



BitVoice manual

An open-source phone system in one program, set up entirely in the web interface: extensions, queues, conference rooms, the staff portal, hosted organisations, and the public-safety lab track.

76 chapters · 63 figures · 39 thousand words · BioDream LLC

Contents

1. Part 1. Getting started

1. Getting started
2. How mature each part is
3. Extensions
4. Connecting phones
5. Skinny phones and MGCP gateways
6. Telephony cards
7. Trunks
8. Call routing
9. PIN sets
10. Caller ID lookup
11. Emergency calls
12. Sites and office 911
13. Health
14. Settings

2. Part 2. Where calls go

15. Call flow
16. Ring groups
17. Queues
18. Outbound dialer
19. Sounds: prompts, languages and hold music
20. Menus (auto attendant)
21. Dial-by-name directory
22. Dial-in (DISA)
23. Voice bots (AudioSocket)
24. Day/night toggles
25. Schedules
26. Scripts (JavaScript and Lua)
27. AI receptionist, voicemail triage and asking
28. Voicemail
29. Callbacks
30. Call links and website calling

3. Part 3. Everyday calling

31. Feature codes, forwarding and speed dials
32. The staff portal
33. Conference rooms
34. Paging and overhead speakers
35. Hot desking
36. Hotel and wake-up calls
37. Contacts on every phone
38. Calendars and do not disturb
39. Mobile apps
40. Video calls
41. Fax: sending and receiving, fax machines and analogue phones
42. Inbox: web chat, WhatsApp and Messenger
43. Business texting (SMS)
44. Transfer briefings

4. Part 4. Customers and insight

45. Live calls
46. Call history
47. Customer timeline
48. CRM: caller cards and call logging
49. Call insight for supervisors
50. Call recording and card payments
51. Recording consent
52. Call quality
53. Costs and budgets
54. Caller ID reputation and branded calls
55. Spam protection and STIR/SHAKEN

5. Part 5. Running BitVoice

56. Administrators
57. Single sign-on
58. Security
59. Changes, backups and import
60. Updates
61. Several servers as one phone system
62. Where the data is kept (SQLite or PostgreSQL)
63. Hosting other organisations
64. Codecs and sound
65. Microsoft Teams (Direct Routing)
66. Import
67. Configuration as code
68. Asterisk compatibility (AMI and ARI)
69. Archive
70. Legal holds
71. Metrics: Prometheus, OpenTelemetry, Grafana, SNMP and call records
72. Call generator
73. SIP trace

6. Part 6. Public safety: the lab track

- 74. 911 calls (the 911 profile)
- 75. Radio dispatch console
- 76. NG911 Core Services (ECRF, ESRP, LVF, BCF, LIS, logging)

Getting started

From a new install to a working phone system: extensions, phones, a phone company, where calls go, emergency calling from your offices, and how to tell everything is ready.

Getting started

In the web interface: 

BitVoice is a complete phone system in one program: the exchange that phones sign in to, the menus and queues callers move through, voicemail, recording, conference rooms, faxes, and the connections to phone companies. Everything is set up in the browser, on the screens this manual describes. There is no configuration file to edit and nothing else to install.

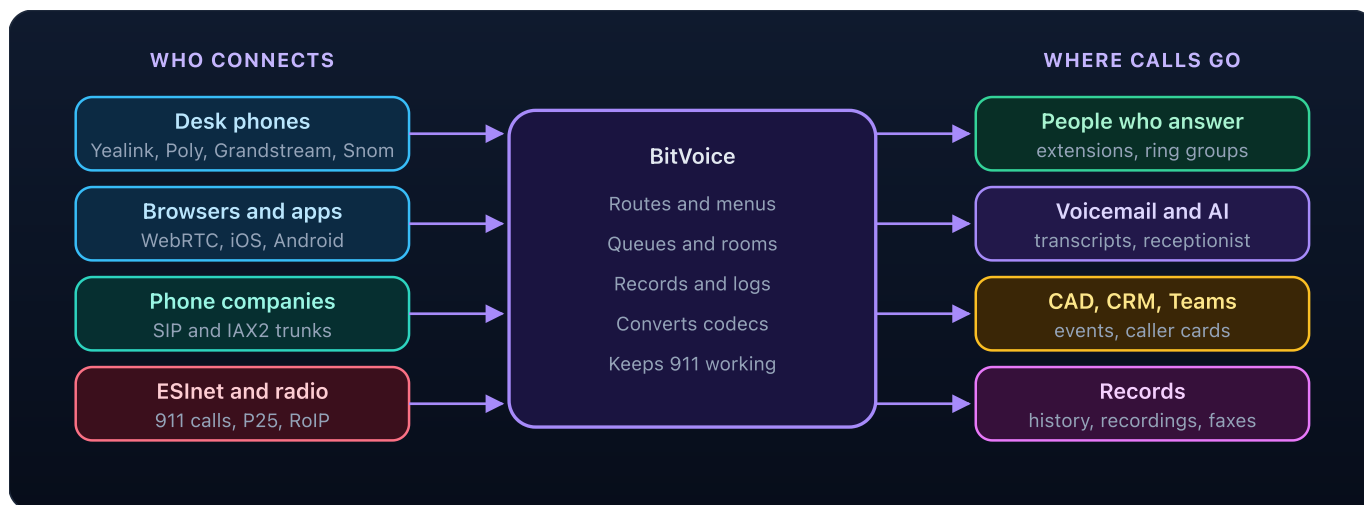


Figure 1. Where BitVoice sits: phones and carriers on one side, people and systems on the other

What you need

- **A computer to run it on:** a small office server, a virtual machine, a cloud instance, or for a first look your own laptop. Linux, macOS and Windows all work. A Raspberry Pi 4 runs a small office.
- **Phones:** desk phones from Yealink, Poly, Grandstream or Snom are set up automatically; any other SIP phone or softphone works with its own settings; and every browser is a phone too.
- **A trunk** (an account with a phone company) to call outside numbers and be called. You can start without one: extensions call each other and the echo test.

The first start

Start BitVoice and open the address it prints (on a desktop it opens the browser itself). The web interface is **HTTPS only**, on this computer too. Until you install a certificate of your own (Settings, Certificate), BitVoice uses one it made itself, so the browser

warns once: choose to continue to the site. The **setup wizard** asks four things: the organisation's name and country, the first administrator's password, the first extensions, and your phone company if you have one. The country sets the emergency numbers, the tones callers hear and the way phone numbers are written.

Choose the country carefully. It decides which numbers are emergency numbers (911 and 933 in the US, 112 and 999 elsewhere), and emergency numbers always work from every extension.

The first hour

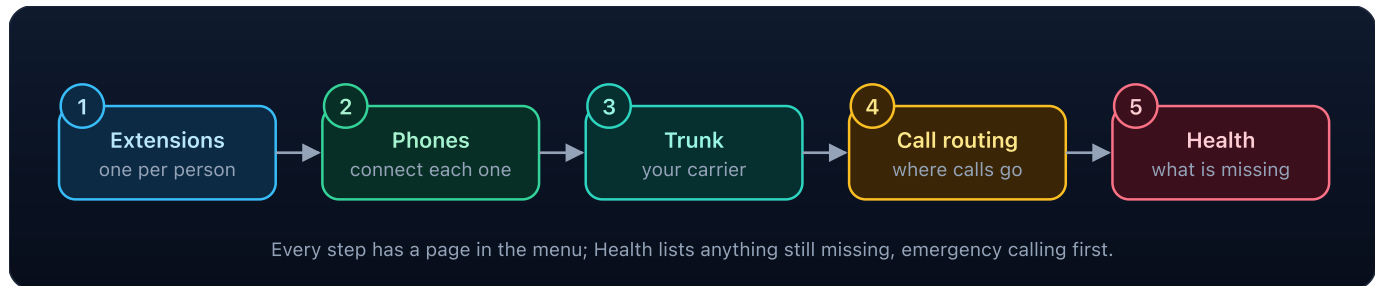


Figure 2. The first hour: five steps from a new install to a working phone system

1. **Extensions:** one per person or desk. Each has a number and a SIP password. [Open Extensions](#) (chapter 3)
2. **Phones:** connect a desk phone, softphone or the browser phone to each extension. [Open Phones](#) (chapter 4)
3. **Trunk:** your phone company account, so you can call outside numbers and be called. [Open Trunks](#) (chapter 7)
4. **Call routing:** where calls from outside go (a person, a ring group, a menu), and which trunk outside calls use. [Open Call routing](#) (chapter 8)
5. **Health** tells you what is still missing, emergency calling first. [Open Health](#) (chapter 13)

Then make a test call: from any phone dial ***43**, the echo test. You hear yourself back, which proves sound works both ways.

Installing it in one command

On a Linux server (Ubuntu, Debian, RHEL, Fedora, Alpine with systemd) or a Mac:

```
curl -fsSL https://raw.githubusercontent.com/BioDream-LLC/bitvoice/main/deploy/install.sh | sh
```

On Windows, in PowerShell:

```
irm https://raw.githubusercontent.com/BioDream-LLC/bitvoice/main/deploy/install.ps1 | iex
```

It downloads the latest release for the computer, checks it against the release's SHA-256 sums and BioDream's release signature (with OpenSSL 3, which current Linux, macOS and Git for Windows have), and installs it as a service that starts by itself:

- **Linux:** `/usr/local/bin/bitvoice`, a `bitvoice` system user, data in `/var/lib/bitvoice`, the systemd service `bitvoice` with the web interface on port 443. When ufw or firewalld is on, it opens 443, SIP (5060, 5061) and the calls' audio ports (10000 to 20000 UDP). It prints the address to open and the setup code the wizard asks for from another computer.
- **macOS:** `~/Applications/BitVoice`, started when you sign in, the web interface on port 8443. It runs under `caffeinate`, so the Mac does not go to sleep while BitVoice runs: a sleeping Mac answers no calls, and even its short maintenance sleeps stop everything for many seconds. The screen may still turn off. Run by hand, use `caffeinate -i -s ./bitvoice` for the same.

- **Windows:** as administrator, `C:\Program Files\BitVoice`, started with Windows, data in `C:\ProgramData\BitVoice`, and a firewall rule for private and domain networks; otherwise for you alone, started when you sign in. It works in Windows PowerShell 5.1 (what Windows Server has) and in PowerShell 7. A cloud server's network is one Windows calls Public, where that rule lets nothing in: the installer says so and prints the rule to add (and open the ports in the provider's firewall too). Windows has no OpenSSL, so there the release signature is checked only when one is installed (Git for Windows brings it); the SHA-256 sums always are. Tried on Windows Server 2022 on Amazon EC2: installed, reached from outside, running again after a restart.
- **Linux on a cloud server:** the address it prints is the machine's own, private one; the installer says so. Open the server by its public address or name, with 443, SIP and the audio ports open in the provider's firewall.

Run the same command again to upgrade (or use [Updates](#) (chapter 60)). `sh -s -- --version v0.1.5` (after the pipe) picks a release; `--uninstall` removes the program and service and keeps the data, `--purge` removes the data too. On Windows, save the script and run `.\install.ps1 -Version v0.1.5, -Uninstall` or `-Purge`. For a server kept off the internet, copy the release's archive and `SHA256SUMS` into a folder and set `BITVOICE_FROM` to it.

Where it runs

BitVoice runs on Linux, macOS and Windows.

System	How to start it	Where its data is
Linux	<code>./bitvoice</code> (or the systemd service in the release)	<code>~/.config/BitVoice</code>
macOS	<code>./bitvoice</code>	<code>~/Library/Application Support/BitVoice</code>
Windows	double-click <code>bitvoice.exe</code>	<code>%APPDATA%\BitVoice</code>

On Windows, keep the window it opens while you use BitVoice (closing it stops BitVoice), and when Windows asks whether it may use the network, allow private networks so phones can connect. `-data` puts the data anywhere else; `-https` and `-sip` choose the addresses and ports.

The ports it uses

Port	What for
5060 UDP and TCP	SIP: phones and trunks
5061 TCP	SIP over TLS (encrypted signalling)
4569 UDP	IAX2 trunks, only once one is set up
1720 TCP	H.323 trunks, only once one is set up
2000 TCP	Skinny (SCCP) phones, only when switched on
2727 UDP	MGCP gateways, only when switched on
10000–20000 UDP	the sound of calls (RTP), set in Settings
8443 TCP	this web interface, HTTPS only (plain HTTP to it is sent on to HTTPS)

If another phone system already has 5060, BitVoice takes the next free port and Health says so.

Everything in the web interface

Every setting has a screen, and every screen has a chapter here. Nothing needs a configuration file, a command line or a second tool. For automation, the same screens are a REST API, and the configuration can follow a file in Git ([Configuration as code](#) (chapter 67)), but neither is ever required.

What BitVoice is for

BitVoice is first a phone system for offices and contact centres, and a hosted phone service for providers: everything in the first five parts of this manual. The 911 answering-point profile, the radio dispatch console and NG911 Core Services are a separate **lab and design-partner track** ([Public safety](#) (chapter 74)): they are off the menu until switched on in Settings, and not certified for live use. Emergency calls from your own phones are not part of that track; they work on every phone system from the first start.

Finding your way

- Press **⌘K** (Ctrl+K) anywhere to jump to a screen, or type an extension's name or number to open it.
- Every screen has a **Help** link at the top right that opens its chapter of this manual.
- **Changes** keeps every version of the configuration: you can see what changed and undo it.
- Three themes (Midnight, Dark and Light) are at the foot of the menu.

CHAPTER 2

How mature each part is

How far each part has been proven, plainly. **Field-tested**: run against real systems over a real network (Amazon EC2, the internet, real providers' services). **Checked against a reference**: tested against an independent implementation or the standard's own data. **Software-tested**: BitVoice's own tests and browser tests only, not yet with real equipment or accounts. **Lab**: public-safety parts, not certified.

Area	Maturity	What it rests on
SIP trunks: registration, calls in and out, caller ID, busy, keys (RFC 4733)	Field-tested	Asterisk as the carrier across the internet on EC2; MOS 4.4. No commercial carrier account tried yet
Desk phones over SIP	Field-tested	Asterisk as a desk phone across the internet. Real Yealink, Poly or Grandstream handsets not tried yet
Browser phone (WebRTC)	Field-tested	from outside to EC2; Chromium browser tests with sound both ways
Two servers, one PostgreSQL, failover	Field-tested	kill -9 and planned stop on EC2: the call carries on, the provider line moves
One-command install	Field-tested	Ubuntu 24.04 (x86, Arm), Debian 12, Amazon Linux 2023, Windows Server 2022, macOS; back after a reboot
Capacity	Field-tested	200 calls at once, no loss, on c7g.large, c7i.large, t4g.small, an M4 and the Lightsail nano
FreePBX migration (Import)	Field-tested	a real FreePBX 17.0.33 backup (70 MB) imported whole; others from FreePBX's own formats
AI transcripts, summaries, receptionist	Field-tested	a real model through OpenRouter
Archive to S3, secrets from AWS Secrets Manager	Field-tested	real AWS, our own SigV4
SNMP monitoring and traps	Field-tested	net-snmp's snmpwalk and snmptrapd
Let's Encrypt	Field-tested	issues and renews the public demo's certificate
Machine images (AWS) and configuration as code	Field-tested	an EC2 image built by <code>deploy/cloud/build-images.sh</code> from Ubuntu 24.04: each server from it starts new; <code>deploy/terraform</code> with OpenTofu 1.13 (create, change, no drift, destroy). Azure images not tried
IAX2 trunks	Checked against a reference	Asterisk 23
H.323 trunks	Checked against a reference	H323Plus 1.28 (its simph323 playing and recording WAV files) on EC2: calls both ways with Fast Connect and G.711, each end hearing the other's tone, MOS 4.39; call set-up also with Asterisk's chan_ooh323 (whose own sound is broken in Asterisk 22)
SIP over WebSocket	Checked against a reference	JsSIP and SIP.js
Codecs: GSM, G.726, iLBC, MP3 hold music	Checked against a reference	ffmpeg, the RFC 3951 reference program and minimp3, sample for sample
Codecs: G.711, G.722, Opus, G.729, AMR, Speex	Software-tested	own round-trip tests; Opus and G.729 through their pinned Go libraries; Opus calls with the Android app

Area	Maturity	What it rests on
ISDN PRI and BRI, T1/E1	Checked against a reference	libpri (Asterisk's), both sides; and on EC2 through the real DAHDI 3.4 kernel driver (two spans wired together in the kernel, Asterisk 20's chan_dahdi as the phone company): calls both ways with numbers and names, all 23 channels at once, MOS 4.39. No physical card or line yet
MFC-R2, E&M	Checked against a reference	OpenR2; MF tones by our own tests. No real card or line tried
SS7 (ISUP, MTP, link sets of several links with changeover)	Checked against a reference	libss7 (Asterisk's chan_dahdi library), ITU and ANSI. No real carrier link tried
Fax (T.30, T.38)	Software-tested	own tests between BitVoice ends; no fax provider or gateway tried
Provisioning: Yealink, Poly, Grandstream, Snom, Fanvil, Cisco, Mitel, Sangoma, Algo, CyberData	Software-tested	the vendors' documented formats; not on real handsets
Skinny (SCCP) phones, MGCP gateways	Software-tested	own tests; no Cisco phone or gateway tried
Microsoft Teams Direct Routing	Software-tested	own tests; no Teams tenant tried
Hotel (FIAS)	Software-tested	own tests; not against Opera
Texting, WhatsApp, Messenger, web chat	Software-tested	stand-ins for each provider's API
CRM caller cards, Azure Blob, Azure Key Vault	Software-tested	stand-ins
iPhone and Android apps	Field-tested (Android)	Android: a Google Pixel 11 (Android 17) in AWS Device Farm against BitVoice on EC2 with a Let's Encrypt certificate: paired by link, rang, answered from the notification, called out; Opus, MOS 4.1, about 5% of packets lost between the phone lab and the server. iPhone: Simulator only; push wake-up needs our own Apple and Firebase accounts
911 profile, radio console, NG911 Core Services, caller video	Lab	NENA's schemas and sample data, Wireshark, spandsp, OpenSSL, ExifTool, ffmpeg, a NIST P25 capture; never on a live ESInet, P25 system or PSAP CAD, and not certified
P25 Phase 2 half-rate voice (AMBE+2)	Not offered	patented until 2028
Ed448 signatures (NENA)	Not offered	not in Go's standard library, and BitVoice writes no cryptography of its own; Ed25519 is used

Area	Maturity	What it rests on
SS7 signal transfer (STP), SCCP/TCAP	Not offered	a PBX is an end point; number portability and toll-free lookups are the carrier's
MFC-R2 China variant	Not offered	OpenR2's own China variant disagrees with itself, so there is nothing sound to test against
ALI over PAM or E2	Not offered	through a gateway; BitVoice speaks the request-and-text ALI links

CHAPTER 3

Extensions

In the web interface: `/extensions`

An extension is a number for a person or a desk. Phones sign in with the extension number and its **SIP password**; calls to the number ring every phone signed in as it, at once.

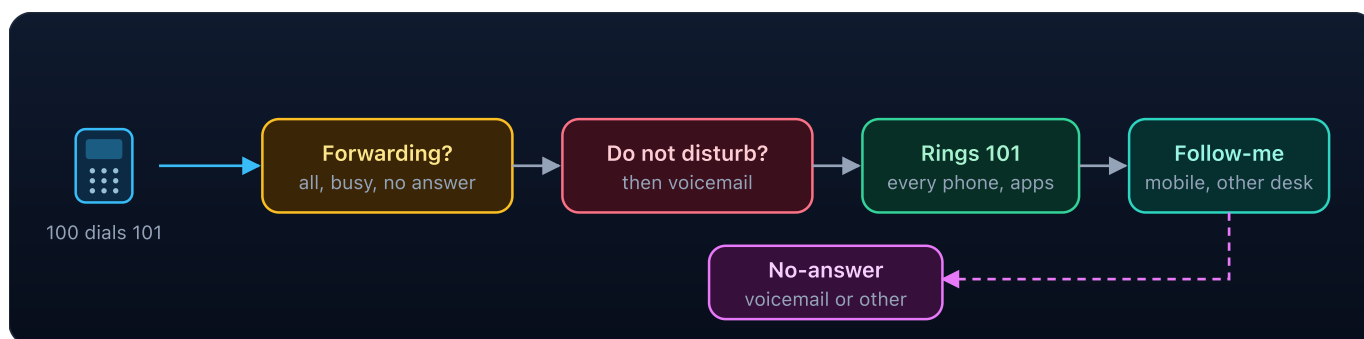


Figure 3. A call between two extensions

Adding an extension

1. Open [Extensions](#) (chapter 3) and press **Add extension**. The next free number and a strong SIP password are filled in.
2. Enter the person's **name** (shown on other phones when they call) and **email** (for voicemail by email).
3. Fill in the **911 location**: the site, floor and room. It is sent with emergency calls.
4. Choose what happens **If nobody answers**: voicemail by default, or another extension, a group or a menu.
5. Save. The extension can sign in at once.

The fields

Field	What it does
Number	2 to 8 digits; no other extension, group, queue, room or feature code may have it
SIP password	what the phone signs in with; at least 12 characters; Generate makes a new one
Ring for	seconds before If nobody answers happens (25 by default)
Outside caller ID	the number shown on calls out; empty uses the trunk's
Can call outside numbers	off keeps the phone to internal calls and emergency numbers
Record calls	when Settings records only selected extensions
Hot desk	a shared phone that becomes whoever signs in at it
Encrypt calls to this phone	media encryption (SRTP); use it with TLS so the keys stay private
Department	groups this extension's call costs

Fill in the 911 location for every desk. An emergency call carries it to the answering point, and the RAY BAUM'S Act requires a dispatchable location for calls from offices in the US.

Follow-me, pickup and intercom

Each person's own call handling is in its own box on the extension:

- **Follow-me** rings more numbers when the phones here are not answered: a mobile, another desk. They ring one after another, or all together, for their own ring time. With **Outside numbers press 1 to take the call** on, whoever answers on a mobile presses 1 first, so the mobile's own voicemail never takes the call. See [Feature codes](#) (chapter 31).
- **Pickup group**: anyone in the same group answers this extension's ringing calls by dialling ***8**.
- **Allow intercom**: ***80** and this number answers on the speaker by itself. Switch it off for people who never want that.
- **Own speed dials**: short codes for this person only, dialled after ***0**.

Forwarding right now

Below the settings, **Forwarding right now** shows and changes what the person set from their phone or the staff portal: do not disturb, and forwarding of every call, of calls while busy, and of calls not answered. These are live state, not configuration: changing them makes no revision.

What is connected

- **Phone**: a desk phone, softphone or app.
- **Door station**: an intercom at a door. Whoever answers presses the key code, or **Open door** on Live calls. With a door controller address, BitVoice calls that address (2N, Axis, Akuvox, relay boards); without one, the code is sent to the station, which opens the door itself.
- **Overhead speaker**: calls to it are answered at once and heard over the speaker.
- **Analog adapter port**: an old phone, fax or alarm panel on an adapter. Calls use G.711 only, which faxes need.

Connected devices

An extension that is open shows every device signed in as it: the phone's make and model, how it connects (UDP, TCP, TLS or the browser) and since when. The address and port to give a phone are shown there too.

CHAPTER 4

Connecting phones

In the web interface: `/phones`

Any SIP phone or app works: point it at this server's address and port 5060, with the extension number as user and its SIP password. Desk phones from the four big makers are set up automatically instead, so nobody types a password into a phone's keypad.

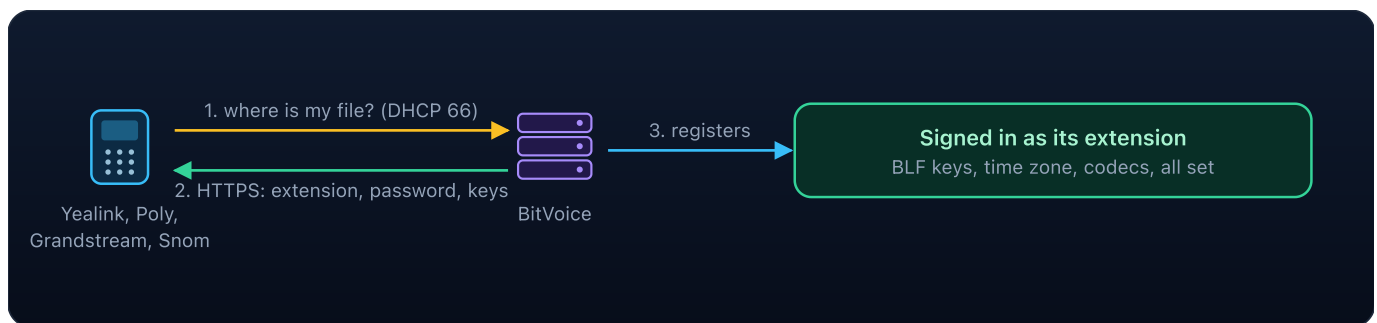


Figure 4. Provisioning: a desk phone fetches its settings by itself

Automatic set-up (provisioning)

1. On [Phones](#) (chapter 4) press **Add phone** and enter the phone's **MAC address** (on a sticker underneath) and the **extension** it belongs to.
2. Choose the make: Yealink, Poly, Grandstream (including the HT analog adapters), Snom, Fanvil, Cisco, Mitel, Sangoma, Algo or CyberData.
3. Tell the phone where BitVoice is: in your DHCP server, set **option 66** to the provisioning address shown on the screen, so every phone on the network finds it by itself; or type the same address in the phone's own web page, under auto provisioning.
4. Restart the phone. It fetches its file over HTTPS, signs in, and its line keys show who is on the phone.

HTTPS or plain HTTP. Phones fetch their file over HTTPS when they can. Some older phones cannot use HTTPS at all, and many do not trust the certificate BitVoice made itself; for those, the screen also shows a plain `http://` address on the same port. Plain HTTP is only for these files: every other page still goes to HTTPS. Over plain HTTP the phone's SIP password crosses the network unencrypted, so keep it to a trusted office network, or install a certificate from an authority and switch on **HTTPS only for phones**.

The file carries the extension, its password, the time zone, the codecs, busy lamp keys and the address of this server. Provisioning is protected by a user name and password the phones are given once.

Make	Models tested	What it gets
Yealink	T3x, T4x, T5x	account, BLF keys, time, codecs
Poly	VVX, Edge E	the same, in Poly's own format
Grandstream	GRP, GXP, HT802/814	account per port on adapters
Snom	D7xx	account, keys, time

These makes are provisioned too, with files written from each maker's documented format but **not yet tried on their devices** (the Phones screen marks them untested):

Make	Models	File the device asks for
Fanvil	X and V series	its MAC address .cfg
Cisco	6800, 7800, 8800 multiplatform (3PCC), SPA adapters	its MAC address .xml (enterprise firmware needs the multiplatform firmware first)
Mitel (Aastra)	6800, 6900 SIP	its MAC address .cfg, in capitals
Sangoma	S-series	the Fanvil-format file
Algo	paging speakers, strobes, adapters	algot and its MAC address .conf
CyberData	speakers, intercoms, paging adapters	its MAC address .xml

Each gets its account, the server (and the backup server, with several servers), the transport and encryption, and the voicemail number where it has one. Set Algo and CyberData speakers' extensions to **Overhead speaker** so calls to them are answered at once.

Phones without provisioning

Softphones and other makes take the same three things: server (this computer's address, port 5060, or 5061 with TLS), user (the extension number) and password (its SIP password). Health lists phones that tried a wrong password; after a few tries the address is banned for a while (see [Security](#) (chapter 58)).

The browser phone

The phone icon at the bottom of every screen makes this browser a phone for an extension, with your microphone and speakers. It uses WebRTC: the sound is encrypted end to end, and it works through most firewalls without any setting.

Other WebRTC softphones (SIP over WebSocket)

Softphones and web apps built on JsSIP, SIP.js and similar libraries speak SIP over WebSocket (RFC 7118). Switch on **SIP over WebSocket** in [Settings](#) (chapter 14) and point them at `wss://<this server's name>:<web port>/ws`, with the extension number as the user and its SIP password. BitVoice ends their WebRTC sound itself (ICE, DTLS-SRTP, Opus or G.711), so they reach every phone, trunk and feature as desk phones do: calls both ways, keys pressed, hold, transfers by the other party, busy lamps and message-waiting. Settings shows how many are connected.

The WebSocket needs no sign-in of its own: the softphone signs in with SIP, and failed sign-ins and too many requests are counted for its own address, as for any phone. A softphone page served from another site must trust this server's certificate (use

Let's Encrypt), and current Chrome asks the user before a public web page may reach a server on the local network.

Tested with two independent libraries, JsSIP 3.13 and SIP.js 0.21: two softphones of each register, call each other and hear each other through BitVoice. Other softphones (Linphone's web client, Zoiper's) speak the same protocol but have not been tried.

TURN, for networks that block UDP

Some networks (hotels, guest Wi-Fi, strict company firewalls) let no UDP out, or hide computers behind NATs. WebRTC cannot get through. For those, switch on **TURN server** in [Settings](#) (chapter 14): BitVoice then runs its own TURN server (RFC 8656) on port 3478 (UDP and TCP) and, with this server's certificate, on port 5349 over TLS. Browser phones, the staff phone and call links are given short-lived credentials for it, and use it when they cannot reach the media ports directly; the sound is still encrypted end to end, and BitVoice relays it only to its own addresses, so nobody can use it to reach anything else.

Open the TURN port, the TLS port and the relay ports (49152 to 49407 unless you change them) in the firewall. TURN over TLS needs a certificate browsers trust for the name they use, such as one from [Let's Encrypt](#) (chapter 14): until there is one, BitVoice does not offer it to browsers (they would try it, fail and wait), and offers it as soon as the certificate arrives; set its port to 443 on a server with a spare address to get through firewalls that allow only HTTPS.

The staff phone

Open `#/phone` on any computer or mobile browser: people sign in with their extension and SIP password, without an administrator account. It is a full phone: dial, answer, hold, transfer and the keypad. For their own forwarding and voicemail they use the staff portal at `#/me` (see [The staff portal](#) (chapter 32)).

The mobile apps

The BitVoice apps for iPhone and Android are paired from the extension's page with a QR code. Calls wake the app through Apple's and Google's push services, so it rings even when it is closed.

CHAPTER 5

Skinny phones and MGCP gateways

In the web interface: `/desk-phones`

Phones and gateways that do not speak SIP can still be used. **Skinny phones and MGCP gateways** (under Calls) takes two older protocols:

- **Skinny (SCCP)**: Cisco desk phones running SCCP firmware (79xx, 69xx, 89xx). Add each phone by its MAC address and choose its extension; switch on **Accept Skinny phones** and point the phone at BitVoice (TCP port 2000) in its TFTP settings file or menu.
- **MGCP** (RFC 3435): gateways whose ports BitVoice drives one by one, such as analogue telephone adapters and line cards. Add the gateway by the domain name it uses in its endpoint names (aaln/1@name), its address, and the extension of each line; set the gateway's call agent to BitVoice (UDP port 2727).

Each Skinny phone or MGCP line is one more device of its extension: it rings with the extension's other phones, calls out as the extension, and shows in the extension's devices. The page shows each one's state.

What works

- Skinny: registration, the line and its number on the display, dial tone, dialling (the number goes after a pause or #, or at once from the phone's dial pad when lifted after dialling), ringback, ringing with caller ID, answer and hang up by handset, speaker or soft keys (NewCall, Redial, Answer, EndCall), busy and reorder tones.
- MGCP: dial tone, digits collected by the gateway with a digit map, ringing with caller ID (rung without it when the gateway refuses caller ID), ringback, busy and reorder tones, and the audio connection (CRCX, DLCX). A gateway that restarts (RSIP) has its ports set up again.
- Audio is G.711 μ -law on both; BitVoice converts to whatever the other party uses.

Hold, transfer and conference are done from the extension's other phones for now, and a Skinny phone's second line is not used.

Security

Neither protocol has passwords. Skinny phones are accepted only from the local network and only when listed; MGCP commands are obeyed only from the gateway's address (or, without one, from the local network). Keep both ports off the internet.

Both are tested against BitVoice's own test phone and test gateway, which follow the protocols, but not yet against Cisco phones or gateway hardware.

CHAPTER 6

Telephony cards

In the web interface: `/cards`

A telephony card in the BitVoice computer (Digium, Sangoma, OpenVox and other cards with Linux DAHDI drivers) can carry analogue phones and phone company lines. A separate program, **bitvoice-dahdi**, drives the card and joins its ports to BitVoice over the computer's own network connection, so BitVoice itself never touches the kernel driver and a card problem cannot stop the phone system.

Setting it up

1. Install the DAHDI drivers and set the ports up as usual in `/etc/dahdi/system.conf` (`fxoks=1-4` for ports with phones, `fxsks=5-8` for ports with lines, and the country's tones in `loadzone` and `defaultzone`), then run `dahdi_cfg`.
2. In **Telephony cards** (under Calls) switch on **Use telephony cards**. A token is made; copy it into a file only root can read, such as `/etc/bitvoice/cards.token`.
3. Add each port by its DAHDI channel number: a **phone** port (FXS, a phone, fax or door phone plugged in) with its extension, or a **line** port (FXO, a phone company line) with the line's number. Choose A-law for Europe and most of the world, μ -law for North America and Japan.
4. For lines, add a trunk with the protocol **Telephony card lines** on Trunks, and route numbers to it as to any trunk; inbound routes match each line's number.
5. Start `bitvoice-dahdi -server https://127.0.0.1:8443 -token-file /etc/bitvoice/cards.token` as a service user that may open `/dev/dahdi`. It is in the Linux download beside bitvoice. The page shows whether it is running and each port's state; changes reach it within ten seconds. A channel another program still holds (or one `dahdi_cfg` has not set up yet) is tried again every two seconds, so the order things start in does not matter.

What works

- Phones: dial tone, dialling (digits are heard by BitVoice's own tone detector; the number goes after a pause or #), ringback, busy and congestion tones from the card's tone zone, ringing for incoming calls, and keys pressed during a call (sent on as RFC 4733 events too).
- Lines: a ringing line calls its number here and is answered when the call is; calls out take a free line, wait for the exchange's dial tone and dial the number as tones. The far end hanging up is noticed when the line reports it (loop current or polarity, depending on the line). Analogue lines do not say when the far end answers, so a call out counts as answered once the number is dialled.
- Sound is G.711 to and from the card; BitVoice converts to whatever the other party uses.

Digital lines: T1 and E1 PRI

A T1 (23 calls at once) or E1 (30) line run as primary rate ISDN plugs into a digital card (Digium/Sangoma TE-series, OpenVox D-series and the like). BitVoice speaks the PRI itself (Q.921 on the D-channel, and Q.931 call control), so every call comes in with the number dialled and the caller ID, goes out with the extension's caller ID, and the phone company says when the far end answers, rings or is busy.

1. In `/etc/DAHDI/system.conf`, set the span up for PRI as usual (`span=1,1,0,esf,b8zs` , `bchan=1-23` , `hardhdlc=24` for a T1; `span=1,1,0,ccs,hdb3,crc4` , `bchan=1-15,17-31` , `hardhdlc=16` for an E1), and run `dahdi_cfg` .
2. Under **Digital lines** on [Telephony cards](#) (chapter 6), add the span: T1 or E1, its first DAHDI channel, whether BitVoice is the PBX on the phone company's line (the usual) or the network (another PBX connects to BitVoice as if to the phone company), and the switch type the phone company gave you.
3. Add the card trunk on Trunks if there is none yet, and inbound routes for the numbers (DIDs) the phone company sends.

Calls out take a PRI first, the analogue lines after; a busy number, a wrong number or no free channel comes back as busy, not found or unavailable, as the network says. The page shows whether each span's D-channel is up and how many of its channels are in calls. T1 channels set up with robbed-bit signalling (`fxsks` , `fxoks`) work as the analogue ports above.

Names, forwarding and transfers. A caller's name comes in with the call (from the Display element, or on National ISDN-2 from the name the network sends as a facility) and phones show it; calls out send the extension's name the same way. The number a call was forwarded from comes in as its redirecting number (BitVoice passes it on as SIP's Diversion, with the reason: busy, no answer, unconditional), and when an extension's forwarding sends a call back out on the PRI, the extension's number goes with it as the redirecting number, so the far end and the phone company know whose call it is. When an outside call is answered, BitVoice tells the network the number that answered (the connected number).

