NERO PROJECT

Nero - AI Operating System for Home and Work

Human approval at the points that matter. Results verified by outcome.

Paul Schindler | Built with Claude and OpenAI Codex | November 2026

Executive summary

Nero is a personal AI operations system that I designed and run on a home workstation. AI agents from two providers, Claude and OpenAI Codex, write and maintain its code and do much of the daily work, under written operating rules that I set. Every agent starts with read-only access, every change is proposed first in a five-part brief in plain English that a person approves, and each result is checked by what actually happened in the house or on the screen. The most important limits are enforced by permission settings and a command filter that the agents cannot edit. Nero runs my household and my projects every day, and twelve of its parts are described in their own reports.

1,406
changes recorded in version control, as of November 2026

374
notes in the shared memory, as of November 2026

50+
projects on one live dashboard, as of November 2026

5 parts
in the plain-English brief written before every change

What is in this report

Page	Contents
2	The problem I set out to solve, and the designs I ruled out first
3	How Nero works, from the operator to the reports that come back
4	The rules every agent works under, and the code that enforces them
5	The two kinds of AI worker, the tests behind them and the daily reports
6	The twelve component projects, results to date, next steps and what the work shows about how I operate

My role

I set the requirements, wrote the operating rules, approve every change and verify each result. AI agents wrote and maintain the code. I do not write code by hand, and the rules are built around that fact. The work dates to March 2023 in my notes and early designs, and the current version-controlled system dates from June 2026. Each component project on page 6 has its own report, and hardware, network and security details are left out on purpose because Nero runs a private home.

1. The problem

AI agents can now research, draft, write code and run commands on the computers that run a household. I wanted that help for my household and my projects, and those include the systems behind our cameras, our alarm and our home network. I do not write code by hand and cannot judge most code changes line by line, so reading an agent's code was never going to be my control. I had to work out how to give agents real access while a person stays in charge of anything that cannot be undone.

My answer was to write the operating rules before I granted the access. The rules came first, while the agents had nothing dangerous to control, and every later capability was added under them. That order made each new piece cheaper to add, because the briefs, the limits and the written procedures were already in place when it arrived.

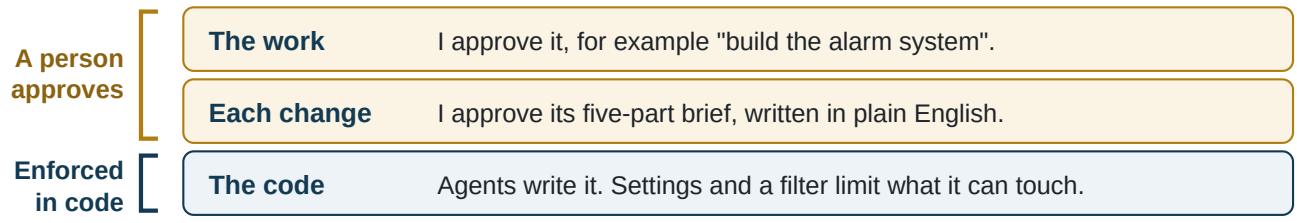
What I ruled out

Option I considered	Why I did not take it
Let the agents act freely	The agents can run commands on the computers that operate our cameras, our alarm and our network. I was not willing to give that access without limits, a written plan and a way to reverse each change.
Approve every code change line by line	I cannot evaluate most code changes, and approving what I cannot read would train me to click "yes" by reflex. I approve the work and its plain-English brief, and structure in the code limits what an agent can reach.
Keep the rules in the agents' instructions only	A capable model can argue its way around written instructions, even when it acts in good faith. The limits that matter most live in permission settings and a command filter outside the model.
Buy store-bought assistants and cloud services	They would cover many features, but the household's data would sit on a manufacturer's servers and a vendor could change or end the service. Running the home systems myself means nothing leaves the house and there is no subscription.
Use one AI provider for everything	One provider would be a single point of dependence. Bulk research goes to isolated OpenAI Codex workers, and a model on my own hardware takes over the messaging line when the main provider cannot be reached.
Have an AI write status reports on demand	It would cost model time on every view and could compose a status that nobody recorded. A plain script draws the dashboard from one project file instead.

