CHAPTER 8
Voicemail

Just leave a message, maybe I'll call.

—Joe Walsh

Before email and instant messaging became ubiquitous, voicemail was a popular method of electronic messaging. Even though most people prefer text-based messaging systems, voicemail remains an essential component of any PBX.

Comedian Mail

One of the most popular (or, arguably, unpopular) features of any modern telephone system is voicemail. Asterisk has a reasonably flexible voicemail system named Comedian Mail.[1] Voicemail in Asterisk is provided in the dialplan by the *app_voicemail.so* module.

Some of the features of Asterisk's voicemail system include:

- Unlimited password-protected voicemail boxes, each containing mailbox folders for organizing voicemail
- Different greetings for busy and unavailable states
- Default and custom greetings
- The ability to associate phones with more than one mailbox and mailboxes with more than one phone
- Email notification of voicemail, with the voicemail optionally attached as a sound file[2]

1. This name was a play on words, inspired in part by Nortel's voicemail system Meridian Mail.
2. No, you really don't have to pay for this—and yes, it really does work.

- Voicemail forwarding and broadcasts
- Message-waiting indicator (flashing light or stuttered dialtone) on many types of phones
- Company directory of employees, based on voicemail boxes

And that's just the tip of the iceberg!

The default version of the */etc/asterisk/voicemail.conf* configuration file requires a few tweaks in order to provide a configuration that will be suitable to most situations.

We'll begin by going through the various options you can define in *voicemail.conf*, and then we'll provide a sample configuration file with the settings we recommend for most deployments.

The *voicemail.conf* file contains several sections where parameters can be defined. The following sections detail all the options that are available.

The [general] Section

The first section, `[general]`, allows you to define global settings for your voicemail system. The available options are listed in Table 8-1.

Table 8-1. [general] section options for voicemail.conf

Option	Value/Example	Notes
`format`	`wav49|gsm|wav`	For each format listed, Asterisk creates a separate recording in that format whenever a message is left. The benefit is that some transcoding steps may be saved if the stored format is the same as the codec used on the channel. We like WAV because it is the highest quality, and WAV49 because it is nicely compressed and easy to email. We don't like GSM due to its scratchy sound, but it enjoys some popularity.[a]
`serveremail`	`user@domain`	When an email is sent from Asterisk, this is the email address that it will appear to come from.[b]
`attach`	`yes,no`	If an email address is specified for a mailbox, this determines whether the message is attached to the email (if not, a simple message notification is sent).
`maxmsg`	`9999`	By default, Asterisk only allows a maximum of 100 messages to be stored per user. For users who delete messages, this is no problem. For people who like to save their messages, this space can get eaten up quickly. With the size of hard drives these days, you could easily store thousands of messages for each user, so our current thinking is to set this to the maximum and let the users manage things from there.

Option	Value/Example	Notes
`maxsecs`	`0`	This type of setting was useful back when a large voicemail system might have only 40 MB[c] of storage: it was necessary to limit the system because it was easy to fill up the hard drive. This setting can be annoying to callers (although it does force them to get to the point, so some people like it). Nowadays, with terabyte drives common, there is no technical reason to limit the length of a message. Two considerations are: 1) if a channel gets hung in a mailbox, it's good to set some sort of value so it doesn't stay there for days, but 2) if a user wants to use her mailbox to record notes to herself, she won't appreciate it if you cut her off after 3 minutes. A setting somewhere between `600` seconds (10 minutes) and `3600` seconds (1 hour) will probably be about right.
`minsecs`	`4`	Many folks will hang up instead of leaving a message when they call somebody and get voicemail. Sometimes this hangup happens after recording has started, so the mailbox owner gets an annoying 2-second message of somebody hanging up. This setting ensures that Asterisk will ignore messages that are shorter than the configured minimum length. You should take care not to set this to a value that is too high, though, because then a message like "Hey it's me, give me a call" (which can be said in less than 1 second) will get lost, and you'll get complaints of messages disappearing. Three seconds seems to be about right. To discourage people from leaving ultrashort messages that might be discarded, your greeting can request callers to identify themselves and leave some information about why they called.
`maxgreet`	`1800`	You can define the maximum greeting length if you want. Again, since storage is not a problem and setting this too low will annoy your more verbose users, we suggest setting this to a high value and letting your users figure out an appropriate length for themselves.
`skipms`	`3000`	When listening to messages, users can skip ahead or backwards by pressing (by default) `*` and `#`. This setting indicates the length of the jump (in milliseconds).
`maxsilence`	`5`	This setting defines the maximum time for which the caller can remain silent before the recording is stopped. We like to set this setting to 1 second longer than `minsecs` (if you set it equal to or greater than `minsecs`, you will get a warning stating "maxsilence should be less than minsecs or you may get empty messages"). This value is probably most useful when you have analog trunks, as far-end disconnect can be dodgy on such circuits. With any sort of digital circuit (PRI or VoIP), a far-end disconnect message will tend to handle the end of the message quite neatly, and in fact you might want to consider increasing this value to 10 seconds, to reduce the chance that quiet folks accidentally get disconnected midmessage.

Option	Value/Example	Notes
silencethreshold	128	This parameter allows you to fine-tune the silence sensitivity of the previous parameter, maxsilence. Valid values are from 0 to 32767. The default value is 128. The value helps app_voicemail decide what amplitude to use as a reference point as to what it will consider "silence." Since this is a linear value, and amplitude needs to be properly considered in terms of decibels (which are logarithmic), we don't recommend adjusting this parameter unless you have a solid understanding of amplitude, decibels, logarithmic scales, C programming, digital signal processing, and so forth. If you have audio problems with your system, this is not the first place to go to attempt to solve them.
maxlogins	3	This little security feature is intended to make brute-force attacks on your mailbox passwords more time-consuming. If a bad password is received this many times, voicemail will hang up and you'll have to call back in to try again. Note that this will not lock up the mailbox. Patient snoopers can continue to try to log into your mailbox as many times as they like, they'll just have to call back every third attempt. If you have a lot of sausage-fingered users, you can set this to something like 5.
moveheard	yes	This setting will move listened-to messages to the *Old* folder. We recommend leaving this at the default.
forward_urgent_auto	no	Setting this to yes will preserve the original urgency setting of any messages the user receives and then forwards on. If you leave it at no, users can set the urgency level themselves on messages that they forward.
userscontext	default	If you use the *users.conf* file (we don't), you can define here the context where entries are registered.
externnotify	*/path/to/script*	If you wish to run an external app whenever a message is left, you can define it here.
smdienable	no	If you are using Asterisk as a voicemail server on a PBX that supports SMDI, you can enable it here.
smdiport	*/dev/ttyS0*	Here is where you would define the SMDI port that messages between Asterisk and the external PBX would pass across.
externpass	*/path/to/script*	Any time the password on a mailbox is changed, the script you define here will be notified of the context, mailbox, and new password. The script will then be responsible for updating *voicemail.conf* (the Asterisk voicemail app will not update the password if this parameter is defined).
externpassnotify	*/path/to/script*	Any time the password on a mailbox is changed, the script you define here will be notified of the context, mailbox, and new password. Asterisk will handle updating the password in *voicemail.conf*. If you have defined externpass, this option will be ignored.
externpasscheck	*/usr/local/bin/voicemailpwcheck.py*	See the sidebar following this table for a description of this option.
directoryintro	dir-intro	The Directory() dialplan application uses the *voicemail.conf* file to search by name from an auto attendant. There is a default prompt that plays, called dir-intro. If you want, you can specify a different file to play instead.