Switch on **Transfers through the phone company** on a span (user side, National ISDN-2, 5ESS, DMS-100 or EuroISDN) and when someone transfers a caller on that line to another number on the same line, BitVoice asks the network to join the two calls itself: Two B-Channel Transfer on National ISDN-2 and 5ESS lines, Release Link Trunk transfer on a Nortel DMS-100, explicit call transfer (the caller held, then joined) on EuroISDN. A DMS-100 gives a transfer identifier only to calls BitVoice places, so there one of the two calls must be a call out (the usual case: a caller forwarded or transferred to an outside number). Both channels are freed as soon as the network has joined them. The line needs the service (ask the phone company for "two B-channel transfer" or "ECT"); if the network refuses, the transfer still works, through BitVoice, using both channels as before. On a span where BitVoice is the network side, another PBX may ask BitVoice to join two of its calls; BitVoice declines (it is not a switch), and the PBX keeps the calls itself.

Trying it without a card

DAHDI's own dynamic spans can stand in for a card: two spans joined inside the kernel, BitVoice on one and another PBX (Asterisk's `chan_dahdi`, say) as the phone company on the other. Load `dahdi` , `dahdi_dynamic` and `dahdi_dynamic_loc` , and in `/etc/DAHDI/system.conf`:

```
dynamic=loc,1:0,24,0
dynamic=loc,1:1,24,0
bchan=1-23
dchan=24
bchan=25-47
dchan=48
```

Leave the last number (timing) at 0 on both: DAHDI's own timer then drives them. Spans that are each other's timing source tick only when the other sends, and the D-channel comes and goes. Add span 1 here as a T1 (first channel 1), and set the other PBX up as the network side on channels 25 to 48.

Basic rate ISDN (BRI)

A BRI carries two calls at once (B1 and B2) and is common in Europe for small offices; it also connects ISDN desk phones. BRI cards (Digium B410P, the HFC-based cards from OpenVox, Junghanns and others) work through DAHDI as the PRI cards do: in `/etc/dahdi/system.conf` the span is `span=1,1,0,ccs,ami` with `bchan=1-2` and `hardhdlc=3`. Add it under **Digital lines** with **BRI** as the line type; its channels are B1, B2 and then the D-channel.

- **BitVoice is the PBX (user side), point to point:** a PBX line from the phone company (Germany's **Anlagenanschluss**): calls in carry the number dialled, calls out the extension's.
- **User side, multipoint:** a line shared with other devices (**Mehrgeräteanschluss**): BitVoice asks the network for its own TEI (terminal identifier) and answers calls to the line beside any other devices on it; the phone company sends the number dialled (the MSN), which the inbound routes match.
- **BitVoice is the network, multipoint:** ISDN phones plug into the card's S-bus as onto a phone company line. Each asks BitVoice for a TEI; a call to the bus rings every phone on it, the first to answer takes it and the others stop ringing. Numbers dialled digit by digit (as ISDN phones dial) are collected until the phone says it is complete, # is pressed, or no digit comes for four seconds. Route the numbers to and from the bus with the card trunk as for any line.

Names, forwarding, transfers on EuroISDN (explicit call transfer) and advice of charge work on a BRI as on a PRI.

E&M trunks on a T1

A T1 whose channels are set up as E&M (robbed-bit signalling: `e&m=1-24` in `/etc/dahdi/system.conf`) gives 24 trunks, each signalled on its own: tie lines to another PBX, channel banks, and long-distance carriers' Feature Group D trunks. Under **Digital lines**, add a T1 and set **Signalling** to **E&M**, then:

- **Start signal: Wink start** (the usual: the far end winks when it is ready for the number, and BitVoice winks back at a call coming in), **Immediate start** (the number follows the seizure at once), **Delay dial** (the far end goes off hook while it gets ready, then on hook), or **Feature Group D** (wink start with MF numbers, the caller's number sent first: KP 00 ANI ST, then KP number ST; the ANI becomes the caller ID).
- **Numbers sent as:** DTMF, or MF (MF R1, KP number ST) as most carrier trunks want.
- **Number when none is sent:** what a call in reaches if the far end seizes the trunk without dialling (a private line straight to the front desk, say).

A call out takes a free trunk, seizes it, waits for the start signal (no wink within five seconds is "no wink", and the call fails), sends the number, and is answered when the far end goes off hook (answer supervision). A call in is answered by going off hook when the extension answers.

CAMA trunks (911, lab track). A PSAP still fed by a 911 selective router over CAMA trunks sets **Start signal** to **CAMA (911 trunk)**. The router seizes, BitVoice winks, and the router sends the caller's number (ANI) in MF: enhanced CAMA as KP, two information digits and all ten digits, ST; traditional CAMA as KP, one numbering plan digit and seven digits, ST. For traditional

CAMA, list under **Area codes for NPD 0 to 3** the area code each numbering plan digit stands for (your router's set-up sheet has them); a digit above 3 means the router could not identify the line. Each call goes to the inbound route for **911** as a 911 call, with the ANI as its callback number, and BitVoice looks its location up by that number (see [911 calls](#) (chapter 74), **Location lookup (ALI)**). The information digits show on the call as **CAMA (II 00)**.

MFC-R2 on an E1

In Latin America, parts of Asia and Africa, E1 lines are often MFC-R2 rather than ISDN: 30 circuits, each with its line signals on the A and B bits of timeslot 16 (ITU-T Q.421) and its number, the caller's category and the caller's number exchanged in compelled multifrequency tones (Q.441), forward and backward. Set the span up as CAS in `/etc/dahdi/system.conf` (`span=1,1,0,cas,hdb3` , `cas=1-15:1101` , `cas=17-31:1101` , `dchan` removed), then under **Digital lines** add an E1, set **Signalling** to **MFC-R2** and choose the **Country variant**: ITU-T, Argentina, Brazil, Colombia, Czech Republic, Ecuador, Mexico, Philippines or Venezuela. Each country assigns some signals differently (what means busy, how the end of the number is told, Mexico's extra Group C); the variant holds those.

Every circuit carries calls both ways. A call in reaches the inbound routes with the number dialled and the caller's number (withheld numbers come in as anonymous); a busy extension or a wrong number is signalled back as busy or unallocated, as the variant says. A call out sends the number and the extension's caller ID, or the "number withheld" signal when the caller hides it. China's variant is not offered.

The MFC-R2 signalling is tested against OpenR2 (the R2 library Asterisk's `chan_dahdi` uses), in every variant offered and in both directions, including a call refused as busy; E&M against BitVoice's own simulated channels.

SS7

A carrier may hand over T1s or E1s signalled by SS7 rather than ISDN: the voice circuits carry calls, each named by a circuit identification code (CIC), and one timeslot carries the signalling link to the carrier's signal transfer point or switch, over which ISUP sets the calls up. BitVoice is then a signalling point of its own, with a point code the carrier assigns. In `/etc/dahdi/system.conf` set the span up as clear channels with the signalling timeslot as `mtp2` or `hardhdlc` (timeslot 16 on an E1, 24 on a T1). Under **Digital lines** add the span, set **Signalling** to **SS7 (ISUP)** and give:

- **SS7 variant**: ANSI (North America; point codes network-cluster-member, like 245-16-1) or ITU-T (most of the world; point codes zone-area-point, like 1-10-2, or a plain number).
- **Our point code** and **Adjacent point code**, the **Network** (national, the usual, or international), the signalling **Link code** (SLC, 0 unless told otherwise), and the **First CIC**: the code of the span's first voice channel, the others following in order. All of these come from the carrier.

Link sets of several links. Spans with the same variant, point codes and network are one link set (up to 16 links): each span's signalling link is one of its links, and needs its own **Link code**, and every span's circuits belong to it (their CICs may not overlap). Traffic is shared between the links in service by the messages' SLS, so one call's messages keep to one link. When a link fails (the link test goes unanswered, or messages go unacknowledged), its traffic changes over to the others (Q.704: COO and COA, the messages the far end had not got sent again on another link; or, with no answer in time, by time), the calls on every span go on, and **Telephony cards** says how many links are in service and how this span's stands. When the link is back and tested, its traffic changes back (CBD and CBA). Only when every link is down is the link set down. Tried against libss7 with two links, one cut in the middle of a call: libss7 takes network management messages only on the link their SLS names, so a changeover message about the cut link (sent on the other, as Q.704 has it) is dropped there, and both ends change over by time instead, which works; between two BitVoice ends it goes by COO and COA.

When the link comes up, BitVoice aligns it (MTP level 2, with the emergency proving period as a single-link link set uses), proves it with the link test (SLTM/SLTA), exchanges traffic restart allowed, and resets every circuit (GRS). Calls in reach the inbound routes with the number dialled, the caller's number (withheld ones as anonymous), the number a call was forwarded from and, on ANSI,

the caller's name; calls out carry the extension's caller ID and name. The carrier blocking or resetting circuits is obeyed. On dual seizure the end with the higher point code keeps the even circuits, as Q.764 says, and BitVoice takes the circuits it keeps first.

SS7 is tested against libss7 (the SS7 library Asterisk's chan_dahdi uses), ITU and ANSI: alignment, link test, traffic restart, circuit reset, calls both ways with numbers and names, release, busy and blocking; and BitVoice against itself with frames lost on the link (sent again by MTP2's error correction) and a dual seizure. Signal transfer (being an STP for others) and SCCP/TCAP (number portability lookups, toll-free translation) are not offered: a PBX is an end point, and those are the carrier's.

Advice of charge. On a EuroISDN line, switch on **Advice of charge** and BitVoice asks the phone company, with each call out, to say what it costs: during the call (AOC-D) and at its end (AOC-E), in money or in charging units. The last figure goes on the call's record (**charged EUR 1.50** in History, and beside the call on Costs); when it is in your currency (set on Costs) it is the call's cost, instead of the figure from the trunk's rates. The phone company must offer the service on the line. SIP trunks may send the same advice, as the 3GPP AOC XML in an INFO or the BYE, and BitVoice keeps it the same way.

The PRI code is tested against libpri (the ISDN library Asterisk uses), both as the user side and as the network side, for National ISDN-2 and EuroISDN calls, names, redirecting and connected numbers, explicit call transfer, the Two B-Channel Transfer and DMS-100 Release Link Trunk requests byte for byte (with libpri's PBX joined by BitVoice as the DMS-100), and advice of charge in both directions; and on BRI, point to point and multipoint, as the terminal and as the network (TEI assignment, calls broadcast to the bus, numbers dialled digit by digit).

Not yet: caller ID shown on analogue phones or read from analogue lines, hook flash, and echo cancellation beyond what the card itself does.

bitvoice-dahdi is tested against a simulated card, not yet against real card hardware. It must run on the same computer as BitVoice: it only talks to 127.0.0.1, and BitVoice answers its token only from there.

CHAPTER 7

Trunks

In the web interface: `/trunks`

A trunk is your account with a phone company. It carries calls to and from outside numbers. Most are SIP trunks; an Asterisk or FreePBX server, or an IAX2 provider, can be reached with IAX2 instead.

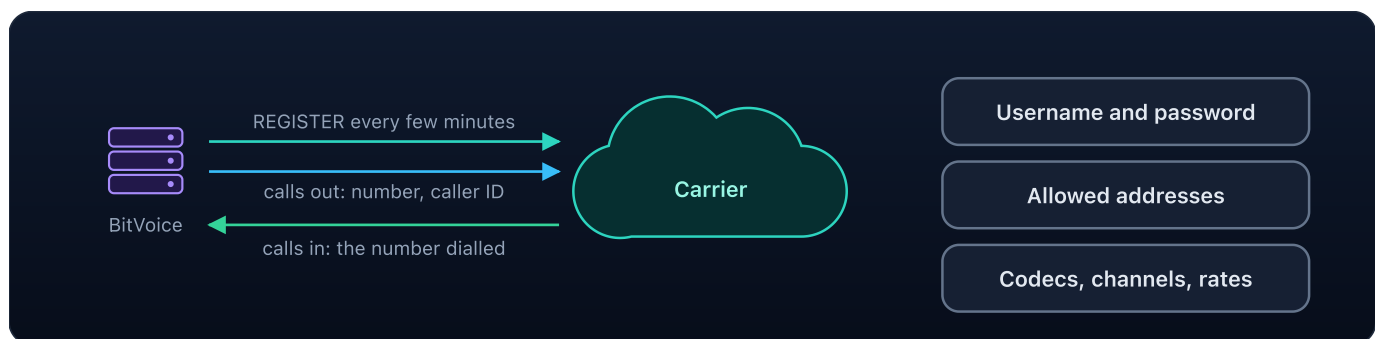


Figure 5. A trunk: your account with a phone company

Adding a SIP trunk

1. On [Trunks](#) (chapter 7) press **Add trunk**.
2. Enter the **provider server** your phone company gave you (`sip.provider.com`), and the port if it is not 5060.
3. Enter the **username** and **password**, and switch on **Register** if the provider says to (most do).

4. Save. Within a few seconds the trunk shows **registered**; if not, the reason is shown beside it.

The trunk health judge

With two or more trunks on an outbound route, BitVoice judges every 15 seconds whether each is used in its place or **tried after the other trunks**. A trunk moves behind the others when its provider stops answering the keep-alive checks, or when its last calls out failed one after another on the provider's side (**Failed calls in a row** within **Within (minutes)**, Settings, **Trunk health judge**), and another trunk is working. It is not moved on a failing keep-alive alone while calls through it still get through; and when every trunk fails at once, the likelier cause is this system's own internet link, so nothing is moved. A change needs two judgements in a row (30 seconds), so a trunk never flaps. A trunk is never taken out altogether: it is still tried when the others fail, so an emergency call always has every way out. The trunk's page shows the judgement and its reasons; a call that went out another way says so in its record.

1. Add an **incoming route** for your number and an **outgoing route** that uses this trunk ([Call routing](#) (chapter 8)).

Field	What it does
Transport	UDP (usual), TCP, or TLS for encrypted signalling
Register	sign in every few minutes, so the provider knows where to send calls
Authentication username	only when the provider gives a separate one
From domain	the domain in the caller's address, when the provider asks for one
Caller ID	the number shown on calls out; an extension's own caller ID wins
Also accept calls from	the provider's other addresses or networks that send calls in
Calls at once	the limit your plan allows; calls beyond it go to the next trunk in the route
Encrypt media	SRTP, for providers that support it
Codecs	leave it on *same as Settings*, or choose what this provider supports, best first
Rates	what calls cost by number prefix, for Costs (chapter 53)

A trunk that accepts calls by address alone should list only the provider's own addresses. Networks wider than a /16 are refused: they would let anyone on the internet pass as your provider.

Cloud providers

Start from fills in Microsoft Teams Direct Routing, Amazon Chime SDK Voice Connector or Azure Communication Services, with a note on what to set on their side. Teams is described in its own chapter ([Microsoft Teams](#) (chapter 65)).

IAX2 trunks to Asterisk and FreePBX

Choose **IAX2** as the protocol to join another Asterisk or FreePBX server, or an IAX2 provider. IAX2 carries signalling and sound on one UDP port, 4569, which makes it easy through firewalls.

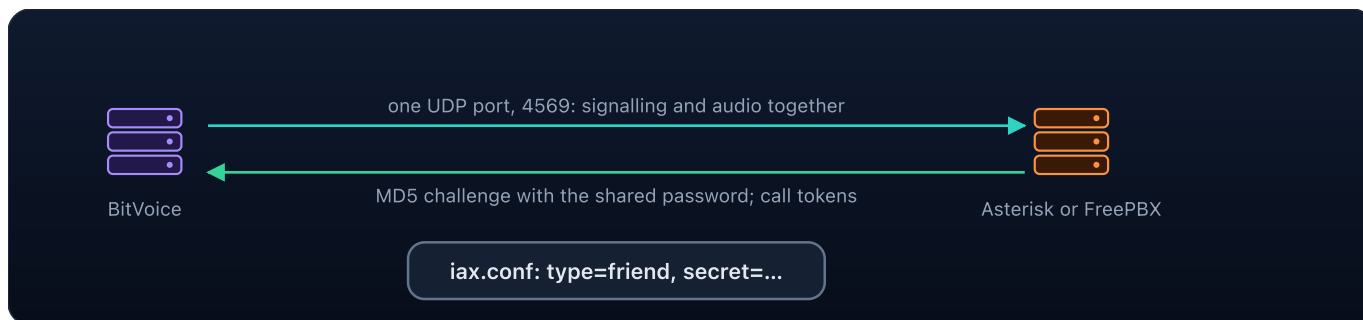


Figure 6. An IAX2 trunk to an Asterisk or FreePBX server

- **Server address:** the other server. Leave it empty when the other server registers to BitVoice instead (a branch office behind a changing address); then the username and password are what it registers with.
- **Username and password:** the same on both sides. BitVoice answers and sends MD5 challenges, and handles call tokens.
- **Register:** sign in to the other server, so it can send calls here.

On the Asterisk side, a peer like this is all it takes:

```
[bitvoice]
type=friend
host=dynamic
secret=the shared password
context=from-bitvoice
```

IAX2 port 4569 opens only when the first IAX2 trunk is added, so an Asterisk running on the same computer keeps it until then.

H.323 trunks

Choose **H.323** as the protocol for older gateways, PBXs and video-conference systems that speak H.323 but not SIP. Give the gateway's address (port 1720); calls are accepted only from it. Calls are set up with **Fast Connect** and G.711 (μ -law or A-law; the trunk's codec list may narrow it to one), and the sound goes straight between the gateway and BitVoice's media ports. BitVoice converts to whatever the other party uses.

H.323 here is deliberately small: no gatekeeper (RAS), no H.245 (so a gateway that will not use Fast Connect gets no call, and is told so), no video, no H.235 security. Caller ID comes from the Q.931 calling number or the caller's dialled-digits alias, the name from the display or its H.323 ID. BitVoice's H.225.0 and H.245 encoding was checked against an independent ASN.1 encoder, but not yet against gateway hardware.

What the status means

Status	Meaning
registered	signed in; calls can come in
ready	does not register, or IAX2 waiting for calls
unreachable	the server's name is not in DNS, or nothing answers
failed	the provider refused the sign-in; the reason is shown
standby	servers share this trunk one at a time and another holds it

Call routing

In the web interface: [/routes](#)

Call routing decides two things: where calls from outside go, by the number they dialed, and which trunk calls to outside numbers use.

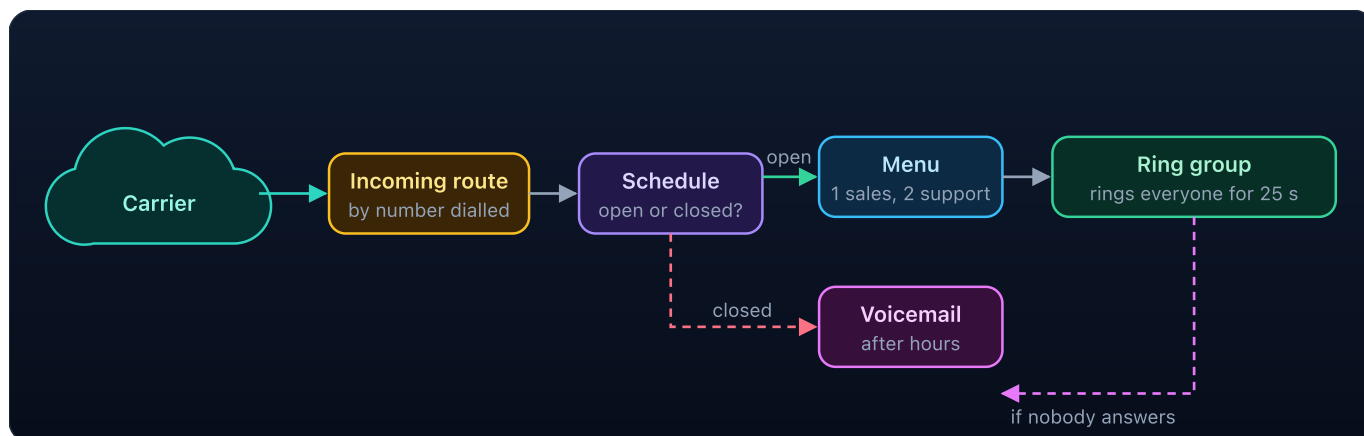


Figure 7. How a call from outside finds its way

Incoming routes

Each incoming route matches the number that was called (or any number) and sends the call to a destination: an extension, ring group, queue, menu, schedule, script, conference room, AI receptionist or voicemail. Numbers match in any form: +1 212 555 0100, 2125550100 and 12125550100 are the same.

- A route for one **number** wins over a route for any number.
- A route can be limited to one **trunk**, for numbers that two providers both send.
- With **emergency mode** on (Settings), every incoming call goes to one place instead, whatever the routes say: an evacuation, a closure, an incident.

Outgoing routes

Outgoing routes match what people dial, by pattern, and send it to trunks in order: if the first is down or full, the next is tried.

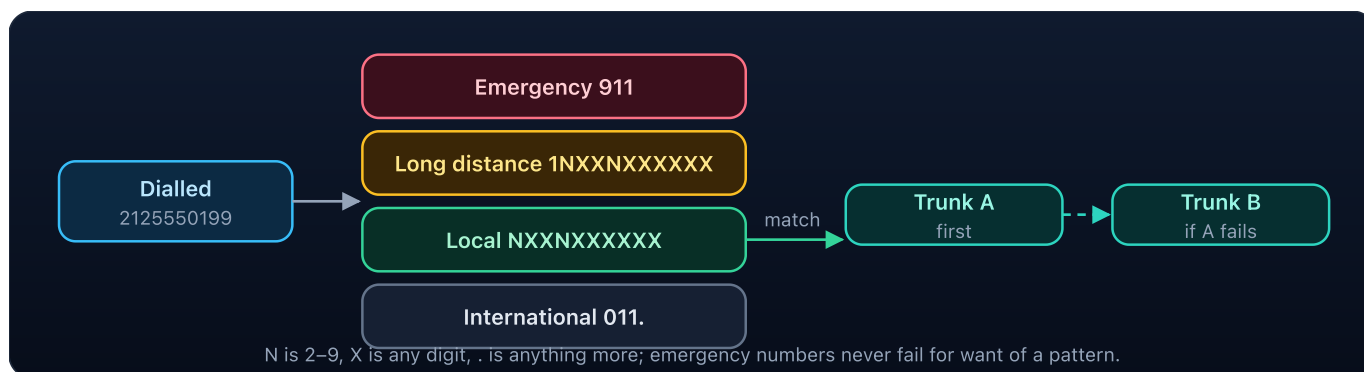


Figure 8. Outgoing routes: the first pattern that matches, its trunks in order

Pattern	Matches
<code>X</code>	any digit
<code>Z</code>	1 to 9
<code>N</code>	2 to 9
<code>[1-5]</code>	one of a set
<code>.</code> at the end	one or more further digits
<code>NXXNXXXXXX</code>	a ten-digit North American number
<code>1NXXNXXXXXX</code>	the same with 1 in front
<code>011.</code>	an international call from the US

- **Remove** takes digits off the front (a 9 dialled for an outside line); **Then add in front** puts some on (a country code).
- The order matters: the first route whose pattern matches is used. Drag routes to change it.
- Premium numbers (900 and the like) are refused unless a route names them.

Emergency numbers

The emergency numbers always work. A route that names the number exactly is used first; with none, every working trunk is tried in turn. A call to them is never refused for an extension's limits, a hosted organisation's limits or a missing route. Health warns when no trunk can carry them.

Test emergency routing with your provider's test number (933 in the US) rather than 911. 933 reads back the address the call carries.

CHAPTER 9

PIN sets

In the web interface: `/pin-sets`

A **PIN set** (chapter 9) is a list of PINs an outbound route asks for before it calls out: to keep international or long-distance calls to people with a PIN, or to bill calls to clients and projects by their code.

1. **Add PIN set** and list its PINs (4 to 10 digits each).
2. Switch on **Keep the PIN in the call record** to write the PIN used into each call's record as its account code (the `accountcode` column of the CSV, syslog and PostgreSQL call record outputs, and `account` in the API).
3. On **Call routing** (chapter 8), open an outbound route and choose the set under **Ask for a PIN**.

Callers dialling a number that route takes hear "Please enter your PIN, then press hash", enter it, and the call goes out. Three wrong PINs end the call, and the attempt is logged as a security event. Emergency numbers never ask for a PIN, whatever route they take, and forwarded calls and follow-me numbers are not asked either: only people dialling from a phone.

CHAPTER 10

Caller ID lookup

In the web interface: `/lookups`

Many phone companies send a caller's number but no name. BitVoice first looks in the [company contacts](#) (chapter 37); after that, an inbound route can ask a [caller ID lookup](#) (chapter 10):

- **A web address:** your CRM or any service that answers with the name. Put `{number}` where the number goes (`https://crm.example.com/api/callers?phone={number}`). A plain-text answer gives the name on its first line; for JSON, give the **name field** as a dotted path (`data.contact.name`). Basic authentication is optional.
- **OpenCNAM:** the North American caller-name database, with your account SID and auth token.

Choose the lookup on an inbound route on [Call routing](#) (chapter 8) (**Caller ID lookup**). Phones show the name while they ring, and it goes into the call record.

A lookup never holds a call up: BitVoice waits at most its **wait** (1.5 seconds unless you change it, 5 at most) and then rings without a name. Answers are kept for a day (or as long as you set, or never with -1), so a regular caller is not looked up on every call; numbers the service does not know are asked again after an hour. **Try it** on the lookup's page asks for a number at once, with the settings as saved.

CHAPTER 11

Emergency calls

Emergency numbers always go out, from every extension, even ones that cannot call outside numbers. This chapter is about emergency calls made from the phones BitVoice runs: an office calling 911. Answering 911 calls is the [911 profile](#) (chapter 74).

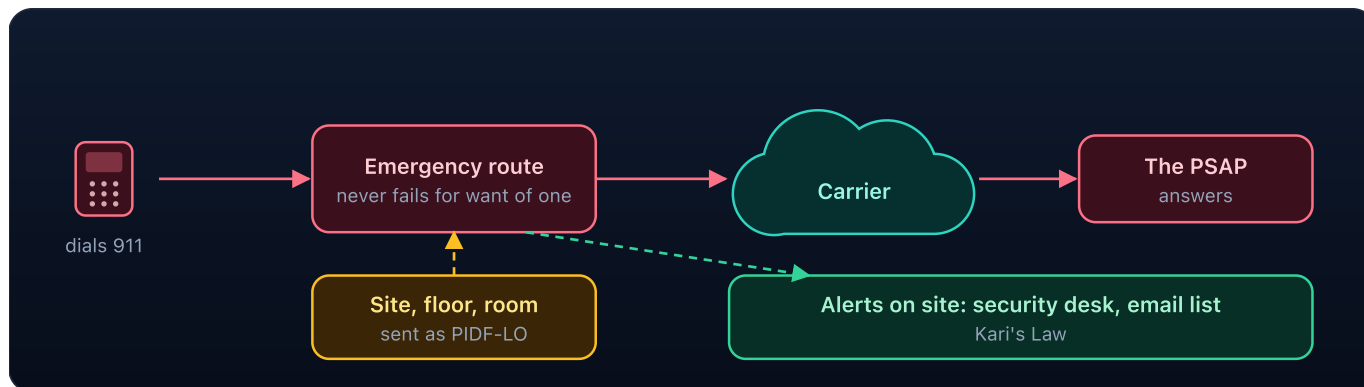


Figure 9. Office 911: location goes with the call, people on site are told

What BitVoice does on every emergency call

1. **Routes it**, always: a route that names the number first; with none, every working trunk in turn. Limits never stop it: an extension not allowed outside calls, a hosted organisation at its call limit or suspended, a trunk at its channel limit.
2. **Sends where the caller is:** with **Send the caller's address** on (Settings), the call carries the site's civic address, floor and room to the provider as PIDF-LO.
3. **Tells people on site** (Kari's Law in the US): the extensions under **Notify** get a message on their phones, every open BitVoice screen shows an alert, and the addresses under **Also email** get an email with the caller's name, extension, site, floor and room, so someone can meet the responders at the door.
4. **Lets the call back through:** for an hour after an emergency call, calls coming in are never refused, so the answering point can always ring back.

Setting it up

- 1. Check the emergency numbers in [Settings](#) (chapter 14) (set by country: 911 and 933 in the US).
- 2. Add every building under [Sites](#) (chapter 12), with its street address and the networks its phones use.
- 3. Give every extension its site, floor and room.
- 4. Make sure an outgoing route and trunk carry emergency calls; [Health](#) (chapter 13) warns when none can.
- 5. Fill in **Notify** and **Also email**.
- 6. Dial **933** (in the US) to hear the address your provider has.

Remote workers

A phone registering from outside its site's networks is a remote worker: their phone page asks where they are, and that address is used instead. Alerts say when it was not confirmed.

Emergency mode

A different thing: one switch that sends every incoming call to one place (an evacuation, a closure, an incident). Set where calls go, an optional code such as ***55** and the extensions allowed to dial it in [Settings](#) (chapter 14); switch it on the [Dashboard](#) (chapter 1), or by dialling the code (rising tones for on, a low tone for off). A red bar shows on every screen while it is on.

CHAPTER 12

Sites and office 911

In the web interface: **/sites**

A site is a building: its street address and the networks its phones use. Sites are how an emergency call says where the caller is, down to the floor and room. In the US, Kari's Law and RAY BAUM'S Act require exactly this from business phone systems.

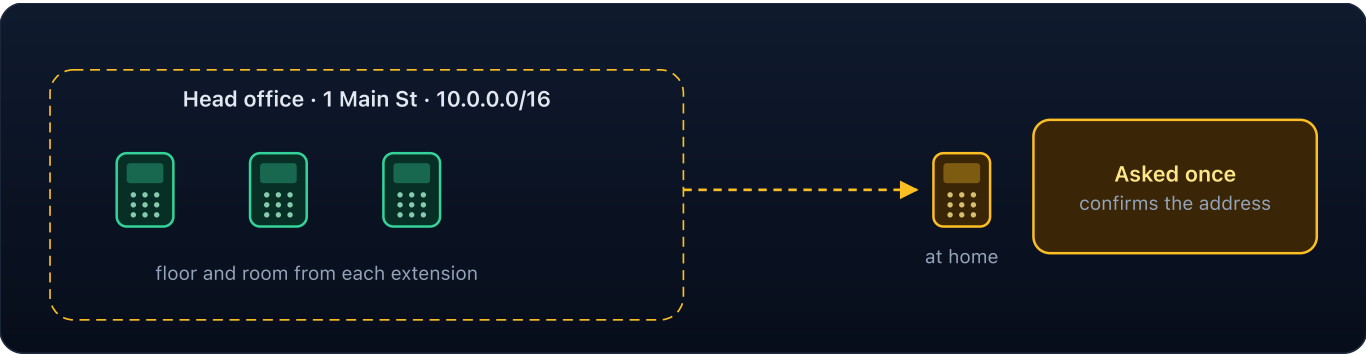


Figure 10. Sites and remote workers: where a phone is, for 911

Adding a site

On [Sites](#) (chapter 12), press **Add**:

Field	What it is
Name	what people call it: Head office
Street, City, State or province, Postal code, Country	the civic address responders are sent to
Networks	the address ranges its phones use, such as 10.1.0.0/16 ; one per line

Then on each extension give its site, **floor** and **room**. [Health](#) (chapter 13) lists extensions without a location.

How a phone's place is found

1. A phone registering from one of a site's networks is at that site, with its extension's floor and room.
2. A hot desk carries the desk's own site, floor and room, not the person signed in ([Hot desking](#) (chapter 35)).
3. A phone registering from outside every site's networks is a **remote worker**. On its first emergency-capable registration from a new place, its user is asked to confirm their address on their phone page; emergency calls then carry that address.
4. If a remote worker never confirmed, the call still goes out, with the extension's usual site, and alerts say the address was not confirmed.

On-site alerts

With **Email when someone calls 911** set in Settings, the people named there get the caller's name, extension, site, floor and room as soon as a call to an emergency number starts, so someone can meet the responders at the door. **Notify** extensions get a message on their phones, and every open BitVoice screen shows an alert. See [Emergency calls](#) (chapter 11).

Check each site's address with your provider: dial **933** (in the US) from a phone at the site to hear the address the provider has.

CHAPTER 13

Health

In the web interface: [/health](#)

Health says whether the system is ready for calls, and if not, what to do. Each line says what is wrong in plain words and links to the screen where it is fixed. Check it after every change of any size.

What it checks

Check	Why it matters
Emergency calls can get out	a route and a working trunk carry the emergency numbers
Extensions without an emergency location	their 911 calls would carry no floor or room
The phone service	SIP is listening, and on which port
Phones not connected	extensions nobody has signed in as
Trunks	registered, or the reason they are not
Menus without a recorded prompt	callers would hear silence
The time zone	schedules follow it
Recordings and their keeping	disk space, and how long recordings are kept
Certificates	when the TLS certificate runs out
Key vaults	secrets a vault cannot read
Other servers	when several share a database, whether each is running

Problems are coloured: red stops calls (no emergency route), amber should be looked at (a phone not connected), and grey is for information.

The emergency checks are always first. A phone system that cannot call 911 is not ready, whatever else works.

CHAPTER 14

Settings

In the web interface: </settings>

Settings holds everything about the system as a whole. Each section saves on its own, and every save is a revision on [Changes](#) (chapter 59).

Section	What is there
System	name, country, time zone, language, the public address phones and providers see
Numbers	the echo test, voicemail and park numbers, and the emergency numbers
Codecs	the codecs phones may use, best first (Codecs (chapter 64))
Call recording	off, every call, or selected extensions; consent rules; how long recordings are kept
Sound	clean-up for voicemail and during calls
Protection	bans after failed sign-ins, trusted networks, calls per extension, premium numbers
Emergency	send the caller's address, who is notified, who is emailed, emergency mode
AI	the AI service, models, and which AI features are on (AI (chapter 27))
Single sign-on	the identity provider (Single sign-on (chapter 57))
Asterisk compatibility	AMI and ARI for programs written for Asterisk (Asterisk compatibility (chapter 68))
Mobile apps	the push service that wakes the apps
Email	the mail server voicemail, faxes and alerts go through
Key vaults	secrets kept in AWS Secrets Manager or Azure Key Vault
Configuration from Git	keep the configuration in step with a file in a repository
Database	SQLite or PostgreSQL (Database (chapter 62))
Certificate	the certificate for this interface and SIP over TLS: from Let's Encrypt, installed by hand, or BitVoice's own

Passwords and keys are never shown again once saved; leave them empty to keep them. Viewers see Settings with every secret removed.

The public address matters when BitVoice is behind a router: phones and providers outside send sound to the address it advertises. Phones, gateways and servers on a private network beside BitVoice (10.x, 172.16-31.x, 192.168.x, 100.64.x, IPv6 fd00::/8) are given BitVoice's own address on that network instead, which they can always reach. Leave it empty to work it out per phone.

The certificate and Let's Encrypt

BitVoice starts with a certificate it made itself, which browsers and phones warn about. For a server with a public name, switch on **Let's Encrypt** under Certificate: enter the names (`pbx.example.com`), optionally an email address, and apply. BitVoice gets a free certificate from Let's Encrypt, uses it at once for the web interface, SIP over TLS and Teams, and renews it about 30 days before it ends (Let's Encrypt certificates last 90 days or less). The status line says when it was issued, when it renews and, if it failed, why; it tries again after an hour, then less often, within Let's Encrypt's limits. **Check now** tries at once.

Let's Encrypt must reach the server under each name from the internet, to prove it is yours:

- **http-01** (the choice unless the web interface is on port 443): a request to port 80. BitVoice answers on port 80 itself (on Linux it needs the right to open low ports, such as `setcap cap_net_bind_service=+ep bitvoice`), or forward port 80 on the router to the web interface's port, which answers too. Anything else arriving on port 80 is sent on to HTTPS, for the certificate's names only. When BitVoice cannot open port 80, the Let's Encrypt box says why in amber and what allows it (the exact `setcap` line for this program, or systemd's `AmbientCapabilities=CAP_NET_BIND_SERVICE` ; or another program already holds the port). **Test the names** then asks each name for a test file over the internet, as Let's Encrypt will, and says for each whether it reached BitVoice, reached something else, or is not in DNS. Some routers do not loop their own public address back to the inside (no hairpin NAT): there the test fails from inside although Let's Encrypt, from outside, succeeds.
- **tls-alpn-01**: a TLS connection to port 443, answered on the web interface's own port: for a server whose web interface is on 443.

Other ACME authorities that need no account binding (a company's own step-ca, Let's Encrypt's staging service for tests) are chosen under **Authority**. Wildcard names need DNS proofs, which BitVoice does not do: install those certificates by hand. Each server of a cluster gets its own certificate.

Where calls go

The building blocks callers move through: groups and queues, menus and schedules, scripts and the AI receptionist, voicemail, and the diagram that shows it all.

