `voicemail.conf`. Here too the file is sub-divided into sections:

```
;cut from here
[default]
101 => 1234,User-1,user-1@somedomain.com
102 => 1234,User-2,user-2@somedomain.com
103 => 1234,User-3,user-3@somedomain.com
104 => 1234,User-4,user-4@somedomain.com
;upto here
```

Asterisk, the easy way

I will only use and cover the “default” context of the voicemail file. The `voicemail.conf` file’s statements are of the form:

```
mailbox => password,name[,email[,pager_email[,options] ] ]
```

I have made use of the `VoiceMail()` and `VoiceMailMain()` applications in the example dialplan in `extensions.conf` to incorporate voicemail. The `VoiceMailMain()` application does not take arguments, but provides prompts for users to enter a username/password combination to access their voicemail and to customise their voicemail response with something like a custom greeting. The `VoiceMail()` application takes the mailbox details as argument, which is in the form:

```
VoiceMail(<prefix><voicemail number>@<voicemail context>)
```

The “prefix” is an optional symbol that indicates to the `VoiceMail()` application which message it should play. If you supply the prefix ‘b’, Asterisk will play the “person busy” message, and if you supply ‘u’, Asterisk will play the “person unavailable” message.

Conclusion

This should provide you with adequate information to read through the provided dialplan and understand it. You can look up more information on Asterisk on the internet—to help you understand this dialplan better and to build you own dialplans. The [VoIP-info website](#) is a mammoth resource on Asterisk and VoIP, and has exhaustive material on the mentioned topics. The [Asterisk Book resource](#) has a small concise task-oriented tutorial on Asterisk, which covers the basics; it also points you to useful resources on the web to accomplish the tasks at hand.

Biography

[Mitul Limbani](#) (/user/26845" title="View user profile.): We have expertise in Asterisk installation, Asterisk configuration, Customized Asterisk Development, Dialplan programming, AGI scripting, manager api, IVR Designing. We have installed,configured and customized various opensource solutions based on Asterisk such as Vicidial , A2Billing , AstBill , FreePBX , TrixBox, etc. We have expertise in PHP,MySQL, AJAX as well. Visit us today at : <http://www.asteriskatoffice.com/> Send us your query on : <http://www.asteriskatoffice.com/contact>

Copyright information

Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

Source URL:

http://www.freesoftwaremagazine.com/articles/asterisk_the_easy_way
