PRESENCEOS

A phone that answers to the person holding it

Whitepaper v1.0 — July 2026

CONTENTS

Paul — creator of PresenceOS, New Zealand
paul@presenceos.email · +64 22 629 2483 · https://presenceos.mobile

1. The problem

Pick up any mainstream phone and ask one question: who is this thing actually working for?

The honest answer is: mostly not you. A modern smartphone is a delivery mechanism for other people's business models. The app store rewards whatever keeps you looking longest. The notifications exist to pull you back. The "free" services pay for themselves with your data. None of this is a conspiracy — it's just the economics of the platform, and every phone in the shop is built on top of it.

Now hand that phone to a child.

The industry's answer is parental controls: screen-time dashboards, app timers, content filters, location trackers. I think of these as band-aids. They ask a parent to configure a phone into being safe, when the phone was designed from the ground up to be engaging. You are fighting the platform's economics with a settings menu, and the platform has more engineers than you do.

Even on a regular mum-and-dad Android phone, with no kids involved, the single most common frustration is the same story in miniature: *what is that app, what is that notification, why is my phone so slow now?* The device fills up with things nobody asked for, working for people nobody invited.

I didn't want to configure my way around that model. I wanted a phone that simply doesn't have it.

2. What PresenceOS is

PresenceOS is a purpose-built operating system experience for a family's phones. I'm Paul, its sole creator, and I build every part of it — the phone software, the parent app, the update pipeline, and the infrastructure behind them — from New Zealand.

Today, PresenceOS ships as **Beta 7 (v1.5.30)**, running on an Android base on a moto g56 5G. I'm upfront about that base, and section 8 explains where it's headed — the destination is not Android at all. But the experience is already nothing like a stock phone.

What it keeps. The things a phone is genuinely for: calls, end-to-end encrypted messaging with the people you've paired with in person, a browser, music (your local files, plus a plain frontend for Spotify and Deezer), photos, and backups that stay inside your own circle. The phone stays fast, does just enough to be genuinely helpful to your family, and stays absolutely useless to anyone who wants to profit from your data.

What it refuses to do. There is no third-party app store and no way to sideload the usual attention machinery. There are zero Google services on the device — no Play Services, no Google messaging framework, no analytics SDKs. Push notifications come from my own server. In the browser, social logins are structurally unavailable — not blocked by a filter you can toggle, but unavailable by construction. The one exception is virtual-device access that a guardian has explicitly granted.

My working principle for every feature decision: functionality before profit, and safety before privacy breaches. If a feature only makes sense because it harvests something, it doesn't go in.

3. Identity without accounts

There are no accounts in PresenceOS. No usernames, no passwords, no two-factor codes, no "verify your email". Nothing to phish, nothing to reset, nothing to leak in the next big breach.

Instead, identity belongs to the device itself. Every PresenceOS phone has a device identity, presented to humans as a **POS-ID** — for example, `POS-IHV105UW`. That identity is what other devices pair with and what the relay recognises. You never type a password to be you; the phone in your hand *is* you.

The obvious question: what happens when things go wrong? Recovery runs through your people, not through a form on a website.

- If a child's phone is locked and needs to be opened remotely, that takes approvals from **two or more** guardians through their Companion apps (section 5).
- If a phone is lost or replaced, your data comes back through myCloud: your recovery passphrase plus the trusted contacts who hold your encrypted shards (section 6).

In development: human-friendly **@presenceos.email names** to replace raw POS-IDs, running on my own self-hosted mail and identity infrastructure. This is in-development work — it is not in shipping builds today, and I'll say so plainly until it is.

4. Contacts by consent

On PresenceOS, a contact is not a row in a database that anyone can request. It's an agreement between two people standing in the same room. Here is the actual text from the pairing screen in v1.5.30:

> "Only one of you presses Start Pairing. The other phone just needs Presence installed and its screen on — then tap the phones together and agree how long the connection lasts."

That's the whole model. Pairing is NFC — physically tapping the phones together. One button, pressed by one person, and both people agree to the connection. There is no username search, no friend-request queue, no way for a stranger to reach into the contact list. If someone isn't standing next to you (or hasn't been approved by a guardian — section 5), they don't exist to your phone.

Two more properties matter here:

Every connection has an agreed time limit. Again, straight from the app: "Each connection has an agreed time limit." A pairing isn't a permanent open door — it lasts as long as both people agreed it should.

Deleting is mutual. When you delete a contact, you don't just remove them from your phone — you remove yourself from theirs. Both ends, one action. There's no ghost copy of you left behind on somebody else's device.