Option	Value/Example	Notes
`charset`	`ISO-8859-1`	If you need a character set other than ISO-8859-1 (a.k.a Latin 1) to be supported, you can specify it here.
`adsifdn`	`0000000F`	Use this option to configure the Feature Descriptor Number.[d]
`adsisec`	`9BDBF7AC`	Use this option to configure the security lock code.
`adsiver`	`1`	This specifies the ADSI voicemail application version number.
`pbxskip`	`yes`	If you do not want emails from your voicemail to have the string [PBX] added to the subject, you can set this to `yes`.
`fromstring`	`The Asterisk PBX`	You can use this setting to configure the `From:` name that will appear in emails from your PBX.
`usedirectory`	`yes`	This option allows users composing messages from their mailboxes to take advantage of the Directory.
`odbcstorage`	*`<item from res_odbc.conf>`*	If you want to store voice messages in a database, you can do that using the Asterisk `res_odbc` connector. Here, you would set the name of the item in the *res_odbc* file. For details, see Chapter 22.
`odbctable`	*`<table name>`*	This setting specifies the table name in the database that the `odbcstorage` setting refers to. For details, see Chapter 22.
`emailsubject`	`[PBX]: New message ${VM_MSGNUM} in mailbox ${VM_MAILBOX}`	When Asterisk sends an email, you can use this setting to define what the `Subject:` line of the email will look like. See the *voicemail.conf.sample* file for more details.
`emailbody`	`Dear ${VM_NAME}:\n\n\tjust wanted to let you know you were just left a ${VM_DUR} long message (number ${VM_MSGNUM})\nin mailbox ${VM_MAILBOX u might\nwant to check it when you get a chance. Thanks!\n\n\t\t\t--Asterisk\n`	When Asterisk sends an email, you can use this setting to define what the body of the email will look like. See the *voicemail.conf.sample* file for more details.

Option	Value/Example	Notes
`pagerfromstring`	`The Asterisk PBX`	We don't actually know anybody who uses pagers anymore (nor can we recall having seen one in many years), but if you have one of these historical oddities and you want to customize what Asterisk sends with its pager notification, presumably you can do that with this. A very practical application of this feature for short-message voicemail notifications is to send a message to an email-to-SMS gateway.
`pagersubject`	`New VM`	As above.
`pagerbody`	`New ${VM_DUR} long msg in box ${VM_MAILBOX}\nfrom ${VM_CALLERID}, on ${VM_DATE}`	The formatting for this uses the same rules as `emailbody`.
`emaildateformat`	`%A, %d %B %Y at %H:%M:%S`	This option allows you to specify the date format in emails. Uses the same rules as the C function `STRFTIME`.
`pagerdateformat`	`%A, %d %B %Y at %H:%M:%S`	This option allows you to specify the date format in `pagerbody`. Uses the same rules as the C function `STRFTIME`.
`mailcmd`	`/usr/sbin/sendmail -t`	If you want to override the default operating system application for sending mail, you can specify it here.
`pollmailboxes`	`no, yes`	If the contents of mailboxes are changed by anything other than `app_voicemail` (such as external applications or another Asterisk system), setting this to `yes` will cause `app_voicemail` to poll all the mailboxes for changes, which will trigger proper message waiting indication (MWI) updates.
`pollfreq`	`30`	Used in concert with `pollmailboxes`, this option specifies the number of seconds to wait between mailbox polls.
`imapgreetings`	`no, yes`	This enables/disables remote storage of greetings in the IMAP folder. For more details, see Chapter 18.
`greetingsfolder`	`INBOX`	If you've enabled `imapgreetings`, this parameter allows you to define the folder your greetings will be stored in (defaults to `INBOX`).
`imapparentfolder`	`INBOX`	IMAP servers can handle parent folders in different ways. This field allows you to specify the parent folder for your mailboxes. For more details, see Chapter 7.
`imapserver`	`localhost`	Defines the IMAP server Asterisk should connect to.
`imapport`	`143`	Defines the port of the IMAP server to connect to.
`imapflags`	`ssl`	IMAP servers typically have different flags that can be passed along with the mailbox name. A common flag to pass is `ssl`, which enables OpenSSL encryption in the communication if the IMAP libraries were compiled with OpenSSL support.
`imapfolder`	`INBOX`	The folder to store voicemail messages in on the IMAP server. The default is `INBOX`.

Option	Value/Example	Notes
`authuser`	`user`	If your IMAP server has been defined with an account that can access all mailboxes, you can define that user for Asterisk to connect to the server with.
`authpassword`	`password`	This option defines the password to be used with the `authuser` attribute.
`imapopentimeout`	`60`	The TCP open timeout in seconds.
`imapclosetimeout`	`60`	The TCP close timeout in seconds.
`imapreadtimeout`	`60`	The TCP read timeout in seconds.
`imapwritetimeout`	`60`	The TCP write timeout in seconds.

[a] The separator that is used for each format option must be the pipe (`|`) character.

[b] Sending email from Asterisk can require some careful configuration, because many spam filters will find Asterisk messages suspicious and will simply ignore them. We talk more about how to set email for Asterisk in Chapter 18.

[c] Yes, you read that correctly: megabytes.

[d] The Analog Display Services Interface is a standard that allows for more complex feature interactions through the use of the phone display and menus. With the advent of VoIP telephones, ADSI's popularity has decreased in recent years.

External Validation of Voicemail Passwords

By default, Asterisk does not validate user passwords to ensure they are at least somewhat secure. Anyone who maintains voicemail systems will tell you that a large percentage of mailbox users set their passwords to something like `1234` or `1111`, or some other string that's easy to guess. This represents a huge security hole in the voicemail system.

Since the *app_voicemail.so* module does not have the built-in ability to validate passwords, the settings `externpass`, `externpassnotify`, and `externpasscheck` allow you to validate them using an external program. Asterisk will call the program based on the path you specify, and pass it the following arguments:

```
mailbox context oldpass newpass
```

The script will then evaluate the arguments based on rules that you defined in the external script and, based on your rules, it should return to Asterisk a value of `VALID` for success or `INVALID` for failure (actually, the return value for a failed password can be anything except the words `VALID` or `FAILURE`). This value is typically printed to *stdout*. If the script returns `INVALID`, Asterisk will play an invalid-password prompt and the user will need to attempt something different.

Ideally, you would want to implement rules such as the following:

- Passwords must be a minimum of six digits in length
- Passwords must not be strings of repeated digits (e.g., `111111`)
- Passwords must not be strings of contiguous digits (e.g., `123456` or `987654`)

Asterisk comes with a simple script that will greatly improve the security of your voicemail system. It is located in the source code under the folder: */contrib/scripts/voicemailpwcheck.py*.

We strongly recommend that you copy it to your */usr/local/bin* folder (or wherever you prefer to put such things), and then uncomment the `externpasscheck=` option in your *voicemail.conf* file. Your voicemail system will then enforce the password security rules you have established.

Part of the `[general]` section is an area that is referred to as *advanced options*. These options (listed in Table 8-2) are defined in the same way as the other options in the `[general]` section, but they can also be defined on a per-mailbox basis, which would override whatever is defined under `[general]` for that particular setting.