Call flow

In the web interface: `/flow`

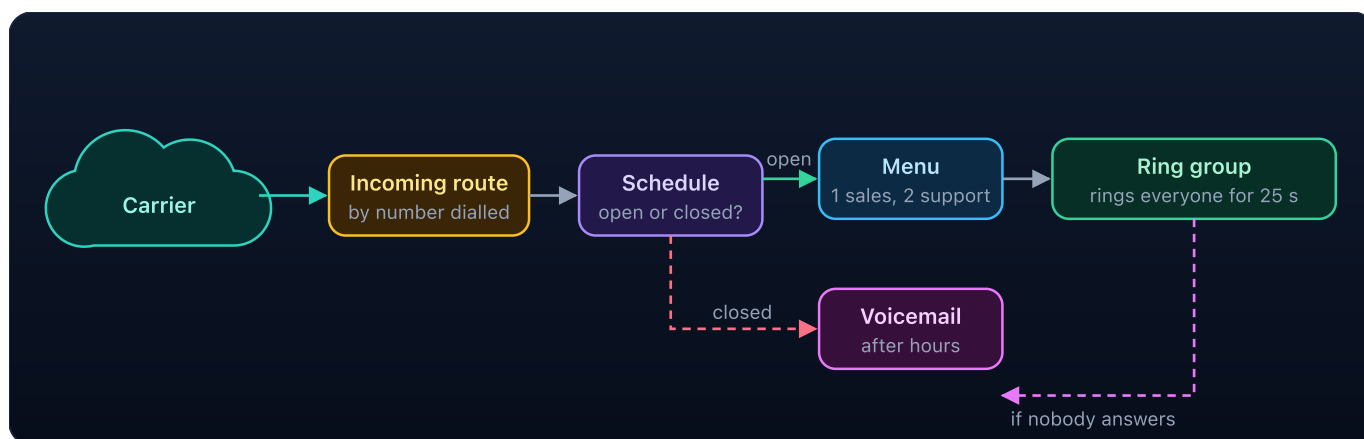


Figure 11. How a call from outside finds its way

Check every situation

Check every situation, at the top of [Call flow](#) (chapter 15), tries your whole call plan in every situation it can be in, not a sample:

- every combination of open and closed that your schedules really reach over the coming year (each holiday included), found in 15-minute steps;
- every combination of day/night toggles (or, with more than six, each one switched alone);
- every incoming route, and every key a caller can press in every menu, every "if nobody answers" and every queue timeout along the way;
- every extension dialling every emergency number.

It lists what it finds, the worst first: **a call goes nowhere** (a group with nobody to ring and nowhere to go next), **a loop, works, but look at it** (a queue with no agents), **nothing routes to it** (a menu or schedule no call reaches), **a branch no situation takes** (a schedule that is never closed, so its closed side is never used), and **emergency calls rest on one trunk** (if it fails, they cannot get out). Each finding names the situations that show it: the time and which schedules were open, the toggles switched over, and the path (**a call to +12125550100 → Press 2 (Billing) → No answer**). **Try this call** fills **Try a call** below with that exact situation and traces it.

The same check runs on changes before they are made: a proposal from **Ask in plain words** and the **Restore this** look at an old revision both list what the change would break that works today.

Try a call

Enter who calls (an extension, or an outside number), what they dial and when, and press **Try it**. BitVoice follows the call exactly as it would for real, without ringing anything: the incoming route, schedules at that time, routing scripts (they run, in their sandbox), menus, groups and queues, and what happens if nobody answers. The path this call takes is marked; other ways on (a menu's keys, a schedule's other side) can be opened. Green means it works, amber means look at it (a phone not connected, a script falling back), red means the call fails there. Tick **Assume every phone is connected** to see the plan as designed rather than as it is right now; **Arrives on** picks the trunk for routes that only take one.

The diagram

The diagram shows where calls from outside go, box by box: each incoming route, then the menus, groups, queues, schedules, scripts and people it leads to.

Click a box (or Tab to it and press Enter) to change where calls go from it: a route's destination, an extension's or group's **If nobody answers**, a schedule's open and closed destinations, a menu's keys (add or remove keys right there), a script's fallback or a receptionist's person. Saving checks the change like any other and makes a revision. **Open its full editor** goes to the item's own page. **Add incoming route** adds a route without leaving the diagram.

CHAPTER 16

Ring groups

In the web interface: `/ring-groups`

A ring group is one number that rings several extensions: a sales team, the front desk and its back-up, everyone in the workshop.

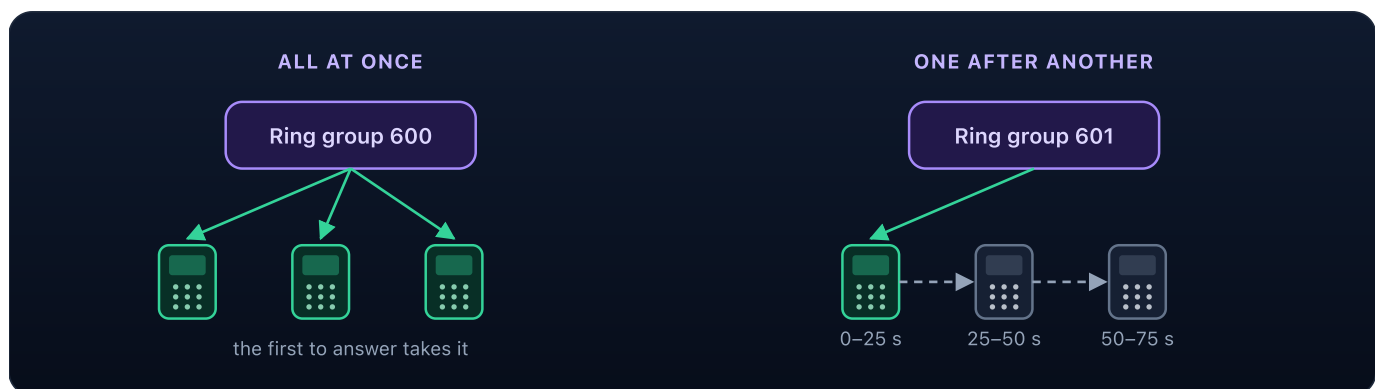


Figure 12. Ring groups: all at once, or one after another

Ringling strategies

- **All at once:** every member's phones ring together, and the first to answer takes the call. The others stop ringing.
- **One after another:** each member rings for the group's ring time, in the order listed. Move members with the arrows, or Alt+↑ and Alt+↓.

Members who are away (in a meeting on their calendar, or with do not disturb on) are skipped; the group rings the others. Members with the mobile app are woken by a push and rung too.

If nobody answers

When nobody answers within the ring time, the call goes to **If nobody answers**: a member's voicemail, a queue, a menu, another group. With nothing set, it rings until the caller gives up.

Groups, queues or pickup?

Use	When
A ring group	a few people, any of whom can take the call, and the call should ring now
A queue	callers may have to wait for someone free, and should hear their place
A pickup group	each person has their own calls, but colleagues may answer for them

CHAPTER 17

Queues

In the web interface: [/queues](#)

A queue holds callers in order, with music, and rings the next free agent. It is the heart of a contact centre, and useful in any office where calls come faster than people can take them.



Figure 13. A queue: callers wait in order, agents take the next one

How calls are offered

Strategy	Who is rung
Longest idle	the free agent who has waited longest since their last call
Round robin	each free agent in turn, carrying on from the last one
Ring all	every free agent at once
Fewest calls	the free agent who took the fewest calls today
In order (linear)	always from the top of the list: the first free agent
Random	any free agent, by chance
Weighted random	by chance, agents with a lower penalty much more often

- **Order:** the agents' list is in order; move them up and down. In order and Round robin follow it.
- **Penalties** rank agents (0 to 100, 0 when empty): calls are offered to the free agents with the lowest penalty, and an agent with a higher one is rung only when nobody with a lower one is free. Use them for a first line and an overflow line in one queue.
- **Weight** ranks queues that share agents: while a queue with a higher weight has someone waiting, its agents take its callers first.
- **Wrap-up** gives an agent time after a call before the next.
- **Longest wait:** callers who wait longer go to the queue's timeout destination. 0 waits for ever.
- **Hold music:** callers hear the queue's music (any class from [Sounds](#) (chapter 19)), the system's, or the queue tone.
- **Tell waiting callers** their place in line and the expected wait (the average wait of answered callers today, in minutes), every so many seconds, in the queue's language. The words come from the queue prompts in [Sounds](#) (chapter 19); until those are recorded, nothing is said.

The **Queues** screen is a live wallboard: who is waiting, for how long, and each agent's state.

Agents

Agents dial ***45** to log in or out of every queue, and ***46** to pause (a break) and resume; paused and logged-out agents are not offered calls. The wallboard has the same buttons, and shows why an agent paused.

State	Meaning
available	free; will be offered the next caller
ringing	being offered a call
busy	on a call
wrap-up	just finished; the wrap-up time is running
paused	on a break, with the reason given
away	in a meeting or on time off, from their calendar
logged out	not taking queue calls at all

Supervisors

On [Live calls](#) (chapter 45), **Supervise** rings a supervisor's phone; when they answer they **listen** (heard by nobody), **whisper** (heard by the agent only) or **barge** (heard by both), and can switch between them during the call.

Sticky queues and VIPs

Switch on **Sticky** in a queue and a returning caller (one answered in the last 30 days, or as many as you set) is offered first to the agent who answered them last time, if that agent is free; if they do not answer, the queue goes on as usual. **VIP callers**, at the foot of Queues, go to the front of every queue they call, after any VIP already waiting, and are marked VIP on the board.

CHAPTER 18

Outbound dialer

In the web interface: **/dialer**

The **outbound dialer** (chapter 18) calls lists of contacts for a **queue** (chapter 17)'s agents: renewals, appointment reminders, collections, surveys. Each **campaign** has its queue, its list and its way of calling:

Mode	How it calls
Preview	an agent asks for the next contact, sees its details, then calls it (their phone rings first) or skips it
Progressive	one call for each free agent
Power	a set number of calls for each free agent (1 to 3)
Predictive	BitVoice adjusts the calls per agent, leaning in while few calls are abandoned and backing off as the abandon rate nears the limit

1. **Add campaign**: its queue, its caller ID (the number contacts see), the hours calls may be made in a time zone, how long a contact's phone rings, how many tries and how far apart.
2. Open **Contacts** and **Add a list**: a CSV file with a header row and a column called **number** (or **phone**); **name** and any other columns are shown to the agent in preview. Duplicate and malformed numbers are skipped.
3. Record the **message** (in the campaign): who is calling and why, and "press 9 if you do not want these calls". It plays to a person answered when no agent is free, and, if you choose, after an answering machine's greeting.
4. **Start** it. It calls only within its hours, and stops calling when it is paused or nothing is due.

People and machines. BitVoice listens to the first seconds after an answer: a long, unbroken greeting is an answering machine; a short "hello?" and a pause is a person. A machine can be hung up on (and tried again), left the message, or passed to an agent like anyone else. When in doubt, it takes the answer for a person.

Abandoned calls. A person answered when no agent is free in two seconds hears the message and counts as **abandoned**. With power and predictive dialling, if the share of abandoned calls over the last hundred answers goes over the campaign's limit (3% unless you change it, the US rule), the campaign dials one call per free agent until it is back under. The campaign shows its abandon rate, calls ringing and free agents live.

Do-not-call. No campaign calls a number on the **do-not-call list**, whatever list it came in on; anyone who presses 9 during the message is added at once. Add numbers by hand or through the API (`POST /api/dnc`). For the national registry (in the US, the FTC's National Do Not Call Registry: download the area codes you call as a flat file), choose the file under **National registry** and **Import**: every line with a number counts (`212,5550100` as the FTC writes them, or one number a line), headings and broken lines are skipped and counted. With **Replace the last import** on, the previous download's numbers come off first, so a number that left the registry is callable again; numbers added by hand or by pressing 9 always stay. The registry must be refreshed at least every 31 days (US rules); the page counts the list and how much of it came from the registry. The API is `POST /api/dnc/import?replace=1` with the file as the body.

Outbound calling is regulated (in the US the TCPA and the Telemarketing Sales Rule; in the UK Ofcom's rules on silent and abandoned calls; elsewhere the local law). BitVoice gives you the tools (hours, do-not-call, abandon limits, caller ID, the message); you remain responsible for consent, the lists you call and the rules where your contacts are.

CHAPTER 19

Sounds: prompts, languages and hold music

In the web interface: `/sounds`

Sounds (chapter 19) holds everything BitVoice says or plays to callers, apart from greetings and menu prompts, which are recorded where they are used.

Prompts, in every language

A **prompt** is one thing BitVoice says: "Please enter your PIN", "Your place in line is", the digits it reads out. The dial-by-name directory, dial-in (DISA), wake-up calls and queue announcements speak with them. BitVoice has the words of every prompt in English, Spanish, French, German, Italian, Portuguese, Dutch and Polish; numbers are read digit by digit, which sounds the same way in each.

- **Callers hear** sets the language for everyone. A queue, a dial-in number, a directory or an extension can choose another.
- Each prompt can be **recorded in the browser**, **uploaded** (any audio file; it is turned into telephone audio) or **made by the speech service**: with a speech model set in Settings, AI, one button makes every prompt of a language from its words. Prompts already recorded are kept unless you ask to replace them.
- **Add a language** for any other language (a code such as sv, ja or pt-BR); its prompts are recorded or uploaded, with the English words shown as a guide.
- A prompt missing in a language is played in the default language, then in English; with none recorded at all, callers hear a short tone, so every feature works before a single prompt is recorded.

Hold music

Hold music is grouped in **classes** (Calm, Jazz, Sales offers...), each a list of tracks played one after another, from the start or **shuffled**. Upload tracks of up to fifteen minutes in any format the browser reads; MP3 and WAV files can also be sent straight to the API (`PUT /api/music/{class}/{name}`), and BitVoice decodes MP3 itself (MPEG-1, 2 and 2.5 Layer III, with a decoder that gives the same samples as minimp3).

Internet radio. A class can play a stream instead of its tracks: enter an MP3 stream's address (Icecast, Shoutcast, most internet radio) under **Internet radio**. BitVoice connects when the first caller is put on hold with that class, plays the one stream to everyone waiting, and disconnects half a minute after the last one leaves. While the stream cannot be reached, the class's own tracks play (or the hold tone, without tracks), and the class shows why.

Who hears it	Which music
someone put on hold	the music chosen on the extension of the person holding, else the default
a parked caller	the music of the person who parked them, else the default
a caller waiting in a queue	the queue's music, else the default
a caller waiting for a conference leader	the default

Default for everyone can be the hold tone, so nothing changes until you choose music. Choosing **the hold tone** on a queue or extension keeps the tone there.

Only play music you have the rights to play to callers: most recorded music needs a licence for on-hold use.

CHAPTER 20

Menus (auto attendant)

In the web interface: `/menu`

A menu (an auto attendant) answers, plays a prompt and sends the caller where they press: 1 for sales, 2 for support, 0 for reception.

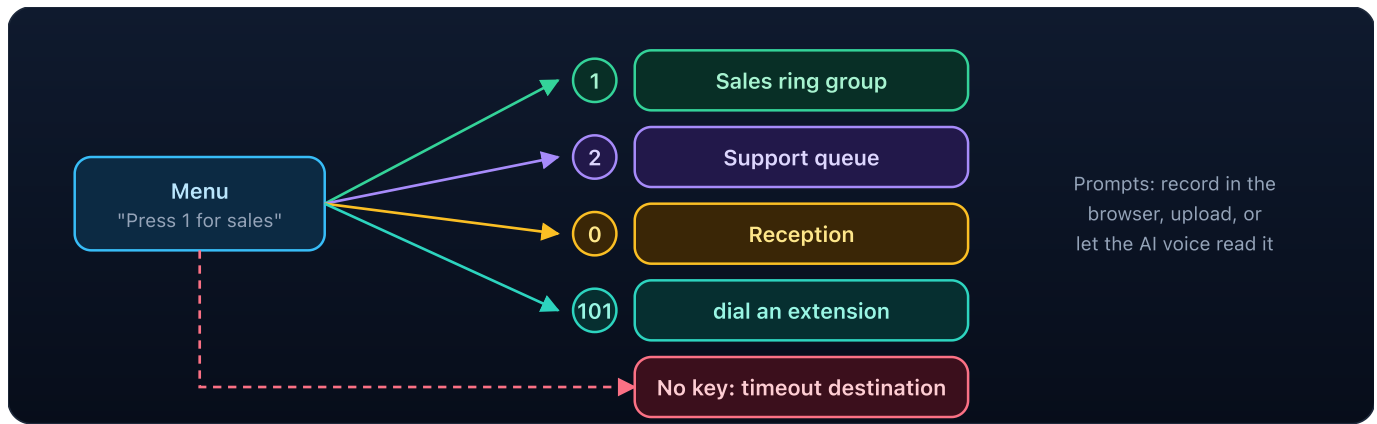


Figure 14. A menu (auto attendant): each key goes somewhere

Setting up a menu

1. On [Menus](#) (chapter 20) press **Add menu** and give it a name (and a number, if staff should be able to dial it).
2. Add **options**: a key, a label, and where the key goes.
3. Record the **prompt** in the browser, or upload a recording in any format (it is converted to telephone sound). Say every option: "For sales press 1, for support press 2."
4. Choose what happens when the caller presses nothing: the **timeout destination**, after the prompt has played the number of **repeats** you set.
5. Point an incoming route (or a schedule) at the menu.

Choices

- **Callers may dial an extension**: callers who know the number type it during the prompt.
- **Wait for a key**: seconds after the prompt before it repeats.
- A key can go to another menu, for a second level ("for billing questions press 1").

Keep menus short: four options at most, the most common first, and always a way to a person. Callers who are lost hang up.

Prompts in other voices

With AI switched on (Settings), a prompt can be written as text and read by the AI voice instead of recorded.

CHAPTER 21

Dial-by-name directory

In the web interface: `/directories`

A [dial-by-name directory](#) (chapter 21) lets callers reach someone without knowing their number: they spell the first letters of the name on the keypad (2 for ABC, 3 for DEF, up to 9 for WXYZ) and are put through.

1. **Add directory**, name it, and give it a number (411, say) or leave the number empty and use it as a menu option.
2. Choose what callers spell: the **last name** (the usual), the **first name**, or **either**.
3. Choose **who is listed**, or nobody to list everyone with a phone.

Callers hear "Please enter the first three letters of the person's name". One match is put through at once. Several are read out, eight at a time: each person's name, then "to call this person, press 1", 2 and so on, with 9 for more names. No match: "No one

matches", and they try again (three tries).

Spoken names: record each person's name on their extension (**Spoken name:** record in the browser or upload a file), and the directory says it; until then it reads out the extension number. The directory's own words come from the directory prompts in [Sounds](#) (chapter 19), in its language.

CHAPTER 22

Dial-in (DISA)

In the web interface: `/dial-in`

A [dial-in](#) (chapter 22) (DISA, "direct inward system access") lets someone call in from outside, prove who they are, and call out again through the office: the call goes out with the office's caller ID, its routes and its rates, as if they were at their desk.

1. **Add dial-in**, and set a **PIN** (4 to 10 digits), the numbers allowed to use it (**Only from these numbers**), or both. One of the two is required.
2. Choose the extension the calls are **made as**: its outside-call permission, caller ID and outgoing routes apply. It must be allowed outside calls.
3. Give it a number to dial inside, and send a phone number to it on [Call routing](#) (chapter 8) (choose the dial-in as the destination).

Callers hear "Please enter your PIN, then press hash", then "Please enter the number to call, then press hash", and are put through. The words come from the dial-in prompts in [Sounds](#) (chapter 19).

A dial-in is a door to your phone bill. Three wrong PINs end the call, every attempt (refused number, wrong PIN, connected call) is logged and shown in Security, and calls follow the extension's limits. Use a long PIN, list callers' numbers where you can (caller ID can be faked, so keep the PIN too), and keep the extension's outside calls to what is needed.

CHAPTER 23

Voice bots (AudioSocket)

In the web interface: `/voice-bots`

A [voice bot](#) (chapter 23) is a program of your own (an AI agent, a speech-recognition IVR, a recorder) that takes calls over **AudioSocket**, the simple protocol Asterisk uses: BitVoice answers the call, opens a TCP connection to the bot, and sends the call's UUID, then the caller's sound as 20 ms frames of 8 kHz 16-bit linear audio and each key pressed; whatever sound the bot sends back is played to the caller. Bots and speech-service connectors written for Asterisk's AudioSocket work with BitVoice unchanged.

1. **Add voice bot**: its name, the bot's address (`10.0.0.9:9092`) and, optionally, a number to dial it.
2. Choose where the caller goes **if the bot cannot take the call, or breaks off** (a person, a queue, voicemail): required, so no caller is ever dropped because a bot is down.
3. Send calls to it from [Call routing](#) (chapter 8), a menu option, a schedule or a toggle.

When the bot sends an AudioSocket hang-up, the call ends; when the caller hangs up, the bot is sent one. A bot that cannot be reached within three seconds, or whose connection breaks without a hang-up, hands the caller to the fallback, and the failure is logged.

To bring a program into a call already in progress (a listening transcriber, an assistant that whispers to no one), use ARI's external media: see [Asterisk compatibility](#) (chapter 68).

CHAPTER 24

Day/night toggles

In the web interface: `/toggles`

A [day/night toggle](#) (chapter 24) is a switch for a call flow (FreePBX calls it call flow control). Calls sent to it go to its **normal** destination, or to the other one while someone has **switched it over**: the office closes early for a storm, someone dials `*281` on the way out, and calls go to voicemail until it is switched back in the morning. It stays as it was left, across restarts.

- 1. **Add toggle**, give it a code (1 to 3 digits: dialled after `*28`) and choose where calls go normally and when switched over. Either side may be a schedule, so a toggle can sit in front of the opening hours and override them.
- 2. Choose who may switch it from a phone (nobody chosen lets any extension).
- 3. Send calls to it from [Call routing](#) (chapter 8), a menu option or a schedule, like any other destination.

Switch it by dialling `*28` and its code (a rising tone: switched over; a falling tone: back to normal), with the switch beside it on the Toggles page, or through the API (`POST /api/toggles/{id}/state`). A busy lamp key watching `*281` is lit while that toggle is switched over, so the person who closes up can see it from their desk. The toggle code itself is on [Feature codes](#) (chapter 31).

[Call flow](#) (chapter 15) shows both sides of every toggle and which one calls take now; the call-path simulator follows the side that is in force.

CHAPTER 25

Schedules

In the web interface: `/schedules`

A schedule sends calls one way when you are open and another when you are closed. Point an incoming route at a schedule, and the schedule at the menu (open) and voicemail (closed).

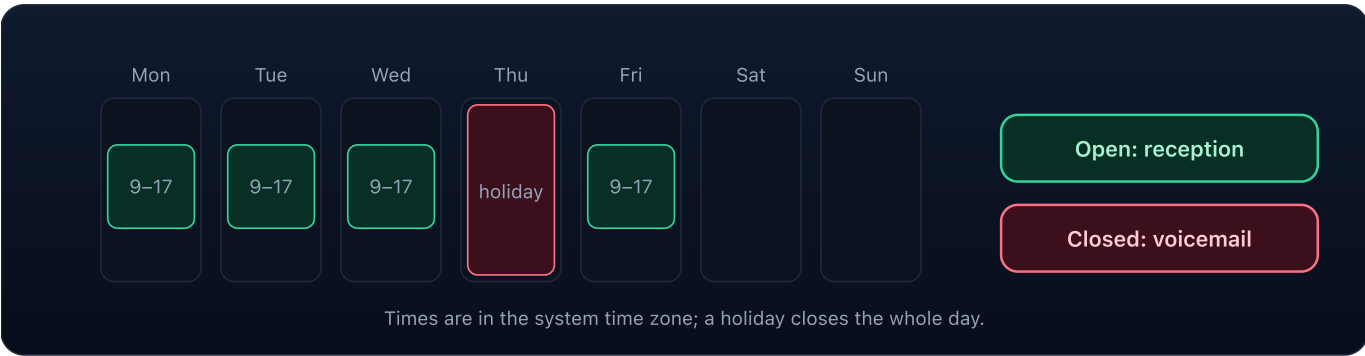


Figure 15. A schedule: open hours, holidays, and where calls go in each

Open hours

Each line of hours is a set of days and a time from and to. Several lines are allowed: Monday to Friday 9:00 to 17:00, and Saturday 10:00 to 13:00. Hours may cross midnight (22:00 to 06:00) for night shifts.

Holidays

Holidays count as closed all day, whatever the hours say. Add each date with a name; the name is shown on [Call flow](#) (chapter 15) when you try a call on that day.

Time zone

The time zone is the one in Settings. When the clocks change, schedules follow the local time.

Try a call on [Call flow](#) (chapter 15) with a time and date (a holiday, a Saturday night) to see which side of the schedule it takes, before it matters.

CHAPTER 26

Scripts (JavaScript and Lua)

In the web interface: `/scripts`

A script decides where a call goes when menus, schedules and groups are not enough: callers from one area to one office, VIP numbers straight to a person, a decision from your own data. Or, set to **Runs the whole call**, it answers the call and runs it, as an Asterisk dialplan or AGI script would: prompts and speech, keys, recordings, ringing people, handing on (below). Scripts are written in JavaScript or Lua on [Scripts](#) (chapter 26).

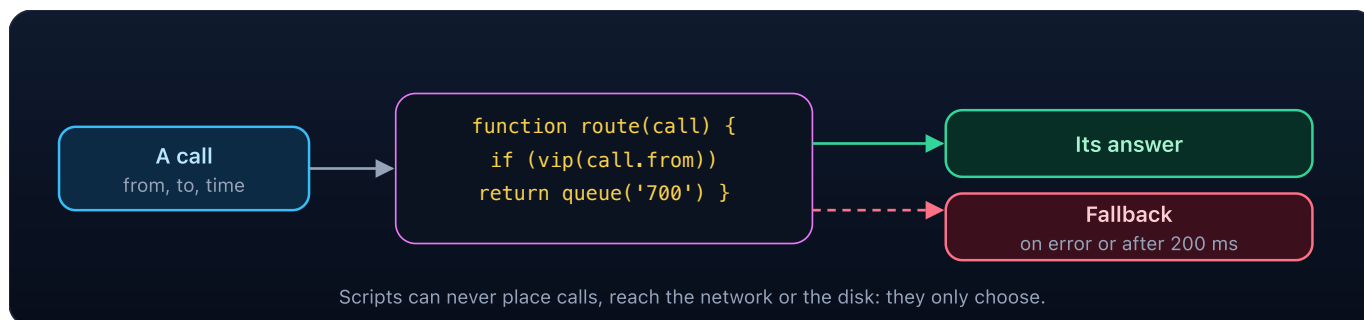


Figure 16. A routing script: code decides, a fallback is always there

Safe by design

- A routing script runs for at most 200 ms. If it fails or takes too long, the call goes to the script's **fallback**, so a mistake never loses a call. (Call scripts have their own limits, below.)
- It runs in a sandbox: it cannot reach the network or the disk. A routing script only chooses; a call script can do only the steps listed below, to its own call.
- Emergency calls never run scripts.

What a script sees

`call` has:

Field	What it is
<code>from</code> , <code>fromName</code>	the caller's number and name
<code>to</code>	the number dialled
<code>direction</code>	<code>inbound</code> or <code>internal</code>
<code>trunk</code>	the trunk it came in on
<code>weekday</code> , <code>hour</code> , <code>minute</code>	the time now (0 is Sunday)
<code>date</code>	YYYY-MM-DD

What it returns

A destination, `{type, target}`: type is `extension` , `ringgroup` , `queue` , `menu` , `schedule` , `voicemail` or `hangup` , and target is its number. Returning nothing uses the fallback. `log("text")` writes to the script's run list.

JavaScript

```
if (call.from.startsWith("212")) {
  log("New York caller")
  return { type: "ringgroup", target: "600" }
}
if (call.weekday === 0 || call.weekday === 6) return { type: "voicemail", target: "100" }
```

Lua

```
if string.sub(call.from, 1, 3) == "212" then
  log("New York caller")
  return { type = "ringgroup", target = "600" }
end
if call.weekday == 0 or call.weekday == 6 then
  return { type = "voicemail", target = "100" }
end
```

Using a script

Point an incoming route, a menu key or a schedule at the script, as at any destination. **Test** runs it with a call you describe, without placing one; the run list shows each real run with its decision, time taken and any error. The [call flow](#) (chapter 15) runs scripts too when you try a call.

Scripts that run the whole call

Choose **Runs the whole call**. The script then answers the call and goes from top to bottom, each step waiting until it is done:

Step	What it does
<code>play(name)</code>	plays a prompt from Sounds (chapter 19) in the caller's language (<code>goodbye</code> , <code>digit-5</code> ...), <code>digits:1234</code> , or a tone: <code>beep</code> , <code>confirm</code> , <code>error</code> , <code>off</code> , <code>hold</code> . A prompt not recorded yet is a beep.
<code>say(text)</code>	speaks the words with the speech service in Settings, AI (a beep without one)
<code>collect({max, timeout, interDigit, end, say, prompt})</code>	gathers keys: at most <code>max</code> (32), waiting <code>timeout</code> seconds for the first (5) and <code>interDigit</code> for each next (3), until <code>end</code> (<code>#</code>). <code>say</code> or <code>prompt</code> asks first. Returns the keys, <code>""</code> when none.
<code>record({box, seconds, beep})</code>	records a message into an extension's mailbox, until <code>#</code> or the time; returns its seconds
<code>wait(seconds)</code>	waits, returning a key pressed
<code>dial(destination, ringSeconds)</code>	rings it with the caller hearing ringback. Answered: they talk and the script is done (<code>true</code>). Not answered: <code>false</code> , and the script carries on.
<code>transfer(destination)</code>	hands the call on; the script is done
<code>hangup()</code>	ends the call
<code>log(text)</code>	writes to the run list

A destination is a number (an extension, ring group or queue here, else an outside number through the outbound routes; never an emergency number) or `{type, target}` as above. A key pressed during a sound stops it; `play` and `say` return it and the next `collect` starts with it. Returning a destination sends the call there.

```
const k = collect({ max: 1, timeout: 5, say: "Press 1 for sales, 2 for support." })
if (k === "1") return { type: "ringgroup", target: "600" }
if (k === "2") {
  if (dial("101", 20)) return
  say("Nobody is free. Leave a message after the tone.")
  record({ box: "101" })
  hangup()
}
```

Lua has the same steps (`collect{max = 1}`). Give the script a **Number to dial it** to call it from an extension.

Its limits keep the call safe: the script's own code may run at most a quarter of a second between two steps; a call takes at most 1,000 steps; and the whole script stops after its **Longest run** (10 minutes unless you say, 60 at most). When it fails, is stopped, or ends without hanging up or handing the call on, the call goes to its **fallback**; a caller who hangs up stops it at once. **Try it** runs the script with the keys you say the caller presses and lists each step, with no call placed.

CHAPTER 27

AI receptionist, voicemail triage and asking

In the web interface: `/receptionists`

AI features use the AI service set in [Settings](#) (chapter 14): an OpenAI-compatible service (OpenRouter by default) or one you run yourself. What they need is sent there, so choose a service you trust with calls; for a 911 centre, its own government cloud counts as local.

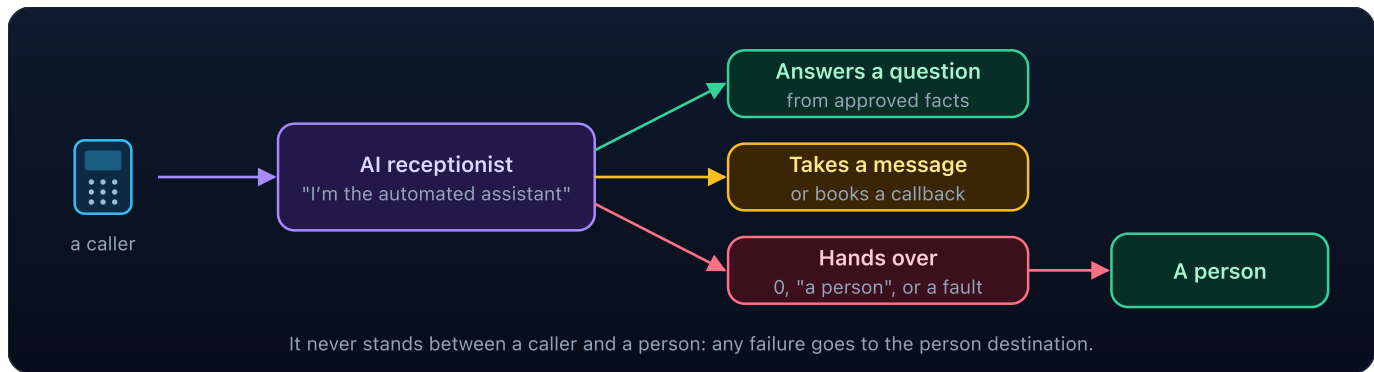


Figure 17. The AI receptionist: answers, helps, and always hands over to a person

AI receptionist

Choose a receptionist as where calls go: a route, a menu option or an extension's **If nobody answers**. It says it is automated, answers only from **Approved information**, takes a message, books a callback (see [Callbacks](#) (chapter 29)), or transfers to **A person is**. Pressing **0** or asking for a person always reaches someone, and so does any problem with the AI service or the speech model: callers are never stuck with it.

Models

Model does the text work: summaries, the receptionist's replies and asking in plain words. Turning speech into text needs a model that takes audio; set it as **Speech-to-text model** (for example google/gemini-2.5-flash on OpenRouter) to use a text-only model for the rest, such as meta-llama/llama-3.3-70b-instruct.

Voicemail triage

With **Transcribe and summarise voicemail** on, each message is also marked urgent, normal or low priority, with its kind (sales, support, billing...). Urgent messages raise an alert on every administrator's screen.

Ask in plain words

On [Changes](#) (chapter 59), describe a change ("send sales calls to voicemail after 5 pm"). The AI proposes it, BitVoice checks it like any other change and shows exactly what would change. Nothing changes until you press **Apply**, and it becomes a revision you can restore. Passwords are never sent to the AI.

CHAPTER 28

Voicemail

In the web interface: `/voicemail`

Each extension can have a mailbox. Callers who are not answered hear its greeting and leave a message; the person hears it on their phone, reads it on the screen, or gets it by email.

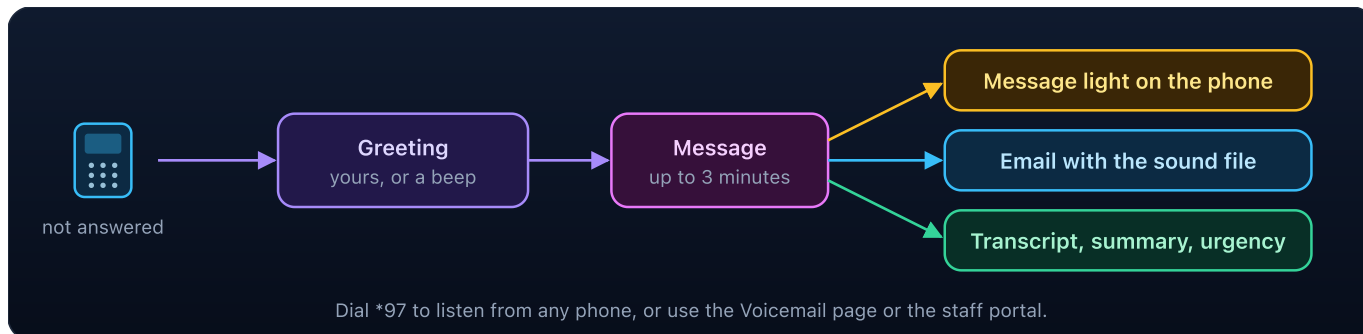


Figure 18. Voicemail: from a missed call to a message you can read

Switching it on

On the extension, switch on **Voicemail**. **If nobody answers** is then set to the mailbox. Give it a **PIN** (4 to 10 digits): it opens the mailbox from another phone, and signs the person in to the staff portal and at hot desks.

Setting	What it does
Longest message	10 to 900 seconds (180 by default)
Email new messages	to the extension's email address, with the message attached
Greeting	recorded in the browser, uploaded, or recorded from the phone; without one, callers hear a beep

Listening

- From the extension's own phone, dial ***97** : messages play newest first; keys replay, delete or keep each one.
- From any other phone, dial ***97** , enter the mailbox number and its PIN.
- On **Voicemail** (chapter 28) (administrators) or in the staff portal (the person): play, mark heard, delete.
- The phone's message light is on while there are new messages.

With AI

With **Transcribe and summarise voicemail** on (Settings, AI), each message is written out, summarised in a line, and marked urgent, normal or low priority with its kind (sales, support, billing). Urgent messages raise an alert on every administrator's screen. See **AI** (chapter 27).

Sound clean-up

Sound clean-up (Settings) removes steady background noise and evens out levels as messages are saved, so a message from a car or a busy street is easier to hear.

Messages under a **legal hold** (chapter 70) cannot be deleted, by anyone, until the hold is released.