Between the phones sits **presenceos.services**, the pairing tracker and message relay. Its job is narrow: it keeps devices from forgetting who they're linked to, and it carries encrypted packets that it cannot read. It has no accounts to mine and no plaintext to look at.

5. Guardian Relay

Guardian Relay is the parent side of PresenceOS, delivered through the **Guardian Companion** app (v1.3.50) — an ordinary app a parent installs on their own existing phone. A child's device supports up to **three** guardians, and the model is consent-based throughout.

Event-based alerts, not surveillance. The Companion tells a guardian when something happens that they should know about. It is not an always-on tracker; there is no live feed of a child's location or screen to sit and watch. I think constant surveillance teaches kids to route around their parents. Events, alerts, and conversations work better — and they're honest.

Calling. A guardian can approve regular phone numbers that the child can call — grandparents on a landline, a parent at work, whatever fits your family. And one rule sits above everything: **111 always works.** Emergency calling is never gated behind approvals, pairings, or anything else.

Remote connections. Guardians can also approve remote contact connections between the child and family members and friends who install Guardian Companion — the aunt in another city, the friend who moved schools. For those remote connections, I have a novel, secure and very personal identity-verification method in active development: it stays entirely between you, your child, and the third party, relying on no external source of truth. To be clear and honest: it is not in shared builds yet — it is actively evolving in my development environment.

Remote unlock. If a child's phone needs to be opened remotely, that requires approvals from **two or more** Companions. No single adult — including me — can quietly open a child's device alone.

6. myCloud distributed backup

Every phone backup product I know of ends the same way: your family's photos on somebody else's servers, under somebody else's terms. myCloud is my answer, and it's live today on Android.

How it works. Your data is encrypted on your device, then split into shards across your trusted contacts' devices. Each contact holds ciphertext they cannot read — to them it's opaque bytes, nothing more. To restore, you need two things: your recovery passphrase, and your people. No company sits in the middle, and no data centre holds a readable copy — because no readable copy exists anywhere except in your hands.

Because your friends' devices are the storage, the backups themselves carry no storage fee by design. (To be precise about what *is* paid for: the Guardian Relay service is the paid part of PresenceOS — see section 9. The backups aren't.)

What this looks like in practice:

- The old iPhone in a drawer becomes a backup vault for the things you actually care about *(when the iOS Companion ships)*.

- Buying a new phone: move your important data offline, between you, a friend, and the new device — no cloud middleman.
- Back up sets of holiday photos knowing they sit on hardware you own.

The longer vision — and I label it as vision, not product: a home hub that acts as a family's personal cloud, so you back up to your own devices or your own home network, never to a third party.

On the intellectual-property side: I have filed a **provisional patent application with IPONZ** (the Intellectual Property Office of New Zealand) covering this backup system. That is an application, not a granted patent, and I won't pretend otherwise. I don't claim to own distributed backup as a concept — the moat here is execution and integration, not the filing.

7. Infrastructure independence

A privacy product that depends on other people's infrastructure is making a promise it can't keep. So PresenceOS depends on as little of it as I can manage.

Zero Google services. No Play Services, no Google push messaging, no analytics SDKs — nothing on the device phones home to Mountain View. Push notifications are delivered by my own server.

My own update pipeline. Updates are built, signed and shipped by me, straight from my development environment to the devices. The update system enforces a **minimum version**: builds below the floor must update before they are allowed to reconnect. That floor is a security feature, not a nuisance — it's how end-to-end encryption stays locked in as the *only* way to chat on PresenceOS. There is no legacy mode to downgrade into.

One download source. presenceos.mobile is the only legitimate download source for the Guardian Companion beta APK. Nowhere else — if you found it somewhere else, don't install it. (Inside PresenceOS, devices talk to presenceos.services, the pairing relay described in section 4.)