Table 8-2. Advanced options for voicemail.conf

Option	Value/Example	Notes
`tz`	`eastern`, `european`, etc.	Specifies the `zonemessages` name, as defined under `[zonemessages]` (discussed in the next section).
`locale`	`de_DE.utf8`, `es_US.utf8`, etc.	Used to define how Asterisk generates date/time strings in different locales. To determine the locales that are valid on your Linux system, type **`locale -a`** at the shell.
`attach`	`yes`, `no`	If an email address is specified for a mailbox, this determines whether the messages are attached to the email notifications (otherwise, a simple message notification is sent).
`attachfmt`	`wav49`, `wav`, etc.	If `attach` is enabled and messages are stored in different formats, this defines which format is sent with the email notifications. Often `wav49` is a good choice, as it uses a better compression algorithm and thus will use less bandwidth.
`saycid`	`yes`, `no`	This command will state the caller ID of the person who left the message.
`cidinternalcontexts`	*`<context>`*, *`<another context>`*	Any dialplan contexts listed here will be searched in an attempt to locate the mailbox context, so that the name associated with the mailbox number can be spoken. The voicemail box number needs to match the extension number that the call came from, and the voicemail context needs to match the dialplan context.[a]
`sayduration`	`yes`, `no`	This command will state the length of the message.
`saydurationm`	`2`	Use this to specify the minimum duration of a message to qualify for its length being played back. For example, if you set this to `2`, any message less than 2 minutes in length will not have its length stated. (The option really is `saydurationm` and isn't a typo here.)

Option	Value/Example	Notes
`dialout`	*`<context>`*	If allowed, users can dial out from their mailboxes. This is considered a very dangerous feature in a phone system (mainly because many voicemail users like to use `1234` as their password), and is therefore not recommended. If you insist on allowing this, make sure you have a second level in the dialplan where another password is specified. Even so, this is not a safe practice.
`sendvoicemail`	`yes`, `no`	This allows users to compose messages to other users from within their mailboxes.
`searchcontexts`	`yes`, `no`	This allows voicemail applications in the dialplan to not have to specify the voicemail context, since all contexts will be searched. This is not recommended.
`callback`	*`<context>`*	This specifies which dialplan context to use to call back to the sender of a message. The specified context will need to be able to handle dialing of numbers in the format in which they are received (for example, the country code may not be received with the caller ID, but might be required for the outgoing call).
`exitcontext`	*`<context>`*	There are options that allow the callers to exit the voicemail system when they are in the process of leaving a message (for example, pressing `0` to get an operator). By default, the context the caller came from will be used as the exit context. If desired, this setting will define a different context for callers exiting the voicemail system.
`review`	`yes`, `no`	This should almost always be set to `yes` (even though it defaults to `no`). People get upset if your voicemail system does not allow them to review their messages prior to delivering them.
`operator`	`yes`, `no`	Best practice dictates that you should allow your callers to "zero out" from a mailbox, should they not wish to leave a message. Note that an `o` extension (not "zero," "oh") is required in the `exitcontext` in order to handle these calls.
`envelope`	`no`, `yes`	You can have voicemail play back the details of the message before it plays the actual message. Since this information can also be accessed by pressing `5`, we generally set this to `no`.
`delete`	`no`, `yes`	After an email message notification is sent (which could include the message itself), the message will be deleted. This option is risky, because the fact that a message was emailed is not a guarantee that it was received (spam filters seem to love to delete Asterisk voicemail messages). On a new system, leave this at `no` until you are certain that no messages are being lost due to spam filters.
`volgain`	`0.0`	This setting allows you to increase the volume of received messages. Volume used to be a problem in older releases of Asterisk, but has not been an issue for many years. We recommend leaving this at the default. The *sox* utility is required for this to work.

Option	Value/Example	Notes
`nextaftercmd`	`yes`, `no`	This handy little setting will save you some time, as it takes you directly to the next message once you've finished dealing with the current message.
`forcename`	`yes`, `no`	This strange little setting will check whether the mailbox password is the same as the mailbox number. If it is, it will force the user to change his voicemail password and record his name.
`forcegreetings`	`yes`, `no`	As above, but for greetings.
`hidefromdir`	`no`, `yes`	If you wish, you can hide specific mailboxes from the `Directory()` application using this setting.
`tempgreetwarn`	`yes`, `no`	Setting this to `yes` will warn the mailbox owner that she has a temporary greeting set. This can be a useful reminder when people return from trips or vacations.
`passwordlocation`	`spooldir`	If you want, you can have mailbox passwords stored in the spool folder for each mailbox.[b] One of the advantages of using the `spooldir` option is that it will allow you to define file `#include` statements in *voicemail.conf* (meaning you can store mailbox references in multiple files, as you can with, for example, dialplan code). This is not possible otherwise, because *app_voicemail* normally writes password changes to the filesystem, and cannot update a mailbox password stored outside of either *voicemail.conf* or the spool. If you do not use `passwordlocation`, you will not be able to define mailboxes outside of *voicemail.conf*, since password updates will not happen. Storing passwords in a file in the specific mailbox folder in the spool solves this problem.
`messagewrap`	`no`, `yes`	If this is set to `yes`, when the user has listened to the last message, pressing next (6) will take him to the first message. Also, pressing previous (4) when at the first message will take the user to the last message.
`minpassword`	`6`	This option enforces a minimum password length. Note that this does not prevent users from setting their passwords to something easy to guess (such as `123456`).
`vm-password`	`custom_sound`	If you want, you can specify a custom sound here to use for the password prompt in voicemail.
`vm-newpassword`	`custom_sound`	If you want, you can specify a custom sound here to use for the "Please enter your new password followed by the pound key" prompt in voicemail.
`vm-passchanged`	`custom_sound`	If you want, you can specify a custom sound here to use for the "Your password has been changed" prompt in voicemail.
`vm-reenterpassword`	`custom_sound`	If you want, you can specify a custom sound here to use for the "Please reenter your password followed by the pound key" prompt in voicemail.

Option	Value/Example	Notes
vm-mismatch	custom_sound	If you want, you can specify a custom sound here to use for the "The passwords you entered and reentered did not match" prompt in voicemail.
vm-invalid-password	custom_sound	If you want, you can specify a custom sound here to use for the "That is not a valid password. Please try again" prompt in voicemail.
vm-pls-try-again	custom_sound	If you want, you can specify a custom sound here to use for the "Please try again" prompt in voicemail.
vm-prepend-timeout	custom_sound	If you want, you can specify a custom sound here to use when a user times out while recording a prepend message. The default prompt is, "then press pound" and follows the `vm-pls-try-again` prompt.
listen-control-forward-key	#	You can use this setting to customize the fast-forward key.
listen-control-reverse-key	*	You can use this setting to customize the rewind key.
listen-control-pause-key	0	You can use this setting to customize the pause/unpause key.
listen-control-restart-key	2	You can use this setting to customize the replay key.
listen-control-stop-key	13456789	You can use this setting to customize the interrupt playback key.
backupdeleted	0	This setting will allow you to specify how many deleted messages are automatically stored by the system. This is similar to a recycle bin. Setting this to `0` disables this feature. Up to 9,999 messages can be stored, after which the oldest message will be erased each time another message is deleted.

[a] Yes, we found this a bit confusing, too.

[b] Typically the spool folder is */var/spool/asterisk*, and it can be defined in */etc/asterisk/asterisk.conf*.

The [zonemessages] Section

The next section of the *voicemail.conf* file is the `[zonemessages]` section. The purpose of this section is to allow time zone–specific handling of messages, so you can play back to the user messages with the correct timestamps. You can set the name of the zone to whatever you need. Following the zone name, you can define which time zone you want the name to refer to, as well as some options that define how timestamps are played back. You can look at the *~/src/asterisk-complete/asterisk/11/configs/voicemail.conf.sample* file for syntax details. Asterisk includes the examples shown in Table 8-3.

