

Personal OS Wiki Integration Plan

Proposed workflow design for review

Prepared by Hermes · 2026-09-03
Source: existing Reyna family brain and workflow

Contents

1. Executive Recommendation	4
Recommendation	4
Decision boundaries	5
Adoption decision	5
2. Inspected Current State and Planning Assumptions	5
Existing workflow facts	5
Upstream project facts used as ideas, not instructions	6
Important assumptions to validate before implementation	6
3. Target Model: Two Complementary Stores, Existing Authorities Preserved	6
3.1 The core split	6
3.2 Authority matrix	7
3.3 Proposed minimal storage topology	7
4. Concrete Workflow and State Mapping	7
4.1 Intake classification	7
4.2 Proposed canonical fields	8
Shared work-item fields	8
Task fields	9
Claim, run, and review fields	9
4.3 Task lifecycle and lease policy	9
4.4 Definition of Done templates	10
5. Context Packets and Retrieval	10
5.1 Context packet contract	10
5.2 Retrieval strategy: optional semantic layer	11
6. Notification and Apple Reminders Adapter Plan	11
Adapter boundary	11

Initial adapter order	11
Adapter requirements	11
7. Agent Safety, Review, Privacy, and Failure Handling	12
7.1 Claim eligibility and agent identity	12
7.2 Review gates	12
7.3 Evidence rules	12
7.4 Failure policy	13
8. Phased Adoption Roadmap	13
Phase 0: Governance, authority, and pilot selection	13
Phase 1: Local work-state contract and shadow ledger prototype	13
Phase 2: Hermes and Herdr read-only context packets	14
Phase 3: Controlled low-risk claims, review workflow, and Apple Reminders dry run	15
Phase 4: Project-specific integration for factory, infrastructure, and ministry	15
Phase 5: Optional operational hardening and retrieval evaluation	16
9. Migration and Rollout Strategy	16
Non-disruption principles	16
Suggested rollout cohort	17
Legacy linking protocol	17
10. Measures of Success and Operating Metrics	17
11. Explicit Open Decisions for User Review	17
12. Ideas Deliberately Rejected or Deferred	19
13. Final Review Checklist	19
Source Notes	20

For Hermes: Treat this as a design and review document. Do not implement any phase until the user approves the relevant decision gates. When implementation is approved, use the subagent-driven-development workflow task-by-task for code-bearing work.

Goal: Add an explicit, auditable work-state layer to the existing Reyna family workflow while preserving the Markdown brain, existing project/factory records, Git/Gitea, Hermes, Herdr, and Apple-facing tools as their current authorities.

Architecture: Adopt the useful boundary from lawyer112/personal-os-wiki: durable Markdown knowledge is not the same thing as current execution state. Start with a small local structured state ledger plus Markdown evidence links, rather than deploying the upstream Next.js/Postgres/Wiki stack. Hermes and Herdr become controlled clients of that ledger; existing tools remain the places that execute work, store code, publish, send, or notify.

Tech Stack (proposed, incremental): Existing ~/brain Markdown vault and Gitea backup; Hermes Agent and existing profiles/cron; Herdr as human control surface; repository-local GitHub/Gitea workflows; Reyna CLI native privacy host for Apple adapters; initially a local SQLite/JSON export work-state component only if Phase 1 is approved; optional read-only search/retrieval index later.

1. Executive Recommendation

Recommendation

Adopt the **operating ideas**, not the upstream application:

- 1 Separate **work state** from long-term memory and durable notes.
- 2 Make every executable task have a concrete `nextAction`, `definitionOfDone`, `owner/lease` policy, evidence requirements, and an explicit review gate.
- 3 Use a narrowly scoped structured execution ledger for claims, heartbeats, status transitions, review decisions, and notification delivery records.
- 4 Keep durable narrative, evidence explanations, decisions, and human-readable handoffs in the existing Markdown brain.
- 5 Treat Apple Reminders, WhatsApp, desktop notifications, cron, Herdr, Paseo, GitHub, and Gitea as adapters or execution surfaces—not a competing task truth.
- 6 Keep retrieval keyword/metadata/link-first. Evaluate semantic/vector retrieval only after a real recall benchmark demonstrates a gap.

The first useful release should be a **small, local, reversible overlay**. It should not introduce a public service, Docker stack, PostgreSQL instance, duplicate Wiki, migration of existing notes, automatic task generation, or autonomous side effects.

Decision boundaries

Decision	Recommended boundary
Durable knowledge	Existing Markdown brain remains authoritative. <code>areas/platform.md</code> remains the technical tracker; <code>areas/operations/services_inventory.md</code> remains the human-facing service catalog; <code>project/factory</code> documents remain cross-server project records.
Work state	A proposed local structured ledger owns only live execution state: task lifecycle, claim leases, heartbeats, review status, notification delivery markers, and links to evidence. It is not a replacement wiki or personal memory store.
Stable user context	Hermes persistent memory remains for durable preferences, safety constraints, and stable environment facts—not unfinished work, task status, leases, or approvals.
Code and change evidence	Repository Git history, PRs/issues where used, tests, builds, and source <code>AGENTS.md</code> remain authoritative for software changes. The ledger links to this evidence; it never substitutes for it.
Cross-server project management	<code>projects/FACTORY_OPERATIONS.md</code> and <code>projects/factory/<project-key>.md</code> remain authoritative. Paseo/Herdr are execution and control views, not a project-record replacement.
Notification delivery	Apple Reminders, WhatsApp, desktop notifications, and future adapters only deliver or mirror a nudge. Completing a reminder must never close a work item.
High-impact actions	Human approval remains mandatory for merge, deploy, restart, publishing, external send, credential changes, production configuration, TCC/privacy changes, destructive operations, and migrations. A state transition or agent lease does not authorize these effects.

