Building a Personal VoIP System

May 25, 2023 AD

I've always been a big self-hoster, but had never attempted anything related to VoIP. I recently purchased some IP phones and set up a personal home phone network using Asterisk. This guide will help you set up your own digital telephone system using open-source tools.

This guide is written for those who are experienced with self-hosting, but are totally unfamiliar with VoIP. Therefore, I'm going to gloss over certain uninteresting technicalities for the sake of brevity.

SIP: A Brief Introduction

Before getting your hands dirty with phone configuration and Asterisk dialplans, it's worth taking some time to understand the underlying network protocols involved. This will pay dividends later on when you're debugging the inevitable connectivity issues with your voice calls (trust me).

The Session Initialization Protocol, or SIP, is a signaling protocol used by basically every digital telephony device produced in the last two decades. VoIP phones, softphones, office conferencing devices…they all use SIP. SIP can use either TCP or UDP, and usually listens on port 5060.

There are two really important things you need to know about about SIP.

1. SIP does **not** carry the voice data. It's just a signaling protocol, and it's only used to negotiate which IP address, port, and audio format should be used to carry the audio stream.

2. SIP was initially released in 1999, and was designed with the assumption that each device has its own globally routable public IP address. After all, the IPv6 standard was released back in 1995, and NAT would soon be a thing of the past…right? Unforunately, this did not end up being the case.

For example, let's say we have a standard home network with two VoIP phones in the local 192.168.1.0/24 subnet. Consider what happens when these two phones want to call each other. In plain English, the SIP handshake will go something like this:

> Hello 192.168.1.6.
> I would like to place an audio call using the G.711 codec.
> Please send your audio frames to me at address 192.168.1.5, port 20000.

> Hi there, 192.168.1.5!
> Sure, I support the G.711 codec.
> Please send your audio frames to me at address 192.168.1.6, port 30000.

Each phone randomly chooses an unused UDP port on which to receive the other party's audio stream. After the SIP negotiation, they will both send voice data to each other using the Real-Time Transport Protocol, or RTP. Since they are both on the same local network, this works fine.

NAT Problems

Now consider what happens when we call someone *outside* of our local network. Let's say our router has a public IP of 203.0.113.8, and our friend Alice has address 198.51.100.7.

> Hello 198.51.100.7. Is Alice there?
> I would like to place an audio call using the G.711 codec.

> Please send your audio frames to me at address 192.168.1.5, port 20000.

> Hi there, “192.168.1.5”. I see you have source IP 203.0.113.8…strange!
> Sure, I support the G.711 codec.
> Please send your audio frames to me at address 198.51.100.7, port 30000.

Thanks to network address translation, or NAT, we are able to make the SIP connection to Alice. Unfortunately, you will probably find that audio only works in one direction!

We’ve got two problems. First, we’ve asked Alice to send her audio to our *local* IP address, which won’t route over the Internet. Luckily, most devices are smart enough to use the source address of the incoming packet when these don’t match up. So Alice’s audio stream will probably end up at our router.

But second, our router will still block all of her audio traffic! When it receives a UDP packet from Alice’s audio stream, there’s no stateful NAT connection to match it back to an internal client, so it’s silently dropped. Sad!

NAT Solutions

There are three ways to solve this problem.

1. Give each of your SIP devices its own public IP (probably not feasible).
2. Use a SIP Application Layer Gateway. This is a horrible feature offered by some routers. Basically, it deep-packet-inspects your SIP traffic, rewrites the headers, and creates port forwards on-the-fly to make sure the inbound audio stream makes its way to your device.

   SIP ALGs are a total hack and notoriously buggy. In addition, when you decide to encrypt your SIP traffic using TLS, they quit working altogether.
3. Configure a fixed RTP port range on each of your SIP devices, then create static port forwarding rules on your router for each device. This is really the only decent option.

   Note that since we’ll be using Asterisk, you also have the option of “proxying” all your audio streams through the Asterisk box–then you only have one port forward to set up (we’ll get to that later).

Asterisk: What *is* it?

Asterisk is an open-source implementation of a private branch exchange, or PBX. If you’re a VoIP novice like I was, this is probably meaningless to you.

