How a request becomes work

Who decides where a turn runs, what runs it, and how every model on every machine plugs in. AI Staff for Everyone.

> Clairvoyance is a desktop workspace that pairs people with AI Staff across design, planning, development, support, and operations. A person asks a Staff member for something. A headless Orchestrator records the turn and admits it. A local Concierge chooses the model and the machine. A runtime carries out the work under one shared set of context, tools, and policy. Local models run through Locama, a single inference host. What the user sees is a view over those events. A closed window does not stop the work.

1. Product shape

Clairvoyance runs on the user's computer so Staff can reach the files that already live there. Each Staff member has a name, a role, and memory. They work in parallel, hand work to one another, and can keep going on a schedule. They connect to the coding agents already installed on the machine, to Hosted AI, and to models that run on local silicon.

The work product is durable. Staff write Notes and Reports, draw Canvases, publish interactive Exhibits, and keep structured Binders. Those artifacts stay in the workspace. The same views can render in a desktop window or in a remote browser.

Three components sit at the center of the architecture:

- **Orchestrator.** Owns coordination in a dedicated, headless process: turns, Staff runtimes, missions, peer messaging, schedules, and side effects.
- **Concierge.** Routes every turn to the right model and machine. A small local classifier describes the request. The application decides the route.
- **Locama.** The single local inference host. Every GPU, NPU, and extra machine plugs in as a hardware guest.

2. Six planes

COORDINATION
Orchestrator
One owner for turns, runtimes, missions, peer messages, schedules, and side effects. Runs headless.

INTELLIGENCE
Concierge
A local classifier describes the request. The host derives a capability level and picks the route.

MODELS

Locama and guests

One inference host. GPUs, NPUs, Apple silicon, the user's other machines, and Hivemind peers plug in as guests.

EXECUTION

Runtimes and Meta Harness

Cloud coding agents and Clairvoyance's own agent loop sit behind one interface, shaped by one context, tool, and policy plane.

EXPERIENCE

Views over events

Staff, tabs, Modules, and artifacts render Orchestrator events and survive reloads.

FOUNDATION

Platform and cloud

A thin desktop host plus identity, sync, relay, a model catalog, and billing.

3. How a request flows

One turn, end to end:

1. A person asks a Staff member for something. The view sends the turn to the Orchestrator and receives a persisted receipt.
2. The Orchestrator logs the turn, admits it against runtime, silicon, spend, and entitlement budgets, and queues it by priority.
3. The Concierge classifies the request locally, derives a capability level, and picks a route: model, runtime, and machine.
4. The Meta Harness composes context, memory, and the tool grant for that level, and applies policy.
5. The runtime supervisor dispatches to the chosen runtime: an ACP coding agent, or Clairvoyance's Core Harness.
6. If the route is local, Locama runs the model on the best guest for this machine: a GPU, an NPU, or another desktop in the user's fleet.
7. Tool calls that change the world pass the Effects broker, which applies them once and attributes them.
8. Projections stream every event to windows, remote browsers, and notifications. A window can reload at any time and resubscribe.

4. Orchestrator

The Orchestrator is the single owner of coordination. It runs headless in its own process. Windows, remote viewers, and agents are clients. They send commands and subscribe to events. Schedules, triggers, and missions keep running with no window open.

Four guarantees

- **Headless.** Work continues when every window is closed.

- **Durable first.** Every command is in the event log before it is acknowledged. A receipt means the command is recorded. Execution is observed through events.
- **Exactly once.** Side effects carry an idempotency key and apply once, so retry and recovery cannot double-write a file or send a message twice.
- **Views are projections.** A window reload resubscribes from a cursor and loses nothing.

Inside the process

Clients speak a two-verb contract: submit a command, subscribe to events. Inside, a command API validates and assigns an id. Admission checks four budgets (runtime concurrency, silicon slots, spend, and entitlement) and places the turn on a priority queue. A runtime supervisor starts, dispatches, and restarts agent runtimes. A work engine runs missions, plans, and loops. A peer bus delivers inbox items with receipts. Recruitment owns the roster of hired and recruited Staff. A time service owns schedules, triggers, and timers. An Effects broker applies side effects once.