Adoption decision

Proceed through Phase 0 and Phase 1 only if the user approves the proposed authority model and the initial state vocabulary. Do not evaluate a separately hosted dashboard/service until the local overlay has been used successfully for a limited pilot.

2. Inspected Current State and Planning Assumptions

This plan is grounded in read-only inspection performed on 2026-09-03. Assertions labeled “needs verification” are intentionally not treated as live facts.

Existing workflow facts

- 1 The brain is an active PARA-style Markdown vault at `/Users/adolforeyna/brain`. Its index explicitly separates Projects, Areas, Resources, Archives, and Inbox. `brain/rules.md` directs uncertain material to `brain/inbox/`, requires YAML frontmatter for new notes, and uses WikiLinks.
- 2 The brain's service inventory calls the brain Markdown source of truth for family, projects, operations, EMI, maintenance, finances, media, and journals. It marks `areas/platform.md` as the canonical technical tracker.
- 3 `areas/platform.md` identifies the Mac mini private-host layer, Hermes, the Reyna CLI native privacy host, local services, existing cron, and Gitea-backed projects. It records a deliberate migration from MacMiniMCP to Reyna CLI for privacy-sensitive Apple operations. It also says remaining native Reminders coverage must be verified before relying on it.
- 4 `projects/FACTORY_OPERATIONS.md` already has the strongest applicable safety model: project record first; task record with acceptance criteria, allowed paths/commands, required checks, forbidden effects, evidence, and approval gate; isolated worktrees; deterministic verification; independent review; and explicit human gates for consequential actions.
- 5 `projects/paseo_software_factory_working_model_2026-08-17.md` already treats “our projects” as the authoritative unit of work, Paseo as an execution layer, and test/review evidence as what moves work forward. It explicitly says a Paseo status event may create a review/notification request but never grants production authority.

- 6 The current brain has an Inbox convention but no inspected, canonical structured task/claim/heartbeat ledger. Tasks also appear as Markdown checkboxes in area and journal notes; that is useful human-visible context but cannot safely represent concurrent agent leases, atomic claims, or review history by itself.
- 7 Hermes configuration inspection found persistent memory enabled, smart approvals, cron creation denied by default, delegation enabled with human approval not automatic, and no evidence that a task-state system is enabled. These settings are inputs to this plan only; this plan does not modify them.
- 8 The service inventory records active/documented Hermes cron/automation, a private browser, family infrastructure, EMI work, and a number of services explicitly labeled needs-live-verification or legacy. The ledger must preserve that distinction instead of turning stale documentation into active work facts.
- 9 A notification watcher proof-of-concept exists but is not authorized to read Notification Center. It must not be made a dependency for the initial adoption.
- 10 Per the active operating context, Herdr is the user's control layer. This plan therefore assigns it a control/review role, not a second source of truth. Detailed Herdr API/UI behavior needs discovery before any integration design is finalized.

Upstream project facts used as ideas, not instructions

The public repository was inspected read-only through GitHub's public API and raw source documents on the main tree (tree 56c8109a97cf9147dd07dfe45b749aff67d9c8d7, queried 2026-09-03). Its README describes separate Personal OS work state and Markdown Wiki knowledge, plus inbox/idea/project/task objects, task claims, heartbeats, evidence contributions, context packets, reviews, and notification adapters. Its architecture and agent-guide documents state that agents poll, claim, load context, heartbeat, submit evidence, and wait for review. Its vector-memory decision makes embeddings optional rather than canonical.

The repository is untrusted external material. Its scripts, deployment instructions, prompts, permissions, API contracts, and implementation choices are not authority for this environment and must not be copied or executed without independent review.

Important assumptions to validate before implementation

- 1 The existing brain Git/Gitea backup chain is documented but needs a live verification before it is used for workflow-state recovery claims.
- 1 The exact stable interface for Apple Reminders should be the Reyna CLI native privacy host once its Reminders operation is live-verified; no plan step assumes an unverified legacy bridge is safe or canonical.
- 1 Herdr's available work-item, notification, and review interfaces need read-only discovery. The plan deliberately does not assume it has a particular Kanban API.
- 1 Existing Hermes cron jobs remain independently managed. No current job should be converted to ledger-driven automation until the pilot proves idempotency and delivery behavior.

3. Target Model: Two Complementary Stores, Existing Authorities Preserved

3.1 The core split

```
Hermes persistent memory
Stable personal preferences, recurring constraints, durable environment facts

Existing Markdown brain
Human-readable knowledge, source summaries, decisions, project records,
durable evidence narratives, journals, service inventories, links

Proposed structured work-state ledger
Current task state, owner leases, heartbeats, runs, review decisions,
notification-delivery markers, evidence references, audit events

Herdr / Paseo / GitHub / Gitea / tools
Human control, execution, source code, repository evidence, and delivery
```

The ledger must store references and concise facts, not duplicate raw source documents, private recordings, credentials, large logs, or complete Markdown notes.