CHAPTER 29

Callbacks

In the web interface: **/callbacks**

People who asked to be rung back. An AI receptionist books one when a caller asks: it collects their name, the number to call (offering the one they are calling from), a good time and what it is about, reads them back, and books it only when the caller agrees. It never promises an exact time. Administrators can also add callbacks by hand.

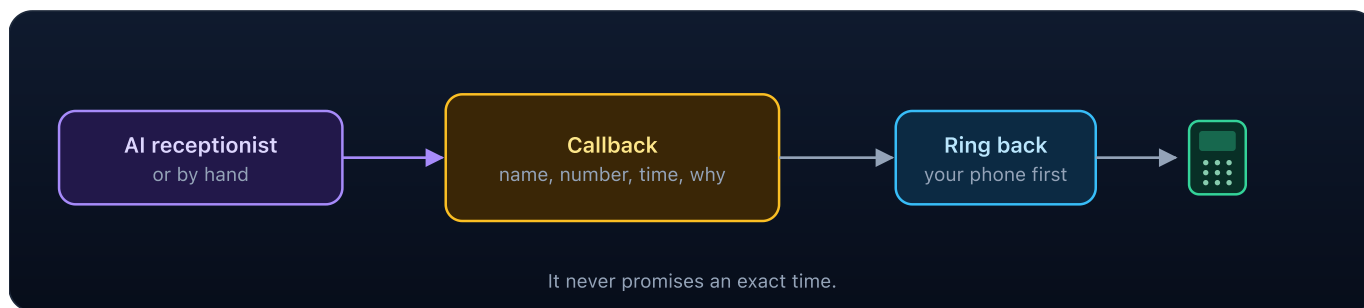


Figure 19. Callbacks: booked by the receptionist, rung back by a person

A new callback shows an alert on every open screen, and is emailed to the owner of the receptionist's **Take messages for** mailbox when that extension has an email address and email is set up in Settings.

To ring back, choose your extension and press **Ring back**: your phone rings first, and when you pick up BitVoice calls the number for you, with your normal routes and caller ID. Then add a note if you like and press **Done** (or **Cancel it**). Finished callbacks are kept below the open ones and can be opened again.

CHAPTER 30

Call links and website calling

In the web interface: `/call-links`

A call link is a web address that rings your business from any browser: no phone number, no app. Put it in an email signature, print its **QR code** on a flyer or a shop window, or paste the **website button** into your site.

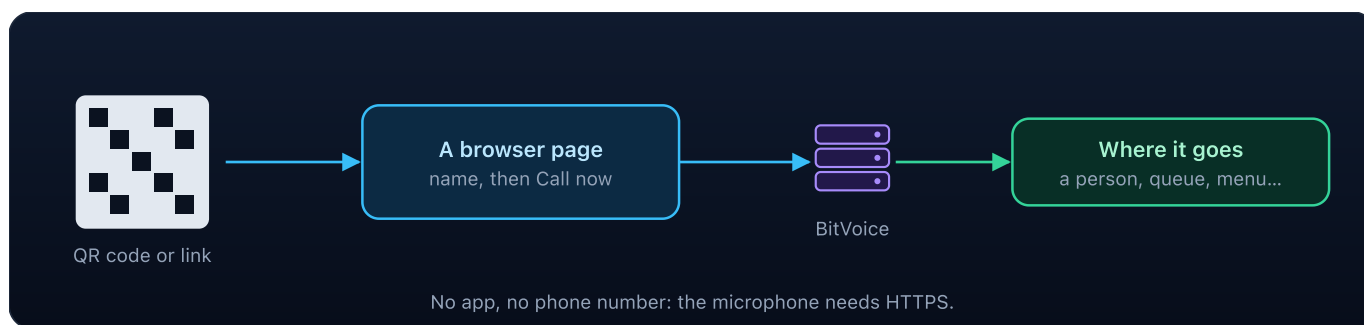


Figure 20. Call links: a web address that rings the business

On [Call links](#) (chapter 30), add a link, choose where its calls go (an extension, a group, a queue, a menu, the AI receptionist, a schedule or voicemail), and save: the page shows its address and QR code. The caller sees the title and words you set, gives their name if you ask for it, allows the microphone and presses **Call now**; the keypad works for menus. Whoever answers sees **Web**: and the caller's name.

Switch **Open** off to close the line (the page says so). **Calls at once** limits how many people can call through one link (5 unless set). The address holds a long random part so it cannot be guessed; repeated wrong addresses are blocked like failed sign-ins. Callers' microphones need the page to be served over HTTPS, so give out the HTTPS address.

Everyday calling

What people do on their phones every day, and what they set for themselves: forwarding, follow-me, pickup, conference rooms, paging, hot desks, faxes and texts.

CHAPTER 31

Feature codes, forwarding and speed dials

In the web interface: `/codes`

Feature codes are what people dial on their phones to change their own settings and reach features. They work on every phone, desk phone or softphone, with nothing set up on the phone itself. A rising tone means done; a falling tone means refused.

The codes at a glance

Code	What it does
*72	forward every call (then the number and #, or the number straight after: *722125550199)
*73	stop forwarding every call
*90 / *91	forward when on a call / stop
*52 / *53	forward when not answered / stop
*78 / *79	do not disturb on / off
*76	switch do not disturb (rising tone: now on)
*21	switch follow-me on and off
*8	answer a call ringing in your pickup group
** and an extension	answer the call ringing that extension
*80 and an extension	intercom: that phone answers on its speaker
*0 and a code	a speed dial
*62	wake-up call: hear it, book one (four digits, such as 0630) or cancel it (star)
*68 and a digit	room status, for housekeeping in hotels
*28 and a toggle's code	switch a day/night toggle (chapter 24) over, or back
*43	the echo test
*97	your voicemail
70	park a call (transfer it here); 71 to 79 pick it up
*45 / *46	queue agents: log in or out / pause or resume
*33 / *34	sign in / out at a hot desk
*7 / *8 during a call	pause and resume the recording (card payments)

Forwarding

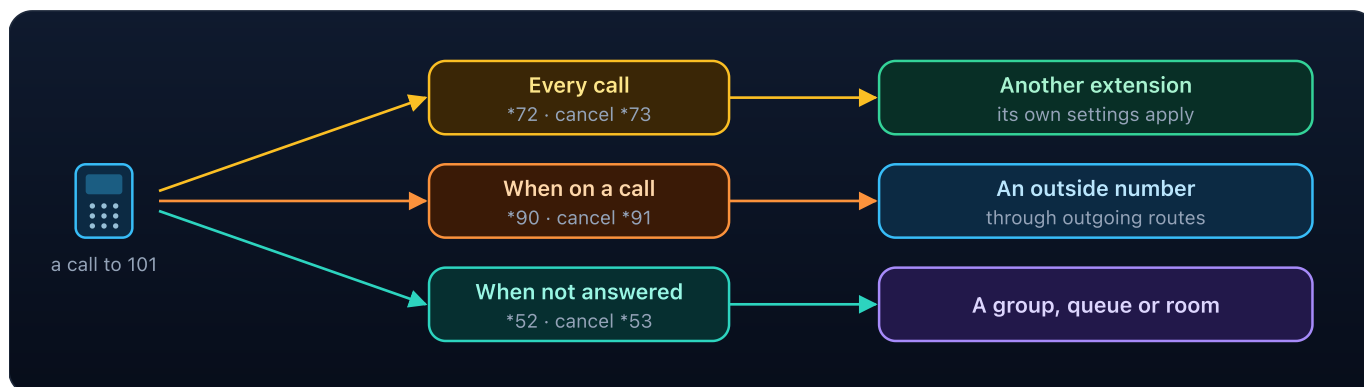


Figure 21. Forwarding: every call, when busy, when not answered

- ***72** forwards every call. Dial it, hear a tone, then dial the number and #. Or dial the number straight after the code.
- ***90** forwards calls that come while you are on a call. ***52** forwards calls you do not answer within your ring time.
- ***73** , ***91** and ***53** cancel them.

Forwarding to another extension uses that extension's own settings (its voicemail, its follow-me), up to four hops so two people forwarding to each other cannot loop. Forwarding to an outside number goes through the outgoing routes, as if the person dialled it, and needs the extension to be allowed outside calls.

Forwarding is live state, not configuration: setting it makes no revision, and it is kept across restarts. [Extensions](#) (chapter 3) and the staff portal show and change it too.

Do not disturb

***78** on, ***79** off, ***76** switches. With do not disturb on, calls go to voicemail, or to the extension's no-answer destination. Do not disturb set by hand wins over the calendar ([Calendars](#) (chapter 38)). Emergency call-backs still ring.

Follow-me

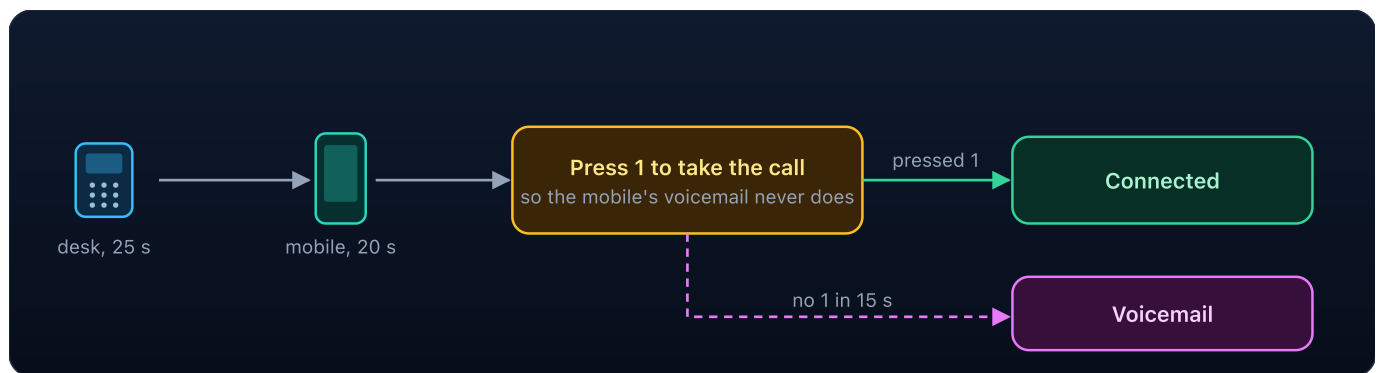


Figure 22. Follow-me: the desk first, then the mobile, then voicemail

***21** switches follow-me on and off, for extensions that have follow-me numbers set (on the extension, or in the staff portal). The desk rings first for its ring time, then the follow-me numbers, one after another or all together. On an outside number with **press 1** on, the person hears a tone and presses 1 to take the call; a voicemail or a silence never does, and after 15 seconds the call moves on.

Answering someone else's call



Figure 23. Call pickup: answer a colleague's ringing phone

- ***8** answers the oldest call ringing anyone in your **pickup group** (set on each extension; names are compared without regard to capitals).

- ****** and an extension answers a call ringing that extension: ****101** .

Intercom

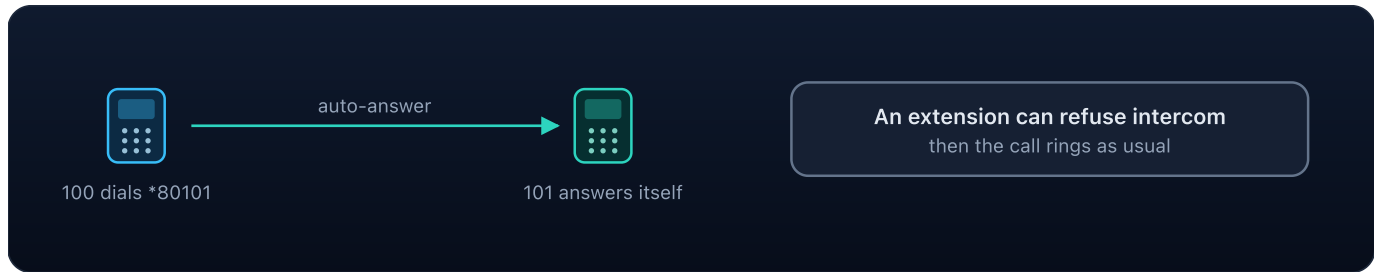


Figure 24. Intercom: the called phone answers on its speaker

***80** and an extension calls it with auto-answer: its phone answers on the speaker, and both sides talk. Yealink, Grandstream and Snom read BitVoice's Call-Info header, Poly reads Alert-Info; some phones need intercom allowed in their own settings. An extension can refuse intercom; then the call rings as usual. Door stations never take intercom.

Speed dials

***0** and a code dials a saved number: ***05** . The company's speed dials are on this page; each person can have their own on their extension or in the staff portal, and theirs win over the company's with the same code. Codes are 1 to 3 digits.

Changing the codes

Every code can be changed on [Feature codes](#) (chapter 31), or switched off with the word **off** . Two features cannot share a code, and no code may be an extension's or a group's number: BitVoice says which one clashes. The echo, voicemail and park numbers are in [Settings](#) (chapter 14).

CHAPTER 32

The staff portal

The staff portal lets everyone look after their own calls without an administrator: do not disturb, forwarding, follow-me, speed dials, voicemail and its greeting, their PIN, and a list of their own calls. It is at **#/me** on this server's address, and linked from the sign-in page.

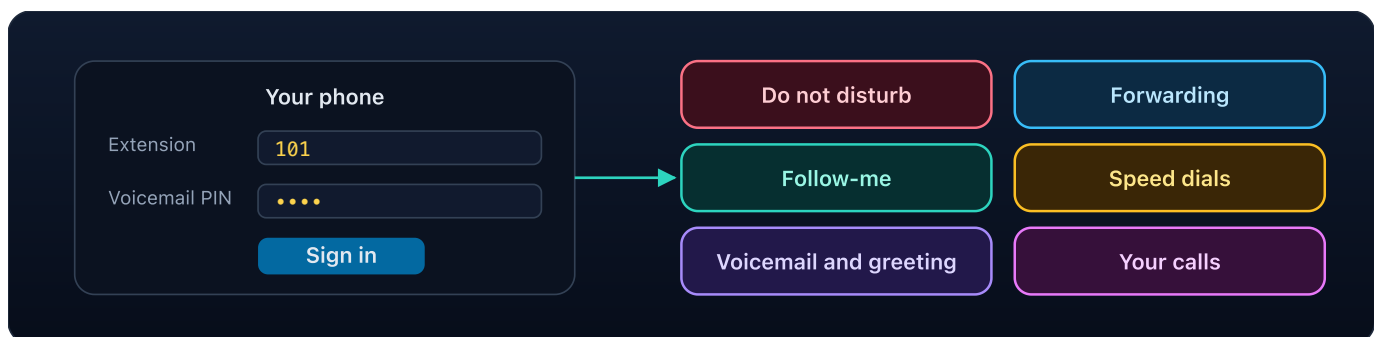


Figure 25. The staff portal: everyone looks after their own calls

Signing in

People sign in with their **extension number** and their **voicemail PIN**: the same PIN that opens their voicemail from a phone and signs them in at hot desks. An extension without a PIN cannot sign in; an administrator sets one on the extension.

Repeated wrong tries from one address lock it out for ten minutes. A portal session reaches only the person's own settings: never another extension, the configuration or anyone's recordings.

What people can do

Tab	What is there
Where calls go	do not disturb; forwarding of every call, when busy and when not answered; follow-me numbers; own speed dials
Voicemail	messages to play and delete, with transcripts and summaries when AI is on; record or upload the greeting
Your calls	the person's own calls, newest first, without recordings
PIN	change the PIN, giving the current one

Forwarding and do not disturb set here are the same as those set from the phone with feature codes ([Feature codes](#) (chapter 31)): whichever was set last is in force. Follow-me numbers and speed dials are part of the extension's configuration, so changing them makes a revision on [Changes](#) (chapter 59), recorded as the person's own.

Forwarding to an outside number works only for extensions allowed to call outside numbers. The portal says so rather than saving a forward that could never ring.

Chat and presence

The portal's **Chat** tab lists colleagues with how they are right now: **available**, **on a call**, **ringing**, **away** (do not disturb, a meeting or time off from their calendar, with the reason) or **offline** (no phone connected), and the status they typed ("At lunch, back at 2"). Choose someone to message them; messages arrive at once, and a number beside a name counts the unread ones. **Team rooms** (Settings, Staff chat) are conversations for a group, or for everyone when a room lists no members.

Messages are only ever seen by the people in the conversation: administrators manage the rooms, never read them. Settings, **Staff chat** switches chat off, and sets how long messages are kept (0 keeps the newest thousand of each conversation).

CHAPTER 33

Conference rooms

In the web interface: `/conferences`

A conference room is a number people dial to all talk together. Staff dial it from their desks; outside callers reach it through an incoming route or a menu option that points at the room.

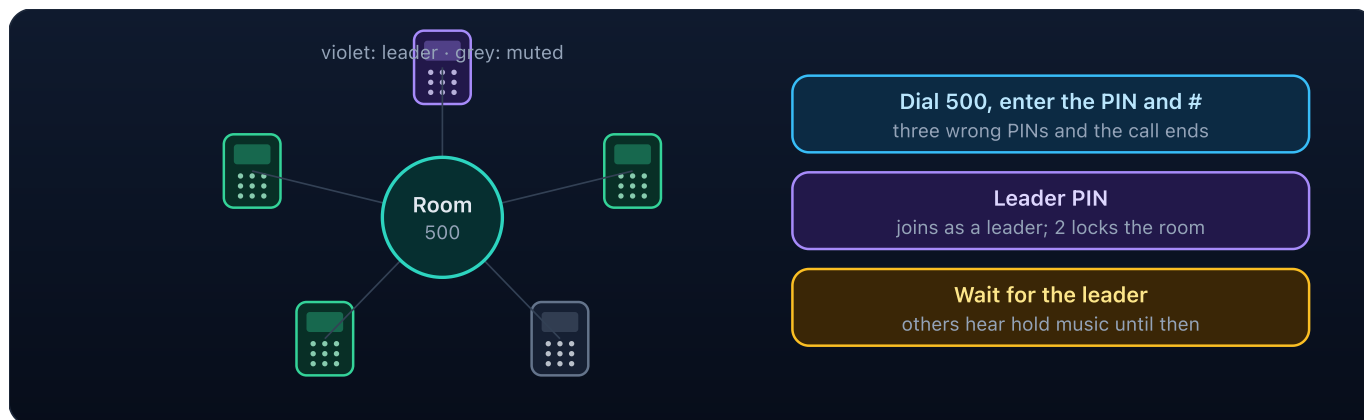


Figure 26. A conference room: PIN, leaders, and who is talking

Setting up a room

1. Open [Conference rooms](#) (chapter 33) and press **Add conference room**.
2. Give it a **number** (any free number, 500 by default) and a **name**.
3. Optionally set a **PIN**. Callers hear a tone, enter the PIN and press #. Three wrong PINs and the call ends.
4. Optionally set a **leader PIN**, different from the PIN. Whoever enters it joins as a leader.
5. To let outside callers in, point an incoming route (or a menu key) at the room.

Choices

Choice	What it does
Wait for the leader	everyone else hears the hold tone until a leader joins; when the last leader leaves, the room closes and everyone is hung up. Needs a leader PIN.
Members join muted	for briefings and lectures: leaders talk, members listen. A member presses 1 to ask to speak, and a leader unmutes them.
Record the room	the whole room is kept as one recording in Call history (chapter 46), unless recording is off for the whole system
Tones when people join and leave	on by default
Most people at once	50 by default, up to 200

Keys during a call

- **1** mutes and unmutes yourself (in a room where members are muted, it asks to speak).
- **2**, for a leader, locks the room so nobody new can join, and unlocks it.

While a room is in use

The page shows each room in use: who is in, who is talking (the green light), who is muted, who has asked to speak, and who is still waiting for the leader. An administrator can mute or unmute anyone, remove someone, and lock or unlock the room.

Everyone in a room hears everyone else who is not muted, mixed by BitVoice; each person's phone can use its own codec. A call record is kept for each person's call, and one for the room itself with its length and recording.

Conference sound is mixed at telephone quality (8 kHz), whatever codecs the phones use: it is what every phone can hear, and it keeps a room of fifty cheap to run.

HD audio

Switch on **HD audio** in a room (new rooms have it on) and it is mixed at 16 kHz instead of 8: people on wideband phones (G.722, Opus, the browser phone) hear each other in HD, with the crispness of a modern call, while people on ordinary phones (G.711, mobiles through a provider) hear the same room as usual. The live view marks the room and each member who is in HD. HD needs G.722 among the codecs the phones use (it is first in the system's list, Settings, Codecs). Recordings of the room stay in telephone quality.

Video rooms

Set **Video** on a room and video phones (and softphones) that join it send their camera and see one picture, switched as people talk, as in Asterisk's ConfBridge:

- **Follow the speaker:** everyone sees whoever has been talking for a second, once the one on screen has been quiet for a second; the person on screen sees the one before.
- **Show the leader:** everyone sees the leader (who joins with the leader PIN); the leader sees whoever spoke last.
- **Grid:** everyone sees everyone at once, side by side (one picture fills the screen, two side by side, then 2 by 2 and 3 by 3, up to nine), each person in their own place in the order they joined, themselves included.

BitVoice switches the pictures without decoding them, so it costs next to nothing, and it asks each new picture's phone for a full frame (RTCP PLI and FIR, and the SIP INFO picture-fast-update most desk phones answer) so the switch shows at once. Every member's video uses the room's **video codec** (H.264 unless you choose VP8); a phone without it joins with sound only. Each viewer gets one steady stream (one source, its own payload type), whoever is on screen. The live view marks who has video and who is **on screen**.

A grid is made differently: BitVoice gives every picture to **ffmpeg** on the server, which decodes them, places them in a 1280 by 720 picture at 15 frames a second, and encodes that once for everyone. Install ffmpeg (Settings, Video conversion, says whether it is found); without it a grid room follows the speaker instead, and the room's screen says so. Each picture costs processor time: on a small server, keep grids to a few rooms at once. Only members whose camera is sending are shown: someone joining without video, or whose camera stops, leaves the grid (it never freezes on a still camera), and the grid is remade within a couple of seconds when that changes, with a short pause in the picture. The tenth member and later are heard but not shown. The live view marks the grid's size and who is **in the grid**.

CHAPTER 34

Paging and overhead speakers

In the web interface: `/paging`

Paging speaks through many phones at once: dial a paging number, and every phone in the group answers on its speaker by itself. Use it for announcements to a warehouse, a school, a shop floor.

Setting up a paging group

1. On [Paging](#) (chapter 34) press **Add paging group** and give it a number (800 by default) and a name.
2. Choose the phones. Overhead speakers set up as extensions (kind **Overhead speaker**) can be members too.
3. Optionally, a **multicast** address and port.

Multicast

Multicast also sends the page to a multicast address and port (**239.0.1.10:5000**), as G.711 μ -law, which overhead speakers and many desk phones can listen to without a call each. A group may have only a multicast address. It works on one network: routers do not pass multicast on unless they are set up to.

Paging and intercom

	Paging	Intercom
Who hears	everyone in the group	one extension
Talk back	no, one way	yes, both ways
How	dial the group's number	*80 and the extension

Phones that do not answer a page by themselves within a few seconds are left out, so one phone without auto-answer never holds a page up. Some phones need auto-answer (intercom) allowed in their own settings.

CHAPTER 35

Hot desking

In the web interface: **/phones**

A hot desk is a shared phone that becomes whoever signs in at it: for offices where people do not have fixed desks. Make one by switching on **Hot desk** in the extension the phone is set up as (the desk's own extension, such as 900 Desk 3).

Signing in and out

1. At the phone, dial ***33** and wait for the tone.
2. Enter your extension and #, and wait for the tone.
3. Enter your voicemail PIN and #. A rising tone means you are signed in; a falling tone means it was refused.

From then on, calls to your extension ring this desk as well as your other phones, and calls you make from it show your number and name and count as yours on Costs. Signing in at another desk moves you there. ***34** signs you out. Five wrong tries lock the desk for ten minutes.

Emergency calls from a hot desk always carry the desk's own site, floor and room, not your usual ones: responders go to where the phone is.

Managing hot desks

On **Phones** (chapter 4), **Hot desks** shows who is signed in where; administrators can sign someone in (without their PIN) or out. The codes, and a time each day when everyone is signed out, are in Settings. Sign-ins are kept across restarts.

CHAPTER 36

Hotel and wake-up calls

In the web interface: **/hotel**

Wake-up calls

Anyone can book a wake-up call: guests, night shifts, people on call. From their own phone they dial ***62** : BitVoice says the one set (if any), then they enter the time as four digits (**0630**), or press star to cancel it. On [Hotel](#) (chapter 36) an administrator books one for any extension, once or **every day**.

At the time, the phone rings with the system's name; answered, it hears "Good morning. This is your wake-up call." (the wake-up prompt in [Sounds](#) (chapter 19), in the guest's or extension's language). Unanswered, it is tried again (3 tries, 5 minutes apart, both set under Settings on the page), and after the last the **reception** extension can be rung. With several servers, only one makes wake-up calls.

Guest rooms

Switch on **Guest room** on the extension of each room's phone. On Hotel each room shows its guest, wake-up call and room status, with **Check in** and **Check out** buttons.

- **Bar vacant rooms:** rooms with nobody checked in call only extensions and emergency numbers. Emergency calls always work.
- **Clear on check-out:** deletes the room's voicemail and cancels its wake-up call, forwarding and do not disturb, ready for the next guest.
- A guest's language (from the check-in) is the language the room's prompts are said in.
- **Room status:** housekeeping dial ***68** and a digit from the room's phone (***683**); the hotel system decides what each digit means.

The hotel system (PMS)

- **FIAS:** BitVoice connects to the FIAS interface of the hotel system (Oracle Opera and the many systems that speak FIAS) at its address and port. It receives check-ins, room moves, check-outs and wake-up bookings, and sends wake-up results, room status and call charges. While the link is down, what it would send waits and goes when it is back; the page says whether it is linked.
- **REST API:** the hotel system calls **POST /api/hotel/rooms/<room>/checkin** (name, account, language), **/checkout** , and **PUT /api/wakeups/<room>** , with an API key from Settings, Automation. Charges are posted as JSON to the address you give.
- **Charge calls:** answered outside calls from occupied rooms go on the guest's bill, at the trunk's rates (from [Costs](#) (chapter 53)) plus the **markup**.

CHAPTER 37

Contacts on every phone

In the web interface: **/contacts**

Contacts is the company directory: the suppliers, clients and other offices people call. Every phone lists it beside the extensions, and when one of these numbers rings in, phones and the caller card show the contact's name (when the provider sent none). A number matches however it is written: +1 212 555 0100 and 2125550100 are the same line.

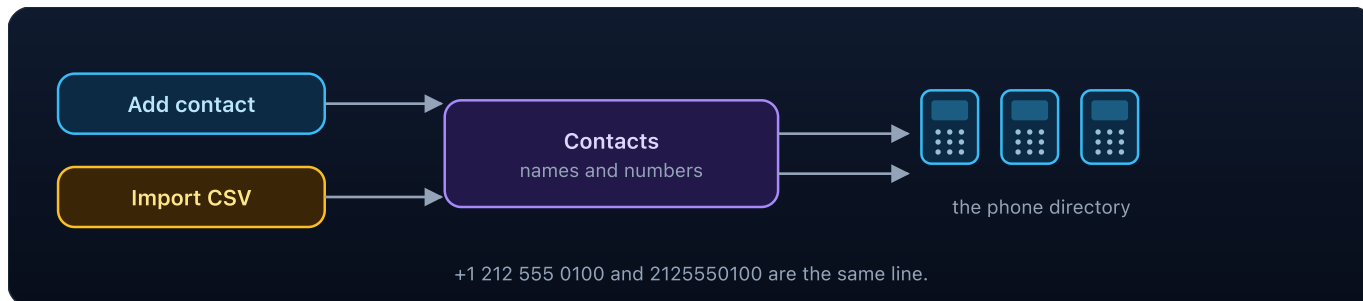


Figure 27. Contacts: one directory on every phone

Adding contacts

Add contact for one at a time, with as many numbers (work, mobile, home, other) as they have. **Import** takes a spreadsheet saved as CSV whose first row names the columns: name, company, work (or phone), mobile, home, other, email, notes. BitVoice checks the whole file first and says which lines are wrong; a contact with the same name as one there is updated rather than added twice, and **Replace every contact** starts the list afresh. **Export** downloads the list as CSV.

On desk phones

Phones set up by [Phones](#) (chapter 4) (provisioning) get the directory with no more setup:

- **Yealink**: a remote phonebook named Company, refreshed every hour.
- **Grandstream**: the phonebook download, every hour.
- **Polycom**: the directory file, read when the phone starts.
- **Snom**: the list is written into the phone's settings, so it updates when the phone next checks them.

In the browser phone

Contacts under the keypad finds a name or number and puts it ready to call.

CHAPTER 38

Calendars and do not disturb

In the web interface: [/extensions](#)

An extension can follow its person's calendar: during meetings and time off, its calls go to voicemail, or are forwarded (to another extension, or a mobile), instead of ringing a phone nobody will answer. Ring groups ring the others meanwhile, and in queues the agent shows as **Away**.

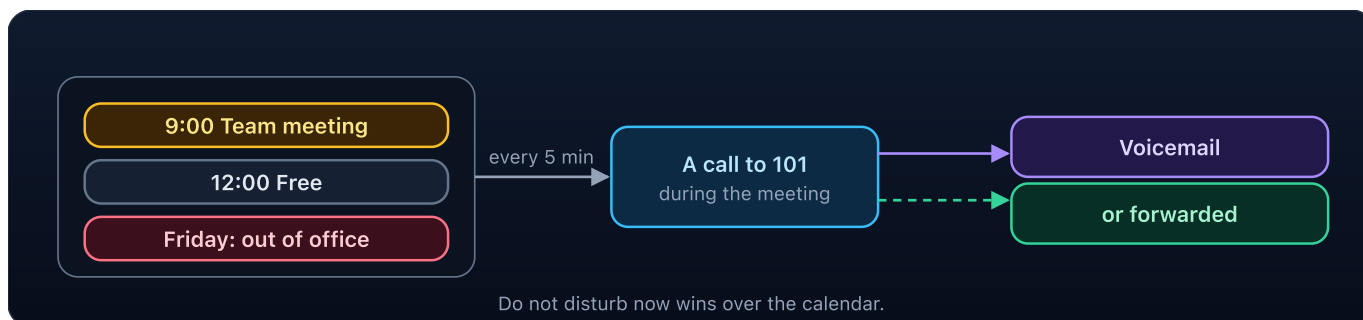


Figure 28. Following a calendar: no ringing in meetings

In the extension, **Calendar and do not disturb**, paste the calendar's private address:

- **Outlook:** Settings, Calendar, Shared calendars, Publish a calendar; choose the calendar and "Can view all details", then copy the ICS link.
- **Google Calendar:** Settings, the calendar on the left, then **Secret address in iCal format**.

Check reads it at once and lists the next busy times. BitVoice reads each calendar every five minutes. Time off (all-day events, and Outlook's "out of office") always counts; meetings count when **Meetings count as busy** is on; events marked free never count, nor do cancelled ones. Recurring meetings, with skipped and moved occurrences, are understood. If a calendar cannot be read, Health says so.

Do not disturb now stops the extension ringing until switched off, whatever the calendar says. The extensions list shows who is away and until when.

The private address works like a password: anyone with it can read that calendar, so it is kept with the other secrets.

CHAPTER 39

Mobile apps

The BitVoice apps for iPhone and Android make a mobile phone one of an extension's phones. It rings like a desk phone, on the phone's own call screen, even when the app is closed.

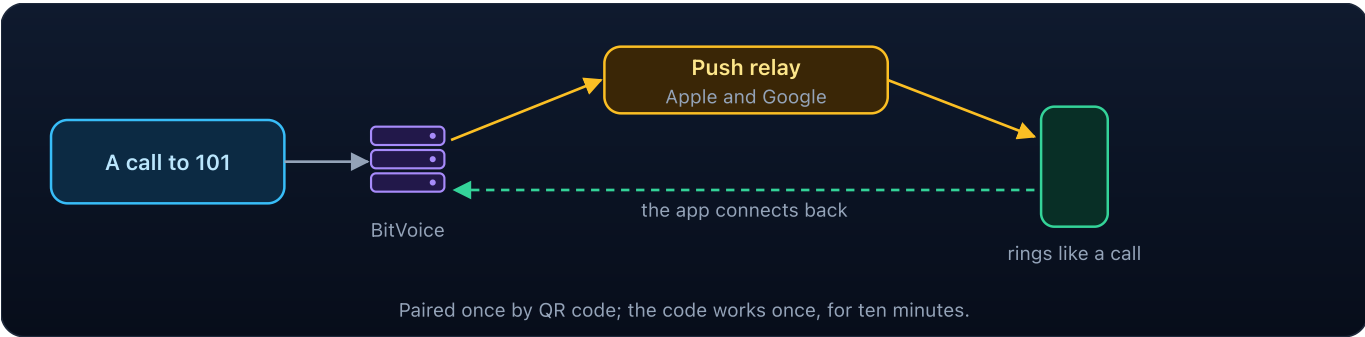


Figure 29. The mobile apps: woken by a push, rung like a desk phone

Setting up push

A closed app is woken for calls by a push message. In [Settings](#) (chapter 14), **Mobile apps**, choose one:

Push	What you need
Push relay	BioDream's relay (with hosting or support) or your own <code>bvpush</code> : its Relay address , Key ID and Secret
Own apps	your own builds: for Apple, the Team ID , Bundle ID , Key ID and Key (.p8) ; for Google, the Firebase service account (JSON)

Set **Address the apps use** to this server's https address as it is reached from anywhere, not only the office.

With push off, the app rings only while it is open.

Pairing a phone

1. Open the extension and press **Add a phone**.
2. Scan the QR code in the app. The code works once, for ten minutes.

When a call comes in

BitVoice wakes the app; it connects, rings on the phone's own call screen, and is answered like any other phone of the extension: ring groups and queues ring it with the extension's other phones. The extension's page lists its paired phones, whether each **Can be woken for calls**, when it was last seen and last woken, and any push problem. **Unpair** stops a phone ringing at once, for a lost phone.

CHAPTER 40

Video calls

In the web interface: `/settings`

When both phones in a call send video (video desk phones, softphones, video doorbells), the picture passes between them through BitVoice. Nothing to set up: it is on unless **Settings → Video calls** is switched off.

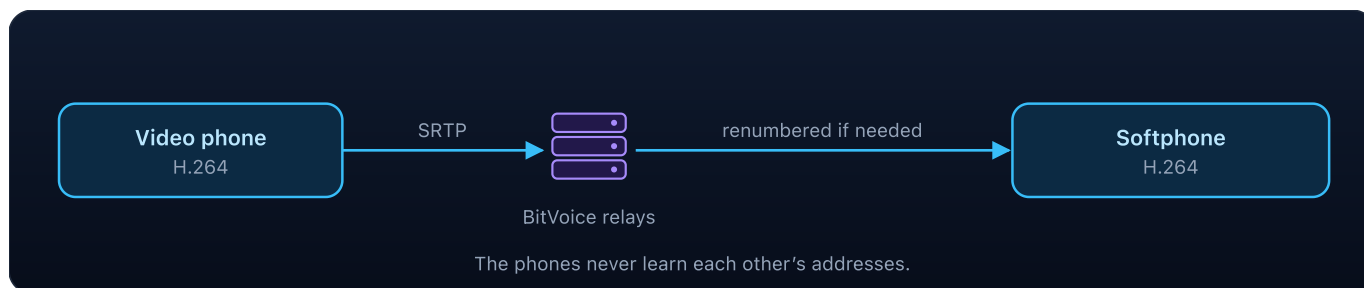


Figure 30. Video calls: relayed as they are

- BitVoice relays the video as it is, on a port of its own for each phone, so the phones never learn each other's addresses. With a codec in common (nearly every video phone has H.264) it never decodes the video; without one, it can convert (below). When they number the same codec differently, BitVoice renumbers it on the way.
- The picture is encrypted (SRTP) to each phone that encrypts its sound.
- A phone's request for a fresh picture (after lost packets) reaches the other phone, so the picture recovers in a moment.
- A camera switched on during a call (a softphone's video button) works too: the other phone is asked first, and the picture starts when it says yes.
- A phone without video, or a call to an outside number, gets sound only. On hold or while parked the picture pauses, and comes back when the call does. After a transfer it carries on only when the new party sends video too.
- Video uses far more of the network than sound (about 0.5 to 2 Mbit/s each way per call): allow for it on slow links and in the media port range.