Self-hosted source code. In May 2026, GitHub confirmed a breach: an employee installed a poisoned VS Code extension, and roughly 3,800–4,000 GitHub-internal repositories were exfiltrated, with the group behind it offering the archive for sale.[1] The same wave of attacks poisoned over 170 npm and PyPI packages and hijacked a VS Code extension with around 2.2 million installs, spreading a self-replicating worm that stole CI/CD credentials.[2] The same month, a copycat campaign hit more than 5,000 public repositories with malware-laden commits.[3]

On 9 July 2026, I moved all PresenceOS source off GitHub and onto my own git server, on my own hardware — I verified a complete capture first, then deleted the originals. In hindsight, hosting the

1. TechRadar: "GitHub confirms breach — thousands of internal repositories hit after employee installs malicious VS Code extension" — https://www.techradar.com/pro/security/github-confirms-breach-thousands-of-internal-repositories-hit-after-employee-installs-malicious-vs-code-extension
2. TechRadar: "Malicious AI-made extension with ransomware capabilities sneaks on to Microsoft's official VS Code marketplace" — https://www.techradar.com/pro/security/malicious-ai-made-extension-with-ransomware-capabilities-sneaks-on-to-microsofts-official-vs-code-marketplace
3. TechRadar: "GitHub hit with another major attack — Megalodon hits over 5,000 repos with malware-laden commits" — https://www.techradar.com/pro/security/github-hit-with-another-major-attack-megalodon-hits-over-5-000-repos-with-malware-laden-commits

code for a privacy product on someone else's platform was a mistake, open source or not. With AI-generated malware now flooding package registries and extension marketplaces, my view is simple: AI can't be trusted yet. Maybe one day.

8. Beyond Android

Here is the part I want to be most careful about, because this is where a document like this could blur what runs today with what I'm building next. So, plainly:

Today, PresenceOS Beta 7 rides on an Android base on the moto g56 5G. That is what pilot families receive, and it is the shipping vehicle while the firmware underneath matures.

On my bench, there is a Motorola Edge 50 Fusion that boots a Linux userspace **completely void of Android and Google**. The stack is: the hardware, the vendor's drivers and hardware-interface layer, and then my own systemd userspace running the PresenceOS UI. No Android framework, no Google anything. Think postmarketOS, but it's PresenceOS — only PresenceOS. It is early — a first iteration — and it boots on my bench today. It is not what ships yet, and I won't claim it is.

Why does this matter? Because Google is tightening its grip on Android — developer-verification requirements, restrictions on sideloading. Every Android-adjacent project is downstream of those decisions. PresenceOS's answer is that it is **not Android**: no tie to Google, no tie to AOSP, so there is no jurisdiction there. My devices get their updates from my internal ecosystem, straight from my development environment — no store review, no platform gatekeeper.

One honest caveat: vendor and OEM proprietary code — the drivers — is still required today to make the hardware work. One day it won't be. But I'd rather tell you the real dependency now than have you discover it later.

9. Business model and pilot

I want the money side of this to be as legible as the technology. There are exactly three things you can pay for:

- **Guardian Companion app — $5/month**, standalone, through the app stores. The store-distributed version follows Apple's and Google's guidelines so it can be sold there; it is a separate flavour from the self-updating beta that pilot families use.
- **PresenceOS beta device — $600 one-time.** That's a **new moto g56 5G (256 GB)** flashed with PresenceOS, and the price covers the device, shipping, and the flashing service. It comes with a **6-month hardware guarantee from me** — unlocking the bootloader voids the manufacturer's warranty, so my guarantee replaces it. The parent buys the Companion app from their own app store; PresenceOS updates come from my internal ecosystem, Companion updates through the stores.
- **Founder feedback loop — $75/month per family**, with a scheduled monthly feedback session. This is the pilot programme, and the point of it is that the founding families decide what grows: you tell me, I bake it in, sign it, publish it, and push the update to you.

The pilot launches **Q1 2027** with **five founding families**. Applications are open now at presenceos.mobile.

Funding. The project has received **$10,000 of government funding**. I'll frame that honestly: it's a little funding — miles away from targets — but it proves the project is fundable. Every step of development is documented; every thought and action is stored and available for scrutiny by serious parties.

On source code. PresenceOS is closed source, at least during development. That isn't ideology — it protects me, the codebase, and my workstation while this is very much beta. How the licence looks later is a decision I'll make together with the people who invest in me and in this pilot.

10. How to take part

If you've read this far, there are three ways in:

1. **Apply for the pilot.** Five founding families, Q1 2027, applications open at https://presenceos.mobile.
2. **Talk to me.** I'm one person, and I answer my own email and my own phone: paul@presenceos.email, +64 22 629 2483.
3. **Scrutinise me.** If you're an investor, an agency, or an engineer who wants to kick the tyres — every step of this project is documented, and I'll walk you through any of it.

PresenceOS exists because I think a family should be able to buy a phone that answers to the person holding it — not to an advertiser, not to an app store, and not to me either. This whitepaper describes exactly where that idea stands in July 2026: what runs today, what boots on the bench, and what's still in development, each labelled as what it is.

— Paul, New Zealand, July 2026