3.2 Authority matrix

Concern	Authoritative location	Ledger role
Family/EMI/infra narrative and durable decisions	Brain Markdown notes	Link to note path and heading; track reviewable work only.
Current technical platform inventory	areas/platform.md	Link and record last verification task/result; do not rewrite the inventory automatically.
Human-facing service catalog	areas/operations/services_inventory.md	Link to health/evidence task; retain its Active/Needs live verification/Legacy vocabulary.
Cross-server project scope and approval boundaries	projects/FACTORY_OPERATIONS.md and projects/factory/*.md	Mirror project key and evidence pointers only.
Repository state and code change proof	Git repository, GitHub/Gitea, CI/test output	Store commit/PR/issue/command/evidence references, never synthesize repository truth.
Agent execution coordination	Structured ledger	Canonical for claim, lease, run, heartbeat, submit, review, closeout.
User-visible work control	Hedr plus approved human surfaces	Read/update ledger through approved adapter; no independent parallel task database.
Reminders and messaging	Apple Reminders / WhatsApp / desktop / approved adapters	Delivery-only mirror with idempotency key; never a source of task completion.

3.3 Proposed minimal storage topology

Phase 1 should be designed around a **local state ledger**, not a service stack:

- 1 Primary runtime database: a local, owner-only SQLite ledger under an Application Support location, outside the Markdown vault and outside Git working trees.
- 1 Backup/recovery: transaction-safe backups plus compact, redacted export snapshots that can be reviewed and linked from the brain. The exact backup destination must be approved after the existing Gitea/brain backup chain is live-verified.
- 1 Evidence: Markdown remains where explanations belong. Ledger records point to brain paths/headings, Git refs, test command outcomes, file artifacts, URLs, and review decisions.
- 1 Access: begin with one local coordinator/adaptor on the Mac mini. Do not expose a LAN or public HTTP API in the pilot. Any later multi-host adapter requires separate authentication, authorization, secret storage, rate limits, and audit design.

This gives atomic claims and expirations without deploying upstream Postgres, Docker, Next.js, a second Markdown Wiki, or a public dashboard.

4. Concrete Workflow and State Mapping

4.1 Intake classification

Every inbound item is classified before action; one input may yield multiple linked records.

Input category	Create or update	Example	What remains authoritative
Inbox	Raw trace, source metadata, triage state	a voice transcript, link, email summary, reMarkable capture, or user message	Original source and/or brain Inbox note.
Idea	Non-committal opportunity needing shaping	"Could a Kindle dashboard show review queue?"	Brain idea/project note; ledger holds only its lifecycle.
Project	Bounded outcome/workstream with existing project key	emi, reyna-cli, tactility, family-voice, remarkable-hermes	Factory project definition and related brain notes.

Input category	Create or update	Example	What remains authoritative
Task	Smallest executable, reviewable unit	"Run read-only discovery of the reMarkable watcher health path"	Ledger for live state; brain/repository for scope and evidence.
Evidence	Durable proof or result	test output summary, screenshot path, live health result, source note section	Markdown/repository/original system; ledger stores reference and verification metadata.
Notification	A nudge derived from current state	morning Apple Reminder or private WhatsApp summary	Adapter destination; ledger stores delivery marker only.

Rules:

- 1 An Inbox item is not automatically a task.
- 1 An Idea is not automatically promoted; it can remain captured, shaping, someday, Or archived.
- 1 A Project is not a task queue. It groups tasks, decisions, risks, and evidence.
- 1 A Task must be independently executable and reviewable; vague "advance project" records are rejected at intake.
- 1 The first structured state change never edits the original source. It adds linked work state only.

4.2 Proposed canonical fields

The exact schema is an implementation decision, but the following fields are the proposed minimum contract. Stable IDs should be opaque and immutable; human-readable titles and paths are not IDs.

Shared work-item fields

Field	Meaning
id, kind, createdAt, updatedAt	Immutable identity and record type (inbox, idea, project, task, notification).
title, summary	Human-readable, redacted description.
projectKey	Existing factory/brain project key when applicable; no invented duplicate project registry.
domain	family, home-infra, emi, coding, tactility, remarkable, voice, personal, or explicit future value.
sourceRefs	Original message/file/note/repository references with provenance and access classification.
brainRefs	Canonical Markdown paths and headings; never copied note bodies by default.
tags	Retrieval and routing labels, not access control.
sensitivity	normal, private, restricted, or highly-restricted; controls what may enter exports, notifications, and reviewer packets.
createdBy, lastChangedBy	user, named human, Hermes profile, Herdr adapter, or named worker.

Task fields

Field	Meaning
status	draft, ready, claimed, active, waiting, blocked, submitted, accepted, closed, or cancelled. submitted is distinct from accepted.
priority	P0 through P3; priority does not override approval policy.
nextAction	One specific physical/digital next action, with an accountable actor.
definitionOfDone	Observable conditions that a reviewer can check.
requiredEvidence	Required proof types and references: e.g., command exit/result, screenshot, reviewed Markdown note, human confirmation, or target-system readback.
executionMode	human_only, agent_suggest, agent_prepare, agent_execute_low_risk, approval_required, or blocked_until_user.
riskLevel	low, medium, high, or restricted; drives claim eligibility and gate requirements.
allowedEffects / forbiddenEffects	Explicit scope from the project/factory task record.
reviewGate	none, peer, human, human_before_effect, or human_before_and_after_effect.
dueKind, dueAt	Optional soft, hard, or none; due date does not cause autonomous execution.
contextVersion	Incremented when scope/policy/DoD changes; invalidates a stale claim.