Converting video

Phones and apps that share a video codec (most use H.264) get each other's video as it is. When two parties have none in common (a desk phone with only H.264 and a browser with only VP8), switch on **Convert video between H.264 and VP8** in Settings: BitVoice then offers both codecs, answers each party in its own, and converts each way with **ffmpeg**, which must be installed on the server (apt install ffmpeg, brew install ffmpeg; or give its path). Conversion costs about a processor core per call and adds a little delay, so it is off by default and used only when needed. Settings says whether ffmpeg was found.

Fax: sending and receiving, fax machines and analogue phones

In the web interface: `/faxes`

Fax machines, analogue phones, door phones and old lines connect through a SIP gateway (or a telephony card in the BitVoice computer: see [Telephony cards](#) (chapter 6)): an ATA (Grandstream HT series, for one or two analogue phones or a fax), or a bigger gateway (Patton SmartNode, AudioCodes MediaPack and Mediant) for many analogue ports, an ISDN line (BRI/PRI) or old analogue lines (FXO). Each port registers to BitVoice as an extension, like a phone.

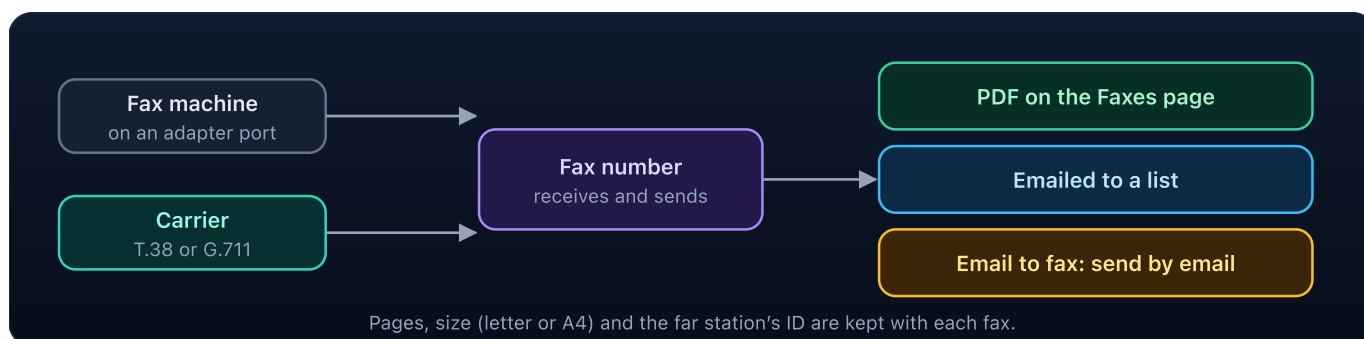


Figure 31. Fax: machines, email and PDFs

Receiving faxes without a fax machine. On **Faxes**, add a fax number (300, say), with the email addresses each fax goes to. Then in [Call routing](#) (chapter 8) send a phone number's calls to it. BitVoice answers as a fax machine, switches the call to T.38 and takes the fax; each fax becomes a PDF, kept on Faxes (**View, PDF**) and emailed. A fax that breaks off part way keeps the pages that arrived and says so. This needs a provider that supports T.38 (most do); a fax sent as sound cannot be received this way.

Sending a fax. On **Faxes**, **Send a fax**: choose the fax number it goes from, type the number to send to, write a note if you like (it becomes the first page) and add documents. Text files and pictures (PNG, JPEG, GIF) work everywhere. PDF needs poppler on the server (`apt install poppler-utils` , `apk add poppler-utils` , `brew install poppler`); Word, Excel, OpenDocument and RTF need LibreOffice as well. The page says which the server can do. Every page gets a header line, as fax machines print it: the time, the fax number's name, its identity and the page count. Pages are US letter in North America and A4 elsewhere, from the country in Settings.

BitVoice calls the number over your outbound routes, as an extension would, and sends at the fastest rate the line carries (14,400 bit/s, slower if the line is poor). Faxes from one fax number go one at a time. A busy number, or one that does not answer, is tried again five minutes later, three times in all. Sent faxes are listed with the received ones: **Sent**, **Waiting to send**, **Failed** (with **Try again**), or **Part sent** when the line broke after some pages.

Email to fax. Below the list, set up a mailbox for BitVoice to read (IMAP over TLS, port 993; for Google and Microsoft mailboxes use an app password), then **Test** and **Save**. Anyone allowed to send writes to that mailbox with the fax number in the subject (`Fax 212 555 0199`), or to a plus address (`fax+2125550199@example.com`) if the mail server takes them. Attachments become the pages and the email's own words the cover note (the signature and quoted mail are left out). It goes from the fax number that lists the sender: its **Email each fax to** addresses may always send, and **May also send faxes by email** adds others (whole addresses, or `@example.com` for a domain). The sender gets an email when the fax has gone through, or why it could not.

Email is easy to forge, so BitVoice faxes a message only when your mail server found it really came from the sender's domain (DKIM, SPF or DMARC, in the Authentication-Results line it adds). Other messages are dropped without a reply. Switch **Accept mail the sender's domain did not vouch for** on only for a mailbox no outside mail can reach.

Fax machines on gateways. BitVoice carries fax over T.38, the fax standard for the internet: when the gateway hears a fax tone it switches the call to T.38, BitVoice asks the other side (your provider, or another gateway) to switch too, and passes the fax

between them. If the other side cannot do T.38, the call stays on G.711 and the fax goes as sound, which works on a good network. Live calls show **fax (T.38)** while a fax is going through.

For reliable faxes:

- Use T.38 on the gateway and choose a provider that supports it.
- Make the port's extension an **Analog adapter port** (Extensions, Kind): BitVoice then offers it only G.711, which faxes sent as sound need.
- On the gateway's fax port: Fax Mode T.38, only PCMU or PCMA, line echo canceller off, call waiting off, and a fixed, high jitter buffer.

CHAPTER 42

Inbox: web chat, WhatsApp and Messenger

In the web interface: `/inbox`

The **Inbox** (chapter 42) holds every written conversation in one list: the chat box on your website, WhatsApp, Facebook Messenger and **texts** (chapter 43). Anyone signed in to the web interface can answer; the first to reply takes the conversation (anyone can hand it on with **Answered by**), and **Close** puts it away until the person writes again. Unread conversations are counted, and the list refreshes by itself.

The chat box on your website

Under **Channels**, switch on **Web chat**, choose its title, colour and first line, and put the line it shows in your site's pages:

```
<script src="https://pbx.example.com:8443/webchat.js" async></script>
```

It draws a chat button in the corner of every page; visitors write (with their name if they like) and see your answers as they come. The conversation stays in their browser, so it carries on as they move between pages. List your sites under **Only on these sites** so the box runs nowhere else. Visitors must reach this server's address from the internet, with a certificate browsers trust (**Let's Encrypt** (chapter 14)). Ten new chats from one address in ten minutes is the most BitVoice takes.

WhatsApp

WhatsApp Business numbers on Meta's Cloud API. In Meta's developer site, make an app with WhatsApp, add your number, and make a system user with a permanent access token. In **Channels**, switch on WhatsApp and give the **phone number ID** and the **business account ID** (both from API setup), the **access token** and the app's **app secret**, then save. In the app's WhatsApp configuration, set the webhook: the callback address and verify token BitVoice shows, subscribed to ***messages***.

WhatsApp lets you answer freely for 24 hours after the customer's last message; after that it accepts only **message templates** approved in WhatsApp Manager. The reply box then turns into a template picker: choose a template (it lists your account's templates, with their language and kind), fill in its values (**{{1}}**, **{{2}}** ...), check the preview, which is what the customer will read, and **Send template**. Templates the inbox cannot fill are listed greyed out with the reason: one with a picture, video or document header, a variable in its header or a button's link, named parameters, or a code button; and those not yet approved. Make templates in WhatsApp Manager; Meta reviews each one, usually within minutes.

Facebook Messenger

For a Facebook Page: in the Meta app, add Messenger, connect the page and make a page access token. In **Channels**, give the **page ID**, the **page access token** and the **app secret**; then set the webhook (the address and verify token shown), subscribed to ***messages*** and ***messaging_postbacks***. Replies are allowed for 24 hours after the person's last message. From then until 7

days, Messenger takes replies written by a person (sent with Meta's *human agent* tag: the app needs Meta's Human Agent permission, asked for in App Review); the automatic greeting never goes then. After 7 days the inbox takes no reply until they write again.

Every webhook is checked with the app secret (Meta's signature) before BitVoice believes it, and a message Meta sends twice is kept once. Pictures, voice notes and other attachments show as what they are ("[image]"); open them in Meta's own apps.

Automatic greeting, when set, answers the first message of every new conversation at once.

CHAPTER 43

Business texting (SMS)

In the web interface: `/texts`

Send and receive text messages from your business numbers on **Texts**. Texts go through a texting provider. Built in: TextMagic, Twilio, SignalWire, Telnyx, Bandwidth, Vonage, Plivo, Flowroute and VoIP.ms, plus **Your own webhook** for any other service or a gateway of your own. Any number of provider accounts can be used at once: each number says which one carries it.

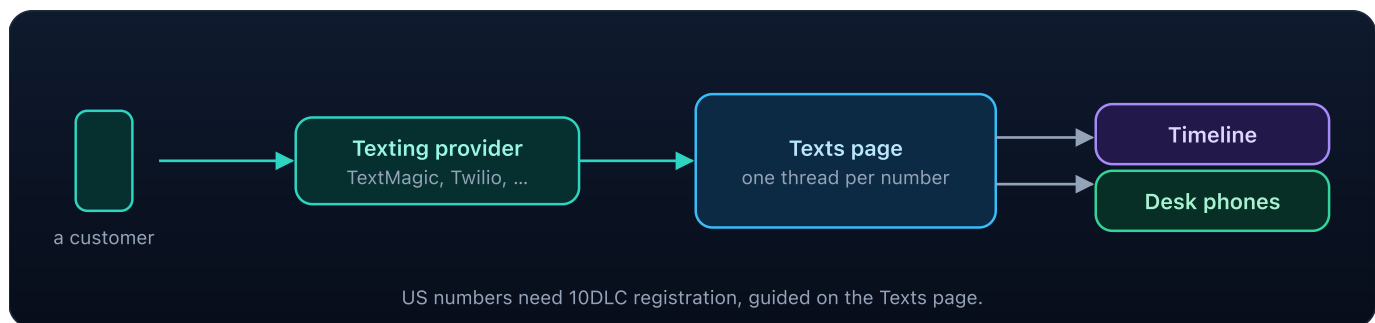


Figure 32. Business texting: the same numbers, one conversation view

Setting up

1. In the provider's portal, find the account's API details. The form asks for exactly what that provider calls them (Twilio: Account SID and auth token; Telnyx: an API key; Bandwidth: API user, password, account ID and messaging application ID; SignalWire: also your space address) and says where to find them.
2. On Texts, **Texting setup**, add the provider with them, and save. BitVoice shows an address for received texts: set it in the provider as where received messages are sent (its callback, webhook or message URL). For VoIP.ms, add `?from={FROM}&to={TO}&message={MESSAGE}&id={ID}` to the end. Delivery receipts sent to the same address are ignored. BitVoice must be reachable from the internet there.
3. Add each number you text from (with its country code), the provider that carries it, and the extensions to tell by email when a text comes in.

US 10DLC

US carriers block texts from ordinary business numbers that are not registered. Register your brand and then a campaign with your texting provider; the checklist on Texts tracks where they stand, and warns when your sample message lacks the opt-out words carriers expect.

Conversations

Each conversation shows who it is (from Contacts), unread texts, and the thread. Enter sends; the counter shows how many texts a long message takes. When someone replies **STOP**, nothing more can be sent to them from that number until they reply **START**. Received texts also appear as a notice in BitVoice.

CHAPTER 44

Transfer briefings

In the web interface: `/settings`

So a caller never has to repeat themselves: with **Brief whoever a call is transferred to** on (Settings, AI), when a call is transferred, BitVoice writes a two-sentence summary of the call so far (who is calling, what they want, anything already promised) for the person it goes to. It appears on their browser phone with the ringing call, on [Live calls](#) (chapter 45), and as a text on their desk phone's screen (for phones that show SIP messages).

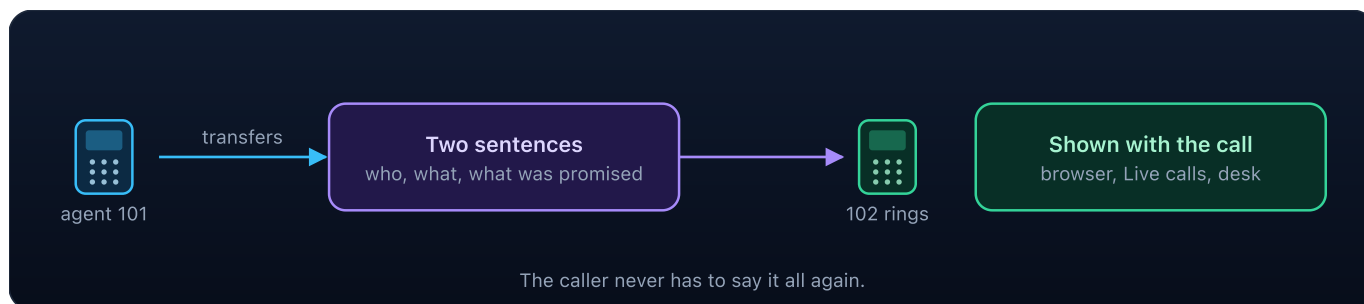


Figure 33. Transfer briefings: the next person knows already

It works the same when the AI receptionist hands a caller to a person: the briefing comes from what the caller told the assistant.

It needs the AI service and a speech-to-text model (Settings, AI), and the call's audio so far (up to five minutes) is sent to that service to be written out. A summary takes a few seconds, so it may appear just after the phone starts ringing.

Customers and insight

Knowing who called and how it went: the live calls and history, the customer timeline, CRMs, recordings and consent, quality, costs and caller ID reputation.

CHAPTER 45

Live calls

In the web interface: `/calls`

Live calls shows every call in progress, updated as it happens: who called whom, how long it has been going, its codecs and its quality.

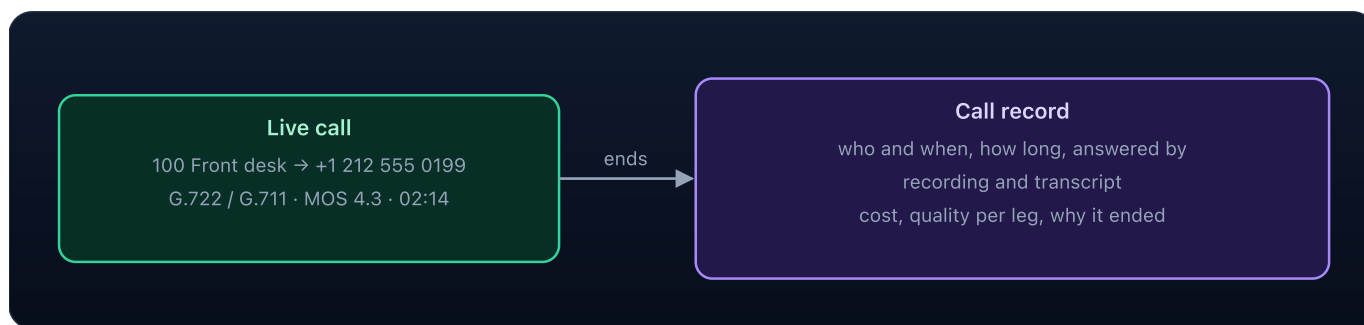


Figure 34. Live calls and history: every call, while it happens and after

What administrators can do

Action	What it does
Hang up	ends the call for both sides
Open the door	for a call from a door station
Pause recording	for a card number: the paused stretch is silent in the recording and marked in the history (Call recording (chapter 50))
Listen	hear the call silently
Whisper	talk to the agent only; the caller does not hear
Barge	join the call; everyone hears you

To listen, whisper or barge, press **Supervise**, choose the phone to use under **From phone** and press **Ring me**: your phone rings, and when you answer you join the call. You can switch between listening, whispering and barging while you are in it.

What else shows

- A caller card while an outside call rings (from your CRM, or past calls; see [CRM](#) (chapter 48)).
- The briefing written for the person a call was transferred to ([Transfer briefings](#) (chapter 44)).
- The caller ID verdict: **Verified**, **Partly verified** or **Unverified** ([Spam protection](#) (chapter 55)).
- With several servers, calls on the others too, read only: hang up or listen on the server that has the call.

When a call ends, it moves to [Call history](#) (chapter 46).

CHAPTER 46

Call history

In the web interface: `/history`

Every finished call: who called whom, when, for how long and how it ended. Sort the columns, play recordings (a pause for card numbers is marked), and download the list as CSV. How long records are kept is set in Settings, Call recording; a [legal hold](#) (chapter 70) keeps them longer.

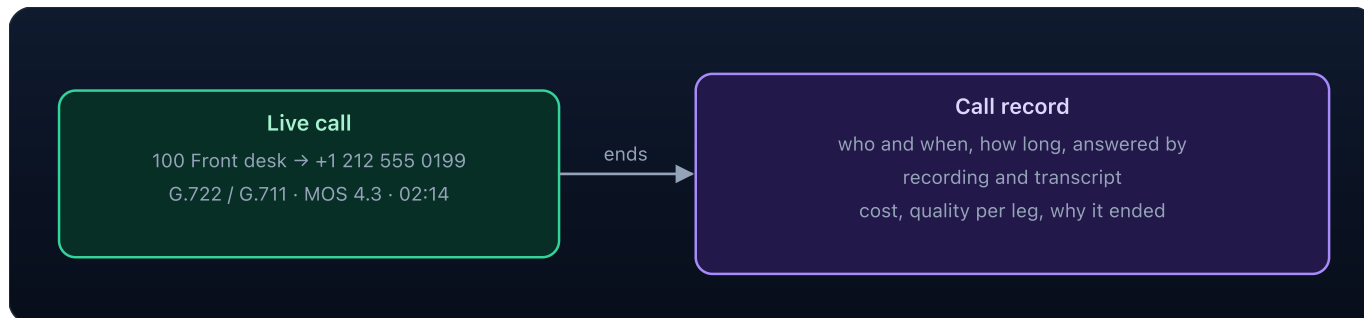


Figure 35. Live calls and history: every call, while it happens and after

How it was routed

Under each call's result, **How it was routed** lists, in plain words and in order, what decided where the call went: the incoming route that took it, whether a schedule was open or closed and the time, a toggle's side, what a routing script chose (or that its fallback took the call), the keys pressed in a menu, who was away, who had no phone connected, who did not answer, the outbound route and trunks used (and a trunk the health judge moved behind the others), the toll-fraud guard's judgement, and, for a recorded call whose parties are under different consent rules, which rule applied.

Transcripts and search

With AI on and **Transcribe recorded calls** switched on (Settings, AI), every recorded call is written out, with a short summary and its action items: what someone agreed or was asked to do. Each side of the call is written out separately from the recording, so the transcript shows who said what, and when. The transcript button next to a recording opens it; administrators can also transcribe a single call, or transcribe one again.

Search what was said finds calls by words from their transcripts, summaries and action items. Only recorded calls are transcribed. A transcript is deleted with its recording, and reading or searching transcripts is noted in the audit log, like playing a recording.

CHAPTER 47

Customer timeline

In the web interface: `/timeline`

The customer timeline shows everything with one phone number over the last year, newest first, so whoever picks up knows the whole story.

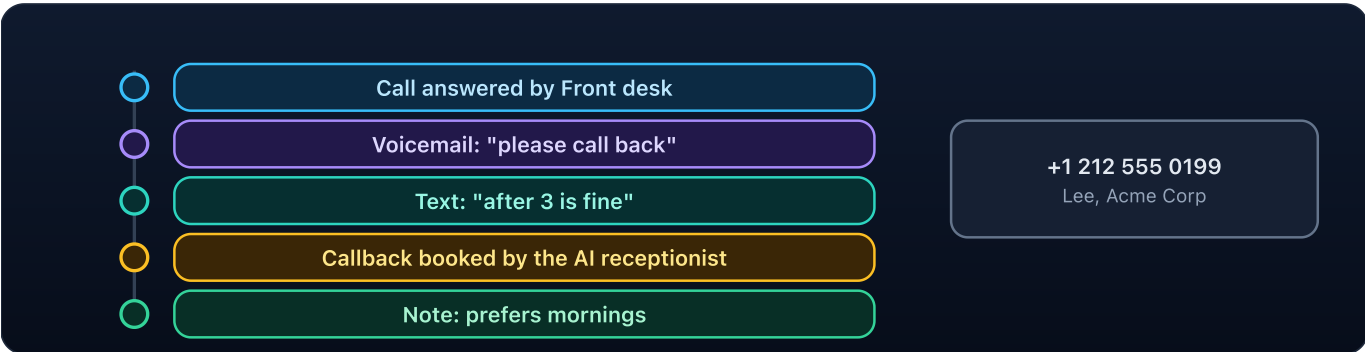


Figure 36. One customer timeline: everything about a number in one place

What it shows

Item	With
Calls	who took them, how long, and with a transcript, its summary, the recording and the transcript
Voicemails	their summaries
Texts	both ways
Callbacks	asked for, and whether done
Notes	the team's own notes

A number matches however it is written: +1 212 555 0100 and 2125550100 are the same.

Opening it

From a caller card (**Timeline**), from Contacts, from an incoming call in Call history, from a text conversation, or type a number at the top of **Timeline** (chapter 47).

Notes

Administrators can **Add a note for the team**: it shows on the timeline for everyone, with who wrote it and when, and can be removed.

CHAPTER 48

CRM: caller cards and call logging

In the web interface: `/settings`

Connect a customer system in **Settings** (chapter 14), CRM. It works with any system that has a web address, without code written for each product.

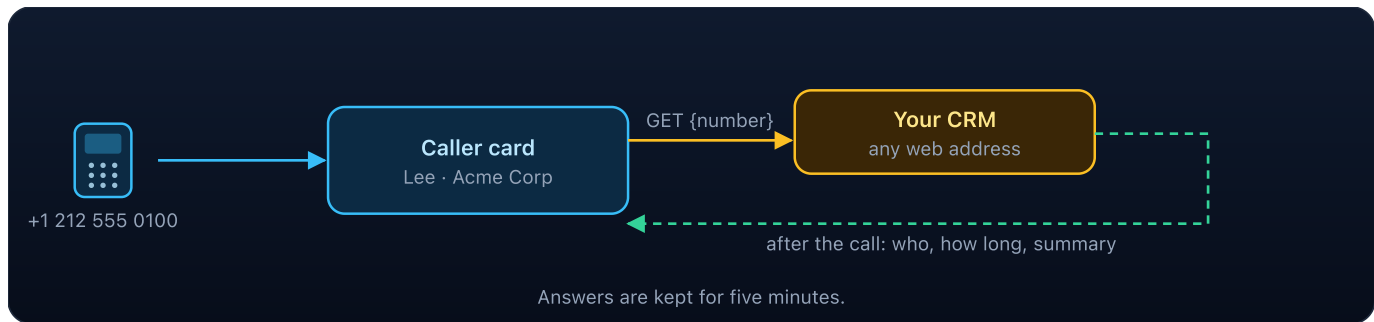


Figure 37. CRM: a card while it rings, a log after it ends

Caller cards

While a call from outside rings, Live calls shows a card: who it is (from your CRM, or the extension's name), how often they have called in 90 days, any [callback](#) (chapter 29) they are waiting for, and the summary of their last transcribed call. The browser phone shows a shorter card to the person answering: who it is, past calls and waiting callbacks, but not other people's call summaries.

For the name, set **Lookup address**: BitVoice asks it (GET) with the caller's number in place of {number}, sending **Lookup header** with its value (a vault reference works), and expects JSON with name, company, url and notes; 404 means unknown. Answers are kept for five minutes. **Open in CRM link** puts a link to the caller on the card. Use **Look it up** to try a number before saving.

Call logging

Set **Call log address** and **Signing secret**, and every call is sent there as it ends (event call.ended), again with its summary, action items and transcript once transcribed (call.summarized), and each booked callback (callback.new). Each POST carries X-BitVoice-Event, X-BitVoice-Timestamp and X-BitVoice-Signature: sha256= followed by the HMAC-SHA256, with the secret, of the timestamp, a dot and the body. A failed delivery is tried four times over about a minute; if it still fails, Health says so. **Send a test event** checks the address and the secret.

CHAPTER 49

Call insight for supervisors

In the web interface: `/quality`

For contact centres: evidence about how calls went, for a supervisor to listen to, never an automatic score.

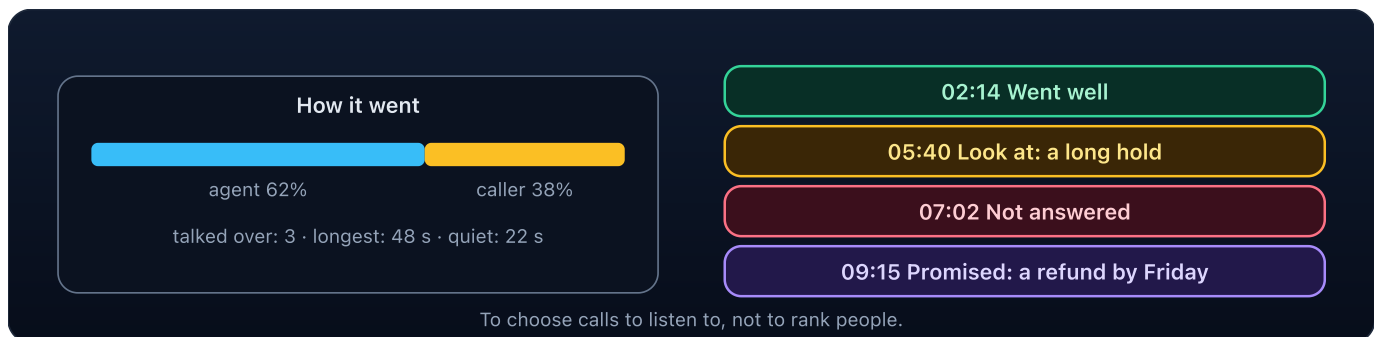


Figure 38. Call insight: evidence for a supervisor, never a score

In a call's **Transcript**, **How it went** shows, measured from the recording's sound alone: how much each side talked, each side's longest stretch, how often each started talking over the other, how long both spoke at once, and the quiet (a long silence is often a hold or dead air).

With **Note how each transcribed call went** on (Settings, AI), the AI also reads the transcript and says how the caller seemed at the start and the end, and lists up to six moments: **Went well**, **Look at** (talking over the caller, a long hold, an unclear answer), **Not answered** (a question the caller asked that got no answer) and **Promised** (something someone said they would do), each with the time and the words said, so you can go straight to it. An administrator can ask for it on any transcribed call with **Look at this call**.

On **Call quality** (chapter 49), **Conversations, by who took them** sums the transcribed calls of the period per person: their share of the talking, their longest stretch, how often they talked over the caller, quiet time, calls that ended with the caller upset, and the moments flagged. Use it to choose calls to listen to, not to rank people.

Coaching notes are your own words on a call, at the moment they are about: under **How it went**, write the **Moment** (**1:05** , or leave it empty for the start) and the **Coaching note**, and press **Add note**. Notes are listed in the order of the call with who wrote them and when; administrators write and remove them, viewers read them with the transcript, and adding or removing one is noted in the audit log. They go with the recording, so its retention and legal holds cover them too. **Coaching notes** on Call quality counts them per person.

CHAPTER 50

Call recording and card payments

Recording is set in **Settings** (chapter 14), **Call recording**: off, every call, or only the calls of extensions marked **Record calls**. Recordings play from **Call history** (chapter 46), and with AI on they are written out and summarised.

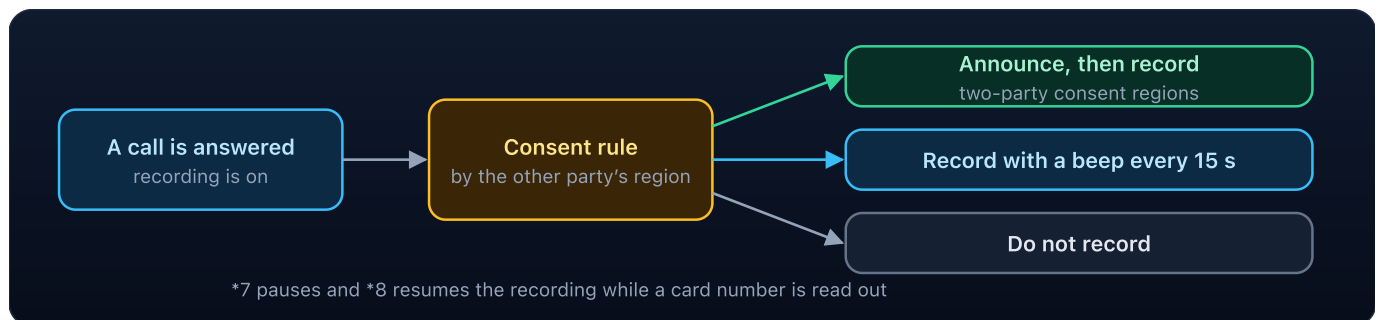


Figure 39. Recording with consent: announce, beep, or not at all

Telling people

Some places need everyone on a call to be told it is recorded: **Announce, Beep** and rules by region are in **Recording consent** (chapter 51).

Card payments (PCI DSS)

Card numbers must not be kept in recordings. Before a caller reads one out, the person on the extension presses ***7** ; ***8** resumes. **Pause recording** on **Live calls** (chapter 45) does the same. The paused stretch is silent in the recording and listed in the call history, so an auditor can see where and for how long.

Conference rooms

A room with **Record the room** on is kept as one recording of the whole room (**Conference rooms** (chapter 33)).

How long things are kept

Settings sets how many days recordings, call history and voicemail are kept; older ones are deleted within the hour. Anything under a [legal hold](#) (chapter 70) is kept whatever its age. The [archive](#) (chapter 69) can keep copies elsewhere.

Administrators and viewers can play recordings and read transcripts; only administrators can transcribe a call, or transcribe it again. Playing a recording or reading its transcript is noted in the audit log.

None of this is legal advice: recording laws differ by country and state. Check the laws where you and your callers are.

CHAPTER 51

Recording consent

In the web interface: `/settings`

Some places let a call be recorded when one person on it knows; others need everyone to be told, or to agree. In Settings, **Call recording**, choose how people on a recorded call are told:

- **No notice.**
- **Announce:** as the call is answered, both sides hear the notice (record it in the same place, for example "This call is recorded for training and quality"), and recording starts only after it has played. Until one is recorded, three beeps are played, and Health says so.
- **Beep:** a short tone to both sides every 15 seconds while the call is recorded.

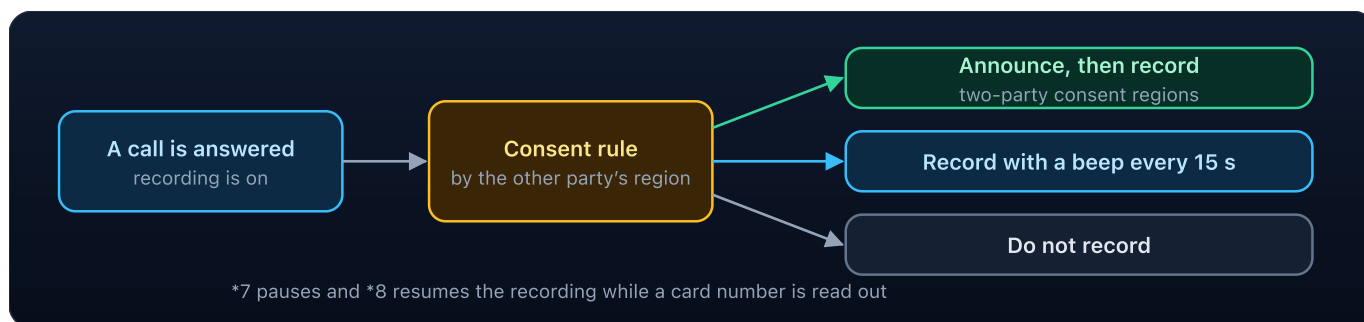


Figure 40. Recording with consent: announce, beep, or not at all

Rules pick by the outside number (the caller on calls in, the number dialled on calls out), longest match first: for example numbers starting 1415 (California) **Announce**, 49 (Germany) **Never record**, everything else as above. A rule's **No notice** records without one; **Never record** leaves those calls out of recording altogether. Calls between extensions follow the main setting.

Your own number counts too: the rules are matched against both the outside number and yours (the number called on calls in, your caller ID on calls out), and the **most protective** of them wins (**Never record** before **Announce** before **Beep** before **No notice**). So an office in a place that needs every party's consent announces whoever it calls, wherever they are. When the parties fall under different rules, the call's record in [Call history](#) (chapter 46) says which rule applied and why.

Recording can also be paused during a call with the pause code (card payments). None of this is legal advice: check the laws where you and your callers are.

CHAPTER 52

Call quality

In the web interface: [/quality](#)

How calls sounded over the last day, week, month or 90 days, from the sound BitVoice itself received on each side of each call. Phones are grouped by their [site](#) (chapter 12) (or "no site"), and calls from outside by the provider's line they came in on.

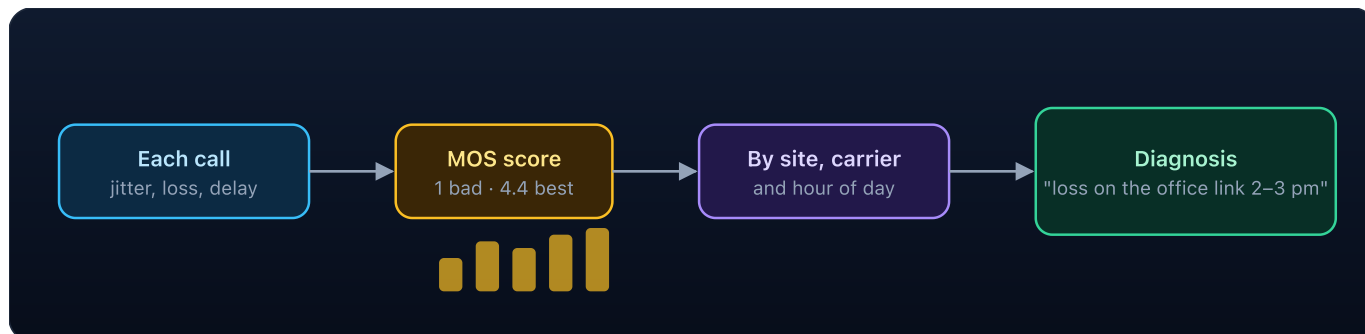


Figure 41. Call quality: from packets to a plain diagnosis

Each place shows its calls, its average score (MOS, out of 5: above 4 is good, under 3.6 sounds poor), the share of sound packets that never arrived, and jitter (uneven delay). Below that, BitVoice says what is wrong in plain words, for example "Packet loss on Head office's network: 3.2% of the sound never arrived, mostly between 14:00 and 16:00", with what to check. The bar chart shows the sound lost by hour of the day.

The diagnosis compares the two halves of the same calls: the sound from the site's phones and the sound from the provider the same calls went through. When the other half arrived clean, the loss is on this place's side, and it says so; when both halves lost sound, the place is not to blame alone and it points at what the calls share, this system's own internet link. If every place loses sound at once, the cause is likewise what they share. **How this was worked out** lists the reasons, step by step (the call-quality rulebook, see [Security](#) (chapter 58), "How BitVoice decides"). Places with fewer than three calls are not judged. Per-call quality is also in [Call history](#) (chapter 46) (the Quality column) and, while calls are live, on [Live calls](#) (chapter 45).

CHAPTER 53

Costs and budgets

In the web interface: [/costs](#)

BitVoice prices each call from what your provider charges, and shows the month's spend by department, person and trunk, with the most expensive calls.

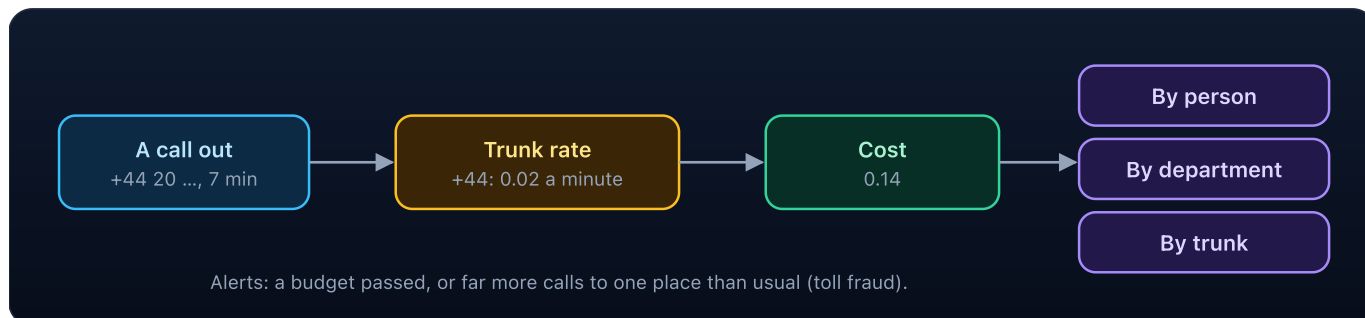


Figure 42. Costs: every call priced, budgets watched

Prices

On [Trunks](#) (chapter 7), open a trunk and its **Call prices**. Each price covers numbers that start with some digits, as the number is sent to the provider (digits only, so 44 for the UK, 1900 for US premium rate); the longest match wins, and a price with no digits covers every other number. Give the price per minute, any charge per answered call, the billing step (60 seconds for whole minutes, 6 or 1 for finer) and any minimum. **Calls in, per minute** is for toll-free numbers you pay to receive. **What would it cost** checks a number before anyone dials it. The currency is set on Costs.