The event log is append-only, with a single writer per conversation. It is the source of truth. Conversation, roster, mission, inbox, and audit views are projections served by cursor. After a crash, recovery replays, resumes, and restores from that log.

Priority

CLASS	SOURCE	BEHAVIOR
Interactive	A person waiting in a conversation	Dispatched first. May preempt deferred work on shared guests.
Delegated	Peer instructions and mission tasks	Fair-shared across Staff, with bounded fan-out per lead.
Deferred	Schedules, triggers, overnight work	Runs on idle silicon. Prefers local and fleet guests. Resumable.
Guest	Hivemind requests from other people	Capped share of one slot. Never displaces the owner's work.

Spend exhaustion defers work rather than forcing a more expensive model. Entitlements and data-boundary rules are checked once at admission.

5. Concierge routing

The Concierge decides where each turn runs. A small local model describes the request. The application turns that description into a capability level and picks the best route across this machine, the user's other machines, and the cloud.

Three rules never break:

- The host owns routing. No model is asked to judge its own adequacy.

- No hosted model sees a message in order to decide where it belongs. The classifier is local, or it does not exist.
- Uncertainty resolves upward into the middle of the ladder and stops there. Writing an email does not consume a frontier model.

Four inputs shape the route: how capable the work needs to be, which tags it requires (vision, web, code, data boundary), whether the person is waiting or the work can wait until overnight, and which machine should run it.

The capability ladder

Levels are defined by tool surface, which makes membership a checkable fact. Each model's level is curated in the model catalog.

LEVEL	NAME	TOOL SURFACE
1	Incidental	Chat and relay only
2	Basic	Read-only tools
3	Standard	Structured writes plus Clairvoyance tools
4	Technical	Reads technical sources, writes prose
5	Connected	Full global and workspace tools
6	Advanced	Writes and runs code
7	Beyond	Planning and decomposition

The highest two levels require an affirmative tag. A tool request outside the grant is a deterministic escalation signal. High-level work that can wait is inverted: it goes to local or fleet silicon overnight instead of a short burst on a frontier model.

No model session owns the transcript. The conversation lives in the Orchestrator's event log and is projected into whichever runtime handles the next turn. Switching models mid-conversation is invisible to the user.

Pipeline

Deterministic rules run first (pins, slash commands, referenced artifacts). A resident local router then emits a grammar-constrained requirement vector. The host maps tags to a tool surface and level, filters the catalog by level, tags, boundary, and home, and hands a route to the Orchestrator with a fallback ladder.

6. Locama and guests

Locama is the local inference host. Every accelerator plugs in as a guest behind a single API, a single model store, and a single scheduler. The user's other machines and Hivemind peers plug in the same

way.

Three kinds of guest:

- **In-process backends** loaded into the Locama server (CUDA, ROCm, Vulkan, SYCL, Metal, CPU, and related stacks). Lowest overhead. They share the model store and KV cache directly.
- **Sidecars** with their own model formats and drivers, supervised and restarted by the Guest Manager. Examples include a Qualcomm NPU through GenieX, Intel OpenVINO, Apple MLX, and AMD Ryzen AI.
- **Remote guests** on other machines, over an authenticated peer channel with bounded payloads: the user's fleet, and Hivemind peers.

A guest describes its silicon, accepts a placement, and streams normalized tokens. Locama owns everything else: scheduling, the model store, and Concierge eligibility. App code does not branch on vendor.

The Guest Manager probes the machine, benchmarks each candidate, and places resident workloads. A tiny router classifier can stay resident on an NPU. Interactive chat uses a local GPU when it is free. Embeddings can run on CPU and pause on battery. Level 6 and above can move to a fleet desktop or to the cloud by policy.

Interactive turns preempt deferred and Hivemind work. Preempted work resumes from its checkpoint. On battery, deferred work pauses and NPU guests are preferred. A failed sidecar leaves the Concierge candidate set within one heartbeat. Hivemind requests never displace the owner's work.