Claim, run, and review fields

Object	Required fields and rule
Claim	claimId, taskId, agentId, leasedAt, leaseUntil, lastHeartbeatAt, state, policySnapshot. Only one active claim per task.
TaskRun	runId, taskId, claimId, startedAt, endedAt, status, resultSummary, contextVersion. A retry is a new run, not overwritten history.
EvidenceRef	type, location, description, capturedAt, capturedBy, verificationMethod, redactionState. It points outward; it does not embed secret/raw data.
Review	reviewId, taskId, reviewer, decision, comment, reviewedAt, evidenceRefs, nextState. Worker and reviewer must be distinct for medium/high-risk work.
AuditEvent	Append-only timestamped state change with actor, previous/new state, reason, and correlation ID.
NotificationDelivery	adapter, destinationClass, idempotencyKey, payloadClass, attemptedAt, deliveredAt, result. No reminder completion may alter task status.

4.3 Task lifecycle and lease policy

```

Draft -> Ready -> Claimed -> Active -> Submitted -> Accepted -> Closed
|   |   |   |
v   v   v   v
Waiting Blocked Changes Cancelled/Archived
requested

```

- 1 Only ready tasks with a complete nextAction, definitionOfDone, risk policy, and required evidence can be claimed.
- 2 A worker reads a current context packet before claiming or immediately after successful claim; it must stop if the packet says policy is stale or incomplete.
- 3 A claim has a bounded lease. Heartbeats extend it only while the task policy and context version remain valid.

- 4 Expired claims become ready only after the ledger records an expiry event. No second worker may assume the first worker's unfinished side effects were safe or absent.
- 5 Policy/scope/DoD changes revoke an active claim, preserve the existing run history, and require a fresh context packet and claim.
- 6 A worker submits evidence; it does not self-accept its own medium/high-risk work. Acceptance moves the task to accepted; closeout is only after linked durable records are updated or explicitly declared unnecessary.
- 7 waiting and blocked must include the external dependency, owner if known, last verified time, and smallest release condition.

4.4 Definition of Done templates

Work class	Minimum definitionOfDone
Research/plan	Source references are recorded; recommendation and uncertainty are clear; user decision is identified; no unapproved implementation occurred.
Brain/document edit	Required Markdown note is created/updated with frontmatter and wikilinks; source preserved where required; links resolve; reviewer confirms intended scope.
Repository change	Isolated worktree/branch is named; required tests/builds run with recorded exit outcomes; independent review passed; no merge/push/deploy without an explicit gate.
Family infrastructure	Target identity was verified; approved change/observation and readback are recorded; health check/rollback evidence exists; service inventory update is reviewed.
EMI/publication	Required ministry/brand/legal review is recorded; target preview verified; explicit final send/publish approval received; published state read back.
Reminder/nudge	Destination/list/channel was verified; idempotency marker recorded; delivery result recorded. This is never the DoD of the underlying task.

5. Context Packets and Retrieval

5.1 Context packet contract

Before a worker acts, it receives a bounded, assembled packet rather than searching the entire brain or relying on chat memory.

Required packet sections:

- 1 **Identity:** task ID, title, project key/domain, context version, claim/lease status.
- 2 **Execution contract:** next action, DoD, required evidence, risk level, execution mode, allowed/forbidden effects, approval gate, stop conditions.
- 3 **Relevant durable context:** selected brain note paths/headings, project-definition pointers, source references, previous decisions, and known stale/uncertain facts.
- 4 **Current execution state:** related tasks, prior runs, active blockers, current reviewer comments, notification history when relevant.
- 5 **Evidence plan:** exact target readbacks/tests/outputs expected; no vague “verify it works.”
- 6 **Retrieval provenance:** query terms, source selection rationale, unavailable/partial sources, and any access restrictions.
- 7 **Privacy boundary:** sensitivity label and data that must not enter output, prompts, logs, notifications, or exports.

Packet design rules:

- 1 Default to narrow, cited retrieval from the brain, project record, and source system.
- 1 Do not scrape or inject the entire vault, private raw audio, credentials, message archives, or notification database.

- 1 State whether retrieval is complete, partial, unavailable, or not-needed; absence of a result is not proof that no knowledge exists.
- 1 Expire/rebuild a packet after material source or policy changes.

5.2 Retrieval strategy: optional semantic layer

Adopt keyword search, paths, tags, WikiLinks, project keys, task IDs, and recent verified evidence first. This aligns with the existing brain/MCP plan, which already proposes Markdown as source of truth and any indexes/databases as rebuildable cache.

Do not require embeddings/vector memory in any early phase. Consider hybrid retrieval only when all are true:

- 1 at least ten real, privacy-safe retrieval questions show repeated keyword/path misses;
- 1 a benchmark compares keyword/metadata retrieval to hybrid retrieval with cited sources;
- 1 results improve recall without degrading provenance, latency, or privacy;
- 1 costs and failure fallback are documented;
- 1 keyword/metadata retrieval still works if the embedding provider is unavailable.