Table 8-3. [zonemessages] section options for voicemail.conf

Zone name	Value/Example	Notes
`eastern`	`America/New_York|'vm-received' Q 'digits/at' IMp`	This value would be suitable for the eastern time zone (EST/EDT).
`central`	`America/Chicago|'vm-received' Q 'digits/at' IMp`	This value would be suitable for the central time zone (CST/CDT).
`central24`	`America/Chicago|'vm-received' q 'digits/at' H N 'hours'`	This value would also be suitable for CST/CDT, but would play back the time in 24-hour format.
`military`	`Zulu|'vm-received' q 'digits/at' H N 'hours' 'phonetic/z_p'`	This value would be suitable for Universal Time Coordinated (Zulu time, formerly GMT).
`european`	`Europe/Copenhagen|'vm-received' a d b 'digits/at' HM`	This value would be suitable for Central European time (CEST).

The Contexts Section

All the remaining sections in the *voicemail.conf* file will be the voicemail contexts, which allow you to segregate groups of mailboxes.

In many cases, you will only need one voicemail context, commonly named `[default]`. This is worth noting, as it will make things simpler in the dialplan: all the voicemail-related applications assume the context `default` if no context is specified. In other words, if you don't require separation of your voicemail users, use `default` as your one and only voicemail context.

The format for the mailboxes is as follows (you should enter all of this on a single line):

```
mailbox => password[,FirstName LastName[,email addr[,pager addr
[,options[|options]]]]]
```

The pipe character (|) used to be more popular in Asterisk. For the first few years, it was used as the standard delimiter. More recently, it has almost completely been replaced by the comma; however, there are still a few places where the pipe is used. One of them is in *voicemail.conf*: for example, as a separator for any mailbox-specific options, and also as the separator character in the `format=` declarative. You'll see this in our upcoming example, as well as in the *voicemail.conf.sample* file.

The parts of the mailbox definition are:

`mailbox`
: This is the mailbox number. It usually corresponds with the extension number of the associated set.

`password`

This is the numeric password that the mailbox owner will use to access her voicemail. If the user changes her password, the system will update this field in the *voicemail.conf* file.

If the password is preceded by the hyphen (-) character, the user cannot change their mailbox password.

If you are storing passwords in the spool (by use of the `passwordlocation` parameter), this field is ignored. However, the parser still requires there to be a field here, so if you are going to specify any other options for this mailbox, a comma will be needed as a placeholder for the password field.

`FirstName LastName`

This is the name of the mailbox owner. The company directory uses the text in this field to allow callers to spell usernames.

`email address`

This is the email address of the mailbox owner. Asterisk can send voicemail notifications (including the voicemail message itself, as an attachment) to the specified email box.

`pager address`

This is the email address of the mailbox owner's pager or cell phone. Asterisk can send a short voicemail notification message to the specified email address.

`options`

This field is a list of options for setting the mailbox owner's time zone and overriding the global voicemail settings. There are quite a few valid options:

`tz`, `locale`, `attach`, `attachfmt`, `saycid`, `cidinternalcontexts`, `sayduration`, `saydurationm`, `dialout`, `sendvoicemail`, `searchcontexts`, `callback`, `exitcontext`, `review`, `operator`, `envelope`, `delete`, `volgain`, `nextaftercmd`, `forcename`, `forcegreeting`, `hidefromdir`, `tempgreetwarn`, `passwordlocation`, `messagewrap`, `minpassword`.

These options should be in *`option`* = *`value`* pairs, separated by the pipe character (`|`). See Table 8-4 for more details about what each of the options do.

The `tz` option sets the user's time zone to a time zone previously defined in the `[zonemessages]` section of *voicemail.conf*. The other options override the global voicemail settings with the same names.

Table 8-4. Mailbox options

Option	Description
`attach`	Whether to attach the voicemail to the notification email versus the pager email. If set to `yes`, will attach to the email defined by the email address field.
`attachfmt`	Sets the `format` to attach to the email. Normally this is the first value defined by the format option, but you can override that per mailbox by using this option. *Option can only be set per mailbox.*
`callback`	If defined, this option will allow the receiver of the email to call back the sender of the voicemail directly from the `Voicemail()` application. This option defines which context the call will be sent from. If not set, calling the sender back will not be permitted.
`cidinternalcon texts`	This is a very old option from 2004, but essentially, you can define multiple contexts (separated by a comma) that will tell Asterisk to check if the call came from an internal context. If so, it will play back the person's name recording instead of saying their extension number. It is unclear if this option is still valid or functional. Likely best used in the `[general]` section of voicemail rather than per mailbox.
`delete`	After sending the voicemail via email, the voicemail is deleted from the server. This option is useful for users who only want to receive voicemail via email. Valid options are `yes` or `no`. *Option can only be set per mailbox.*
`dialout`	If defined, option 4 from the advanced menu will allow you to dial out from the `VoicemailMain()` application. The argument defines which context the dialing will be performed from. If not defined, the option to dial out will not be prompted to the caller.
`envelope`	Turns on or off envelope playback prior to playback of the voicemail message. Valid options are `yes` or `no`. Default is `yes`.
`exitcontext`	The context to exit to when pressing `*` or `0` from the `Voicemail()` application. Works in conjunction with the `operator` option as well. Must have an extension `a` in the context for exiting with `*`. Must have an extension `o` in the context for exiting with `0`.
`forcegreeting`	Forces the recording of a greeting for new mailboxes. A new mailbox is determined by the mailbox number and password matching. Valid values are `yes` or `no`. Default is `no`.
`forcename`	Forces the recording of the person's name for new mailboxes. A new mailbox is determined by the mailbox number and password matching. Valid values are `yes` or `no`. Default is `no`.
`hidefromdir`	If set to `yes`, this mailbox will be hidden from the `Directory()` application. Default is `no`.
`locale`	Allows you to set the locale for the mailbox in order to control formatting of the date/time strings. See *voicemail.sample.conf* for more information.
`messagewrap`	Allows the first and last messages to wrap around; e.g., allow last message to wrap back to the first on the next message, or first message to wrap to the last message when going to the previous message. Valid options are `yes` or `no`. Default is `no`.
`minpassword`	Sets the minimum password length. Argument should be a whole number.
`nextaftercmd`	Skips to the next message after pressing the `7` key (delete) or `9` key (save). Valid values are `yes` or `no`. Default is `yes`.
`operator`	Will allow the sender of a voicemail to hit `0` before, during, or after recording of a voicemail. Will exit to the `o` extension in the same context, or the context defined by the `exitcontext` option. Valid options are `yes` or `no`. Default is `no`.