A call's cost is fixed when it ends, with the prices of the time; changing a price does not change past calls. Costs says how many answered calls out had no price.

Departments and budgets

Give extensions a **Department** (in the extension). On Costs, add a budget for every call or for one department: an alert appears in BitVoice, and is emailed to the budget's address and the alert address, at 80% and again at 100% of the month.

The alarm

Calls out cost more than this in an hour raises an alarm when outgoing calls pass that amount within any hour. A sudden run of expensive calls is the usual sign of toll fraud (a stolen phone password): check Call history, then disable the extension or the trunk.

CHAPTER 54

Caller ID reputation and branded calls

In the web interface: `/reputation`

Phone companies run spam filters that can label a business number "Spam likely", or block it, and then hardly anyone answers.

Caller ID reputation watches each number you call out from, from your own calls, and says in plain words how it is doing:

- **Answered**: the share of calls answered, and what it was the period before. A fall by half, or under 20%, is the usual sign of a new label.
- **Hung up at once**: answered calls that ended within 10 seconds.
- **Refused as spam**: calls a carrier's filter refused (SIP 608) or that the person marked unwanted (607). One 608 means the number is labelled.
- Heavy use: more than about 150 calls a day from one number is what robocallers do, and filters notice.

Numbers with fewer than 20 calls are shown as too few to judge.

Your outgoing numbers gives each number a **name** (sent as the caller's name), a **reason** for calling and a **logo** (an https:// image). On a trunk with **Send branded caller ID** on, the logo and reason go out as rich call data (RFC 9796 Call-Info); carriers that support branded calling show them on the called phone. Most phones show the name from the caller-name (CNAM) database instead, which your provider sets: ask them. **Watch** puts a warning on Health when a watched number starts to look labelled.

The page also says what to do when a number is labelled.

CHAPTER 55

Spam protection and STIR/SHAKEN

In the web interface: </settings>

In the US and Canada, the provider a call starts with signs its caller ID (**STIR/SHAKEN**): **A** when it knows the caller may use the number, **B** when it knows the caller but not the number, **C** when it only passed the call on. Your provider either tells BitVoice its verdict (verstat) or passes the signed Identity header on, which BitVoice then checks itself: the signature, that it is under a minute old, and that it names the numbers of this call. The verdict shows as **Verified**, **Partly verified**, **Unverified** or **Not verified** on Live calls and in Call history.

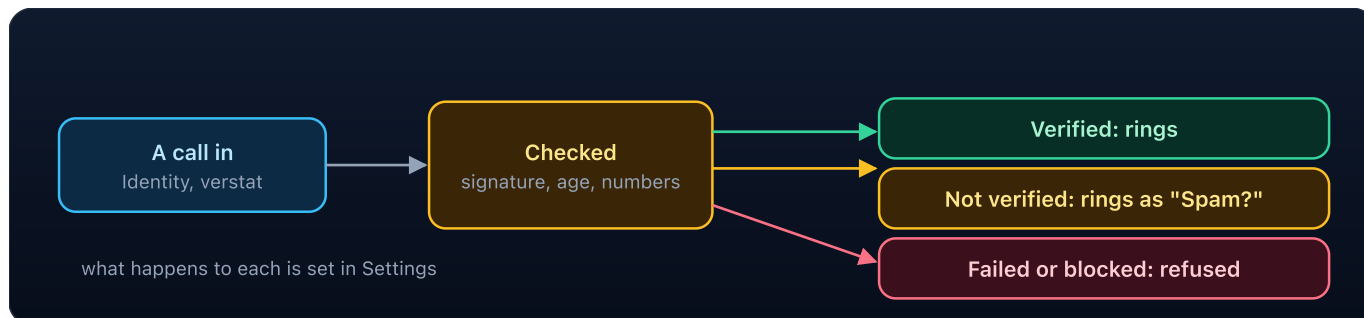


Figure 43. Spam protection: the caller ID is checked before it rings

In Settings, **Spam protection**, choose what happens when a caller ID fails verification (ring, ring labelled, or refuse), when it is not verified at all, and when the number is withheld. A labelled call rings with **Spam?** (or your own label) before the caller's name.

Blocked numbers are refused; an entry ending in * blocks a range (+1900* is every US premium-rate number). **Always allowed** numbers are never refused or labelled. In Call history, the block button beside a caller adds them to the list; refused calls are listed as **Blocked**.

Within an hour of an emergency call from any extension, no call is refused (blocked ones ring labelled instead), in case it is the emergency service calling back.

To check certificates against the STI authorities, paste their certificates under **STI certificate authorities**; without them, BitVoice checks that the call was signed with the certificate it names but not who issued that certificate.

Signing your own calls

Most providers sign their customers' calls themselves. If yours lets you sign (or you are a provider running BitVoice), switch on **Sign caller ID** on the trunk ([Trunks](#) (chapter 7)) and give it your **STI certificate** (from an STI certification authority, with your service provider code token) and its **private key** (P-256). Every call out through the trunk then carries a signed Identity header (ES256, as SHAKEN requires).

- **Certificate address (x5u)** is where the called side fetches the certificate. Leave it empty and BitVoice publishes it at </stir/<trunk>.pem> on its public address (Settings), which must then be reachable from the internet over HTTPS.
- **Attestation**: automatic gives **A** when an extension calls with one of this system's numbers (a number routed in, or a caller ID set on the trunk or the extension), **B** when an extension calls with any other number, and **C** for calls passed on from outside (a forwarded caller). Or fix it at A, B or C.
- Call history marks each signed call with **signed A**, **B** or **C**. A call is never stopped because it could not be signed; the reason is in the log.

Running BitVoice

Keeping it safe and running: administrators and sign-in, security, changes and updates, several servers, hosted organisations, codecs, Teams, importing, automation and the tools for finding faults.

CHAPTER 56

Administrators

In the web interface: `/users`

Administrators are the people who sign in to this web interface. Staff who only look after their own calls use the staff portal instead and need no account here ([The staff portal](#) (chapter 32)).

Roles

Role	Can
Administrator	change anything, play recordings, see every secret field's existence (never its value)
Viewer	see everything, change nothing; secrets are hidden

A change of role, or of password, signs the person out everywhere at once.

Two-step sign-in

Each person turns on two-step sign-in for themselves: **Set up** shows a key for an authenticator app; entering the code it shows turns it on. From then on, sign-in asks for a code as well as the password. Turning it off needs the password.

Turn on two-step sign-in for every administrator of a system reachable from the internet.

Single sign-on

With single sign-on (Settings), people sign in with the organisation's account (Microsoft Entra, Google, Okta, any OpenID Connect provider), and an account is made for them the first time. See [Single sign-on](#) (chapter 57).

API tokens

Programs (Terraform, scripts) use API tokens instead of a password: made on this page, shown once, and revoked at any time. Each token is an administrator or a viewer.

CHAPTER 57

Single sign-on

Single sign-on lets people sign in to BitVoice with their work account, through any OpenID Connect provider: Microsoft Entra ID, Google, Okta, Keycloak and others. Leaving the organisation then removes their access here too.

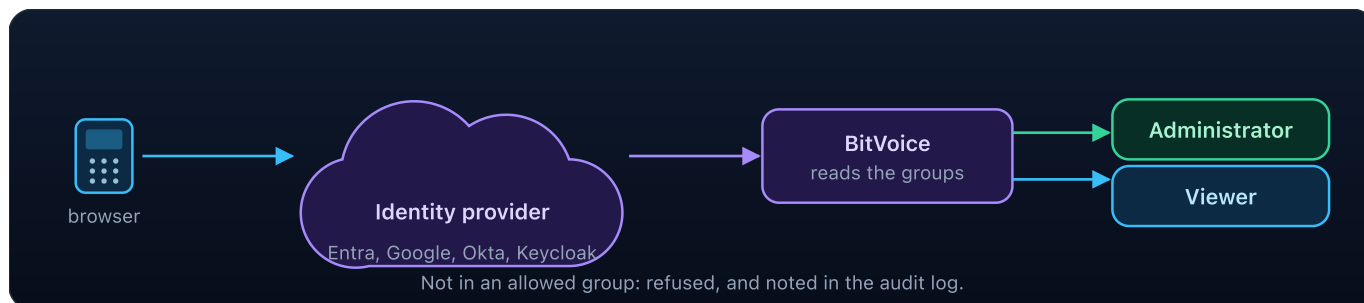


Figure 44. Single sign-on: the work account decides who gets in

Setting it up

In [Settings](#) (chapter 14), **Single sign-on**:

1. Register an application with the provider, with the **Redirect address** shown here.
2. **Fill in for a provider** sets the **Issuer** for the common ones; copy the **Client ID** and **Client secret** from the provider.
3. Name the provider's **Administrator groups** and **Viewer groups**. **Groups claim** is where the provider puts them (`groups` unless set).
4. Optionally change the **Button text** on the sign-in page.
5. Switch on **Use single sign-on**.

How it works

- The account here is made at the first sign-in; the groups set the role at every sign-in, so moving someone between groups at the provider changes their role here.
- Someone in neither kind of group is refused, and the refusal is in the audit log.
- With **Also allow passwords** off, passwords work only on this computer, so a provider outage never locks you out.

Two-step sign-in is then the provider's job: switch it on there.

CHAPTER 58

Security

In the web interface: `/security`

A phone system on the internet is attacked within hours: scanners try passwords on every SIP port, and a broken-into system runs up calls to premium numbers. BitVoice is built to stand up to that by default, in layers.

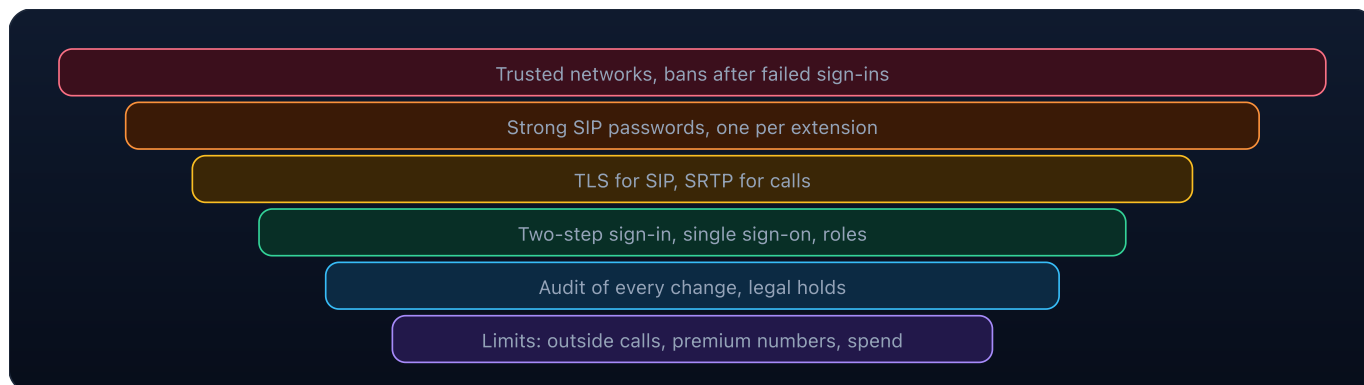


Figure 45. Layers of protection

Against password guessing

- Addresses that keep failing to sign in are banned for a while (Settings, Protection): by default ten failures in a minute ban the address for an hour. Banned addresses are listed on [Security](#) (chapter 58) and can be unbanned.
- **Trusted networks** are never banned: your office network.
- SIP passwords are generated, at least 12 characters, one per extension. A phone signing in as one extension can never register another.
- A scanner that is banned gets no answer at all, so it learns nothing.

Against fraud

- Extensions can call outside numbers only when allowed.
- Premium-rate numbers are refused unless a route names them.
- The **toll-fraud guard** (below) judges every call out of the country before it is placed.
- [Costs](#) (chapter 53) alerts on budgets and on runaway spend: many calls to one country in an hour.
- Each extension may have at most four calls at once (Settings, Protection).

The toll-fraud guard

Switch it on in Settings, **Toll-fraud guard**. Every call out of the country (by +, 00 or 011, and the places that share +1 with North America but are other countries, such as Jamaica, a classic toll-fraud destination) is then judged before it is placed:

Situation	What it does
A call abroad to a country this extension called in the last 90 days	lets it through
A first call to that country in 90 days	lets it through and reports it
A first call there outside Business hours , or placed from outside (a dial-in, a forward)	asks for the PIN asked for (with none chosen, refuses it)
More than Calls abroad in 10 minutes from one extension	refuses it
A first call abroad while someone is guessing passwords (three failures in a row from one address, or a ban, in the last hour)	refuses it
An emergency number	always lets it through

Countries always allowed (calling codes, such as 44) and **Exceptions** (an extension excused until a day, for a trip) take away "first call" and "outside hours"; they never excuse many calls in a short time or an attack, because a stolen password looks

approved too. If the call history cannot be read, the guard does not know whether a call is a first, and something not measured is never taken for safe: it asks for the PIN. Every judgement is listed on [Security](#) (chapter 58) with its reasons, and is on the call's record in [Call history](#) (chapter 46).

How BitVoice decides

BitVoice's judges (the toll-fraud guard, the trunk health judge, the call-quality diagnosis and the 911 caller location check) each follow a **rulebook**: facts that are measured, conclusions they support, the exceptions that overrule a conclusion, and the choices.

How BitVoice decides, on [Security](#) (chapter 58), shows each rulebook. The rules the judges keep:

- Every answer comes with its reasons: what held, what was argued against it and what overruled that.
- Something nobody measured is unknown, never "no". When an unknown could change the answer, the judge takes its cautious choice and says what it could not measure.
- "Cannot determine" is an answer: when the rules leave a situation undecided, the judge takes its cautious choice and says so.
- Each rulebook is checked in every combination of its facts before it is used (when BitVoice starts, and by its tests): never two choices at once, and every rule reachable. The screen shows how many situations each was checked in.

Encryption

- **TLS** encrypts signalling (port 5061), so passwords and numbers are not readable on the network.
- **SRTP** encrypts the sound of calls; the browser phone and apps always use it.
- The web interface has HTTPS on port 8443 with its own certificate, or one you install.

Who can do what

- Administrators change everything; viewers see everything but change nothing; staff use the portal for their own calls only.
- Turn on **two-step sign-in** for every administrator ([Administrators](#) (chapter 56)): a code from an authenticator app.
- [Single sign-on](#) (chapter 57) signs people in through your identity provider.
- The **audit log** records every administrative action and who did it, and cannot be edited.

CHAPTER 59

Changes, backups and import

In the web interface: [/changes](#)

Every saved change is a revision: numbered, with who made it, when, and a one-line summary. Nothing in the configuration is ever lost.



Figure 46. Every change is a revision you can compare and undo

Comparing and restoring

- **What changed** shows the difference between a revision and the one before, field by field.
- **Restore** shows what restoring would change and asks first. Restoring is itself a new revision, so it can be undone too.

Export and import

- **Export** downloads the whole configuration as one file (secrets left out unless you ask for them).
- **Import** previews a file's changes before applying them.
- **Import** (chapter 66) also reads FreePBX and Asterisk configurations, 3CX user lists, and the CSV exports of Cisco Unified CM, Avaya and Mitel.
- **Archive** (chapter 69) copies backups, recordings and voicemail to Amazon S3 or Azure.

Changes made at the same time

Two administrators editing the same thing at once cannot overwrite each other: the second save is refused with a message to reload, and nothing is lost.

CHAPTER 60

Updates

In the web interface: `/updates`

Updating BitVoice is one click, and safe: it checks the new version, installs it beside the old one, and goes back on its own if the new one does not stay up.

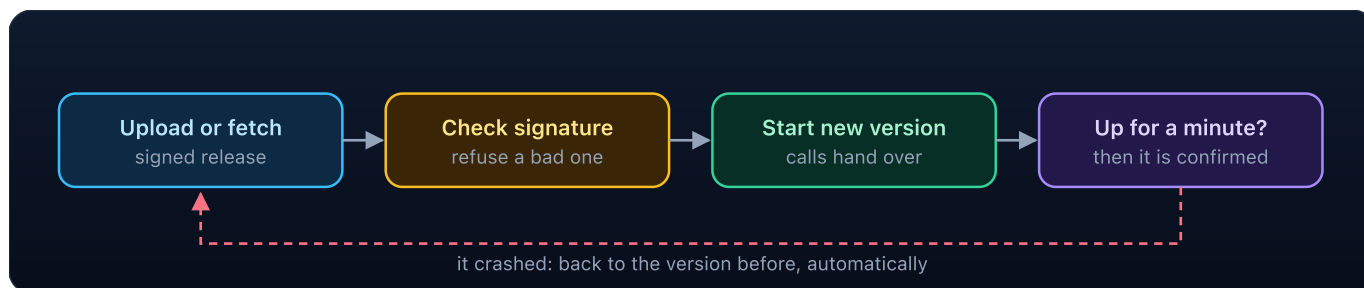


Figure 47. Updates: install, check, and fall back on their own

Installing an update

1. Download the new release and unpack it. Beside the program is its signature from BioDream: `bitvoice.sig`, or `bitvoice.exe.sig` on Windows.
2. On **Updates** (chapter 60), upload the program and its `.sig` file.
3. Press **Install**. By default BitVoice waits until no calls are in progress; you can also install at once.
4. BitVoice restarts on the new version. With several servers, calls in progress are handed to another server and carry on.

Going back

The version you left is kept: **Go back** returns to it in one click. A new version that fails to start twice, or does not stay up for a minute, is replaced by the previous one by itself.

Signatures

Every release from v0.1.3 carries BioDream's release key. Such a build installs only programs signed with that key: an unsigned file, one signed by anyone else, or one changed by a single byte is refused with **the signature does not match**. The page says whether this build checks (**Required**) or not.

You run	Updates accepts
a release, v0.1.3 or later	only programs signed by BioDream
v0.1.2 or earlier, or a build of your own	any program: with no key, a .sig file cannot be checked and is ignored

Install v0.1.3 or later by hand once (download, replace the program, start it): from then on the Updates page checks every update.

CHAPTER 61

Several servers as one phone system

In the web interface: `/servers`

Two or more BitVoice servers that share one PostgreSQL database work as one phone system:

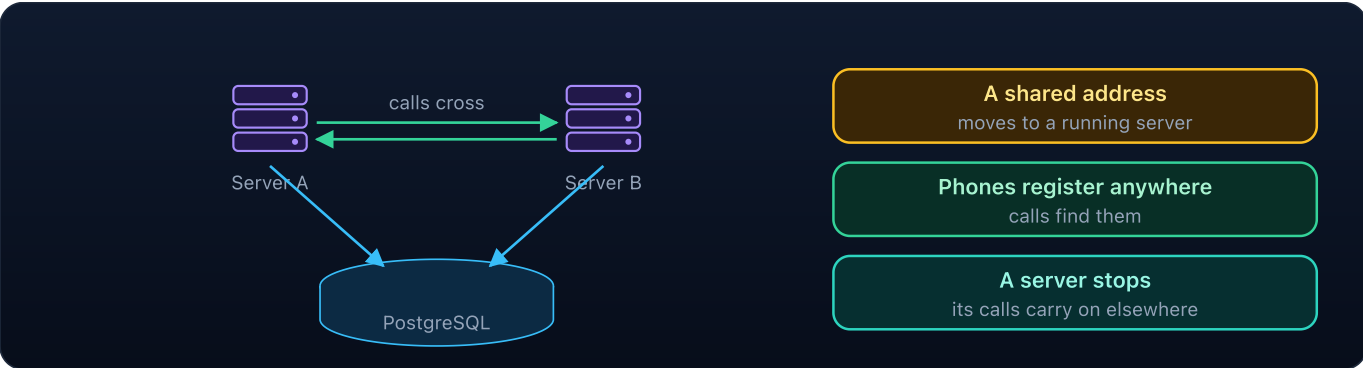


Figure 48. Several servers as one phone system

- A call reaches a phone **whichever server it is registered with**: ring groups, queues, transfers to an extension and calls from provider lines all ring phones on every server.
- **Settings** changed on one server apply on all within five seconds; users, contacts, call history, recordings' transcripts and texts are shared.
- **Busy lamps** (BLF) show phones on every server ringing and busy; **parking** slots are shared, so a call parked on one server is picked up from a phone on another; **paging** reaches phones on every server; **texts** reach desk phones wherever they are registered.
- A queue agent on a call on one server counts as busy on the others, and a provider line's channel limit counts calls on every server.
- **Live calls** on each server lists the calls on the others too (read-only: hang up or listen in on the server that has the call).
- The call record is written once, by the server that has the caller, and a call is recorded there too.

Setting it up

1. Move the first server's data into PostgreSQL: Settings, **Database**.

2. Start each other server with the same database: `bitvoice -database postgres:///...` (or the BITVOICE_DATABASE environment variable). It joins at once; **Servers** lists it.
3. On **Servers**, check where each server is reached (its address and SIP port on the network the servers share), and give each a name. Servers that are each reached from the internet at an address of their own (cloud servers with public IPs, no shared address) get **its own public address** too: providers and phones outside are told that one, so when a server takes over a call, the provider's sound follows it. Left empty, a server uses the public address in Settings.
4. Tell phones how to find the servers (below).
5. Provider lines (below).

How phones find the servers. Phones BitVoice sets up (Yealink, Polycom, Grandstream) are given another running server as their backup by themselves. For the others, Snom and phones set up by hand, use one of these on **Servers**:

- **A DNS name for every server:** a name such as pbx.example.com with SRV records (`_sip._udp`, `_sip._tcp`, `_sips._tcp`) listing each server. Phones BitVoice sets up are given the name, and Servers checks the records against the running servers, lists what is missing and gives the records to create, ready to copy.
- **A shared address:** an unused address on the servers' network, such as 192.168.1.50/24. One running server holds it at a time and tells the network at once (gratuitous ARP); if it stops, the next server takes the address within seconds, carries on its calls and answers its phones at the same address, without them registering again. It needs Linux and the NET_ADMIN and NET_RAW permissions (in a container: `--cap-add NET_ADMIN --cap-add NET_RAW`), and the servers on one network. Restart each server after setting it.

Provider lines. In each trunk, **Shared between servers** says how:

- **Each server uses it:** every server registers and places its own calls. For providers that allow several registrations on one account, or none.
- **One server at a time:** one server registers and carries the line; the others show it as **Standby, held by** that server and send their calls out through it. If it stops, the next server registers within seconds. For providers that allow one registration per account.

When a server stops

- **Calls carry on.** Each server keeps a note of its calls in the database; when one stops, a running server picks up each call within seconds, moves the sound to itself and tells both ends (a re-INVITE). There is a gap of a few seconds in the sound. A planned stop (an upgrade, a restart) hands calls over at once, with no gap.
- A call whose phone is registered with the server that takes over is carried on there without disturbing that phone: only the other party is moved.
- Its phones carry on through their backup server, the DNS name or the shared address.
- A server that only restarts (an upgrade, a reboot) still knows where its own phones are registered: it keeps them in its data folder (`registrations.json`), so calls reach them before they register again.
- Health and **Servers** show it as stopped; when it comes back it takes nothing back from the server that took over.

Between servers. The servers reach each other at their SIP port over TCP, or TLS when they have a certificate (each accepts only the certificate the other published, and the sound between them is then encrypted with SRTP), and send sound to each other's media ports (UDP, the range in Settings). Calls between servers are signed with a secret they share in the database, and only accepted from a running member at its own address. A firewall between them must let those ports through.

Try it first: the lab. BitVoice comes with a lab that runs two servers, their database and test phones on one computer with Docker, each on an address of its own: `deploy/lab/run.sh up` builds and starts it and sets the shared address; `run.sh call` calls between phones on different servers; `run.sh failover` kills the server carrying a call through the shared address and checks the call carries on (the sound comes back in about five seconds); `run.sh impaired` calls from a phone on a bad link (delay, jitter and 5% loss) and checks the call quality BitVoice reports; `run.sh down` removes it. The servers' pages are at `https://127.0.0.1:18501` and `:18502` (user admin).

Where the data is kept (SQLite or PostgreSQL)

In the web interface: `/settings`

BitVoice keeps its settings and their saved versions, users, contacts, call records, the audit log, texts, notes and transcripts in a database. Recordings and voicemail stay as sound files in the data folder.

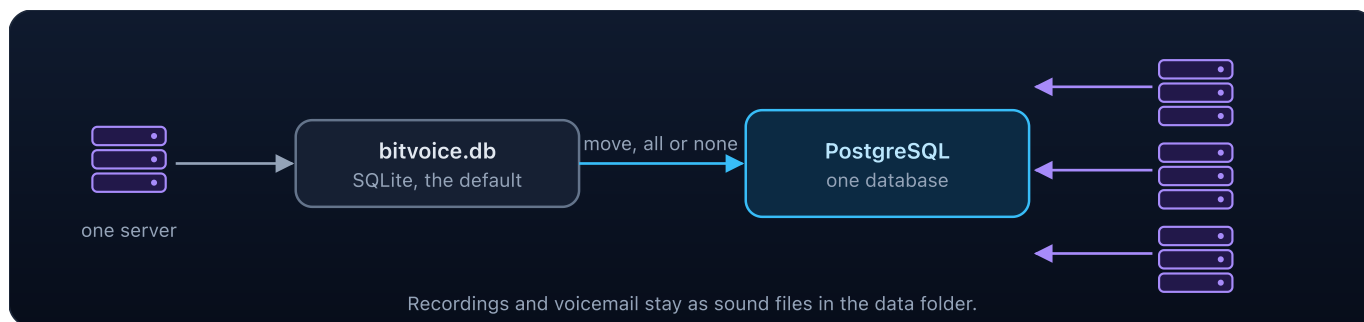


Figure 49. Where the data is: one file, or PostgreSQL

This server's own file (SQLite, the default): a single file, `bitvoice.db` in the data folder, with nothing to install. Back up the data folder and you have everything.

PostgreSQL: for several BitVoice servers sharing one set of data, or for reports in tools such as Power BI, Grafana or Metabase. In Settings, **Database, Move the data into PostgreSQL:** make an empty database and a user who owns it, enter its address (`postgres://user:password@host:5432/database`), press **Test**, then **Copy the data and restart on PostgreSQL**. Everything is copied in one go (all of it or none of it) and BitVoice restarts on it; the old file stays where it was. Moving back into the server's own file works the same way. Moves are refused while calls are in progress.

Servers sharing a PostgreSQL database see each other's settings changes within five seconds. Each still answers its own phones and calls; for a second server to take over a first one's phones, the phones need both addresses (or one address that moves).

The database can also be named when BitVoice starts, with `-database postgres://...` or the `BITVOICE_DATABASE` environment variable; it is then changed there, not in Settings.

Older installations: on the first start of this version, BitVoice copies its old JSON files into the database and moves them into `pre-database` in the data folder.

Hosting other organisations

In the web interface: `/tenants`

One BitVoice can host several organisations. Each one is a phone system of its own: its own extensions, numbers, trunks, routes, administrators, voicemail, recordings and call history. No organisation can see another's. They share only this server's SIP port, IAX2 port and media ports.

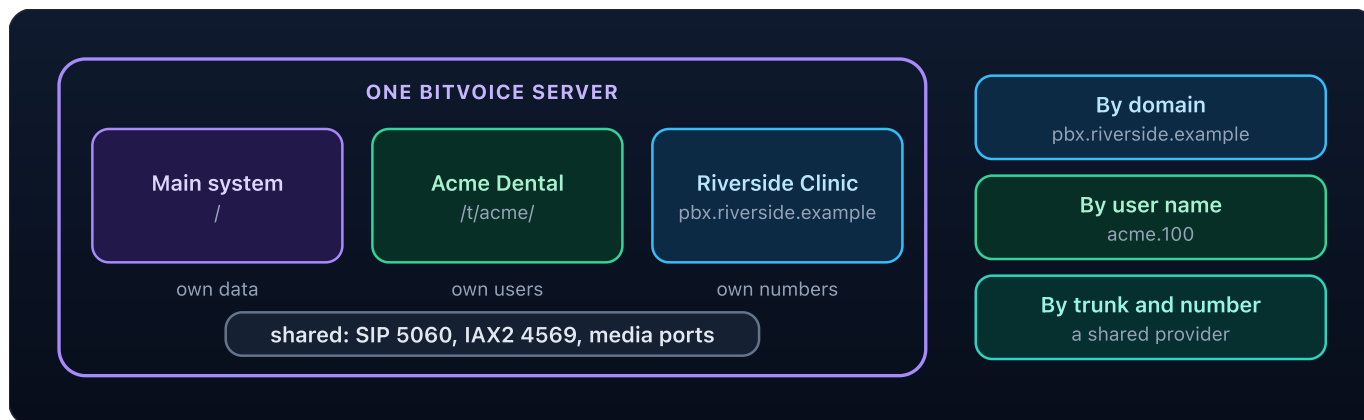


Figure 50. Hosted organisations: one server, separate phone systems

When to use it

- A provider, or an IT company, running phone systems for many small customers on one server.
- A group of offices or clinics that want separate systems, administered separately, on shared hardware.
- Several answering points sharing one system, each keeping its own call-takers and records.

The main system is an organisation too: the one this server was set up as. Its administrators manage the others.

Adding an organisation

1. Open [Organisations](#) (chapter 63) and press **Add organisation**.
2. Give its **name** and a **short name** (lowercase letters, digits and dashes). The short name is part of its web address, `/t/acme/`, and of its phones' sign-ins, `acme.100`. It cannot be changed later.
3. Optionally give its **own domain**, such as `pbx.acme.example`, pointed at this server. Phones register to that domain, and its administrators open it in a browser.
4. Set its **first administrator**: a username and a password of at least 12 characters, and its country.
5. Optionally set **limits**.

The organisation starts at once with an empty phone system, ready for its administrator to set up as described in [Getting started](#) (chapter 1).

How calls find the right organisation

A request	goes to
from a phone registering to <code>pbx.acme.example</code>	the organisation with that domain
from a phone signing in as <code>acme.100</code>	Acme, extension 100
from a provider, for a number	the organisation whose trunk it comes from and whose incoming route has that number
anything else	the main system

So two organisations can both have extension 100, and several can share one provider account with different numbers.

Where its data is

On SQLite, each organisation has its own folder under `tenants/` in the data folder, with its own database file. On PostgreSQL, each has its own schema in the same database (`tenant_acme`). Backups, archives and exports are per organisation.

Limits

Limit	What happens at it
Extensions	adding one more is refused with a message
Calls at once	new calls are refused (503) until one ends
Storage	new recordings and voicemail are not kept
AI requests a day	transcripts, summaries and the AI receptionist stop until midnight UTC

0 is no limit. Emergency calls are never refused: an organisation at its call limit, or suspended, can still call 911 and be called back.

Opening an organisation

Open signs you in to its web interface as the provider, without its own account; it shows in its audit log as the provider. Its administrators sign in at its own address with their own accounts. Each organisation has its own sign-in cookie, so being signed in to one never signs anyone in to another.

Organisations never see this server's own screens: updates, the database, the other servers, the certificate, the SIP trace (which shows every organisation's messages), the call generator or this list.

Suspending and removing

Suspended stops sign-ins and every call except emergency calls; nothing is deleted. **Remove** stops the organisation and moves its folder aside (`tenants/.removed-<name>-<time>`), so a mistake can be undone by hand.

CHAPTER 64

Codecs and sound

A codec is how sound is turned into data on a call. Phones, browsers and carriers each support some; BitVoice converts between any two of its codecs when two parties differ, so everyone can talk to everyone. When both use the same codec the sound passes through untouched.

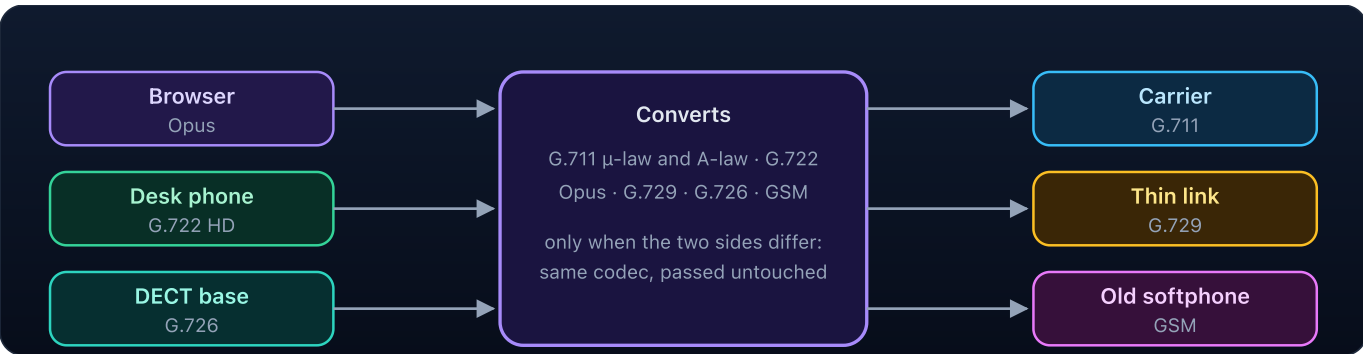


Figure 51. Codecs: each side speaks its own, BitVoice converts in between

The codecs

Codec	Sound	Bandwidth	Used by
G.722	HD (wideband)	64 kbit/s	most desk phones: the best choice between them
G.711 μ -law	telephone	64 kbit/s	North America and Japan; every carrier
G.711 A-law	telephone	64 kbit/s	the rest of the world
Opus	HD, copes with loss	24 kbit/s	browsers and the mobile apps
G.729	telephone, a little rougher	8 kbit/s	thin links, some carriers
G.726 (40 kbit/s)	telephone; carries modem tones best of the G.726 rates	40 kbit/s	gateways, some DECT systems
G.726 (32 kbit/s)	near telephone	32 kbit/s	DECT bases, older gateways
G.726 (24 kbit/s)	between 32 and 16	24 kbit/s	some DECT systems
G.726 (16 kbit/s)	rough	16 kbit/s	very thin links
AMR-WB (G.722.2)	HD	6.6 to 23.85 kbit/s	mobile networks' HD voice, VoLTE interconnects
AMR	telephone, mobile	4.75 to 12.2 kbit/s	mobile networks, some carriers and gateways
iLBC	telephone, copes with loss	13.3 or 15.2 kbit/s	older softphones and apps
Speex	telephone	15 kbit/s (2.15 to 24.6)	older softphones, links to other Asterisk systems
GSM	rougher	13 kbit/s	older softphones and gateways
G.723.1	rough; converted only with ffmpeg	5.3 or 6.3 kbit/s	very old video phones and gateways

Settings lists the codecs phones may use, best first; each trunk may have its own list. Analog adapter ports always use G.711, which faxes, modems and alarm panels need.

G.723.1 passes untouched between two parties that both use it. To let a G.723.1-only device talk to anything else, switch on **Convert G.723.1 with ffmpeg** in Settings and install ffmpeg (apt install ffmpeg, brew install ffmpeg): BitVoice then runs ffmpeg as a separate program for each direction that needs converting (heard at either rate, sent at 6.3 kbit/s, the rate ffmpeg's encoder makes), about 60 ms later than other codecs, and stops it a few seconds after the call. BitVoice carries no G.723.1 code of its own: the ITU's reference code has no licence that allows including it, and ffmpeg is installed and run on its own.

BitVoice's G.726 is bit-exact with the ITU-T's own reference implementation at all four rates (ffmpeg's differs at 24 and 40 kbit/s, so equipment built on ffmpeg may sound slightly different there). G.726 comes in two packings: RFC 3551's (G726-32) and the older AAL2 order. iLBC is bit-exact with the RFC 3951 reference (its enhancer and concealment included); BitVoice offers 20 ms frames and follows a phone that sends 30 ms ones. Speex is narrowband (8 kHz, RFC 5574) and bit-exact with libspeex 1.2.1; BitVoice sends Speex's mode 5 (15 kbit/s) or the mode the other side's mode= asks for, and conceals lost packets with Speex's own concealment. G.723.1 is passed through untouched between two parties that both use it but never converted: it is offered only to parties whose caller offered it, and a call between a G.723.1-only phone and one without it does not connect. (Its ITU reference code carries no licence that lets BitVoice include a converter.) AMR and AMR-WB follow RFC 4867: BitVoice offers octet-aligned framing, answers in whichever framing the other side asked for, and sends the highest mode the other side's mode-set allows (12.2 and 23.85 kbit/s when it gives none). Most equipment uses the first; choose **G.726 (AAL2 order)** only for equipment that sounds garbled with it.