A PBX is the central brain of any private telephone network. The general idea is this: You plug a bunch of dumb telephones into one side of the PBX, and you plug one or more uplinks to the PSTN into the other side. The PBX lets the internal phones dial each other using private internal extensions, while multiplexing calls to and from the PSTN. It also handles advanced features like voicemail, call queues, hunt groups, and interactive menus for callers.

In the old days, the PBX would be a huge box down in the server room with a rat’s nest of of RJ-11 telephone cables sprawling out, connecting it to every single phone in the office. These monstrosities have since been replaced with software-based PBXes that talk SIP over standard Ethernet networks. Asterisk is the de-facto open source PBX.

Jargon Glossary

Before we get started, you need to know some lingo. Asterisk is an *old* project, with an unintuitive configuration. Hopefully this section will help you decipher some of the documentation and setup

guides you find out there.

Extensions
: Technically, in Asterisk parlance, any dialed number is an extension. In practice though, an *extension* typically refers to a *local* phone number (like 6001) with an associated SIP account. While extensions can be alphanumeric, virtually no one does this–most people use 3- or 4-digit numbers.

Queues
: Asterisk queues are a mechanism for managing incoming callers. You can configure which local extensions should ring for each queue, and even play custom music while callers are waiting.

SIP Trunk
: A "SIP Trunk" is just a fancy term for your upstream telephone provider. For example, if you subscribe to a provider like VOIP.ms, they'll give you a public phone number and a SIP server to connect to, along with a username and password. This is your "SIP Trunk."

DID
: A DID, or "Direct Inward Dial," is the technical term for a public phone number.

Contexts
: In Asterisk, every call is assigned a *context*. The context is simply a name of your choosing, and you specify it when you configure each extension or SIP trunk. For example, I give all my internal phones a context called `from-home`, and I give my SIP trunk a context called `from-pstn`.

Dialplan
: The Asterisk dialplan is an arcane scripting language where you define what happens when you dial a number or an incoming call is received. The dialplan is the glue between the *extensions* and *contexts* described above. The syntax is fairly unintuitive, but don't worry! We'll walk through it in a later section.

Codecs
: A codec is a type of digital encoding used for the audio stream over the wire. Every VoIP device supports one or more codecs, and you need at least one common codec with the other party to make calls. When no shared codecs are supported, Asterisk has the ability to do man-in-the-middle transcoding.
G.711 is basically the only codec with universal support. It has two versions: `ulaw` (North America/Japan) and `alaw` (everywhere else). I use a higher quality codec (G.722) for internal calls, and let Asterisk transcode down to G.711 when I call out to the PSTN.

BLF / Presence
: The BLF, or busy lamp field, is a little LED on your VoIP phone that lights up when one of your contacts is using their line. In Asterisk, this functionality is enabled by something called a presence subscription.

SIP / PJSIP
: On some guides online, you will see references to the `chan_sip` and `chan_pjsip` modules. `chan_sip` is the legacy Asterisk SIP implementation. You should be using PJSIP for everything these days.

Step 1: Acquire an IP Phone

First, you will need one or more VoIP phones. Any device that supports SIP calling should work fine with Asterisk, so this mostly comes down to personal preference.

As a beginner, you'll probably want to select for ease of configuration. Most modern VoIP phones expose a friendly web GUI where you configure all your SIP accounts, ring settings, etc. I'd also recommend avoiding any devices that rely on proprietary setup tools.

Personally, I've had zero problems with Yealink devices. I recommend the T54W model for a desk phone, or the W73P if you prefer a cordless model. You can buy these new at the usual online retailers, but eBay often has used models for sale if you want to save some money.

If you don't want to buy a physical device, you can also use a softphone app like Linphone. I use it on my Android phone running GrapheneOS, and it works well with the right settings. I'll cover Linphone in a later section.

Step 2: Subscribe to a VoIP Provider

Next, you need to subscribe to a VoIP service so you can make and receive calls over the PSTN. You'll often see people call this a "SIP Trunk."

Basically, you pay a VoIP company a small monthly fee, usually $3-$5 per month for a single DID. In exchange, they give you a public telephone number, along with credentials for their SIP server. After configuring Asterisk to connect to that SIP server, you can make and receive calls via the PSTN with that number.