7. Agent runtimes and the Meta Harness

Two kinds of agent loop run side by side behind one runtime interface.

- **ACP runtimes** host the cloud coding agents that bring their own loop: Claude Code, Codex, Gemini, Copilot, Cursor, and Grok.
- **The Core Harness** is Clairvoyance's own loop for Hosted AI, local models Locama can place, and bring-your-own OpenAI-compatible endpoints.

Each runtime declares how it receives composed system context. A runtime that cannot accept instructions is not eligible as a Staff runtime, so no Staff member runs without its Knowledge Base.

The Meta Harness shapes both identically:

- **Context composer.** Knowledge Base, persona, workspace library, and applicable memory are rendered once and delivered through each runtime's declared channel.
- **Tool plane.** Clairvoyance tools, Module tools, and workspace tools, filtered to the grant for the turn's level. Calls go to the Orchestrator.
- **Policy.** Permissions, server lists, credentials, and the data boundary are enforced at spawn and on every request, in code, at the boundary.
- **Memory.** Applicability is checked before the turn. Distillation is written after it. Both run as Orchestrator work items on local models.

8. Experience: views over events

Everything the user sees is a view over Orchestrator events and workspace content. Windows can close, reload, or crash without losing work. The same views render in a remote browser.

- **Staff.** One conversation surface for every runtime, subscribed to turn events. Panels display work. The Orchestrator coordinates it.
- **App shell.** Tab types register a file matcher and a view, the same way for built-in editors and Modules.
- **Modules.** A Module can contribute tabs, sidebars, Home Desk tiles, AI tools, and Exhibit-style apps through one extension format.
- **Artifacts.** Binders, Exhibits, Canvases, Notes, and Reports are durable content with capability-gated brokers. Agent writes arrive through the Effects broker, so every change is attributed, idempotent, and undoable.

A Binder is a self-contained collection of records and assets that can be viewed many ways. An Exhibit is a persistent interactive page. Notes and Reports are Markdown documents. Canvases are visual boards. These are the lasting objects. Tools come to the data.

9. Platform, cloud, and security

The desktop host is thin: windows, protocols, and a typed IPC router. Coordination, tools, and inference each run in their own supervised process. Views, Modules, and Exhibits run with capability grants. A native core holds graph, index, and desktop-control work.

Cloud services sit beside the desktop, chosen for a reason the Concierge can state:

- Identity: accounts, devices, and entitlements.
- Content: journaled personal sync with conflict-aware merges, cloud-authoritative domain notes, and share links.
- Presence: a relay gateway, multiplayer rooms, fleet inventory for the Concierge, and Hivemind signaling.
- Models: the hosted catalog, the single source of Concierge candidates.
- Commerce: subscriptions, with entitlements checked at Orchestrator admission.
- Web: app, viewer, and gallery builds that render the same views over the same events.

Security is enforced at process and API boundaries. The Orchestrator accepts commands only from authenticated channels. Tool servers run one worker per session behind authenticated loopback, with server lists sealed by the host. Guests run with least privilege. Remote guests use authenticated, payload-bounded channels. Modules and Exhibits receive capability grants per id. Cloud APIs use row-level security first, with service-role access only in audited, named code paths. Restricted data stays on the user's devices, even when that means deferring the work.

10. What this adds up to

The Orchestrator coordinates every turn, durably and headless. The Concierge routes each turn to the right model and machine. Locama runs local models on any silicon through pluggable guests. The Meta Harness gives every runtime the same context, tools, and policy. Every window is a view. Nothing is lost when one closes.

That is how Clairvoyance turns a request into work on the machine where the files already are.

About Clairvoyance

Clairvoyance is developed by Stardock Software, a desktop software company with three decades of shipping products. It ships on Windows, macOS, Linux, and the web. The core app is free. Founder and CEO: Brad Wardell. Tagline: AI Staff for Everyone.

Press: press@stardock.com Site: https://www.clairvoyanceai.com