What converting costs

Converting takes processor time, only on calls where the two sides differ. Per call, about 0.3 ms of one core every 20 ms for Opus and 0.12 ms for G.729 (some 60 and 170 converted calls per core); G.726 and GSM cost less than G.729; G.711 and G.722 almost nothing. G.729 is sent without Annex B (no silence frames).

Sound clean-up

Sound clean-up (Settings) turns down steady background noise and evens out levels, for voicemail and, if you like, during calls.

CHAPTER 65

Microsoft Teams (Direct Routing)

In the web interface: `/trunks`

BitVoice can be the session border controller (SBC) for a Microsoft Teams tenant, so Teams users call BitVoice extensions and your phone lines, and BitVoice rings Teams users.

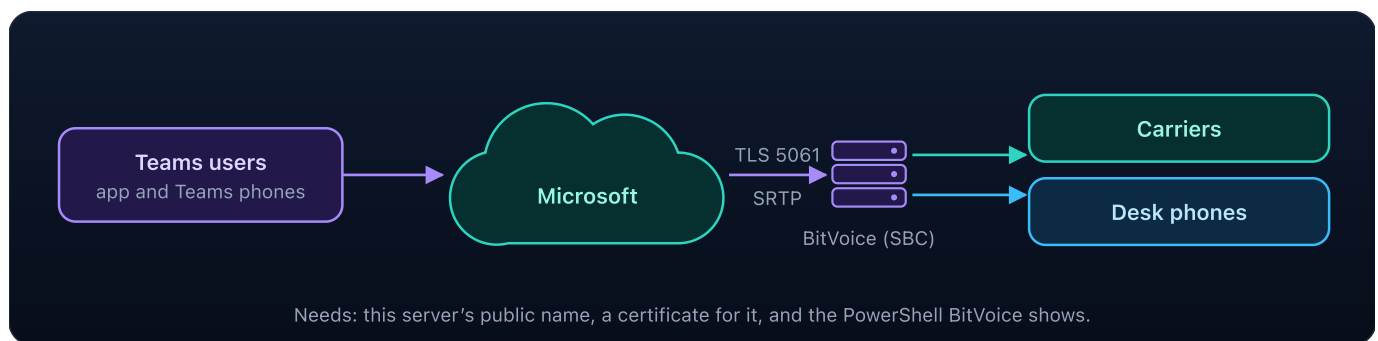


Figure 52. Microsoft Teams Direct Routing: BitVoice is the SBC

What you need: a public name for this server in a domain your Microsoft 365 tenant has verified (sbc.example.com), port 5061 open both ways to Microsoft's networks 52.112.0.0/14 and 52.120.0.0/14, and a certificate for that name from a public authority. Teams users need a Phone System licence.

Steps

1. Settings, **Certificate:** install the certificate and its private key (PEM). It is used at once, for this interface and for SIP over TLS, and BitVoice presents it to Microsoft.

2. Trunks, **Add**, start from **Microsoft Teams**: enter the server's public name. The drawer says whether the certificate covers it.
3. Copy the **PowerShell** the drawer shows and run it as a Teams administrator: it adds the SBC (New-CsOnlinePSTNGateway), a voice route and policy, and a dial plan rule so short extension numbers reach BitVoice as dialled; then give each Teams user the policy and their number.
4. To ring Teams users from BitVoice, add an outbound route for their numbers through the Teams trunk.

BitVoice sends Microsoft OPTIONS every minute with the SBC name in Contact, as Microsoft requires, and tries sip, sip2 and sip3.pstnhub.microsoft.com in turn. Media is always encrypted (SRTP). Health warns when the certificate does not cover the name, is self-signed or ends soon, and when Microsoft stops answering.

This follows Microsoft's published SBC requirements and is tested against a simulated Microsoft proxy; BitVoice is not on Microsoft's list of certified SBCs.

CHAPTER 66

Import

In the web interface: `/import`

Brings extensions, ring groups, queues and trunks over from FreePBX, Asterisk or 3CX; and people, hunt and ring groups, SIP trunks, incoming numbers and dial patterns from Cisco, Avaya and Mitel. Upload the exported files, several at once (a .zip or .tar of them works too); BitVoice shows exactly what would change before anything is applied, and the import becomes a revision you can undo on [Changes](#) (chapter 59).

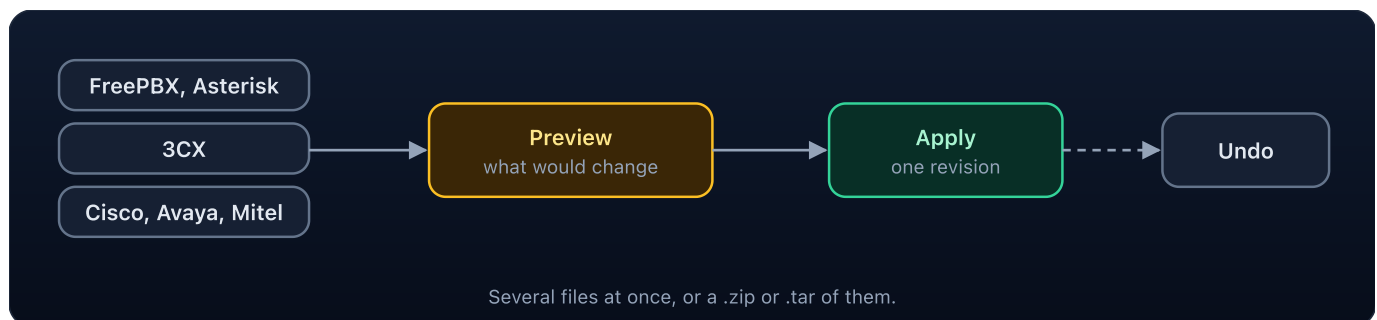


Figure 53. Import: another system's export, previewed before anything changes

Moving from FreePBX

The best file is a FreePBX backup: it holds everything, SIP passwords included.

1. In FreePBX: **Admin → Backup & Restore → Add Backup**. Tick at least **Core**, **Ring Groups**, **Queues** and **Voicemail** (all modules is fine), save, and **Run**.
2. Download the backup from its row (the file is named like `20261001-100000-1790000000-17.0.19-1234567.tar.gz`).
3. Install BitVoice on the new server and finish the setup with one extension that FreePBX does not use (say 199) and no phone line: the import brings the lines.
4. On [Import](#) (chapter 66), drop the .tar.gz as it is. Read the preview: the warnings list everything that cannot come over.
5. Choose **Add**. It is one revision on [Changes](#) (chapter 59): undo it there if anything looks wrong, and import again.
6. Move the phones: point each at BitVoice (or provision them from [Phones](#) (chapter 4)), with the same extension number and SIP password they had.
7. Test calls in and out, then move the carrier's registration (or the number porting) last.

Tried on a real system: a FreePBX 17.0.33 (the official Debian 12 install) with four extensions and voicemail, a PJSIP trunk that registers, two outbound routes (one with a dialling prefix), a ring group, a queue with penalties and three DIDs, backed up by

Backup & Restore with every module ticked. The backup is 70 MB, nearly all of it sound files and call history: BitVoice reads it as it comes and keeps only the configuration, so a backup of up to 512 MB uploads. Everything above came over: the passwords, the voicemail PINs and email, the trunk's registration and codecs, the routes with their prefixes, the ring group's voicemail when unanswered, the queue's penalties and the DIDs to the group, the queue and the extension.

What comes over:

- **Extensions:** the number, name, outbound caller ID, ring time and SIP password (BitVoice makes a new one when FreePBX's is shorter than 12 characters: re-enter it in that phone). Voicemail with its PIN and email address (and whether messages are sent attached).
- **Trunks:** PJSIP and chan_sip carrier accounts: server, port, username, password, registration, from domain, codecs, caller ID, channel limit. Switched-off trunks stay off. DAHDI trunks become the Telephony cards trunk.
- **Outbound routes,** in FreePBX's order: each pattern's prefix becomes digits dropped, its prepend digits added. A route whose patterns drop or add different digits becomes one BitVoice route for each.
- **Ring groups:** members, ring time, ring-all or one-after-another (hunt, memory hunt, first available), and where unanswered calls go.
- **Queues:** members with their penalties, strategy (ring all, least recent, fewest calls, round robin, linear, random), ring time, wrap-up time and longest wait.
- **Incoming routes:** each DID to its extension, ring group, queue, voicemail box, or hang-up.

What does not, and is said in the preview: IVRs, time conditions, announcements, follow-me, custom trunks and contexts, caller-ID matching on routes, and outside numbers in ring groups. Build those on [Menus](#) (chapter 20), [Schedules](#) (chapter 25) and the extension's **Follow me**. Older backups (FreePBX 13 and before) and a plain `mysqldump asterisk` of the FreePBX database are read the same way.

Other systems

Where each export is:

- **FreePBX (extensions only):** Admin → Bulk Handler → Export → Extensions (CSV).
- **Asterisk:** pjsip.conf or sip.conf.
- **3CX:** Users → Export (CSV).
- **Cisco Unified CM:** Bulk Administration → Phones → Export Phones, all details; or Users → Export Users. The phones' first line becomes the extension, with the line's alerting name or label as its name.
- **Avaya IP Office:** Manager → File → Import/Export → Export, Users.
- **Avaya Communication Manager:** `list station` saved as CSV; virtual stations and VDNs are left out.
- **Mitel MiVoice Business:** System Data Management → User and Device Configuration → Export.

Groups and routes. Upload them with the people file, so each group's members are found by number or by name:

- **Cisco Unified CM:** Bulk Administration → Import/Export → Export, with Line Group, Hunt List, Hunt Pilot, SIP Trunk and Route Pattern ticked; upload the .tar as it comes. Each hunt pilot becomes a ring group with the members of its hunt list's line groups, Top Down and Circular as one after another, Broadcast as everyone at once, the RNA reversion timeout as the ring time. Route patterns become outbound routes: `9.[2-9]XXXXXX` with PreDot is pattern `9[2-9]XXXXXX` dropping the 9; `!` becomes a final dot. Patterns using `@` (the national plan) are listed for writing by hand.
- **Avaya IP Office:** export Hunt Group as well as Users (both are CSV with no header row). Group mode is everyone at once; Hunt, Rotary and Longest Waiting ring one after another.
- **Avaya Communication Manager:** `list hunt-group` saved as CSV, with the members in a Members column or a separate members file.
- **Mitel MiVoice Business:** Ring Groups and Ring Group Assignment (Cascade is one after another, Ring All everyone at once), and Hunt Groups.

Trunks from these exports come in switched off and without passwords, as the exports do not carry them. Routes go out on the imported trunk of the same name; a route whose trunk is not here is left out and said. Rotating and longest-idle ringing is closer to a queue than a ring group: the preview says so, so you can make it one.

These exports (all but a FreePBX backup) hold no SIP passwords, so each extension gets a new one; voicemail PINs, names and email addresses come over when the export has them. Cisco, Avaya H.323/DCP and Mitel phones run their maker's own software and cannot be set up by BitVoice: the import says how many were listed. Cisco phones can be re-flashed with Cisco's multiplatform (SIP) firmware; Avaya J100 and Mitel 6800/6900 phones can register to BitVoice by hand in SIP mode.

CHAPTER 67

Configuration as code

Everything the screens do goes through BitVoice's REST API, so anything can be scripted, kept in Git, or managed with Terraform. The API is described for tools at `/api/openapi.json` (/api/openapi.json).

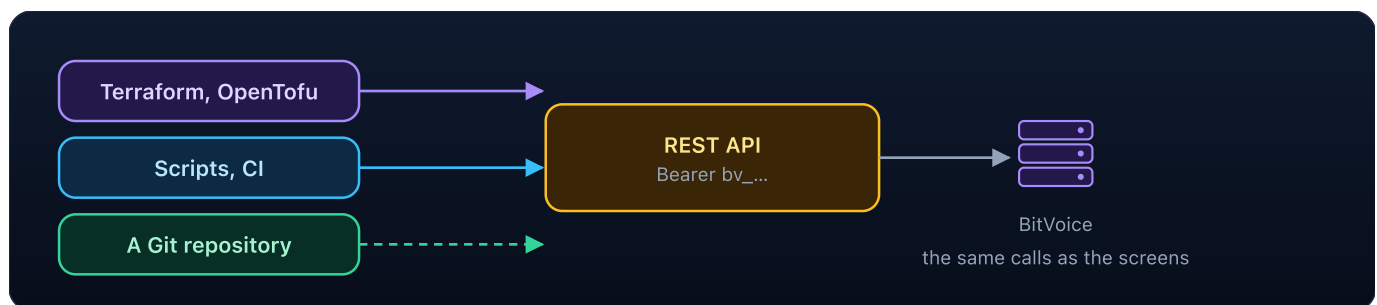


Figure 54. Configuration as code: the API, Terraform and Git

API tokens

Made on [Administrators](#) (chapter 56): one per program, read only or able to change everything. Send it as `Authorization: Bearer bv_...`. It is shown once; revoke it when no longer needed. A token's changes show in Changes and the audit log under its name.

```
curl -H "Authorization: Bearer bv_..." https://pbx.example.com/api/extensions
```

Terraform or OpenTofu

The repository's `deploy/terraform` shows extensions, ring groups and routes managed with the generic REST provider: `plan` shows what would change, `apply` makes the change. The provider checks BitVoice's certificate; while BitVoice still uses its own self-signed one, either get one from Let's Encrypt (**Settings, Certificates**) or set `bitvoice_insecure = true` on a network you trust (on Linux, `SSL_CERT_FILE` pointing at BitVoice's certificate works too). Resources are matched by the id BitVoice gives them, so a second plan shows no changes until the file or the screens change something.

Configuration from Git

In [Settings](#) (chapter 14), **Configuration from Git**, BitVoice reads the configuration from a file in a repository every few minutes and either shows what differs or makes the changes. Start the file with **Export for Git**, which leaves out passwords; BitVoice keeps its own, so secrets never go into the repository.

With BitVoice making the changes, a change made in the screens is undone at the next read if the file says otherwise: choose one place to make changes.

Asterisk compatibility (AMI and ARI)

Tools written for Asterisk can control BitVoice: CRMs, wallboards, click-to-dial buttons, call recorders' controllers. Switch it on in [Settings](#) (chapter 14), **Asterisk compatibility**. Tools see calls as channels named like `PJSIP/100-...`, as on Asterisk.



Figure 55. Asterisk compatibility: tools written for Asterisk drive BitVoice

AMI

The manager interface on TCP port 5038 (**AMI listens on:** this computer only by default). Supported:

Kind	Names
Actions	Login (plain or MD5 challenge), Ping, Events, Filter, ListCommands, CoreShowChannels, Status, CoreStatus, CoreSettings, PJSIPShowEndpoints, ExtensionState, ExtensionStateList, DeviceStateList, Getvar, Originate, Redirect, BlindTransfer, Hangup, Logoff
Queues	QueueStatus, QueueSummary, QueuePause, QueueAdd and QueueRemove (they sign an agent in to, or out of, all of their queues), Agents
Other	ParkedCalls; ConfbridgeListRooms, ConfbridgeList, ConfbridgeKick, ConfbridgeMute, ConfbridgeUnmute, ConfbridgeLock, ConfbridgeUnlock; MailboxCount, MailboxStatus; PauseMonitor, UnpauseMonitor; UserEvent; Reload (nothing to reload: changes apply at once)
Events	Newchannel, Newstate, Hangup, DialBegin, DialEnd, Hold, Unhold, PeerStatus, ContactStatus, MessageWaiting, ConfbridgeJoin, ConfbridgeLeave, ConfbridgeEnd, UserEvent

Filter keeps only the events a regular expression matches (`Event: Newchannel`), or drops them with `!` first, as on Asterisk. There is no Asterisk command line: **Command** answers with an error.

ARI

At `/ari/`, with basic authentication or `?api_key=user:secret` :

Request	What it does
<code>GET /ari/channels</code>	the calls in progress
<code>POST /ari/channels?endpoint=PJSIP/100&extension=5551234</code>	ring 100, then call the number
<code>DELETE /ari/channels/{id}</code>	hang up
<code>POST /ari/channels/{id}/redirect?endpoint=PJSIP/102</code>	send the call to 102
<code>/ari/endpoints</code> , <code>/ari/bridges</code> , <code>/ari/asterisk/info</code>	phones, bridges, version
<code>/ari/events</code>	a WebSocket of events
<code>POST /ari/channels/externalMedia?external_host=host:port&encapsulation=rtp&format=ulaw</code>	a channel sending sound to a program by RTP (ulaw or alaw over UDP); its local port is in the channel's <code>UNICAST RTP_LOCAL_PORT</code>
<code>POST /ari/channels/externalMedia?external_host=host:port&encapsulation=audiosocket&transport=tcp&format=slin&data=<uuid></code>	the same by AudioSocket
<code>POST /ari/bridges/{call}/addChannel?channel={id}</code>	the external media channel joins a call: it hears both parties and both hear it; with <code>role=listener</code> nobody hears it (for live transcription)
<code>POST /ari/bridges/{call}/removeChannel?channel={id}</code>	it leaves the call; the two parties talk directly again

An external media channel not added to a call within a minute is closed.

ARI applications

An ARI application (Asterisk's `Stasis()`) runs calls from your own program. Add it under **ARI applications** in Settings, Asterisk compatibility: a **name**, an optional **number** extensions dial, and a **fallback**. Then send calls to it like any destination: an incoming route, a menu key, a schedule.

Your program connects to `/ari/events?app=NAME` (a WebSocket, with the login in `?api_key=user:secret`). Each call reaching the application is answered and arrives as **StasisStart**, with the channel; the program then runs it:

Request	What it does
<code>POST /ari/channels/{id}/play?media=sound:goodbye,digits:42</code>	plays prompts from Sounds (<code>sound:</code>), <code>digits:</code> , <code>number:</code> , <code>tone:beep</code> or a stored recording (<code>recording:</code>); PlaybackStarted and PlaybackFinished follow. <code>DELETE /ari/playbacks/{id}</code> stops it.
<code>POST /ari/channels/{id}/record?name=msg1&maxDurationSeconds=60&maxSilenceSeconds=5&beep=true&terminateOn=%23</code>	records (wav) into the stored recordings; RecordingStarted and RecordingFinished . <code>POST /ari/recordings/live/{name}/stop</code> ends it, <code>DELETE</code> discards it.
<code>GET /ari/recordings/stored , .../{name} , .../{name}/file , DELETE .../{name}</code>	the stored recordings
<code>POST /ari/channels/{id}/dtmf?dtmf=123</code>	sends keys to the caller
<code>GET / POST /ari/channels/{id}/variable?variable=X&value=Y</code>	the channel's variables
<code>POST /ari/bridges?bridgeId=room1 , .../addChannel?channel=a,b , .../removeChannel , .../play , DELETE</code>	mixing bridges: everyone in one hears everyone else
<code>POST /ari/channels/{id}/continue?extension=101</code>	lets go of the call: to that extension, ring group or queue, else the fallback
<code>DELETE /ari/channels/{id}</code>	hangs up
<code>POST /ari/channels?endpoint=PJSIP/101&app=NAME</code>	rings 101; answered, it enters the application
<code>GET /ari/applications</code>	the applications, whether a program is connected, and their channels

Keys pressed arrive as **ChannelDtmfReceived**; a caller hanging up as **StasisEnd**. When no program is connected the call goes to the fallback; when the program's last connection closes, its calls go there too after five seconds (time to reconnect). For call logic without a program of your own, use [Scripts](#) (chapter 26) that run the whole call; to hand the audio to a program, [Voice bots](#) (chapter 23).

Logins

Each login has a **Username** and **Secret**. Logins are **read only** unless marked **Can change calls**. Keep AMI on this computer unless the tool runs elsewhere: AMI is plain, unencrypted TCP.

CHAPTER 69

Archive

In the web interface: `/archive`

The archive keeps copies of recordings, voicemail and the configuration somewhere else: an Amazon S3 bucket (or any S3-compatible store) or an Azure Blob Storage container. If this server's disk is lost, the archive still has them.

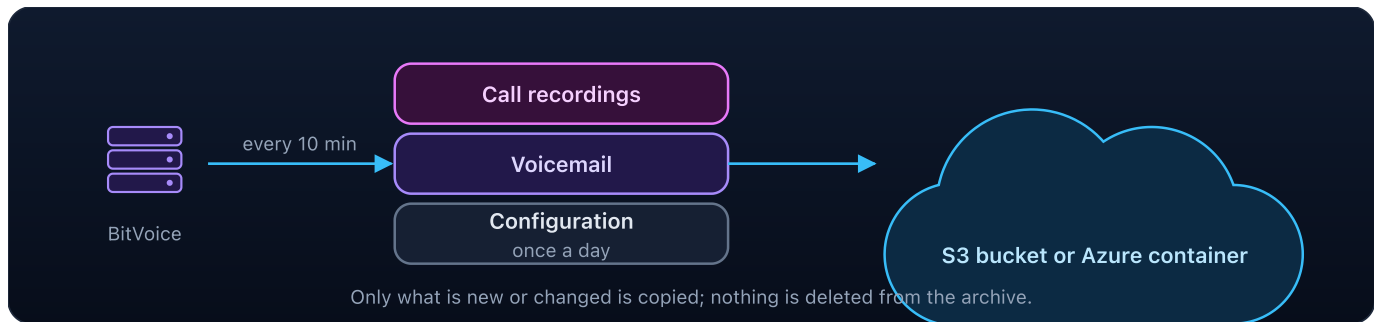


Figure 56. The archive: copies kept somewhere else

Setting it up

1. On [Archive](#) (chapter 69), choose **Where to archive**: Amazon S3 or Azure.
2. For S3: the **Bucket**, its **Region**, an optional **Folder in the bucket** and **Endpoint** (for S3-compatible stores), and an **Access key ID** and **Secret access key**. **Copy the policy** gives the only permission BitVoice needs, to attach to the user or role.
3. For Azure: a **Container SAS URL** with write permission, and an optional **Folder in the container**.
4. Choose what to copy besides the configuration: **Call recordings**, **Voicemail**, or both.

What happens

Every ten minutes BitVoice copies new recordings and messages that are not in the archive yet, and once a day a dated copy of the configuration (`config/2026-10-13.json`). The page shows the last run, the last success and any error.

Keys can be references to a key vault instead of the keys themselves (Settings, Key vaults), so they are never stored in BitVoice.

Deleting something here (by retention or by hand) does not delete the archive's copy: set the bucket's or container's own lifecycle rules for how long the archive keeps things.

CHAPTER 70

Legal holds

In the web interface: `/holds`

A legal hold keeps call records, recordings and voicemail that a matter may need, and stops anyone deleting them, until it is released. Use it when a lawyer asks for records to be preserved.

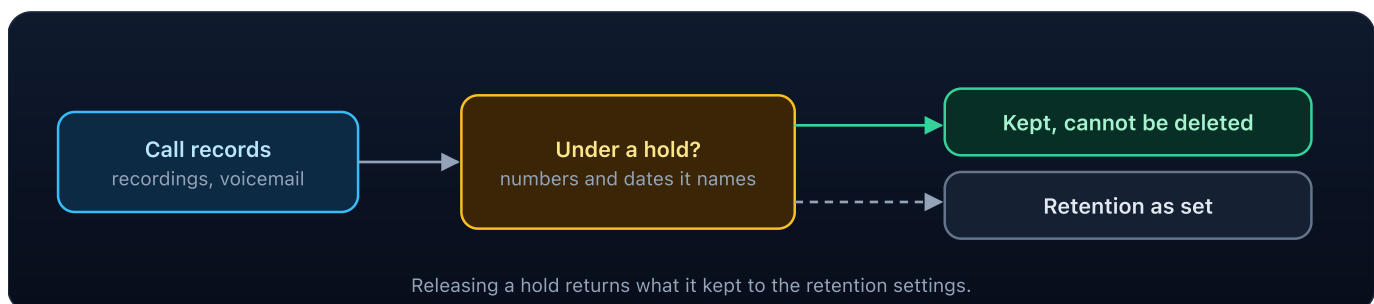


Figure 57. A legal hold: kept, whatever the retention says

Adding a hold

On [Legal holds](#) (chapter 70), press **Add**:

Field	What it is
Name	what the matter is called
Reason	why, for the record
Numbers	the extensions and outside numbers it covers (101 , +15551234567), or Everything
From, To	the dates it covers; empty for no limit

While a hold is on

- Retention leaves what it covers alone, whatever its age.
- Nobody can delete a held call, recording or voicemail, administrators included.
- Held calls show a lock in [Call history](#) (chapter 46).

Releasing

Release makes what the hold kept subject to the retention settings again: things older than the retention are deleted at the next clean-up. Released holds stay listed, with when and by whom, and adding and releasing holds is in the audit log.

CHAPTER 71

Metrics: Prometheus, OpenTelemetry, Grafana, SNMP and call records

In the web interface: [/settings](#)

Prometheus scrapes [/metrics](#) on the web address. Make an API token under Administrators (a viewer token is enough) and give it to Prometheus as its bearer token:

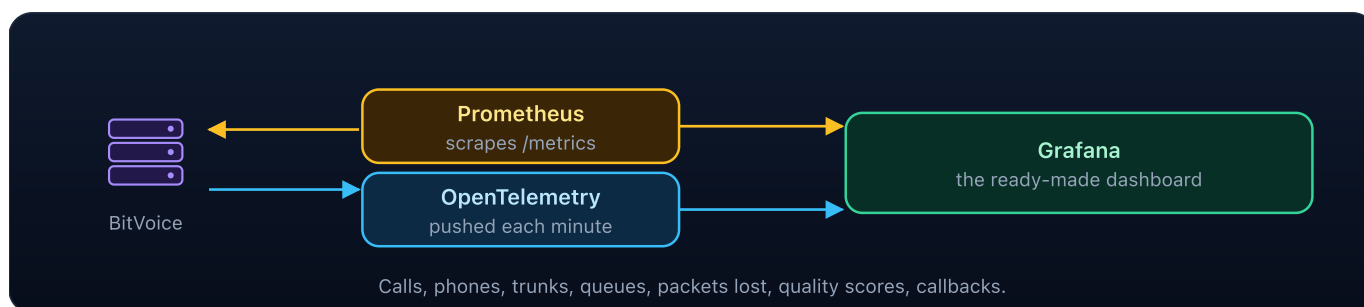


Figure 58. Metrics: Prometheus pulls, OpenTelemetry is pushed

```
scrape_configs:
- job_name: bitvoice
  scheme: https
  authorization: { credentials: bv_... }
  static_configs: [{ targets: ['pbx.example.com'] }]
```

It reads calls in progress (by state), calls ended (by direction and outcome), talk time, phones registered, each trunk's state and round trip, callers waiting and agents in each queue, sound packets received and lost, a histogram of call quality scores (MOS), and callbacks booked. Counters start again from zero when BitVoice restarts, as Prometheus expects.

OpenTelemetry: in Settings, Metrics, enter a collector's OTLP/HTTP address (and a header such as Authorization, with its value) to have the same measurements pushed every minute (or as often as you set, from 15 seconds). **Push now** checks the saved settings.

Grafana dashboard downloads a ready-made dashboard: import it in Grafana and pick your Prometheus data source.

SNMP

For monitoring systems that speak SNMP (Zabbix, PRTG, LibreNMS, Nagios, SolarWinds), switch on the **SNMP agent** in Settings. It answers SNMPv2c GET, GETNEXT and GETBULK, read only, on UDP port 161 (or another you choose: port 161 needs the right to open low ports on Linux), and only from the networks you list. Choose a **community** that is not public or private: it is a password that crosses the network in clear.

It serves the standard system group (sysDescr, sysUpTime, sysContact, sysName, sysLocation) and BitVoice's own values under the **base OID**: calls in progress and ringing, phones registered, trunks up and set up, callers waiting, calls ended, talk seconds, lost packets, average call quality, a trunk table (name, up or down, latency) and a queue table (waiting, agents, answered, abandoned). Download **BITVOICE-MIB** beside the settings for your monitoring to load. BioDream LLC has no enterprise number yet, so the default base OID is in Net-SNMP's experimental space; put one under your own enterprise number to keep it stable.

Traps (SNMPv2) go to each receiver you list when a trunk goes down or comes back, an emergency call is made, or a server of a cluster stops answering. SNMPv3 is not supported yet.

Call records elsewhere

Beside Call history, each record can go, as its call ends, to:

- **Syslog** (RFC 5424, as JSON) over UDP, TCP or TLS, for a log server or a SIEM.
- **A CSV file a day**, with Asterisk's Master.csv columns, so the reports and billing tools made for Asterisk read it. Download the files at the foot of [Call history](#) (chapter 46). Cells that a spreadsheet would run as a formula are written safely.
- **A PostgreSQL table** in another database, made the first time with Asterisk's `cdr` columns (calldate, clid, src, dst, duration, billsec, disposition, uniqueid...) plus BitVoice's direction, trunk, cost and MOS. Each call is written once.

Send a test record tries the syslog server and the database with the settings saved.

CHAPTER 72

Call generator

In the web interface: `/loadtest`

The call generator places real calls with real audio into BitVoice from this computer: to see how many calls a server carries and how well, or to check after a change that calls still go where they should.

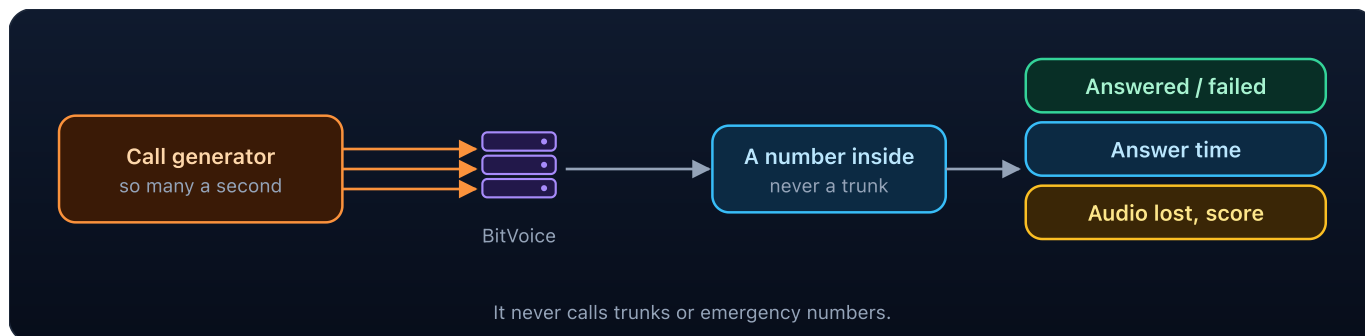


Figure 59. The call generator: real calls, real audio, measured

Running a test

Setting	What it is
Number to call	a number inside BitVoice: an extension, a group, a queue, a menu
Calls come from	the caller number the calls show
Answer as	an extension the generator answers for itself; or nobody, when the number answers by itself (a menu, the echo test)
Calls	how many in all
At once	the most in progress together
New per second	how fast they start
Each lasts (s)	how long each talks

Reading the results

Result	What it says
Answered, Failed	and Why calls failed , by reason
Answer time, median and 95%	how long callers waited
Audio lost	the share of sound packets that never arrived
Audio score	an estimate of the sound (MOS): 4 and over is good, under 3.6 is poor
Calls in progress over time	a chart of the load

It only calls numbers inside BitVoice, never trunks or emergency numbers, so a test can never cost money or reach 911. It is for this server's administrators only.

SIP trace

In the web interface: `/trace`

The SIP trace shows the messages BitVoice sends and receives, live: the first place to look when a phone will not register or a call fails, and what a provider's support desk will ask for.

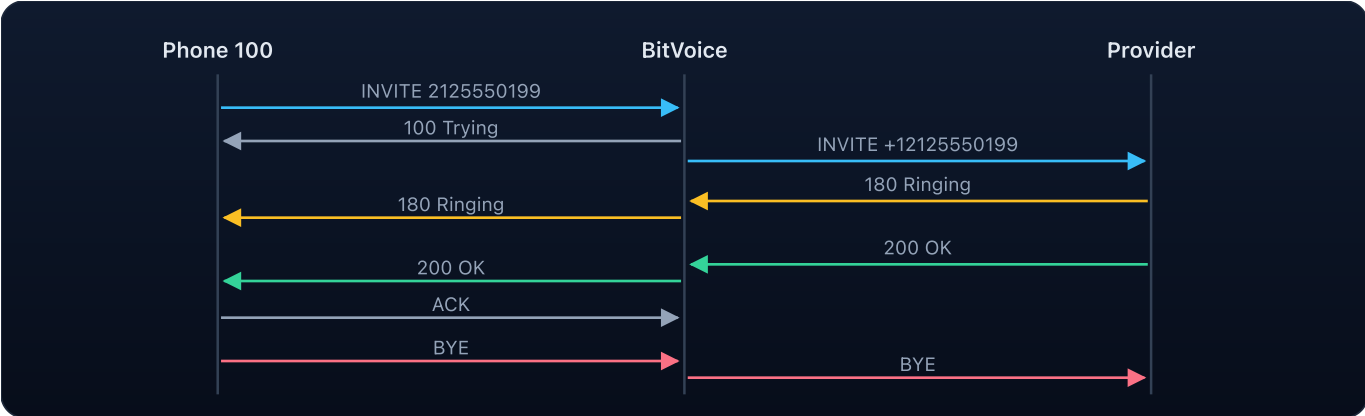


Figure 60. A SIP trace: the messages of one call, in order

Finding a message

BitVoice keeps the last 2,000 messages in memory (nothing is written to disk, and they are gone after a restart). Type part of a number, an address or a Call-ID in the filter. Press **Refresh** for the newest.

Choose a message to see it in full and, beside it, the **call flow**: every message of the same call in order, between the phone, BitVoice and the provider, as in the figure.

Reading a trace

You see	It usually means
REGISTER answered 401, then nothing	the phone does not have the password, or has the wrong one
REGISTER answered 403	the address is banned, or the phone signs in as another extension
INVITE answered 404	no route matches the number dialled (Call routing (chapter 8))
INVITE answered 403 by the provider	the provider refuses the caller ID or the number
486	the other side is busy
488	the two sides have no codec in common (Codecs (chapter 64))
200 OK, then BYE after about 30 seconds	the ACK never arrived: a router in between rewrites or drops messages (switch off its "SIP ALG")
A call connects but nobody hears anything	the sound is blocked: check the public address in Settings, and the media ports in the firewall

A trace shows phone numbers and sign-in names. Only people signed in to this server see it; hosted organisations never do, since it would show every organisation's messages.

Public safety: the lab track

The 911 answering-point profile, the radio dispatch console and NG911 Core Services. They are a lab and design-partner track: built and tested against recorded captures, sample data and stand-ins, but not certified for live 911 or radio dispatch and not yet tried with a live ESInet, P25 system or CAD. They are off the menu until switched on in Settings. Emergency calls from your own phones (Getting started) are not part of this track: they are part of every phone system.

911 calls (the 911 profile)

In the web interface: `/psap`

Lab and design-partner track. Not certified for live 911 or radio dispatch, and not yet tried with a live ESInet, P25 system or CAD. Use it in a lab or in shadow mode beside your current equipment. Switch on **Public-safety profiles (lab)** in [Settings](#) (chapter 14) to show it in the menu.

BitVoice answers 911 calls as a PSAP's call-handling system: calls from the emergency services network (the ESInet, NENA i3) arrive with the caller's location and the network's data, the call-taker sees them at once, and CAD is told when the call is answered.