I can personally recommend two providers.

1. VOIP.ms is an inexpensive provider with servers in the USA, Canada, and western Europe. You can get "unlimited" inbound calls with a single DID for about $5 a month, or less if you pay by the minute.

   VOIP.ms also supports call encryption via TLS/SRTP, which is nice.

2. JMP.chat (USA/Canada only) is an XMPP-based service that lets you make and receive voice calls and SMS/MMS messages using your existing XMPP account. I especially like JMP because there's already tons of high-quality native XMPP apps for both desktop and mobile devices. In addition, their entire stack is open source.

   While JMP is primarily focused on XMPP calling, they also provide you with a SIP account that you can use with Asterisk.

There are *tons* of other VoIP providers out there, so shop around.

Step 3: Configure Asterisk

Now you're ready to set up Asterisk. These instructions are written for RHEL-based distributions, but should be applicable to other Unixen.

Installation

First, install Asterisk:

```
dnf install asterisk asterisk-pjsip asterisk-voicemail-plain
```

You'll also need the sound files. Some distributions include these with their Asterisk package (EPEL does not).

```
for codec in g722 g729 gsm sln16 ulaw wav; do
  curl -sL "https://downloads.asterisk.org/pub/telephony/sounds/asterisk-core-sounds-
        en-${codec}-current.tar.gz" \
  | tar xvz -C /usr/share/asterisk/sounds
done
```

Network Configuration

Now we're ready to edit some config files. If you're behind a NAT (likely), you'll need to start by telling Asterisk about your local network. Edit `/etc/asterisk/pjsip.conf` and add some transport

templates.

```
; /etc/asterisk/pjsip.conf

; This template contains default settings for all transports.
[transport-defaults](!)
type = transport
bind = 0.0.0.0

; For communication to any addresses within local_nets, Asterisk will not apply
; NAT-related workarounds.
local_net = 127.0.0.0/8
local_net = 10.0.0.0/8
local_net = 172.16.0.0/12
local_net = 192.168.0.0/16

; If you have a public static IP for your Asterisk server, set it here.
external_media_address     = 203.0.113.8
external_signaling_address = 203.0.113.8


; The following UDP and TCP transports will inherit from the defaults.
[transport-udp](transport-defaults)
protocol = udp

[transport-tcp](transport-defaults)
protocol = tcp
```

Remember our discussion about NAT and RTP audio streams above? We also need to set up a static port range for incoming UDP audio traffic. Choose a suitable range for your Asterisk server in `rtp.conf`. Then, set up port forwarding on your router/firewall to forward all incoming UDP traffic within that range to the internal IP of your Asterisk server.

If your Asterisk server is behind NAT and you forget to do this, you'll experience one-way audio in your phone calls.

```
; /etc/asterisk/rtp.conf

[general]
rtpstart=20000
rtpend=20999
```

SIP Trunks

Next, we'll configure our SIP trunk(s). I'll be using `pjsip_wizard.conf`, since the PJSIP Wizard configuration style eliminates a ton of boilerplate. For this step, you'll need a server hostname and port, along with the username and password provided by your upstream SIP provider.

```
; /etc/asterisk/pjsip_wizard.conf

; This template contains default settings for all SIP trunks.
[trunk-defaults](!)
type = wizard

; Require SIP authentication.
sends_auth = yes

; Require SIP registration.
sends_registrations = yes

; Send media to the address and port on the incoming packet, regardless of what
```

```
; the SIP headers say (NAT workaround).
endpoint/rtp_symmetric = yes

; Rewrite the SIP contact to the address and port of the request (NAT workaround).
endpoint/rewrite_contact = yes

; Send the Remote-Party-ID SIP header. Some providers need this.
endpoint/send_rpid = yes

; Force the ulaw codec, which should work for everything in North America.
endpoint/allow = !all,ulaw

; Call encryption is out of scope for this guide.
endpoint/media_encryption = no

; If registration fails, keep trying ~forever.
registration/max_retries = 4294967295

; Don't assume an authentication failure is permanent.
registration/auth_rejection_permanent = no

; Perform a connectivity check every 30 seconds.
aor/qualify_frequency = 30


; Your SIP trunks go here. For this example, we'll assume you're using VOIP.ms.
; You can pick whatever section name you like, as long as its unique.
[voipms](trunk-defaults)
; You almost certainly want to use TCP.
transport = transport-tcp

; Hostname and port for your SIP provider.
remote_hosts = atlanta2.voip.ms:5060

; Choose a context name for incoming calls from this account. You'll use this
; name in your dialplan.
endpoint/context = from-pstn

; Your SIP provider will give you these credentials.
outbound_auth/username = 555555
outbound_auth/password = s3cret
```