Vector results, if ever added, are candidates—not authority. Every result must resolve to a source note, task, artifact, or original system record.

6. Notification and Apple Reminders Adapter Plan

Adapter boundary

Ledger decides that a nudge is due and creates a delivery request.
Hermes/Herdr/scheduler decides whether and when to run an adapter.
Reyna CLI or another approved adapter writes to the destination.
The destination confirms delivery only; it never decides task truth.

Initial adapter order

- 1 **No new adapters in Phase 0–1.** Continue existing manually authorized workflows and existing cron behavior unchanged.
- 2 **Read-only notification candidate generation in Phase 2.** Produce reviewable proposed payloads only; no sending, task creation, or reminder writes.
- 3 **Apple Reminders pilot in Phase 3, only after Reyna CLI Reminders operations and list identity are live-verified.** Use the native privacy-host route described as canonical in `areas/platform.md`, not a new unreviewed Apple automation path.
- 4 **Existing WhatsApp/Hermes delivery in Phase 4.** It remains explicit, privacy-scoped, and idempotent. Use a private DM by default for operational/email-derived content; honor the brain rule that email summaries/notifications do not go to the Family Group unless explicitly requested.
- 5 **Herdr notification/review adapter only after read-only interface discovery.** Herdr can surface state and approval requests but must not silently turn a status change into a deployment, merge, send, or config change.

Adapter requirements

- 1 Every delivery request has a stable `idempotencyKey`, destination class, sensitivity classification, and expiration time.
- 1 The adapter verifies the configured Apple list/channel/contact before its first write. Duplicate or missing destination identities are a hard stop, not an invitation to guess.
- 1 Reminder content contains no secrets, bearer tokens, private vault paths, raw recordings, private server inventory, or unnecessary family/medical/ministry detail.
- 1 The adapter records a delivery result or failure, but it cannot mark a task accepted/closed.
- 1 Completing an Apple Reminder means only that the nudge was handled. A task closes only through its DoD and review path.

- 1 Adapter failures back off and report a concise blocked delivery; they never create uncontrolled duplicate messages.
- 1 A notification can request a human decision. It cannot grant approval based on a reply parser alone unless the user has separately approved a secure, unambiguous command protocol.

7. Agent Safety, Review, Privacy, and Failure Handling

7.1 Claim eligibility and agent identity

- 1 Each worker has a stable logical identity (for example, Hermes profile plus role, Herdr worker role, or named review worker); display/model names alone are insufficient.
- 1 Tags and declared capabilities help routing but are not an authorization system. Runtime tool permissions, repository AGENTS.md, Hermes approvals, and task policy remain authoritative.
- 1 A worker may claim only a ready task whose execution mode/risk level it is allowed to perform and whose required context is accessible.
- 1 No agent claims high-risk or restricted work for execution. It may prepare evidence, analysis, and a review packet only.
- 1 Claims are exclusive, bounded, heartbeat-backed, and auditable. If an agent stops responding, expiry must leave a visible handoff record rather than silently declaring failure or success.

7.2 Review gates

Risk / effect	Worker may do	Required gate
Low-risk read-only research, local analysis, draft plan	Gather sources and prepare evidence	Human review if user-facing decision or durable brain modification is proposed.
Medium-risk brain/edit/repository preparation	Work only inside defined scope/worktree; run deterministic checks	Independent reviewer before acceptance; user approval before any external effect.
High-impact code/service/publication change	Prepare plan, diff, review packet, rollout/rollback evidence only	Explicit human approval before merge, deploy, restart, publish, send, credential/TCC change, migration, or production configuration. Verify target readback afterward.
Restricted family voice, medical, finance, child data, credentials	Perform only explicitly authorized minimal handling	Human-only gate; no raw data in work ledger, generic agent packet, notification, or external service without named approval.

The existing factory rule remains controlling: planner and reviewer are read-only; implementers use isolated worktrees; verifiers report exact outcomes; no status transition self-authorizes production.

7.3 Evidence rules

A task submission must state:

- 1 what changed or was observed;
- 1 exact authoritative location(s) affected;
- 1 verification command/readback and outcome, including failure or partial result;
- 1 evidence references with timestamps;
- 1 remaining risk, uncertainty, and required human decision;
- 1 whether every DoD clause is met.

Do not store raw terminal logs by default. Store stable command/result summaries and a pointer to retained local logs when genuinely needed. Evidence that includes secrets, personal data, recordings, private messages, or access tokens must be redacted, classified, or omitted from the ledger.

7.4 Failure policy

Failure	Required behavior
Context source unavailable	Mark packet partial/unavailable ; do not infer absence; retry or block according to task risk.
Claim conflict or expired lease	Stop mutation, refresh state, then re-claim or hand off. Never continue on an expired lease.
Scope/policy/DoD changed mid-run	Revoke or abandon claim; preserve work as a contribution; build a new packet.
Evidence missing	Submit as incomplete/blocked, not complete.
Verification failed	Preserve failure evidence; move to blocked or changes-requested ; no optimistic closeout.
Notification failure	Record delivery failure and retry according to bounded policy; do not alter underlying task state.
Service/repository/device identity uncertain	Treat as blocked; perform read-only identity verification before any mutation.
Privacy/sensitivity conflict	Stop, minimize retained metadata, and request a human decision.
Ledger unavailable/corrupt	Fail closed for claims/reviews/side effects; continue only read-only source inspection; restore from transaction-safe backup after human review.

