01 / Customer delivery and reliability

Restoring a telecom qualification response

Diagnosed a response-path regression during customer UAT, delivered corrected integration sequences and coordinated a successful retest within 13 hours of escalation.

Nokia · September 2026 · Java / WSO2 · REST APIs · AI-assisted diagnosis

12h 33m	Response restored	Reusable fix
Escalation to confirmed UAT retest	Synchronous qualification retest passed	Generic release packaging followed

The problem and my responsibility

At Nokia, I own integrations connecting telecom operators' CRM, ordering and sales systems to Nokia FlowOne, its orchestration, provisioning and activation suite, and network elements. During one customer's UAT rollout, a qualification API built its response but failed to send it. An internal data-service call had overwritten the outgoing HTTP status with a null value.

How I worked through it

- **Gathered evidence across the boundary.** Compared working and failing execution paths using logs, installed versions and live configuration supplied by a delivery colleague. An installation issue initially obscured the separate response regression.
- **Used AI assistance with active verification.** Used OpenCode for log and code analysis, challenged an interim explanation when the failure persisted, and restored the intended HTTP status at the response boundary in two qualification sequences. Both XML files passed syntax parsing.
- **Closed the loop with delivery.** Provided the corrected sequences and reload guidance, requested a retest and received confirmation that synchronous qualification worked. I subsequently incorporated the fix into generic release packaging.

Result

The colleague-confirmed retest established recovery of the reported UAT scenario. This example demonstrates fault isolation, ownership of a corrective change and coordination through verification.

Evidence: internal incident timeline and colleague-confirmed UAT retest. The elapsed interval includes overnight and communication gaps; it is not hands-on effort or production downtime.

02 / Released developer tooling

Making macOS authentication easier to manage

Created AuthCompanion, a Swift CLI coordinating Touch ID and Apple Watch approval for GPG signing and sudo through independently installable companion tools.

Personal / open source · Swift · macOS · GPG / PAM · Release engineering

AuthCompanion	pinentry-companion	pam-companion
Coordinates setup, restore and diagnostics	Handles GPG authentication integration	Handles sudo authentication integration

The problem and my ownership

Configuring authentication across GPG and sudo involves different components, configuration files and privilege boundaries. I created the AuthCompanion coordinator and extended and maintained the companion GPG/PAM forks, bringing them together through explicit machine-readable contracts.

Design decisions

- **Keep component responsibilities clear.** AuthCompanion composes the tools through versioned contracts. Each component remains independently installable, while the coordinator provides a consistent setup and diagnostic workflow.
- **Make system changes inspectable and reversible.** An unprivileged plan lets a developer inspect intended changes before privileged setup. Setup and restore operations, state inspection and actionable diagnostics make the lifecycle easier to reason about.
- **Treat packaging as part of the product.** Added stable JSON output, CI and release verification, with universal archives for Apple Silicon and Intel Macs. Initial release validation recorded 39 passing tests, strict-concurrency compilation and archive tamper rejection.

Result and public evidence

Released AuthCompanion, with v0.1.2 published in July 2026, and entered the project in OpenAI Build Week 2026. The public repository, release and initial implementation are available for review.

Repository · v0.1.2 release · Initial release PR · Build Week project

GPG companion · PAM companion

Release evidence: the linked initial PR records the test and packaging results. The public release and Build Week submission provide dated examples of the shipped work.

03 / AI workflows and upstream engineering

Making AI-assisted development repeatable

I designed and maintain a private development harness around OpenCode and Codex, complemented by merged TypeScript contributions to public LLM provider integrations.

Personal harness / open source · Model routing · Scoped tools · OAuth · TypeScript

Task and context	Agent execution	Verification
Reusable skills and explicit work boundaries	Task-based routing and scoped permissions	Tests, runtime evidence and behavioral evals

The private harness

I use coding agents to investigate unfamiliar systems and deliver software. I built a reusable environment that makes task scope, context and completion checks explicit, with OpenCode as the primary runtime and Codex as a compatible adapter.

- **Centralize shared behavior.** Keep portable instructions and skills separate from host-specific configuration, select models by task, and give delegated work a bounded deliverable and tool permissions.
- **Check behavior and continuity.** Combine deterministic configuration checks with behavioral evaluations for tool use, specialist routing, compaction and handoffs. The continuity cases check what an agent recalls after a context transition.

Public contributions: accepted upstream

- **Worker capacity and failure handling.** Bounded bridge workers, made capacity reusable before callbacks and returned retryable HTTP 503 responses at saturation. Cursor adapter PR #33, merged September 2026.
- **OAuth and plugin lifecycle.** Completed and hardened a V2 migration begun by another contributor, adding refresh deduplication, response correlation, concurrency tests and package validation. Anthropic plugin PR #211, merged September 2026.
- **Account-aware model routing.** Discovered available model and parameter combinations from the provider and routed exact selected variants while preserving account restrictions. Cursor adapter PR #11, merged July 2026.

Result and public evidence

The harness provides a repeatable workflow for my own development and troubleshooting. The three merged contributions provide separate, publicly inspectable examples of backend reliability and integration work in the AI tooling ecosystem.

Capacity: PR #33 · OAuth: PR #211 · Model routing: PR #11

Public evidence: the linked PRs were merged in July–September 2026. The harness repository remains private. These projects are separate from Nokia product work.