Local SIP Extensions

In the same file, we'll also configure our local extensions. You'll need an extension for each VoIP handset or softphone within your network. I'll be using a prefix of `6XXX` for local extensions, but feel free to use whatever convention you prefer.

The following example has three extensions: two dedicated VoIP handsets, and one Android softphone. Note that if you want to use a softphone outside of your local network, you'll need to either open your Asterisk instance to the world (not recommended, unless you know what you're doing) or set up some kind of VPN.

```
; /etc/asterisk/pjsip_wizard.conf

; This template contains default settings for all local extensions.
[extension-defaults](!)
type = wizard

; Require clients to register.
accepts_registrations = yes

; Require clients to authenticate.
```

```
accepts_auth = yes

; When simultaneous logins from the same account exceed max_contacts, disconnect
; the oldest session.
aor/remove_existing = yes

; For internal phones, allow the higher quality g722 codec in addition to ulaw.
endpoint/allow = !all,g722,ulaw

; Context name for BLF/presence subscriptions. This can be any string of your
; choosing, but I'm using "subscribe". We'll use this in the dialplan later.
endpoint/subscribe_context = subscribe


;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Extension 6001 - VoIP phone
;;;;;;;;;;;;;;;;;;;;;;;;;;;
[6001](extension-defaults)
; Dialplan context name for calls originating from this account.
endpoint/context = from-home

; Voicemail address (note: I'm using the same voicemail address for all extensions)
endpoint/mailboxes = 6000@default

; Internal Caller ID string for this device.
endpoint/callerid = Living Room <6001>

; Username for SIP account. By convention, this should be the extension number.
inbound_auth/username = 6001

; Password for SIP account (you can choose whatever password you like).
inbound_auth/password = s3cret

; Maximum number of simultaneous logins for this account.
aor/max_contacts = 1

; Check connectivity every 30 seconds.
aor/qualify_frequency = 30

; Set connectivity check timeout to 3 seconds.
aor/qualify_timeout = 3.0

; IMPORTANT! This setting determines whether the audio stream will be proxied
; through the Asterisk server.
;
; If this device is directly reachable by the internet (either by a publicly
; routable IP, or static port mappings on your router), choose YES.
;
; Otherwise, if this device is hidden behind NAT, choose NO.
endpoint/direct_media = yes


;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Extension 6002 - VoIP phone
;;;;;;;;;;;;;;;;;;;;;;;;;;;
; (Same settings as above, except for the extension number and password.)
[6002](extension-defaults)
endpoint/context      = from-home
endpoint/mailboxes    = 6000@default
endpoint/callerid     = Kitchen <6002>
inbound_auth/username = 6002
inbound_auth/password = s3cret2
aor/max_contacts      = 1
aor/qualify_frequency = 30
aor/qualify_timeout   = 3.0
```

```
endpoint/direct_media = yes


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Extension 6003 - Linphone app on Android device
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[6003](extension-defaults)
endpoint/context      = from-home
endpoint/mailboxes    = 6000@default
endpoint/callerid     = Smartphone <6003>
inbound_auth/username = 6003
inbound_auth/password = s3cret3
aor/max_contacts      = 1
endpoint/direct_media = no

; For mobile devices, connectivity checks should be disabled.
; Two reasons for this:
;   1. Poor cellular signal often causes the connectivity check to time out,
;      even when nothing is actually wrong.
;   2. Frequent traffic on the TCP session causes constant wakeups and kills
;      battery life.
aor/qualify_frequency = 0
```

Voicemail

Voicemail recordings are stored on the local filesystem of the Asterisk server. You can access them by dialing a special voicemail extension from a local phone (configured in your dialplan). In addition, if you have a local MTA running, Asterisk can dispatch an email to an address of your choice whenever new voicemails arrive.