Option	Description
`passwordlocation`	By default, the password for voicemail is stored in the *voicemail.conf* file, and modified by Asterisk whenever the password changes. This may not be desirable, especially if you want to parse the password from an external location (or script). The alternate option for `passwordlocation` is `spooldir`, which will place the password for the voicemail user in a file called *secret.conf* in the user's voicemail spool directory. Valid options are `voicemail.conf` and `spooldir`. The default option if `voicemail.conf`.
`review`	When enabled, will allow the user recording a voicemail message to re-record their message. After pressing the # key to save their voicemail, they'll be prompted whether they wish to re-record or save the message. Valid options are `yes` or `no`. Default is `no`.
`saycid`	If enabled, and a prompt exists in */var/spool/asterisk/voicemail/recordings/callerids*, then that file will be played prior to the message, playing the file instead of saying the digits of the callerID number. Valid options are `yes` or `no`. Default is `no`.
`sayduration`	Determines whether to play the duration of the message prior to message playback. Valid options are `yes` or `no`. Default is `yes`.
`saydurationm`	Allows you to set the minimum duration to play (in minutes). For example, if you set the value to `2`, you will not be informed of the message length for messages less than 2 minutes long. Valid values are whole numbers. Default is `2`.
`searchcontexts`	For applications such as `Voicemail()`, `VoicemailMain()`, and `Directory()`, the voicemail context is an optional argument. If the voicemail context is not specified, then the default is to only search the `default` context. With this option enabled, all contexts will be searched. This comes with a caveat that, if enabled, the mailbox number must be unique across all contexts—otherwise there will be a collision, and the system will not understand which mailbox to use. Valid options are `yes` and `no`. Default is `no`.
`sendvoicemail`	Allows the user to compose and send a voicemail message from within the `VoicemailMain()` application. Available as option 5 under the advanced menu. If this option is disabled, then option 5 in the advanced menu will not be prompted. Valid options are `yes` or `no`. Default is `no`.
`tempgreetwarn`	Enables a notice to the user when their temporary greeting is enabled. Valid options are `yes` or `no`. Default is `no`.
`tz`	Sets the time zone for a voicemail user (or globally). See */usr/share/timezone* for different available time zones. Not applicable if `envelope=no`.
`volgain`	The `volgain` option allows you to set volume gain for voicemail messages. The value is in decibels (dB). The *sox* application must be installed for this to work.

The mailboxes you define in your *voicemail.conf* file might look like the following examples:

```
[default]
100 => 5542,Mike Loukides,mike@shifteight.org
101 => 67674,Tim OReilly,tim@shifteight.org
102 => 36217,Mary JonesSmith,mary.jones-smith@shifteight.org

; *** This needs to all be on the same line
103 => 5426,Some Guy,,,dialout=fromvm|callback=fromvm
|review=yes|operator=yes|envelope=yes
```

```
[shifteight]
100 => 0107,Leif Madsen,leif@shifteight.org
101 => 0523,Jim VanMeggelen,jim@shifteight.org,,attach=no|maxmsg=100
102 => 11042,Tilghman Lesher,,,attach=no|tz=central
```

The Asterisk directory cannot handle the concept of a family name that is anything other than a simple word. This means that family names such as *O'Reilly*, *Jones-Smith*, and yes, even *Van Meggelen*, must have any punctuation characters and spaces removed before being added to *voicemail.conf*.

The contexts in *voicemail.conf* are an excellent and powerful concept, but you will likely find that the `default` context will be all that you need in normal use. The primary reason for multiple mailbox contexts is when your system is hosting more than one PBX and you need mailbox separation.

An Initial voicemail.conf File

We recommend the following sample as a starting point. You can refer to *~/asterisk-complete/asterisk/11/configs/voicemail.conf.sample* for details on the various settings:

```
; Voicemail Configuration

[general]
format=wav49|wav
serveremail=voicemail@shifteight.org
attach=yes
skipms=3000
maxsilence=10
silencethreshold=128
maxlogins=3
emaildateformat=%A, %B %d, %Y at %r
pagerdateformat=%A, %B %d, %Y at %r
sendvoicemail=yes ; Allow the user to compose and send a voicemail while inside

[zonemessages]
eastern=America/New_York|'vm-received' Q 'digits/at' IMp
central=America/Chicago|'vm-received' Q 'digits/at' IMp
central24=America/Chicago|'vm-received' q 'digits/at' H N 'hours'
military=Zulu|'vm-received' q 'digits/at' H N 'hours' 'phonetic/z_p'
european=Europe/Copenhagen|'vm-received' a d b 'digits/at' HM

[shifteight.org]
100 => 1234,Leif Madsen,leif@shifteight.org
101 => 1234,Jim Van Meggelen,jim@shifteight.org
102 => 1234,Russell Bryant,russell@shifteight.org
103 => 1234,Jared Smith,jared@shifteight.org
```

Setting up a Linux server to handle the sending of email is a Linux administration task that is beyond the scope of this book. You will need to test your voicemail-to-email service to ensure that the email is being handled appropriately by the Mail Transfer Agent (MTA),[3] and that downstream spam filters are not rejecting the messages (one reason this might happen is if your Asterisk server is using a hostname in the email body that does not in fact resolve to it).

Standard Voicemail KeyMap

Here we'll look at the standard keymap configuration for Comedian Mail. Some options may be enabled or disabled based on the configuration of *voicemail.conf* (e.g., `envelope=no`), but our overview in Figure 8-1 will show the standard options available with minimal configuration.

Dialplan Integration

There are two primary dialplan applications that are provided by the *app_voicemail.so* module in Asterisk. The first, simply named `VoiceMail()`, does exactly what you would expect it to, which is to record a message in a mailbox. The second one, `VoiceMailMain()`, allows a caller to log into a mailbox to retrieve messages.

The VoiceMail() Dialplan Application

When you want to pass a call to voicemail, you need to provide two arguments: the mailbox (or mailboxes) in which the message should be left, and any options relating to this, such as which greeting to play or whether to mark the message as urgent. The structure of the `VoiceMail()` command is this:

```
VoiceMail(mailbox[@context][&mailbox[@context][&...]][,options])
```

The options you can pass to `VoiceMail()` to provide a higher level of control are detailed in Table 8-5.