The home layer is common
Home Assistant, Frigate and local speech models are well-known open-source tools, and many hobbyists run similar setups. What I gain by running them myself is ownership. Nothing leaves the house, there is no subscription and no vendor can switch it off.

The operating layer is the unusual part
Running household systems through AI agents under written rules, with approval steps, undo plans and a shared memory, is the part I could not buy. It is the subject of this report, and the twelve component projects all run under it.

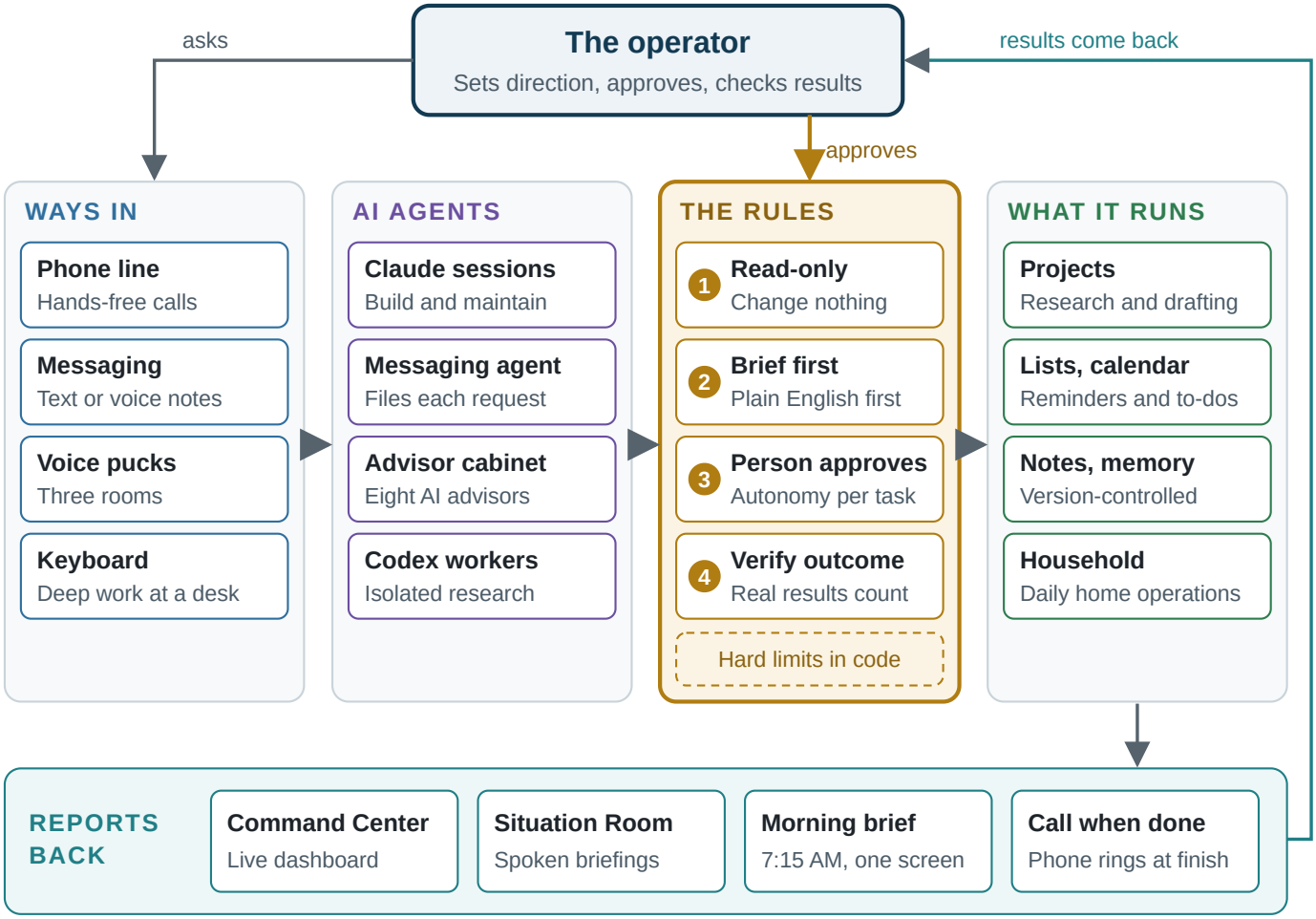
Where my approval sits



I approve at the two levels I can judge. At the level of code, structure does the limiting.