In the previous section, we referenced a voicemail address called `6000@default` for our internal extensions. We'll create the actual voicemail box in `voicemail.conf`.

```
; /etc/asterisk/voicemail.conf

; This section contains global voicemail options.
[general]
; Audio formats to store voicemail files.
format=wav49|gsm|wav

; "From:" address used for sending voicemail emails.
serveremail=asterisk-noreply@example.com

; Whether to attach the voicemail audio file to notification emails.
attach=yes

; Maximum number of messages per mailbox.
maxmsg=100

; Maximum length of a voicemail message in seconds.
maxsecs=300

; Maximum length of a voicemail greeting in seconds.
maxgreet=60

; How many milliseconds to skip forward/back when rew/ff in message playback.
skipms=3000

; How many seconds of silence before we end the recording.
maxsilence=10

; Silence threshold (what we consider silence: the lower, the more sensitive).
silencethreshold=128
```

```
; Maximum number of failed login attempts.
maxlogins=3

; Email template for voicemail notifications.
emailsubject=New voicemail ${VM_MSGNUM} in mailbox ${VM_MAILBOX}
emailbody=Hi ${VM_NAME},\n\nYou have a new voicemail in mailbox
        ${VM_MAILBOX}.\n\nFrom: ${VM_CALLERID}\nDate: ${VM_DATE}\nDuration:
        ${VM_DUR}\nMessage Number: ${VM_MSGNUM}
emaildateformat=%A, %B %d, %Y at %r

; Default timezone for mailboxes (defined in zonemessages section below).
tz=myzone

; Locale for generating date/time strings.
locale=en_US.UTF-8

; Minimum voicemail password length.
minpassword=4


; Custom timezone definitions go in this section.
[zonemessages]
myzone=America/New_York|'vm-received' Q 'digits/at' IMp


;;;;;;;;;;;;;;;;;
; Voicemail Boxes
;;;;;;;;;;;;;;;;;

; Each of the following sections describe a voicemail "context," which is a
; a collection of voicemail boxes. If you don't need multiple contexts, you
; can specify "default".
;
; The format is as follows:
;   VOICEMAIL_NUMBER => INITIAL_PASSWORD,REAL_NAME,EMAIL_ADDRESS,,,
;
; The last three fields aren't relevant for simple configurations.

[default]
6000 => 1234,John Doe,johndoe@example.com,,,
```

Queues

When someone calls our phone number, we'd like all our phones to ring simultaneously (similar to a traditional landline phone). We can emulate this behavior in Asterisk using a `ringall` queue.

Queues are configured in `queues.conf`.

```
; /etc/asterisk/queues.conf

; Global queue settings go in this section.
[general]
; Persist dynamic member lists in the astdb.
persistentmembers = yes

; Some options for more intuitive queue behavior.
autofill                = yes
monitor-type            = MixMonitor
shared_lastcall         = yes
log_membername_as_agent = yes

;;;;;;;;;;;;;;;;;;;
```

```
; Queue Definitions
;;;;;;;;;;;;;;;;;;

; The "home-phones" queue is for incoming calls to our home phone line.
[home-phones]
; For each incoming call, ring all members of the queue.
strategy = ringall

; Max number of seconds a caller waits in the queue.
timeout = 30

; Don't announce estimated hold time, etc.
announce-frequency          = 0
announce-holdtime           = no
announce-position           = no
periodic-announce-frequency = 0

; Allow ringing even when no queue members are present.
joinempty      = yes
leavewhenempty = no

; Ring member phones even when they are on another call.
ringinuse = yes

; Queue Members
;
; Each member is specified with the following format:
;  member => INTERFACE,PENALTY,FRIENDLY_NAME,PRESENCE_INTERFACE
;
; The "penalty" value is not interesting for our use case.
; With PJSIP, the BLF/Presence interface is identical to the standard interface name.
member => PJSIP/6001,0,Living Room,PJSIP/6001
member => PJSIP/6002,0,Kitchen,PJSIP/6002
member => PJSIP/6003,0,Smartphone,PJSIP/6003
```

The Dialplan

At this point, we've configured our upstream SIP trunks, created SIP accounts for some local phone extensions, and even defined a queue for incoming calls. All that's left is to glue all this together in a dialplan!