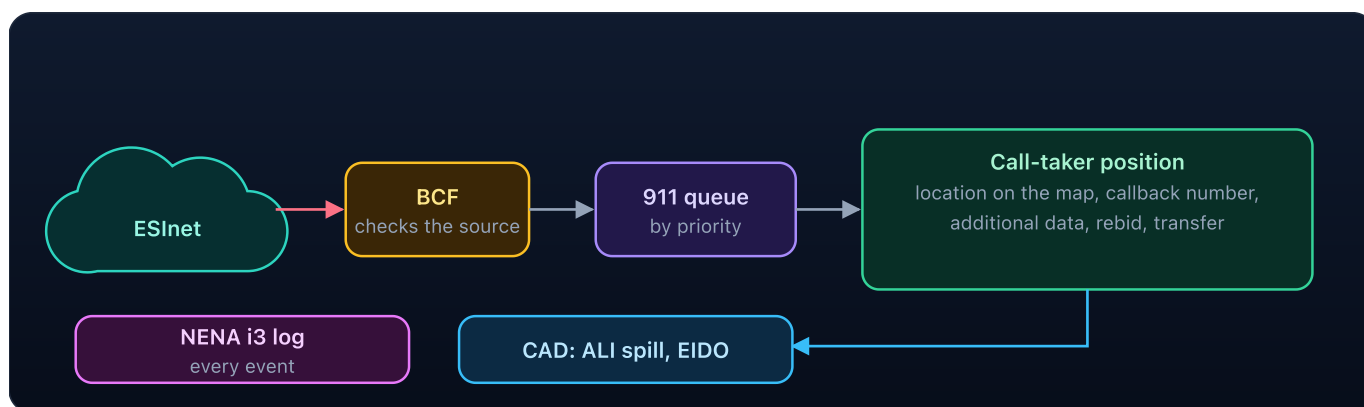


Figure 61. The 911 profile: an NG911 call reaches a call-taker

Setting up.

1. On [Trunks](#) (chapter 7), add the trunk from your ESInet provider (or the NG911 core's ESRP) and switch on **From the ESInet**.
2. In [Call routing](#) (chapter 8), send its calls to your call-takers: a queue, or a ring group.
3. On **911 calls**, enter your agency ID (a domain, like `psap.county.example.gov`) and what CAD takes: the address and port its ALI listener is on, and its EIDO address.

When a 911 call rings, it shows on **911 calls** with:

- **the address** the network sent (civic location), and **the position** with how sure it is (within so many metres), drawn on the map with its circle of uncertainty (once a map is set, below), and a link to OpenStreetMap;
- **the provider**, the kind of service (wireless, VoIP, residence, business), the device and the subscriber, when the call brought them (Additional Data, RFC 7852);
- anything that went wrong reading it.

Location comes in the call (by value), or as a reference BitVoice fetches with HELD. A call with a reference has **Rebid location**: BitVoice asks again, for a caller who is moving or whose first fix was rough.

When a rebid changes the location, the **location check** says which to show first, and why: the new location as a rule; the first one when it was a precise device fix (GPS, within 100 m) less than two minutes old and the new one is rougher; and **both** when two locations disagree by more than their uncertainty allows, so the call-taker asks the caller. The first location stays on the call either way.

When a call-taker answers, CAD gets, each on its own so a CAD that is down never holds up the call:

- **the ALI spill**: the text block CAD systems read (callback number, class of service, address, position and uncertainty, provider), sent over TCP;
- **the EIDO**, the incident document of NENA-STA-024, by HTTPS POST (with a bearer token if set). **EIDO** on a call shows it.

Caller video and photos. On a 911 call, **Send the caller a video link** makes a link only this call opens: texted to the caller from the number under **Text the link from**, or read out. The caller opens it in their phone's browser (no app): **Share live video** sends the back camera live (WebRTC, WHIP: RFC 9725), and **Send photos or videos** sends what they have already taken, dashcam and action-camera files included. On the call:

- live video shows **LIVE**; **Watch the caller's video** plays it **blurred**, and **Reveal (may be distressing)** shows it; **Blur again** hides it;
- each file shows its size, when it was taken (by the camera's own clock), where (from its GPS, when it has any), and its **SHA-256**, taken as it arrived, so a copy can later be shown to be the same file; **Reveal** opens it; **Hold as evidence** keeps the call's files past the retention period;
- the viewing link under it (WHEP) lets CAD, a unit's screen or another position watch the same live video, without signing in: treat it as a key.

Video never takes from the calls: each live video has its own ports, this server takes at most **Live videos at once** of them (4 unless set), each asked to stay under **Video bit rate**, and video that stops for ten seconds is ended. Callers' phones must reach this server at **Link address** (https) and its media ports (UDP). Files are kept **Keep files and the exposure log days** (blank: until deleted), held calls aside.

Recording the live video. With **Record callers' live video** on, the caller's live video is kept with the photos and clips as one file per stretch of video: the same pictures the phone sent, not converted, in an IVF file (H.264 or VP8 with each picture's own time), hashed as it is written. A picture damaged by a lost packet is left out rather than kept broken, and the phone is asked for a full picture to carry on from. **Reveal** plays it (from a copy in WebM or MP4 that ffmpeg makes, the same pictures in another container; without ffmpeg only the original downloads); **Download the original** gives the file the hash is of. ffmpeg, VLC and forensic video tools open the original.

Into CAD. The EIDO (NENA-STA-021) sent to CAD names each photo, clip and recording, and the live video while it lasts, as additional data by reference: `urlPurpose` is `caller-photo`, `caller-video`, `caller-video-recording` or `caller-video-live`, and `additionalDataByReferenceUrl` is the file at `/m/{viewToken}/{n}` or the live video at `/whep/{viewToken}`. BitVoice sends CAD the EIDO again when a file arrives or live video starts or stops. The links use **Link address** (or this server's public address); with neither, the EIDO names no media. Every fetch through them is in the exposure log. The event stream says the same (`GET /api/events` : `call.media` , `call.video`). BitVoice's EIDOS are checked against NENA's published EIDO schema.

Links that do not exist are guessed at no faster than sign-ins: an address that keeps trying is turned away for a while. Files from callers are served in a sandbox, never as a page of this server.

Exposure. Every reveal of a live video and every opening of a photo or clip is counted under **Exposure** on 911 calls: who, how often, over how many calls. It is for supervisors and the centre's wellness programme.

Video a phone sends in the 911 call itself (SIP video over the ESInet) can be watched the same way: **Watch** takes it as the source when no link video is live.

Calls with only a number. A 911 call that brings a callback number and no location (from a CAMA trunk, or an ESInet call whose location did not come) can be looked up: give **Location lookup (ALI)** the address of your ALI database's or location server's HELD interface (RFC 5985, asked by telephone number as RFC 6155 has it). BitVoice asks as the call arrives, without holding it up, and the location appears on the call when the answer comes; the call says when the lookup failed and why.

The legacy ALI link. Without a HELD server, BitVoice can ask an ALI controller, or the ALI database's data link, the old way: it sends a request and reads back the record as text, over the network (**ALI link address** host:port) or out of a serial port (`serial:/dev/ttyUSB0?baud=9600&parity=none` , with `bits` and `stop` too; a USB serial adapter works; on Linux and macOS, not Windows: there, use a serial-to-network converter and its address). **Request** is what to send: `{ani10}` the ten-digit callback number, `{ani7}` its last seven, `{pos2}` the position, `STX` and `ETX` (`{ani10}` when blank). The answer is read up to its `ETX`. ALI layouts differ between providers and states, so you place each field: its line, its column and its length (0: to the end of the line), from a printout of an ALI screen; **Street (whole line)** splits a street line into its parts, or place the parts one by one. Paste a sample record and **Read the sample** to see the lines with a column ruler and what each field reads; **Ask** asks the link itself for a number. The record's address and position become the call's location, its name the caller's name, and its class of service shows on the call. Protocols that are not a request and a text answer (PAM, E2 and other vendors' own) are not spoken: they need a gateway.

ALI spill over a serial port. **ALI spill to** also takes a serial port (`serial:/dev/ttyS0?baud=9600`), for a CAD that reads the spill from a serial cable.

The map. Under **Map** in the 911 settings, give BitVoice its map tiles: an **.mbtiles file** of raster tiles (PNG, JPEG or WebP) on this server, for a map that works with no internet (your GIS team, TileMill, gdal2tiles or a tile vendor makes one of your area), or a **tile server** with `{z}`, `{x}` and `{y}` in its address (your county's, or a commercial one: mind its terms; the public OpenStreetMap servers are not for this). Either way the call-takers' browsers fetch tiles from BitVoice only: it reads the file, or fetches from the tile server and keeps the last tiles in memory. The same map shows radio positions on the radio console. Vector tiles are not supported.

The i3 log keeps the NENA i3 events of each call (call start, answer, location, EIDO sent), one file a day. **Answer times** shows how many calls were answered within your goal (NENA: 90% within 15 seconds, 95% within 20), the abandoned ones, and each call-taker's average. **A row a day (CSV)** downloads the counts state 911 offices usually ask for (calls, answered, abandoned, within the goal and within 20 seconds, average and longest wait) for the days chosen; **A row a call (CSV)** lists every 911 call with when it was answered, by whom, and whether it was text. Each state has its own form: lay yours out under **Report layouts**.

Report layouts. **New report layout** lays out a report as your state or county form has it: what each row is (a day, a week, a month, an hour of the day, a call-taker, or one row in all), a total row, and the columns, in order, each with its own heading if the form words it differently: calls, answered, abandoned (a count or a share), answered within so many seconds (a count or a share; the seconds are yours, so 10 and 20 for one form and 15 and 40 for another), answered after more than so many seconds, the average and longest wait, the time within which 90% were answered, the average call length, and text calls (TTY, real-time text, text to 911). A wait counts from the call's arrival to a call-taker's answer. Choose a layout and dates and **Run** it to read it here, **Spreadsheet (CSV)** to download it, or **Page to print** for a plain page to print or save as PDF. Layouts are kept with the settings, so they go with a backup and into Git.

TTY calls. Deaf and hard-of-hearing callers may call 911 with a TTY (a text telephone, Baudot at 45.45 bit/s). BitVoice listens for one on every 911 call's voice, from the first moment: what the caller types appears on the call as they type it, under **Text**, with a pause or a new line starting a fresh line. A silent call may be a TTY waiting for you: once you answer, **Reply by TTY** sends text

even before any has come. Type the reply and press **Send**: it goes to the caller as TTY tones (the call-taker's voice is muted while it plays, as TTYs take turns), in capitals, and TTY etiquette applies: end each turn with GA ("go ahead"), and SK when signing off. Characters a TTY cannot show are left out. Speech, music and touch tones are never taken for TTY text: a character counts only when every bit of it is a clean TTY tone. The conversation is kept with the call in the call history and in the i3 log (one MessageLogEvent a turn). Untick **Listen for TTYs on 911 calls** in the 911 settings to stop listening.

Transfers. A 911 call sent on to another PSAP keeps its location: the PIDF-LO goes with the call.

Real-time text (RTT). A phone that sends text beside its voice (RFC 4103, as VoLTE phones and NG911 calls do) has its text answered by BitVoice itself, since desk phones show none: each character appears on the call as the caller types it, and your reply goes back as text. Lost packets are made good by the redundancy RTT carries; text lost beyond it shows as .

Text to 911. A text control centre (TCC) sends texts to the PSAP as a session with no voice (MSRP, RFC 4975, over the ESInet trunk). BitVoice answers it and it waits on **911 calls** marked **Text to 911**, with the texts that have come; nobody's phone rings.

Take this text makes you the call-taker (the answer time is counted then, and CAD gets the ALI spill and the EIDO then); reply in the box under the conversation, and **End conversation** when done. The TCC closing the session ends it too. A SIP MESSAGE sent inside a 911 call's dialog is taken as the caller's text the same way, and replies go back by SIP MESSAGE. Texts are taken only from trunks marked **From the ESInet**.

A TCC that offers the session over TLS (`TCP/TLS/MSRP` , msrps: paths, RFC 4975 section 14) gets it over TLS, with this server's certificate (Settings > Certificates; without one such an offer is refused). The TCC's certificate must be the one its offer names (`a=fingerprint` , SHA-256); when it names none, it must be traced to the PSAP Credentialing Agency roots set under Core services, if any are set. BitVoice's answer names its own certificate the same way.

The whole conversation, whichever way it came, is kept with the call in the call history and in the i3 log.

What is not here yet: passing a text session on through the NG911 core (the ESRP routes voice; a text session is answered where it arrives); the EIDO conveyance standard's own subscriptions (NENA-STA-024 WebSocket): BitVoice posts EIDOs to one address.

CHAPTER 75

Radio dispatch console

In the web interface: `/radio`

Lab and design-partner track. Not certified for live 911 or radio dispatch, and not yet tried with a live ESInet, P25 system or CAD. Use it in a lab or in shadow mode beside your current equipment. Switch on **Public-safety profiles (lab)** in [Settings](#) (chapter 14) to show it in the menu.

Dispatchers talk on radio channels from the same headset as their phone calls. BitVoice reaches the radios through radio-over-IP (RoIP) gateways; the radio network itself stays with whoever runs it.

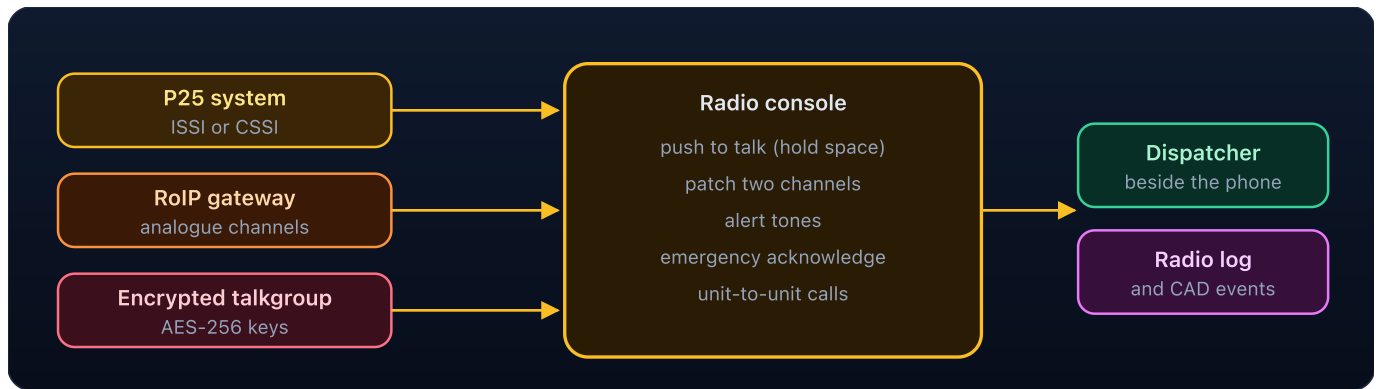


Figure 62. The radio console: talkgroups beside the phone

Setting up. Under **Console setup**, choose a console number (800, say) and add each channel: its name, the port on this server the gateway sends the channel's audio to, and the gateway's address and port for the audio going out. On the gateway: RTP, G.711 μ -law, to this server's address and that port; key the radio on voice (VOX). Each channel uses its port and the next one, outside the range phone calls use.

Joining. A dispatcher dials the console number from their desk phone, or presses **Join with the browser phone**. They then hear every channel's traffic mixed together; **Hearing / Muted** on a channel takes it out of their headset. The lamp on a channel lights while it is receiving.

Talking. Hold a channel's **Hold to talk** button (or focus it and hold the space bar): your voice goes to that channel only, and the channel shows **TX** with your extension. One dispatcher talks on a channel at a time; the button is greyed for the others while someone is talking. Nothing is sent to a channel while nobody is talking, so a voice-keyed radio is never held on the air.

Patching. Press **Patch** on one channel, then **Patch here** on another: the two hear each other (a fire channel and a mutual-aid channel, say) until you press **Unpatch**.

P25 talkgroups. A channel can also be a P25 talkgroup, joined directly over the ISSI (TIA-102.BACA, the interface between P25 subsystems; with **Send the CSSI console PTT word** on, BitVoice adds the console subsystem's word to each request). Choose **A P25 talkgroup (ISSI)**, give the talkgroup ID, the SIP address and ID of the RFSS that is home to the talkgroup, and a port for its voice. Under **This console on the P25 system**, enter the WACN and system ID, the RFSS ID your P25 administrator gave this console, the unit ID dispatchers transmit as, and the SIP port the ISSI uses here; the P25 system must know this server as an RFSS at that address. The tile shows **P25** while the group call is up (**P25 down** and the reason when it is not), and the unit ID of the radio talking, in red with **EMERGENCY** when it is an emergency call. **Hold to talk** asks the home RFSS for the channel (**Asking for the channel...**); your voice goes once it grants it, and if it refuses (someone with a higher priority has it) the tile says so. Voice is converted to and from IMBE, the P25 vocoder, in BitVoice.

Encrypted talkgroups. Under **Encryption keys**, add each traffic key your radio shop gives you: a name, the algorithm (AES-256, DES-OFB or triple DES), the key ID and the key itself in hex (TIA-102.AAAD). A key is never shown again once saved, not even to administrators, and leaving the field empty keeps it. The console hears an encrypted transmission when it has the key with that algorithm and key ID: the tile shows a lock and the algorithm next to the talking unit. Without the key the transmission is muted, never played as noise, and the tile says **no key: muted**. To transmit encrypted on a talkgroup, choose a key under **Transmit** for that channel; the tile then shows **TX AES-256** (or the algorithm you chose). A full configuration export (Settings, an administrator's backup) carries the keys, so keep those files as safe as the keys.

Private calls. With the console on a P25 system, **Call a radio** in your position's bar calls one radio by its unit ID (turn on **Emergency** to call it as an emergency). The call becomes a tile of its own, **Private call: unit 197**, with **Hold to talk** like any channel and **Hang up**. Radios can call the console too, on the unit ID dispatchers transmit as: the call appears as a tile marked **called in** (in red with **EMERGENCY** when the radio called as one), every dispatcher hears it, and it goes when either side hangs

up. Under **Private calls** in the console's P25 settings, give the SIP address and ID of the RFSS that is home to the radios (empty uses the first talkgroup's home) and, if private calls should be encrypted, the key to send them with.

Alert tones. Send an alert tone... on a channel sends one of: **Alert 1**, a steady 1000 Hz tone for 3 seconds; **Alert 2**, a warble between 1000 and 1500 Hz; **Alert 3**, 1000 Hz pulses; or a short **Beep**. The tone holds the channel while it plays (the tile shows **tone** and your extension), every dispatcher monitoring the channel hears it go out, and on a P25 talkgroup it goes once the home RFSS grants the channel.

Emergencies. When a radio transmits as an emergency (its emergency button), its tile turns red with **EMERGENCY** and the unit ID, the server log notes it, and CAD is told at once (below). The emergency stays raised on every console after the radio stops talking, until a dispatcher presses **Acknowledge**: the tile then says who acknowledged it and when, and CAD gets a `radio.emergency.ack` event.

Simulselect. To say one thing on several channels at once, tick **select** on each channel, then hold **Talk on N channels** in the bar above the tiles. It is all or nothing: if another dispatcher is talking on one of them, none is keyed and you are told which.

Radio positions. Radios report where they are over the radio system's data side, not the voice side the console joins: P25 location service, Motorola's LRRP or a vendor's own, collected by the radio shop's data or AVL gateway. Have the gateway (or a small script beside it) POST each report to BitVoice with an API token: `POST /api/radio/locations` with `{"unit": 1205, "lat": 40.2633, "lon": -76.8823, "speed": 42, "heading": 90, "accuracy": 8, "at": "2026-10-15T14:05:00Z"}`, or a list of up to 500 (speed in km/h, heading in degrees, accuracy in metres; all three and the time are optional). **Radio positions** on the console then lists each radio's last fix, with its name, and CAD gets each one as a `radio.location` event. BitVoice does not decode LRRP or the P25 location service itself.

Radio names. Under **Radio names**, name radios by their P25 unit ID (1205 is "Engine 5", say): the tile shows the name beside the unit while it talks, and the radio log and CAD's events carry it.

Radio log and recording. Turn on **Record every transmission** under **Recording and CAD**: each transmission on each channel, heard or sent, is kept as a recording with a line in the **Radio log**: when, the channel, who (the radio's unit, or the dispatcher's extension), how long, and whether it was an emergency, an alert tone, encrypted or a private call. A heard transmission starts with the first speech and ends 1.5 seconds after the last word. Find a unit, dispatcher or channel with the search box, and press **Play** to listen. Recordings are kept as long as call recordings are (Settings, Recording).

Radio data to CAD. Give CAD's address under **Send radio events to CAD at** (and a shared secret if CAD wants one; it is sent as a bearer token and never shown again). BitVoice then POSTs each event (radio.unit, radio.emergency, radio.emergency.ack, radio.transmission and radio.location) as JSON, `{"type": ..., "agency": ..., "data": {...}}`: **radio.unit** when a radio starts talking on a channel (its unit ID, WACN, system, the channel, emergency, encryption), **radio.emergency** for an emergency (tried three times if CAD does not answer), and **radio.transmission** for each finished transmission as it appears in the radio log. CAD can also read the log itself with `GET /api/radio/log` (filters: `channel`, `q`, `since`, `until`). A CAD that is down never holds up the console. P25 radios' GPS positions are not part of this: they travel over P25 packet data (location services), not over the ISSI's voice.

Not here yet: P25 Phase 2 (TDMA) voice in its own half-rate vocoder. A Phase 2 system works with the console when its ISSI talkgroups carry full-rate (Phase 1) voice or its transcoder converts them, which is common: ask your P25 administrator. A talkgroup whose voice arrives in blocks this console does not decode (half-rate voice sent untranscoded, most likely) shows **Silent here** on its tile while a radio talks, so nobody takes the silence for a quiet channel. Decoding it needs two things BitVoice does not have: the half-rate vocoder (AMBE+2, patented until 2028: a licence from its owner, or a licensed hardware vocoder chip), and the ISSI's half-rate voice block format (TIA-102.BACA-B, sold by TIA, not public).

NG911 Core Services (ECRF, ESRP, LVF, BCF, LIS, logging)

In the web interface: `/core`

Lab and design-partner track. Not certified for live 911 or radio dispatch, and not yet tried with a live ESInet, P25 system or CAD. Use it in a lab or in shadow mode beside your current equipment. Switch on **Public-safety profiles (lab)** in [Settings](#) (chapter 14) to show it in the menu.

For a state or regional 911 authority that runs its own NG911 core: BitVoice answers LoST as the ECRF, and routes each 911 call as the ESRP to the PSAP that serves the caller.

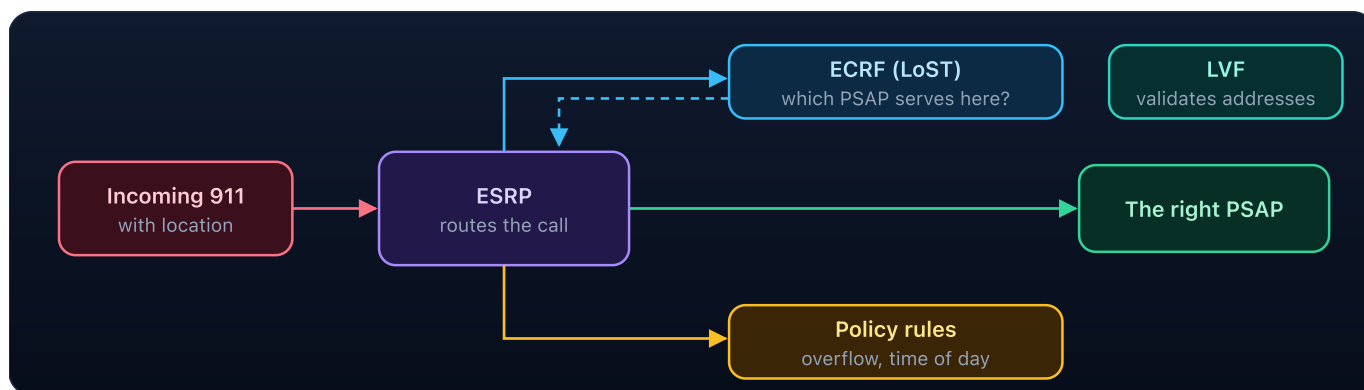


Figure 63. NG911 Core Services: routing a call by where the caller is

Service areas. Export your boundaries from GIS as GeoJSON polygons (PSAP boundaries for `urn:service:sos`; police, fire or EMS areas for `urn:service:sos.police` and the others) and upload them on **Core services**. A polygon's `name`, `uri` and `number` properties are read; give a default PSAP URI for those without one. **Edit** an area to add the cities or counties it serves, for callers whose location is a civic address only. **Try a place** shows which area answers a position or a city, as a LoST query would.

The ECRF. Switch on **Answer LoST queries** and list the addresses that may ask (your ESInet): other elements send `findService` to `/lost` (RFC 5222) and get the PSAP's URI. A specific service (`sos.police`) is answered from its own areas first, then from `sos`. A place none of your areas covers can go to the **Forest guide** (RFC 5582: the national directory of ECRFs): a recursive query is asked there on the asker's behalf and its answer returned (both servers listed in the answer's path), any other query gets a redirect to it, and a query that has already passed through this ECRF is refused as a loop.

The ESRP. List the PSAPs: each one's URI (as the areas name it), the trunk that reaches it and its overflow PSAP, and choose the default PSAP. Then send the ESInet's incoming 911 calls to **ESRP** in [Call routing](#) (chapter 8). Each call goes to the PSAP for its location (from this server's areas, or an **outside ECRF** by LoST), with its location and the service it asked for carried on; when that PSAP does not take it, to its overflow; without a location, or from a place no area covers, to the default. Each decision is in the i3 log on [911 calls](#) (chapter 74).

Policy rules. Once the ECRF has picked a PSAP, the rules under **Policy rules** are checked in order and the first whose conditions all hold decides: send the call to another PSAP, or refuse it as busy. A rule can look at the service (fire calls only, say), the PSAP the ECRF chose, a schedule (while the night shift's centre is closed), calls with no location, calls the BCF marked suspicious, and the trunk the call came in on. Each decision is in the i3 log as a `RouteLogEvent` naming the rule.

Border control (BCF). With **Check calls from the ESInet** on, every call from an ESInet trunk is checked first: addresses under **Always refuse** are turned away, and so is an address that starts more calls a minute, or holds more calls at once, than you allow (a telephony denial-of-service attack sends floods of calls from a few sources). Refused calls get 503 with `Retry-After`. Turn on **let the call through marked suspicious** to send such calls on instead and let a policy rule decide (to a centre that screens them, or refused); a real caller in a flood is then not lost. **Border control: latest** lists what was refused or marked. **Check STIR/SHAKEN**

signatures checks the signed caller ID (the Identity header) of every call that carries one, against the STI roots set under Call screening in Settings: a call whose signature fails goes through marked suspicious, for a policy rule to send somewhere else, and is listed in the BCF log; a 911 call is never refused for it, and unsigned calls (most of them today) are left alone. Each call leaves BitVoice as a new request of its own (BitVoice is a back-to-back agent), carrying only the headers and bodies BitVoice understands and passes on: the location, the Additional Data and the session. **Inspect media** checks every RTP packet of an ESInet call before it is used or passed on: packets from an address other than the one the call's SDP gave, with a payload type the offer did not name, with header lengths (CSRCs, extension, padding) that do not fit the packet, larger than **Largest packet** (1,500 bytes when blank), or past **Packets a second** (150 when blank; G.711 at 20 ms sends 50) are dropped and counted. The first of each kind, then the 10th, 100th and so on, goes in **Border control: latest** (as *media dropped*) and the call's notes. The call is never ended for its media: the caller keeps the good packets. Calls from CAMA trunks come from the local card program and are not inspected.

Address validation (LVF). Upload your 911 GIS address data as GeoJSON, in the NENA GIS data model (NENA-STA-006): road centerlines with `FromAddr_L`, `ToAddr_L`, `Parity_L`, `IncMuni_L` and the rest, and site/structure address points with `Add_Number`, `St_Name`, `Inc_Muni`. A LoST query with `validateLocation="true"` then gets the address's valid, invalid and unchecked parts in its answer (RFC 5222), so carriers and location servers can correct an address before a 911 call ever uses it. **Check an address** does the same here: an exact address point is the best match, then a street segment whose range on the right side of the road holds the number (odd and even sides respected); spellings like Street and St, North and N are treated alike.

The i3 logging service. Switch on **Run an i3 logging service**: the other elements of your core (ESRPs, BCFs, PSAPs) send their log events (NENA-STA-010), each signed as a JWS with the element's certificate in its x5c header, by POST to `/i3log/LogEvents` from the allowed addresses. Each comes as a JWS in the flattened JSON form (signed, or unsigned with algorithm `none`, which a logging service may take: marked as not traced to the PCA), is checked and kept, and the answer is its log event ID. Read back with `GET /i3log/LogEvents/{logEventId}`; list with `GET /i3log/LogEvents ( startTime , endTime , logEventType , limit , start )`; find the IDs of a call or incident with `/i3log/LogEventIds`, `/CallIds`, `/IncidentIds` and `/AgencyIds`; and read a call's text conversation with `/i3log/Conversations?callId=`. `GET /i3log/Versions` says what this service speaks. **Log events received** lists a day's events with who signed each. Every answer is checked against NENA's published OpenAPI definition of the service; the service has not yet been tried with another vendor's elements.

PSAP Credentialing Agency. Paste the PCA's root certificates under **PSAP Credentialing Agency roots**: a signed log event is then taken only when its signer's certificate chains to one of them (and has not expired), and is shown with **PCA**. Without them, the signature alone is checked, and the event is shown as **not traced to the PCA**.

Location records (LIS). Switch on **Answer location requests (LIS)** and upload a spreadsheet (CSV) of location records: a header row, and on each row one of a **number** (the ALI data, exported), a **network** (an address range such as 10.20.3.0/24, or one address: a building, a floor, a VLAN, a Wi-Fi network) or a **mac** address (one device: a desk phone, a wall jack's switch port), then the address as **hno**, **rd**, **sts**, **unit**, **floor** (or **flr**), **a3** (city), **a1** (state), **pc** and **country**, with **lat**, **lon** and **radius** where known (other names, such as **phone**, **street**, **zip**, **subnet** and **ip**, are understood too). PSAPs, legacy network gateways and phones then ask `/held` (HELD, RFC 5985) from the allowed addresses and get a PIDF-LO. A request names its device by telephone number, IP address or MAC address (RFC 6155); a request that names none is answered for the address it came from, as a LIS on an office or campus network does. An address is located by the narrowest network that holds it, so a floor's /24 wins over the campus /16. A device with no record gets HELD's locationUnknown; an identity of another kind (a NAI, a port) gets notLocatable. A BitVoice PSAP points its **Location lookup (ALI)** here. **Look up a number, address or MAC address** checks a record and says which network it came from. A number, network or MAC address listed again replaces its old record; **Replace every record** starts afresh. Only the allowed addresses are answered: a LIS that answers for any address it is asked about must not be open to anyone.

The i3 web services. Under **NENA i3 web services**, switch on what this core offers the other elements of your ESInet, at `/i3/<service>/v1`, for the addresses under **May ask** only. Each answers as NENA's published OpenAPI definitions say (NENA-STA-010.3, the NENA911 repositories), and each has a tab under **i3 web services**:

- **Policy Store** (`/i3/PolicyStore/v1/Policies` , `/PolicyEnums`): agencies store, update and delete their routing policies (Policy Routing Rules), each a JWS signed by the agency that owns it (the signer's certificate must name the owner, and come from the PCA when its roots are set). A policy is kept byte for byte as it came. The ESRP runs them: the **OriginationRoutePolicy** for **This ESRP's incoming queue** first, else the **NormalNextHopRoutePolicy** of the PSAP the ECRF chose. Of the rules whose conditions hold, the highest priority acts; when its route fails (busy, no answer within its **rnaTimer**, 20 seconds when it gives none), the next one does; a **BusyAction** answers 600 Busy Everywhere. Conditions taken: time periods, SIP headers, Additional Data, body types, location (circle or civic), call suspicion, queue state (BitVoice's own PSAPs), LoST service URN and Normal-NextHop, call source, request URI, incoming queue, and the SDP offer's media and languages; service state and security posture are taken as unknown (Unreachable and Green, as the standard says). Body-part, CAP and calling-number-verification conditions are not run: they count as false. Without a stored policy, the **Policy rules** above decide. The tab lists what is stored and lets you paste one, or sign one with this server's certificate.
- **Discrepancy reports** (`/i3/DiscrepancyReports/v1`): other agencies report problems with this server's services; each shows on the tab, where you answer it with a resolution, sent to the reporter. **File a report** sends one to another agency's DR service; its answer and later resolution show under **Reports this server filed**.
- **Bad actors** (`/i3/BadActor/v1/BadActors` , with the BCF): each call the BCF passes on carries its source's ID (`Call-Info: <sip:ID>;purpose=emergency-source`); a PSAP that finds that source attacking reports the ID, and the BCF refuses its calls, or raises their suspicion score for the policies, for **A report lasts** hours. **Lift** ends a report early.
- **Geocode** and **MSAG conversion** (`/i3/Gcs/v1` , `/i3/Mcs/v1`): civic address to position and back from the LVF's address points (the nearest within 150 m), and civic address to MSAG address (`120 N MARKET STREET, HARRISBURG CITY, PA`) and back, from the centerlines' `MSAGComm_L` / `MSAGComm_R` fields. Answers are JSON objects, or XML when the asker asks for it.
- **Additional Data** (`/i3/Adr/v1/AdditionalData?callerUri=`): blocks you keep by caller (RFC 7852 XML), served by value, or by reference for an asker that wants JSON. One block per caller is served by value: the first kept.
- **Locator** (`/i3/ServiceAgencyLocator/v1`): this agency's own record (its ID from the 911 calls page, with its logging and discrepancy addresses) and the agencies you add.
- **Dequeue** (`/i3/DequeueRegistration/v1/Registrations`): positions or PSAPs sign up to take a queue's calls (this ESRP's queue, or a PSAP's URI); the ESRP sends calls for that queue to them, the most preferred first (preference 5 before 1), over the PSAP's trunk.
- **Test calls**: a PSAP asks this core to send it test calls (`/i3/TestCall/v1/SendTestCalls/{agency ID}`): every so many minutes, five at least, between two dates, with a location); BitVoice places each to `urn:service:test.sos` over the PSAP's trunk and keeps what came back. **Place one now** tries one.
- **PSAP states** (SIP, not REST): **Subscribe to each PSAP's queue and service state** has the ESRP SUBSCRIBE (RFC 6665) to every PSAP listed, over its trunk, for `emergency-QueueState` and `emergency-ServiceState` (NENA-STA-010.3 4.2.1.3 and 2.4.2), for an hour at a time, renewed before it ends. What the PSAPs NOTIFY shows under **PSAP states** and is what the policy rules' queue state, service state and security posture conditions read. A PSAP whose queue says Inactive, Disabled, Full or ResourceExhausted (or holds its most calls), or whose service says Unstaffed, ScheduledMaintenanceDown, Overloaded, GoingDown or Down, is passed over for its overflow, and the call's notes say so; when every PSAP in the chain says it cannot take calls, the call is sent as listed anyway, never left with nowhere to go. A PSAP that does not answer the SUBSCRIBE is Unreachable (as the standard says a subscriber must assume) and is tried again each minute; one that refuses it (489, 603 ...) is tried again in ten minutes.
- **This server's states**: **Tell other elements this server's element, service and queue state** answers SUBSCRIBEs for `emergency-ElementState` , `emergency-ServiceState` and `emergency-QueueState` from the addresses under **May ask** (others get 603 Decline; a queue that is not this server's incoming queue, 488). The queue's length is the 911 calls not yet answered here, and it says Full at **Most calls waiting in the queue** (20 when blank); the service says Overloaded at this server's call limit, else Normal. NOTIFYs go out when the state changes, at most one a second, and once a minute regardless (the standard's default rates); their bodies are checked against NENA's notify-body schemas.

Test calls at a PSAP. A BitVoice PSAP answers a test call from the ESInet (`urn:service:test.sos` , NENA-STA-010.3 section 9, RFC 6881) by itself: its answer says the PSAP's name, the service and the location the call brought, the first three packets of

sound go back, and it hangs up. A second test from the same place within two minutes is refused (486), as is a fifth at once. An ESRP passes a test call on like a real one, still marked as a test. Switch off **Answer test calls** on the 911 calls page to refuse them (404).

JWS algorithms: NENA-STA-010.3 asks for EdDSA with Ed448. Go's standard library, which BitVoice uses for every cryptographic step, has Ed25519 but not Ed448, so EdDSA signatures are checked with Ed25519 keys, and ECDSA and RSA signatures are taken too; an Ed448 certificate is refused.

Did you mean. An address that is not valid gets up to five suggestions from the same data, best first, each saying why: a street a letter or two away (a swapped pair of letters counts as one), the same street in the next town, or the nearest number the street has, on the same side when it can (even numbers stay even). **Use** fills a suggestion in. **A position with the address.** Give a latitude and longitude (and the position's radius, when known) with the address: the position is checked against the address's own address point, allowing its radius and 100 m more (the size of a plot, and GIS error). The answer gives the distance and whether the two agree, and, when they do not, the address point nearest the position. An address known only from road centerlines has no point to check against, and the answer says so. LoST's validateLocation has no place for either, so carriers asking by LoST get the validation alone.

BitVoice is a product of BioDream LLC, published under the Apache License 2.0. The same chapters are under **Help** in every BitVoice.