2. How Nero works

Nero has six parts, drawn below in the order a request travels. I sit at the top as the operator. A request comes in through one of four ways in, an AI agent takes it up, the rules decide what that agent may do, the work lands on a project or a household system and the result comes back to me.



The Nero operating model. A request moves left to right, passes the rules before anything changes and comes back to me as a report I can check.

Part	What it is
The operator	I set direction, approve changes and check results. My wife is the second operator, and each working session ends with a handoff note so the other can pick up the work.
Ways in	A phone line for hands-free calls, a Telegram line for text and voice notes, voice pucks in three rooms and keyboard sessions for deep work. The voice and text lines share one memory.
AI agents	Claude working sessions build and maintain each project. An always-on messaging agent files requests, a cabinet of eight AI advisors briefs me aloud and isolated OpenAI Codex workers take bulk research with no access to personal data.
Memory	Plain text notes in version control on one machine. Each note records the file a statement came from or marks it as an inference. Both operators work on that one copy, because a second copy would drift.
What it runs	My projects and research, our lists and calendar, the notes themselves and the household systems described in the component reports.

3. The rules and how they are enforced

Every agent works under one written set of rules, kept in version control with the rest of the system. A change to anything that runs follows the same six steps, and the agent cannot skip a step by deciding to.

1. Read
The agent reads logs, settings and notes. It can change none of them.

2. Brief
It writes a five-part brief in plain English and stops.

3. Approve
One of the two operators reads the brief and approves it or sends it back.

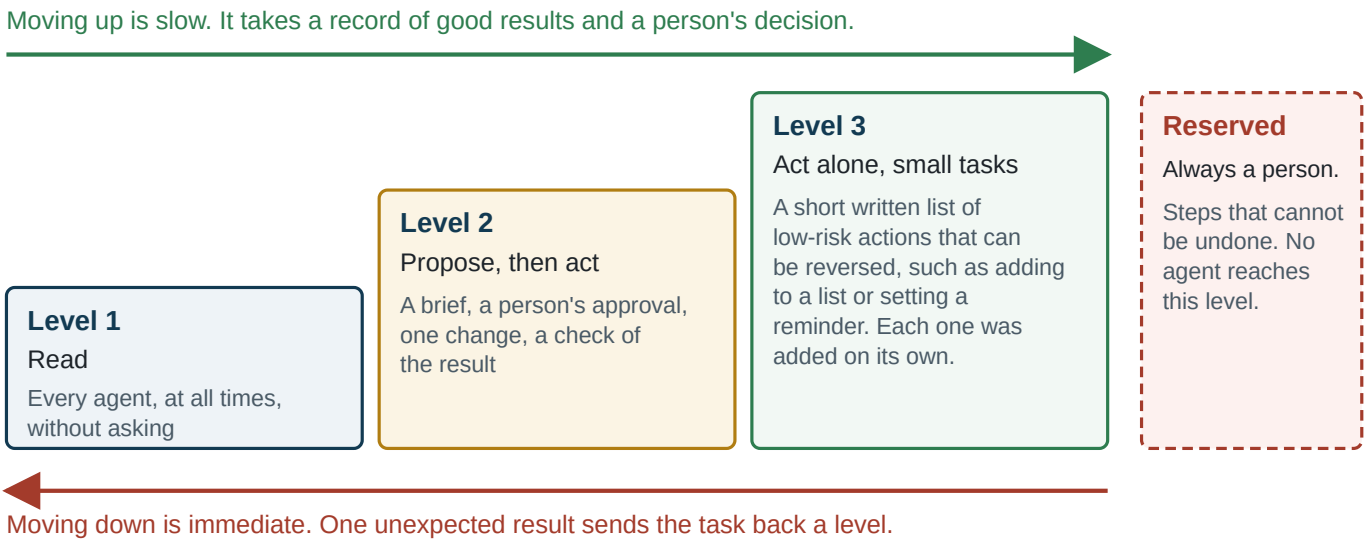
4. Back up
A backup is made first. Without one, the change does not run.