The dialplan is configured in `extensions.conf`, and it's definitely the least intuitive aspect of Asterisk. You'll most likely find its syntax to be confusing at first glance.

The thing to remember is that in the dialplan, everything is an *application*. Hanging up is performed by the `Hangup()` application, and voicemail prompts are handled by the `Voicemail()` application. Dialing a phone number is handled by (you guessed it) the `Dial()` application. Each application can take one or more comma-separated arguments.

Now, for the syntax. Each dialplan context is marked by square brackets. Each line within the context is (confusingly) called an *extension*, and has the following format:

```
[context-name]
exten => NAME,PRIORITY,APPLICATION()
exten => NAME,PRIORITY,APPLICATION()
; etc...
```

The *name* is the number (or pattern) of the extension.

The *priority* defines the order in which the step should be executed.

Finally, the *application* performs some action on the call.

A simple context definition might look something like this:

```
[from-home]
; ${EXTEN} is a macro which expands to the currently dialed number.
exten => _6XXX,1,Dial(PJSIP/${EXTEN})
exten => _6XXX,2,Hangup()
```

This dialplan section allows internal phones to call each other. The `_6XXX` is an extension pattern. When a device in the `from-home` context dials a 4-digit extension beginning with `6`, Asterisk will ring the corresponding PJSIP account.

Because it gets tedious repeating the same extension name over and over, Asterisk provides some syntactic sugar. The exact same dialplan could also be written as the following:

```
[from-home]
exten => _6XXX,1,Dial(PJSIP/${EXTEN})
 same => n,Hangup()
```

Below, I've provided a complete Asterisk dialplan for our example VoIP network. It supports the following features:

1. Internal phones can dial each other via their 4-digit extension.
2. Internal phones can dial out to the PSTN via standard 10-digit numbers.
3. Internal phones can access the voicemail menu by dialing `*99`.
4. Internal phones can show BLF/line status for all other phones.
5. Incoming calls from the PSTN will ring all internal phones simultaneously. Callers will be sent to voicemail if no one answers within 25 seconds.
6. If your phones support the "auto answer" header, you can initiate a whole-house intercom by dialing `6000`.