8. Phased Adoption Roadmap

Each phase is intentionally small and has an approval boundary. None of the phases is authorized by this plan alone.

Phase 0: Governance, authority, and pilot selection

Objective: Ratify the state/evidence boundary before creating a system.

Dependencies: User review of this plan; no code/config/service changes.

Actions:

- 1 Approve or revise the authority matrix and canonical task vocabulary in Sections 3–4.
- 2 Select one low-risk pilot domain, recommended: read-only software-factory planning/review work in a single existing project. Do not start with family voice, medical/financial items, Apple automation, production EMI, or device deployment.
- 3 Define named human reviewers and one logical coordinator identity for the pilot.
- 4 Choose pilot exit metrics: duplicate-work prevention, handoff clarity, evidence completeness, and notification-noise rate.
- 5 Confirm the recovery/backup owner and require a live check of the existing brain/Gitea backup chain before treating it as a recovery guarantee.

Verification and acceptance criteria:

- 1 User approves the authority matrix or records explicit deviations.
- 1 Pilot project and task class are named.
- 1 At least one example task has a non-vague next action, DoD, evidence list, risk level, and review gate.
- 1 No existing brain note, Hermes config, cron job, service, repository, task, or notification is modified in this phase.

Human gate: Written approval to begin Phase 1.

Phase 1: Local work-state contract and shadow ledger prototype

Objective: Establish the smallest local structured state model without replacing existing tools or emitting external effects.

Dependencies: Phase 0 approval; live backup-chain assessment; approved local storage location and backup policy.

Actions:

- 1 Design and test the local ledger schema using the canonical objects above: task, claim, run, evidence reference, review, audit event, and notification delivery.
- 2 Implement an owner-only local storage and transaction/locking strategy. Prefer a local SQLite ledger with a documented export/recovery procedure; do not introduce Docker, Postgres, a network API, or a second UI.
- 3 Define strict input validation: only a task with `nextAction`, `DoD`, required evidence, risk, execution mode, and review gate may enter ready.
- 4 Add a shadow-mode adapter that records state for a small set of manually selected pilot tasks but does not change existing task checkboxes, files, repository state, notifications, or Hermes behavior.
- 5 Produce a compact human-readable state digest that links to the relevant brain/project records without copying their contents.
- 6 Create deterministic test cases for: single claim, conflict, heartbeat extension, lease expiry, policy revocation, submission, independent review, evidence omission, export/import recovery, and redaction.

Verification and acceptance criteria:

- 1 Two concurrent claim attempts produce one winner and one explicit conflict result.
- 1 Expired or policy-revoked claims cannot submit or mutate state.
- 1 Every completed pilot run shows task ID, DoD, evidence refs, reviewer decision, and audit timestamps.
- 1 Ledger backup/export restores into a test location without loss of task/run/review relationships.
- 1 No raw private messages, credentials, recordings, or secret URLs appear in the ledger export/digest.
- 1 Existing brain Markdown remains untouched except for any separately user-approved future evidence note; Phase 1 itself does not backfill or rewrite it.

Human gate: Review pilot ledger evidence and approve Phase 2 read-only context integration.

Phase 2: Hermes and Herdr read-only context packets

Objective: Make explicit work state usable without granting new powers.

Dependencies: Phase 1 acceptance; read-only discovery of Herdr interfaces; task-context/redaction policy approved.

Actions:

- 1 Define a versioned context-packet serializer that joins the ledger task/claim/review state with narrowly selected brain/project evidence references.
- 2 Add read-only Hermes workflow guidance: before any pilot task work, request a current packet; after any observed action, capture evidence rather than inferring completion from chat.
- 3 Design a Herdr read-only representation for project/worker state, review queue, lease health, and blockers. Do not assume a specific UI/API until discovery proves it.
- 4 Add packet-size limits, sensitivity filters, source availability labels, and a visible “stale since” indicator.
- 5 Run a handoff exercise: one worker prepares a task, another worker/reviewer resumes solely from the context packet and source links.

Verification and acceptance criteria:

- 1 A second worker can accurately identify next action, DoD, current owner/lease state, blockers, evidence, and approval gate without reading whole chat history or vault.
- 1 Packets cite source paths/IDs and declare partial/unavailable retrieval rather than fabricating completeness.
- 1 Herdr/Hermes displays cannot update execution state in this phase.
- 1 A restricted-data test confirms prohibited content is excluded from packet and digest.
- 1 The handoff produces no duplicate task execution and no side effect.

Human gate: Approve a controlled write path for low-risk pilot state updates and, separately, approve or decline notification planning.

Phase 3: Controlled low-risk claims, review workflow, and Apple Reminders dry run

Objective: Use claims/reviews in real low-risk work while proving notification adapters remain delivery-only.

Dependencies: Phase 2 acceptance; confirmed agent identities; live verification of the approved Reyna CLI Reminders capability and a single destination list; user approval for any Apple write test.