5. Change
The agent makes that one change and nothing else.

6. Verify
The result is checked in the world. The doorbell rings or the recording exists.

Brief part	What it must say
What	The change, in one sentence a non-engineer can read.
Why	The problem it solves, with evidence.
Risk	The worst realistic outcome.
Undo	The exact way to reverse it, prepared before the change runs.
Blast radius	What else is affected while it runs. A typical entry says the cameras will pause for thirty seconds.

Autonomy is granted one task type at a time



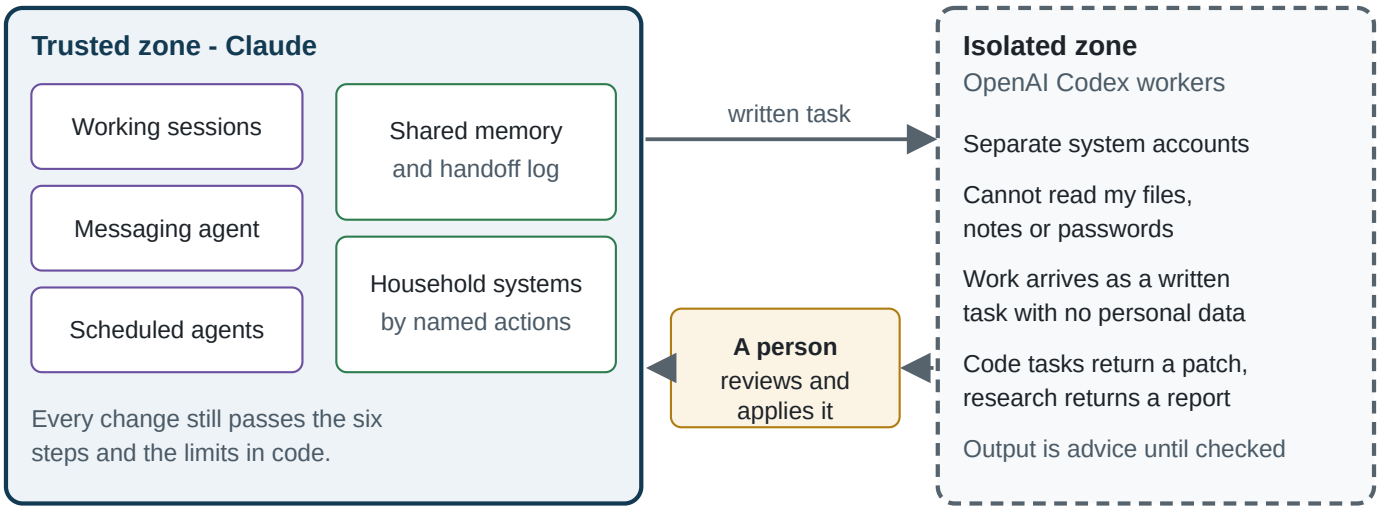
There is no general setting that makes an agent autonomous. Each type of task climbs on its own record, and the steps that cannot be undone stay with a person.

Safeguards

1. **Limits live in code.** Permission settings name the files and commands each agent may use, and a filter blocks destructive commands before they run. Since October 2026 an agent cannot rewrite those settings or the filter by command either, and that protection passed 19 of 19 tests before it went live.
2. **Unattended agents get a named list.** With nobody at a keyboard, the messaging agent can add to lists, set reminders, save notes, create calendar entries and send notifications. Anything larger waits for a keyboard session and an approved brief.
3. **Passwords stay out of the conversation.** No secret is written into a note, a commit or a chat. Agents refer to where a secret is stored, and the tool reads it when it runs.

4. Two kinds of AI worker, and the tests behind them

Nero uses AI from two providers and treats them differently. Claude agents do the trusted work and can read the shared memory. OpenAI Codex workers take bulk research and some code tasks in parallel, and they run in an isolated area where they cannot see my files, my notes or any password.