I'll provide plenty of comments to help you understand the arcane syntax. Hopefully it will be enough to bootstrap your own dialplan!

```
; /etc/asterisk/extensions.conf

; Remember, context names for each SIP account are specified in pjsip_wizard.conf.

; First, some safeguards against abuse of the built-in contexts.
[public]
exten => _X.,1,Hangup(3)
[default]
exten => _X.,1,Hangup(3)


; Next, some global variables. You'll need to change some of these.
[globals]
; Your local area code.
MY_AREA_CODE = 555

; Your real name and public phone number. This will be used for outgoing calls
; to the PSTN.
MY_CALLER_ID = John Doe <+5555555555>
```

```
; Dial this number from a local phone to access the voicemail menu.
VOICEMAIL_NUMBER = *99

; Voicemail address (configured in voicemail.conf)
VOICEMAIL_BOX = 6000@default

; Ring for this many seconds before forwarding the caller to voicemail.
VOICEMAIL_RING_TIMEOUT = 25

; Name of the 'ringall' queue for local phones.
HOME_QUEUE = home-phones

; Number to dial for local intercom.
INTERCOM   = 6000

; Dial pattern for local extensions.
LOCAL_EXTS = _6XXX


; Boilerplate to enable BLF/presence subscriptions. Note that the context name
; corresponds to the "endpoint/subscribe_context" value in pjsip_wizard.conf.
[subscribe]
exten => _XXXX,hint,PJSIP/${EXTEN}


; This context handles incoming calls from our SIP trunk provider. Each call is
; is placed into a ringall queue. If there is no answer, the caller is forwarded
; to voicemail.
[from-pstn]
exten => _X.,1,Queue(${HOME_QUEUE},nr,,,${VOICEMAIL_RING_TIMEOUT})
 same => n,Answer(500)
 same => n,Voicemail(${VOICEMAIL_BOX},su)
 same => n,Hangup()


; This is a function (or "gosub" in Asterisk lingo) that sets the "auto answer"
; SIP header on an outgoing call.
[gosub-intercom]
exten => s,1,Set(PJSIP_HEADER(add,Alert-Info)=auto answer)
 same => n,Return()


; This context handles incoming calls from local phones.
[from-home]
; When the INTERCOM number is dialed, page all members of the "home-phones" queue
; into a conference call.
exten => ${INTERCOM},1,Set(CALLERID(all)=Intercom <${EXTEN}>
 same => n,Page(${STRREPLACE(QUEUE_MEMBER_LIST(${HOME_QUEUE}),",","&")},db(gosub-
        intercom^s^1),10)
 same => n,Hangup()

; For local-to-local calls, ring indefinitely.
exten => ${LOCAL_EXTS},1,Dial(PJSIP/${EXTEN})
 same => n,Hangup()

; When the voicemail number is dialed, dump the caller into the voicemail menu.
exten => ${VOICEMAIL_NUMBER},1,Answer(500)
 same => n,VoiceMailMain(${VOICEMAIL_BOX},s)
 same => n,Hangup()

; The following extensions are used to dial out to the PSTN via our SIP trunk,
; using a personalized caller ID string.
;
; Recall that we named our SIP trunk "voipms" in pjsip_wizard.conf.
;
```

```
; N matches any digit 2-9
; X matches any digit 1-9

; For numbers formatted as +1xxxxxxxxxx, dial as-is.
exten => _+1NXXNXXXXXX,1,Set(CALLERID(all)=${MY_CALLER_ID})
 same => n,Dial(PJSIP/${EXTEN}@voipms)
 same => n,Hangup()

; For numbers like 1xxxxxxxxxx, add a leading "+".
exten => _1NXXNXXXXXX,1,Set(CALLERID(all)=${MY_CALLER_ID})
 same => n,Dial(PJSIP/+${EXTEN}@voipms)
 same => n,Hangup()

; For numbers without a country code, add a "+1".
exten => _NXXNXXXXXX,1,Set(CALLERID(all)=${MY_CALLER_ID})
 same => n,Dial(PJSIP/+1${EXTEN}@voipms)
 same => n,Hangup()

; For 7-digit numbers, add the local area code.
exten => _NXXXXXX,1,Set(CALLERID(all)=${MY_CALLER_ID})
 same => n,Dial(PJSIP/+1${MY_AREA_CODE}${EXTEN}@voipms)
 same => n,Hangup()

; For 3-digit numbers, like 311, 411, 911 (UNTESTED!!!), dial as-is.
exten => _N11,1,Set(CALLERID(all)=${MY_CALLER_ID})
 same => n,Dial(PJSIP/${EXTEN}@voipms)
 same => n,Hangup()
```

Troubleshooting

Once you've got all your configs in place, give Asterisk a kick:

```
systemctl restart asterisk
```

On RedHat-based distributions, you can check `/var/log/asterisk/messages` for errors. You can also check the systemd journal:

```
journalctl -u asterisk
```

You can also get lots of real-time information from the Asterisk console. To try it out, run the following command as root:

```
asterisk -rvvvvv
```

If you're experiencing connectivity issues with your voice calls, you can enable SIP debugging in the console using the following command:

```
pjsip set logger on
```

Step 4: Configure your IP Phones

The final step is to configure your VoIP phones to connect to your Asterisk server, and make a few test calls! The instructions here are for Yealink phones, but they should easily translate to other manufacturers.

SIP Account

Navigate to the IP address of the VoIP phone in your web browser, and log in with the default username and password (usually `admin/admin`). From here, you should be able to add a SIP account. In the Yealink interface, this is found under *Account→Register*.

For the “Living Room” extension we created above, you would set the following:

- **Register Name:** 6001
- **Username:** 6001
- **Password**: s3cret
- **Server Host:** *IP/hostname of your Asterisk server*
- **Server Port:** 5060
- **Transport:** UDP

Codecs

You should also make sure the appropriate codecs are enabled on the device. In the Yealink web interface, you’ll find them under *Account→Codec*. I recommend enabling these codecs in the following priority:

1. **G722**, for high-quality local calls
2. **PCMU** (*ulaw*), for dialing out to the PSTN

Asterisk will automatically transcode G722 down to *ulaw* if the other side doesn’t support it. You can mess around with even higher quality codecs like Opus, but in my experience they are not well supported nor easy to transcode.