3. Also sometimes called a Message Transfer Agent.

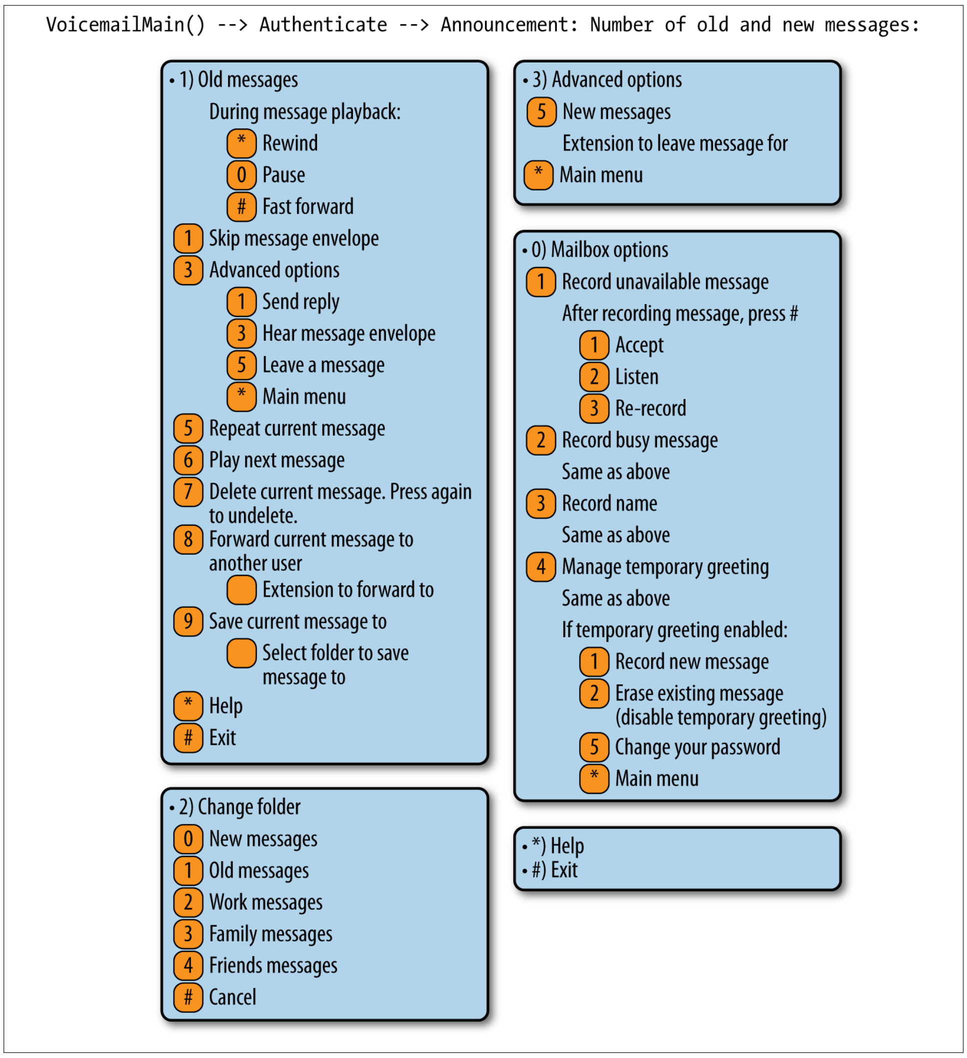

Figure 8-1. Keymap configuration for Comedian mail

Table 8-5. VoiceMail() optional arguments

Argument	Purpose
`b`	Instructs Asterisk to play the busy greeting for the mailbox (if no busy greeting is found, the unavailable greeting will be played).
`d([c])`	Accepts digits to be processed by context `c`. If the context is not specified, it will default to the current context.
`g(#)`	Applies the specified amount of gain (in decibels) to the recording. Only works on DAHDI channels.

Argument	Purpose
`s`	Suppresses playback of instructions to the callers after playing the greeting.
`u`	Instructs Asterisk to play the unavailable greeting for the mailbox (this is the default behavior).
`U`	Indicates that this message is to be marked as urgent. The most notable effect this has is when voicemail is stored on an IMAP server. In that case, the email will be marked as urgent. When the mailbox owner calls in to the Asterisk voicemail system, he should also be informed that the message is urgent.
`P`	Indicates that this message is to be marked as priority.

The `VoiceMail()` application sends the caller to the specified mailbox, so that he can leave a message. The mailbox should be specified as *`mailbox`*`@`*`context`*, where *`context`* is the name of the voicemail context. The option letters `b` or `u` can be added to request the type of greeting. If the letter `b` is used, the caller will hear the mailbox owner's *busy* message. If the letter `u` is used, the caller will hear the mailbox owner's *unavailable* message (if one exists).

Consider this simple example extension 101, which allows people to call John:

```
exten => 101,1,NoOp()
   same => n,Dial(${JOHN})
```

Let's add an unavailable message that the caller will be played if John doesn't answer the phone. Remember, the second argument to the `Dial()` application is a timeout. If the call is not answered before the timeout expires, the call is sent to the next priority. Let's add a 10-second timeout, and a priority to send the caller to voicemail if John doesn't answer in time:

```
exten => 101,1,NoOp()
   same => n,Dial(${JOHN},10)
   same => n,VoiceMail(101@default,u)
```

Now, let's change it so that if John is busy (on another call), the caller will be sent to his voicemail, where he will hear John's busy message. To do this, we will make use of the `${DIALSTATUS}` variable, which contains one of several status values (type **`core show application Dial`** at the Asterisk console for a listing of all the possible values):

```
exten => 101,1,NoOp()
   same => n,Dial(${JOHN},10)
   same => n,GotoIf($["${DIALSTATUS}" = "BUSY"]?busy:unavail)
   same => n(unavail),VoiceMail(101@default,u)
   same => n,Hangup()
   same => n(busy),VoiceMail(101@default,b)
   same => n,Hangup()
```

Now callers will get John's voicemail (with the appropriate greeting) if John is either busy or unavailable. An alternative syntax is to use the `IF()` function to define which of the unavailable or busy messages to use:

```
exten => 101,1,NoOp()
   same => n,Dial(${JOHN},10)
```

```
    same => n,Voicemail(101@default,${IF($["${DIALSTATUS}" = "BUSY"]?b:u)})
    same => n,Hangup()
```

A slight problem remains, however, in that John has no way of retrieving his messages. We will remedy that in the next section.

The VoiceMailMain() Dialplan Application

Users can retrieve their voicemail messages, change their voicemail options, and record their voicemail greetings using the `VoiceMailMain()` application. `VoiceMailMain()` accepts two arguments: the mailbox number (and optionally the context) to be accessed, and some options. Both arguments are optional.

The structure of the `VoiceMailMain()` application looks like this:

```
VoiceMailMain([mailbox][@context][,options])
```

If you do not pass any arguments to `VoiceMailMain()`, it will play a prompt asking the caller to provide her mailbox number. The options that can be supplied are listed in Table 8-6.

Table 8-6. VoiceMailMain() optional arguments

Argument	Purpose
`p`	Allows you to treat the *mailbox* parameter as a prefix to the mailbox number.
`g(#)`	Increases the gain by # decibels when playing back messages.
`s`	Skips the password check.
`a(folder)`	Starts the session in one of the following voicemail folders (defaults to `0`): • `0` - `INBOX` • `1` - `Old` • `2` - `Work` • `3` - `Family` • `4` - `Friends` • `5` - `Cust1` • `6` - `Cust2` • `7` - `Cust3` • `8` - `Cust4` • `9` - `Cust5`

To allow users to dial an extension to check their voicemail, you could add an extension to the dialplan like this:

```
[Services]
exten => *98,1,NoOp(Access voicemail retrieval.)
   same => n,VoiceMailMain()
```

You would then simply need to add an include in the `[LocalSets]` context so that you could dial *98:

```
[LocalSets]
; existing dialplan above here
include => Services
```

Creating a Dial-by-Name Directory

One last feature of the Asterisk voicemail system that we should cover is the dial-by-name directory. This is created with the `Directory()` application. This application uses the names defined in the mailboxes in *voicemail.conf* to present the caller with a dial-by-name directory of users.

`Directory()` takes up to three arguments: the voicemail context from which to read the names, the optional dialplan context in which to dial the user, and an option string (which is also optional). By default, `Directory()` searches for the user by last name, but passing the `f` option forces it to search by first name instead. Let's add two dial-by-name directories to the `incoming` context of our sample dialplan, so that callers can search by either first or last name:

```
exten => 8,1,Directory(default,incoming,f)
exten => 9,1,Directory(default,incoming)
```

If callers press `8`, they'll get a directory by first name. If they dial `9`, they'll get the directory by last name.

Using a Jitterbuffer

When using Asterisk as a voicemail server,[4] you may want to add a *jitterbuffer* in between voicemail and the caller. The purpose of a jitterbuffer is to help deal with the fact that when a call traverses an IP network, the traffic may not arrive with perfect timing and in perfect order. If packets occasionally arrive with a bit of delay (jitter) or if they arrive out of order, a jitterbuffer can fix it so that the voicemail system receives the voice stream on time and in order. If the jitterbuffer detects that a packet was lost (or may arrive so late that it will no longer matter), it can perform packet-loss concealment. That is, it will attempt to make up a frame of audio to put in place of the lost audio to make it harder to hear that audio was lost.

In Asterisk, jitterbuffer support can be enabled on a bridge between two channels in two ways. In the case of voicemail, there is generally only a single channel connected to one of the voicemail applications. The old method (which is required in versions prior to Asterisk 10) is to enable the use of a jitterbuffer in front of voicemail by creating a bridge between two channels using a `Local` channel and specifying the `j` option.