The second provider adds capacity without adding access. Nothing it produces enters the system until a person has reviewed it.

Tests I required before trusting each control

Control	What the test showed
The isolated workers	In July 2026 the isolation was tested with seven attempts to break out, such as reading a planted secret and reaching the network. All seven failed. Two failed because the model declined to try, and the record says so, so I do not count those two as proof of the walls.
The AI reviewer	Before I relied on a review agent, it checked five notes without being told they were a test. Four held planted errors and one was clean. It caught all four errors and passed the clean note.
The rules themselves	In July 2026 an AI reviewer in a fresh session spent a night looking for places where the rules fail. Each finding was checked before I changed any rule.
High-stakes proposals	A second agent reviews them with the claims and the data but without the reasoning behind them, because a reviewer who has read the argument tends to agree with it.
Loss of the main provider	A model on my own hardware takes over the messaging line, and it tells us that it is the backup.

How results reach me

The Command Center is drawn by a plain script from one project file, so the page shows exactly what someone recorded and no model composes a status. A one-screen morning brief arrives at 7:15 AM every day and has run since October 2026. It ends with three overdue items, and each one must be closed, rescheduled or dropped. An item I do not answer keeps asking, because silence never counts as a decision. The Situation Room gives me spoken briefings with time for questions, and the phone line calls me when a long task finishes.

5. The twelve component projects

Each project below has its own report. The second column places it in the operating model on page 3. "Started" is the month concept and design work began, and each report gives the date its project went live.

Project	Part	In one line	Started
Private Camera AI - Nothing Leaves the House	What it runs	Camera intelligence that runs on my own hardware	September 2025
Alarm Revival - AI That Sees, Describes and Confronts Intruders	What it runs	A dealer-locked alarm returned to service	October 2025
Nero's Constitution - Rules AI Agents Cannot Talk Their Way Past	The rules	The rules on page 4, with hard limits in code	November 2025
Agrippina - AI Voice and Personality of the House	Ways in	The house's voice and character, in three rooms	December 2025
Reactive Living Room - Doorbell, Speakers and TV	What it runs	Speakers and TV that respond to house events	January 2026
Situation Room - AI Advisors Brief Me Aloud	Reports back	Eight AI advisors brief me aloud and take questions	February 2026
Command Center - Every Project on One Live Screen	Reports back	Every project, its next step and a nightly check for stale entries	March 2026
Call Nero - Phone Line to AI Staff	Ways in	A phone number for hands-free work sessions	April 2026
Text the House - Always-On AI Aide	Ways in	A Telegram agent that files each request by a declared route	May 2026
Duck Laser - Camera-Triggered, Phone-Confirmed Wildlife Deterrent	What it runs	A laser deterrent that fires only on a person's go. Bench build passed in July 2026, mounting is next.	June 2026
Presence-Aware Home - Arms, Disarms and Adjusts on Its Own	What it runs	The house acts on who is home	July 2026
Deal Desk - AI Reads Every Store Ad	What it runs	Weekly store ads checked against 117 regular purchases	September 2026

6. Results and what is next

As of November 2026 Nero holds 1,406 recorded changes, 374 notes and 67 written procedures, and its dashboard tracks 50+ projects for two operators. It runs my household and my projects every day. A scheduled agent has made 55 commits on its own since July 2026, and a script limits it to the handoff log and the procedure files, so each of those commits is visible in the history and can be reversed.

Nero has limits I accept. It runs on one workstation, with backups but no standby machine. Most of its reasoning comes from commercial AI providers, and its pace is set by how many decisions I can make in a day.

Next, I am handing more of the reversible upkeep to scheduled agents, one task type at a time, so my time goes to the decisions that cannot be undone. The next stop for the Situation Room is the living-room TV.

7. What this shows about how I work

My career has been in partnerships and programs, where an organization relies on written rules for who may commit it to something and how results are checked. I wrote the same kind of rules for AI agents and ran them every day on systems where a mistake would be felt at home. I wrote the rules before I granted access, gave autonomy one task type at a time and accepted a result only when I could see it. That is the order of work I would bring to an organization adopting AI.