Actions:

- 1 Allow a named Hermes/Herdr worker to claim only `low/agent_execute_low_risk` pilot tasks under lease/heartbeat rules.
- 2 Require submission and a distinct reviewer for every pilot task; no self-acceptance.
- 3 Generate reminder candidates from state in dry-run mode. Review their payload, privacy label, destination class, and idempotency key.
- 4 After an explicit user decision, write one harmless test reminder through the verified Reyna CLI adapter to one unambiguous list; rerun to confirm update/no duplicate behavior.
- 5 Confirm that completion of the test Apple Reminder creates no task-state transition.
- 6 Exercise lease timeout, worker crash/handoff, review request changes, and notification failure paths with harmless pilot data.

Verification and acceptance criteria:

- 1 One claimed task has periodic heartbeats and a complete submission/review trail.
- 1 A reviewer can request changes and the task returns to ready/appropriate rework state with the original evidence preserved.
- 1 The Apple adapter reports created/updated/skipped/error outcome and uses a stable marker.
- 1 Re-running the adapter does not duplicate the reminder.
- 1 Completing the reminder leaves the task state unchanged.
- 1 No external send beyond the separately authorized test occurs.

Human gate: Approve limited operational use and decide whether existing Hermes/WhatsApp reminders should become a later adapter.

Phase 4: Project-specific integration for factory, infrastructure, and ministry

Objective: Extend the proven model only where it strengthens existing work, without creating a universal bureaucracy.

Dependencies: Phase 3 acceptance across a sufficient pilot sample; live verification of the target project/repository/service; project-owner review.

Actions:

- 1 **GitHub/Gitea and factory work:** Map repository tasks to existing project keys, worktrees, branch/PR/commit evidence, required checks, independent review, and human merge/deploy gate. Keep repository `AGENTS.md` authoritative.
- 2 **Herdr:** Surface approved ledger views for active leases, review queue, blocked work, and packet links. Herdr remains the control plane; it does not replace project records or auto-authorize effects.
- 3 **Family infrastructure:** Use the existing service-inventory vocabulary. A service verification task must include target identity, read-only health evidence, last verified time, and no service mutation unless separately approved.
- 4 **EMI:** Separate ministry research/drafts from public, governance, financial, or publication changes. Public sends/publishing require the existing explicit approval and target readback; email-derived information defaults to private handling.
- 5 **Family work:** Use the system for ordinary, low-sensitivity household follow-through only after privacy rules are proven. Exclude raw voice, child/private health, financial, and credentials from generic work state.
- 6 Add only approved notification adapters, starting with private summaries and always retaining delivery-only semantics.

Verification and acceptance criteria:

- 1 One software change record references project definition, isolated worktree, tests, reviewer verdict, and explicit merge/deploy gate without claiming those effects occurred.
- 1 One infrastructure read-only verification task preserves Active/Needs live verification/Legacy distinctions.
- 1 One EMI draft/review task demonstrates a public-send gate and does not send or publish without approval.
- 1 One adapter delivery failure is visible but cannot change project/task truth.
- 1 Project leads find the state digest/context packet clearer than the prior status-only approach; otherwise scope is reduced.

Human gate: Review whether a dedicated local dashboard/adapter service is justified by sustained usage, not novelty.

Phase 5: Optional operational hardening and retrieval evaluation

Objective: Decide whether the small overlay should remain local or earn a more durable multi-client implementation.

Dependencies: At least several weeks of useful Phase 4 usage, no unresolved privacy/duplicate-notification defects, and demonstrated recovery tests.

Actions:

- 1 Audit contention, lease expiry, recovery, backup, evidence quality, review latency, adapter duplicates, and user burden.
- 2 Decide whether a local read-only dashboard, authenticated internal API, or Herdr-native view is worth building. Do not deploy a service merely to imitate upstream architecture.
- 3 Run the retrieval benchmark described in Section 5 before introducing vector embeddings.
- 4 If multi-host access is justified, design authentication, least privilege, secret storage, audit retention, backup, revocation, and outage behavior before any LAN exposure.
- 5 Re-evaluate scope: archive fields/processes that were not used; preserve only the minimal proven contract.

Verification and acceptance criteria:

- 1 Recovery drill restores structured state and preserves ledger-to-Markdown/repository references.
- 1 Metrics show fewer duplicate efforts, clearer review handoffs, and acceptable notification noise compared with the baseline.
- 1 Any hosted/LAN service proposal has a written threat model, rollback plan, owner, health check, backup, and approval boundary.
- 1 Vector retrieval is adopted only if benchmarked hybrid retrieval improves real cited recall and keyword fallback remains functional.

Human gate: Separate approval required for any LAN/public service, non-local credential, vector provider, new database host, or production automation.

9. Migration and Rollout Strategy

Non-disruption principles

- 1 **No bulk migration.** Do not parse/rewrite existing brain notes, Markdown checkboxes, journals, project documents, or archives into the ledger.
- 2 **Forward-only pilot.** Begin with manually selected new pilot tasks. Legacy work is linked only when a human explicitly asks to track it.
- 3 **No automatic closure.** Existing checked boxes, completed reminders, Git commits, agent chats, or idle statuses may be evidence but never automatically become ledger acceptance.
- 4 **No source replacement.** The brain, project records, service inventory, repository history, and existing reminder paths remain authoritative throughout rollout.

- 5 **Reversible overlay.** Disabling the ledger must leave original Markdown and existing workflows intact. The ledger can be archived/exported without data loss or source mutation.
- 6 **One-way evidence references first.** Start by linking state to existing sources. Only after review may an approved workflow append a concise evidence section to a brain/project note, following brain formatting rules.
- 7 **No service dependency at launch.** The first version must work while the notification watcher, reMarkable watcher, remote nodes, or optional semantic index are absent/unavailable.