4. This advice applies to any situation where Asterisk is the endpoint of a call. Another example would be when using the `MeetMe()` or `ConfBridge()` applications for conferencing.

Specifying the `n` option for the `Local` channel additionally ensures that the `Local` channel is not optimized out of the call path in Asterisk:

```
[Services]
; old method -- only required in versions prior to Asterisk 10
exten => *98,1,Dial(Local/vmm@Services/nj)

exten => vmm,1,VoiceMailMain()
```

As of Asterisk 10, there exists a new `JITTERBUFFER()` dialplan function which, from the user's perspective, performs the same functionality. Simply by setting values in the dialplan, we can enable a jitterbuffer prior to accessing a dialplan application such as `Voicemail()`:

```
[Services]
; new method -- available in Asterisk 10 and later
exten => *98,1,NoOp()
   same => n,Set(JITTERBUFFER(fixed)=default)
   same => n,VoiceMailMain()
```

There exists both a fixed and an adaptive jitterbuffer, along with several different settings. We've used the fixed jitterbuffer with the default settings, which are as follows. See *core show function JITTERBUFFER* for more configuration options:

- 200 ms buffer length
- If more than 1,000 ms of timestamp difference exists, the jitterbuffer will resync

Storage Backends

The storage of messages on traditional voicemail systems has always tended to be overly complicated.[5] Asterisk, on the other hand, not only provides you with a simple, logical, filesystem-based storage mechanism, but also offers a few extra message storage options.

Linux Filesystem

By default, Asterisk stores voice messages in the spool folder, at */var/spool/asterisk/voicemail/`<context>`/`<mailbox>`*. The messages can be stored in multiple formats (such as WAV and GSM), depending on what you specified as the `format` in the `[general]` section of your *voicemail.conf* file. Your greetings are also stored in this folder.

5. Nortel used to store its messages in a sort of special partition, in a proprietary format, which made it pretty much impossible to extract messages from the system, or email them, or archive them, or really do anything with them.

Asterisk will not create a folder for any mailboxes that do not have any recordings yet (as would be the case with a new mailbox), so this folder cannot be used as a reliable method of determining which mailboxes exist on the system.

Here's an example of what might be in a mailbox folder. This mailbox has no new messages in the *INBOX*, has two saved messages in the *Old* folder, and has *busy*, *unavailable* and name (*greet*) greetings recorded (as shown in Figure 8-2).

```
/var/spool/asterisk/voicemail/default/301
     ./INBOX
     ./Old
          ./Old/msg0000.WAV
          ./Old/msg0000.txt
          ./Old/msg0001.WAV
          ./Old/msg0001.txt
     ./Urgent
     ./busy.WAV
     ./unavail.WAV
     ./greet.WAV
```

Figure 8-2. Sample mailbox folder

For each message, there is a matching *msg####.txt* file, which contains the envelope information for the message. The *msg####.txt* file is also critically important for message waiting indication (MWI), as this is the file that Asterisk looks for in the *INBOX* to determine whether the message light for a user should be on or off.

ODBC

In a centralized or distributed system, you may find it desirable to store messages as binary objects in a database, instead of as files on the filesystem. We'll discuss this in detail in "ODBC Voicemail Message Storage" on page 463.

IMAP

Many people would prefer to manage their voicemail as part of their email. This has been called *unified messaging* by the telecom industry, and its implementation has traditionally been expensive and complex. Asterisk allows for a fairly simple integration between voicemail and email, either through its built-in voicemail-to-email handler, or through a relationship with an IMAP server. We'll discuss IMAP integration in detail in "Voicemail IMAP Integration" on page 503.

Using Asterisk as a Standalone Voicemail Server

In a traditional telecom environment, the voicemail server was typically a standalone unit (provided either as a separate server altogether, or as an add-in card to the system). Very few PBXs had fully integrated voicemail (in the sense that voicemail was an integral part of the PBX rather than a peripheral device).

Asterisk is quite capable of serving as a standalone voicemail system. The two most common reasons one might want to do this are:

1. If you are building a large, centralized system and have several servers each providing a specific function (proxy server, media gateway, voicemail, conferencing, etc.)
2. If you wish to replace the voicemail system on a traditional PBX with an Asterisk voicemail

Asterisk can serve in either of these roles.

Integrating Asterisk into a SIP Environment as a Standalone Voicemail Server

If you want to have Asterisk act as a dedicated voicemail server (i.e., with no sets registered to it and no other types of calls passing through it), the process from the dialplan perspective is quite simple. Getting message waiting to work can be a bit more difficult, though.

Let's start with a quick diagram. Figure 8-3 shows an overly simplified example of a typical SIP enterprise environment. We don't even have an Asterisk server in there (other than for the voicemail), in order to give you a generic representation of how Asterisk could serve as a standalone voicemail server in an otherwise non-Asterisk environment.

Unfortunately, Asterisk cannot send message notifications to an endpoint if it doesn't know where that endpoint is. In a typical Asterisk system, where set registration and voicemail are handled on the same machine, this is never a problem, since Asterisk knows where the sets are. But in an environment where the sets are not registered to Asterisk, this can become a complex problem.

There are several solutions on the Internet that recommend using the `externnotify` option in *voicemail.conf*, triggering an external script whenever a message is left in a mailbox (or deleted). While we can't say that's a bad approach, we find it a bit kludgy, and it requires the administrator to understand how to write an external script or program to handle the actual passing of the message.

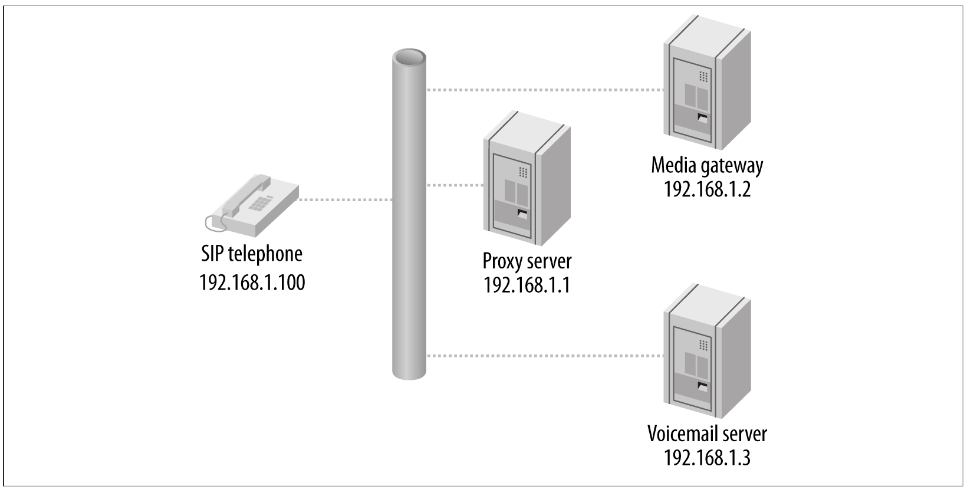

Figure 8-3. Simplified SIP enterprise environment

Instead you can statically define an entry for each mailbox in the voicemail server's *sip.conf* file, indicating where the message notifications are to be sent. Rather than defining the address of each endpoint, however, you can have the voicemail server send all messages to the proxy, which will handle the relay of the message notifications to the appropriate endpoints.