RTP Port Range

To reduce load on the Asterisk server, you may want your VoIP phone to send and receive audio streams directly to and from the other party. To do this, you’ll need to configure your router/firewall to forward all incoming RTP traffic for the device.

First, give the device a static IP on your local network. Then, configure your router to port-forward all UDP traffic on your chosen RTP port range to that IP.

In the Yealink web interface, you can configure a static RTP port range under *Network→Advanced→Local RTP Port*.

If you *don’t* do all this, then make sure `endpoint/direct_media` is set to `no` for the SIP account in `pjsip_wizard.conf`!

Voicemail Number

You’ll also want to configure which number the phone should dial when you press the voicemail button. In the Yealink web interface, set *Account→Advanced→Voice Mail* to `*99` (or whatever you number you chose for `VOICEMAIL_NUMBER` above).

Busy Lamp Field (BLF)

You may want to have your phone’s LED light up when a given line is on a call. In the Yealink web interface, you can configure this under *Dsskey→Line Key*.

For each line you’re interested in, set **Type** to *BLF* and **Value** to the other party’s extension number.

Intercom

For the intercom functionality described in the dialplan above, make sure the phone is configured to allow auto-answer. In the Yealink web interface, you can configure this by setting *Features → Intercom → Intercom Allow* to *on*.

VoIP on Android: Linphone

In the past, Android provided a built-in SIP client. Sadly, as of Android 12, this has been removed.

After trying all of the SIP apps available in F-Droid, I'm only able to recommend Linphone. It works with bluetooth devices, has an intuitive interface, and reliably delivers calls.

To keep Android from killing the app, you'll need to make sure battery optimizations are disabled. Go to *Settings → Apps → See All Apps → Linphone → App Battery Usage* and select *Unrestricted*.

When configuring the SIP account in Linphone, use the following settings:

- **Username:** 6003 *(or your chosen mobile extension)*
- **Password:** *your chosen SIP account password*
- **Domain:** *IP/hostname of your Asterisk server*
- **Transport:** TCP
- **Expire:** 3600
- **Apply prefix for outgoing calls:** *enabled*

You'll definitely want to use the TCP transport for a mobile device. Smartphone radios are exceedingly efficient at keeping a long-running TCP connection open, and TCP will reliably deliver your SIP packets even with a poor cellular signal.

With *Expire* set to 3600, your phone will wake up every hour to re-establish the SIP connection. I've never had any connectivity issues with the TCP transport, but if you're paranoid, you might want to set this to a lower value.

I've also found the following Linphone settings useful:

- **Settings → Audio → Echo cancellation:** *disabled* (most phones have hardware-level echo cancellation)
- **Settings → Audio → Adaptive rate control:** *enabled*
- **Settings → Audio → Codec bitrate limit:** 128 kbits/s
- **Settings → Audio → Codecs:** *enable PCMU and G722*
- **Settings → Call → Improve interactions with bluetooth devices:** *enabled*
- **Settings → Call → Voice mail URI:** *99
- **Settings → Advanced → Start at boot time:** *enabled*

You may be tempted to hide the persistent notification icon while Linphone is running. Don't do it! Whenever I've tried to hide this icon, I get tons of missed calls, and Asterisk reports the device as offline for minutes at a time. As long I keep the icon visible in the notification area, I don't have any issues.

Closing Thoughts

For the past few months, I've used this exact setup for my primary phone number, and I've been pleased with the results.

If you decide to self-host your own VoIP infrastructure, you'll definitely want to configure some type of QoS on your router or firewall to prioritize voice traffic. Otherwise, you'll experience lags and poor audio quality whenever your network is congested.

In addition, you may want to investigate encrypting your calls via STRP and SIP TLS. I don't personally do this, because voice calls are totally in the clear as soon as you connect to the

PSTN anyway.

If you'd like to check out the configs required for encrypting your calls, or if you just want to get some ideas for automating your Asterisk configuration, check out my Ansible role on GitHub.