The voicemail server still needs to know about the SIP endpoints, even though the devices are not registered directly to it. This can be done either through a *sip.conf* file that identifies each SIP endpoint, or through a static realtime database that does the same thing. Whether you use *sip.conf* or the Asterisk Realtime Architecture (ARA), each endpoint will require an entry similar to this:

```
[messagewaiting](!)             ; a template to handle the settings common
                                ; to all mailboxes
type=peer
subscribecontext=voicemailbox   ; the dialplan context on the voicemail server
context=voicemailbox            ; the dialplan context on the voicemail server
host=192.168.1.1                ; ip address of presence server

[0000FFFF0001](messagewaiting) ; this will need to match the subscriber
                                ; name on the proxy
mailbox=0000FFFF0001@DIR1       ; must be in the form mailbox@mailboxcontext
defaultuser=0000FFFF0001        ; this will need to match the subscriber
                                ; name on the proxy
```

Note that Asterisk's dynamic realtime will not work with this configuration, as a peer's information is only loaded into memory when there is an actual call involving that peer. Since message notification is not a call as far as Asterisk is concerned, using dynamic realtime will not allow message waiting to happen for any peers not registered to Asterisk.

You will *not* want to implement this unless you have prototyped the basic operation of the solution. Although we all agree that SIP is a protocol, not everyone agrees as to the correct way to implement the protocol. As a result, there are many interoperability challenges that need to be addressed in a solution like this. We have provided a basic introduction to this concept in this book, but the implementation details will depend on other factors external to Asterisk, such as the capabilities of the proxy.

The fact that no device has to register with Asterisk will significantly reduce the load on the Asterisk server, and as a result this design should allow for a voicemail server that can support several thousand subscribers.

Dialplan requirements

The dialplan of the voicemail server can be fairly simple. Two needs must be satisfied:

1. Receive incoming calls and direct them to the appropriate mailbox
2. Handle incoming calls from users wishing to check their messages

The system that is passing calls to the voicemail server should set some SIP headers in order to pass additional information to the voicemail server. Typically, this information would include the mailbox/username that is relevant to the call. In our example, we are going to set the headers `X-Voicemail-Mailbox` and `X-Voicemail-Context`, which will contain information we wish to pass to the voicemail server.[6]

6. As far as we know, there aren't any specific SIP headers that are standardized for this sort of thing, so you should be able to name the headers whatever you want. We chose these header names simply because they make some sort of sense. You may find that other headers would suit your needs better.

If the source system is also an Asterisk system, you might set the headers using the `SIPAddHeader()` voicemail application, in a manner similar to this:

```
exten => sendtovoicemail,1,Verbose(2,Set SIP headers for voicemail)
    same => n,SIPAddHeader(X-Voicemail-Mailbox: mailbox number)
    same => n,SIPAddHeader(X-Voicemail-Context: voicemailbox)
```

Note that this dialplan does not go on the voicemail server. It would only be useful if one of the other servers in your environment was also an Asterisk server. If you were using a different kind of server, you would need to find out how to set custom headers in that platform, or find out if it already uses specific headers for this sort of thing, and possibly modify the dialplan on the voicemail server to handle those headers.

The voicemail server will need an *extensions.conf* file containing the following:

```
[voicemailbox]
; direct incoming calls to a mailbox
exten =>  Deliver,1,NoOp()
    same =>  n,Set(Mailbox=${SIP_HEADER(X-Voicemail-Mailbox)})
    same =>  n,Set(MailboxContext=${SIP_HEADER(X-Voicemail-Context)})
    same =>  n,VoiceMail(${Mailbox}@${MailboxContext})
    same =>  n,Hangup()

; connect users to their mailbox so that they can retrieve messages
exten => Retrieve,1,NoOp()
    same =>  n,Set(Mailbox=${SIP_HEADER(X-Voicemail-Mailbox)})
    same =>  n,Set(MailboxContext=${SIP_HEADER(X-Voicemail-Context)})
    same =>  n,VoiceMailMain(${Mailbox}@${MailboxContext})
    same =>  n,Hangup()
```

sip.conf requirements

In the *sip.conf* file on the voicemail server, not only are entries required for all the mailboxes for message-waiting notification, but some sort of entry is required to define the connection between the voicemail server and the rest of the SIP environment:

```
[VOICEMAILTRUNK]
type=peer
defaultuser=voicemail
fromuser=voicemail
secret=s0m3th1ngs3cur3
canreinvite=no
host=<address of proxy/registrar server>
disallow=all
allow=ulaw
dtmfmode=rfc2833
context=voicemailbox
```

The other end of the connection (probably your proxy server) must be configured to pass voicemail connections to the voicemail server.

Running Asterisk as a standalone voicemail server requires some knowledge of clustering and integration, but you can't beat the price.

SMDI (Simplified Message Desk Interface)

The Simplified Message Desk Interface (SMDI) protocol is intended to allow communication of basic message information between telephone systems and voicemail systems.

Asterisk supports SMDI, but given that this is an old protocol that runs across a serial connection, there are likely to be integration challenges. Support in various PBXs and other devices may be spotty. Still, it's a fairly simple protocol, so it's certainly worth testing out if you are considering using Asterisk as a voicemail replacement on an old PBX.

The following is not a detailed explanation of how to configure SMDI for Asterisk, but rather an introduction to the concepts, with some basic examples. If you are planning on implementing SMDI, you will need to write some complex dialplan logic and have a good understanding of how to interconnect systems via serial connections.[7]

SMDI is enabled in Asterisk by the use of two options in the `[general]` section of the *voicemail.conf* file:

```
smdienable=yes
smdiport=/dev/ttyS0; or whatever serial port you are connecting your
                   ; SMDI service to
```

Additionally, you will need an *smdi.conf* file in your */etc/asterisk* folder to define the details of your SMDI configuration. It should look something like this (see the *smdi.conf.sample* file for more information on the available options):

```
[interfaces]
charsize=7
paritybit=even
baudrate=1200           ; hopefully a higher bitrate is supported
smdiport=/dev/ttyS0     ; or whatever serial port you'll be using to handle
                        ; SMDI messages on asterisk

[mailboxes]             ; map incoming digit strings (typically DID numbers)
                        ; to a valid mailbox@context in voicemail.conf
smdiport=/dev/ttyS0     ; first declare which SMDI port the following mailboxes
                        ; will use
```

7. If you are not experienced with configuring and troubleshooting serial connections, you may find this whole process far more trouble than it's worth.

```
4169671111=1234@default
4165551212=9999@default
```

In the dialplan there are two functions that will be wanted in an SMDI configuration. The `SMDI_MSG_RETRIEVE()` function pulls the relevant message from the SMDI message queue. You need to pass the function a search key (typically the DID that is referred to in the message), and it will pass back an ID number that can be referenced by the `SMDI_MSG()` function:

```
SMDI_MSG_RETRIEVE(<smdi port>,<search key>[,timeout[,options]])
```

Once you have the SMDI message ID, you can use the `SMDI_MSG()` function to access various details about the message, such as the `station`, `callerID`, and `type` (the SMDI message type):

```
SMDI_MSG(<message_id>,<component>)
```

In your dialplan, you will need to handle the lookup of the SMDI messages that come in, to ensure that calls are handled correctly. For example, if an incoming call is intended for delivery to a mailbox, the message type might be one of `B` (for busy) or `N` (for unanswered calls). If, on the other hand, the call is intended to go to `VoiceMailMain()` because the caller wants to retrieve his messages, the SMDI message type would be `D`, and that would have to be handled.

Database Integration

The Asterisk voicemail application can be integrated into a database. This can be very useful, especially in clustered and distributed systems. It is discussed in detail in Chapter 16.

Conclusion

While the Asterisk voicemail system is quite old in terms of Asterisk code, it is nevertheless a powerful application that can (and does) compete quite successfully with expensive, proprietary voicemail systems.