Suggested rollout cohort

- 1 Start with 5–10 low-risk, non-sensitive, read-only or plan/review tasks in one factory project.
- 1 Use at most one coordinator and one reviewer in the first cohort.
- 1 Do not use scheduled claims, automatic task intake, WhatsApp sending, Apple Reminder writes, remote agent dispatch, or production hosts in the initial cohort.
- 1 Review every task manually. Measure clarity and burden before expanding.

Legacy linking protocol

When the user wants to incorporate an existing note/task:

- 1 Create one new ledger record that contains the existing Markdown path/heading and source provenance.
- 2 Mark it draft until the user or project steward supplies a current nextAction, DoD, risk, evidence, and gate.
- 3 Record “legacy source; current state not yet verified” instead of carrying old prose forward as fact.
- 4 Preserve the original note exactly. Do not rewrite historical journals or archives to match a new state vocabulary.
- 5 If a later durable note update is approved, append/revise according to brain/rules.md and link back to the ledger task ID.

10. Measures of Success and Operating Metrics

Track these only after consent and with privacy-safe aggregate data:

Metric	Desired signal	Guardrail
Claim conflicts/duplicate work	Fewer duplicated agent efforts	Do not inflate task count to improve metric.
Context handoff completeness	A reviewer can resume work from packet/source links	Do not solve by dumping the full vault.
Evidence completeness	Submitted work has verifiable refs and explicit uncertainty	Do not store raw secrets/logs to increase completeness.
Review latency	Clear gate ownership and fewer ambiguous “done” claims	Do not bypass human review for speed.
Lease health	Expired/revoked work is visible and safely recoverable	No unbounded heartbeats.
Notification noise	Useful nudges with few duplicates	Delivery never changes task truth.
Recovery success	Ledger backup/restore preserves links and audit trail	Do not represent backup as verified until drill succeeds.
Retrieval quality	Cited relevant sources, including known partial results	Embeddings are not adopted on anecdote alone.

11. Explicit Open Decisions for User Review

- 1 **Pilot domain:** Is the recommended first pilot a low-risk software-factory planning/review workflow, or should it be another bounded domain?
- 2 **State authority:** Approve the proposed local ledger as authoritative for live task/lease/review state while the brain remains authoritative for knowledge/evidence narrative?

- 3 **Status vocabulary:** Approve the proposed lifecycle (draft, ready, claimed, active, waiting, blocked, submitted, accepted, closed, cancelled), or map it to a preferred simpler vocabulary?
- 4 **Review standard:** Must every low-risk agent task have a distinct human reviewer in the pilot, or can a named independent reviewer agent approve selected non-side-effect work?
- 5 **Storage and backup:** May the initial ledger be local-only with transaction-safe backups pending a live Gitea backup-chain verification, or should the first phase wait for verified private backup integration?
- 6 **Herd scope:** Should Herdr initially be read-only state/review visibility, or should a later phase permit it to submit approved low-risk ledger transitions?
- 7 **Apple Reminders:** Which single list should be used for an eventual pilot, and should it receive only summary nudges or selected P0/P1 task-specific reminders? No adapter will be written until the native Reyna CLI capability and list identity are verified.
- 8 **Messaging:** Should future operational nudges default to private DM only, with the Family Group used only for explicitly family-safe requests? This plan recommends yes.
- 9 **Retention:** How long should detailed ledger runs/audit events be retained, and what redacted digest belongs in the Markdown brain, if any?
- 10 **Semantic retrieval trigger:** What threshold of demonstrated keyword-search misses is sufficient to authorize a benchmark? This plan recommends a documented set of at least ten real retrieval queries before considering embeddings.

12. Ideas Deliberately Rejected or Deferred

Idea	Decision	Why
Deploy the upstream Personal OS + Personal Wiki stack unchanged	Reject for now	It adds Next.js, PostgreSQL, Docker, another UI, tokens, operations, and a duplicate Wiki when the existing brain/Herdr/Hermes/factory records already cover much of that terrain. The valuable missing piece is narrow execution state, not a wholesale replacement.
Replace the Markdown brain with database records	Reject	The brain is the established human-readable, linked, Gitea-backed knowledge base. Markdown is better for durable narrative, review, portability, and editorial work.
Treat Hermes memory as the task database	Reject	Persistent memory is suited to stable preferences and constraints, not current claims, approvals, evidence, review decisions, or recovery-safe status.
Treat Apple Reminders/WhatsApp/desktop notifications as task truth	Reject	They are delivery surfaces. A dismissed/checked nudge proves neither task completion nor DoD verification.
Mandatory vector memory/RAG from day one	Reject	Keyword/metadata/links/context packets are cheaper, more inspectable, and match the current brain MCP direction. Add semantics only after a measured recall benefit.
Automatic intake of every message/note into tasks	Reject	It would create noise, misclassify sensitive material, and violate the existing Inbox/idea-shaping distinction. Capture may be proposed; promotion requires clear action/DoD.
Autonomous “autodrive” across merges, deploys, sends, restarts, and production config	Reject	It conflicts with the current factory operating rules and Hermes approval boundaries. Claims prove ownership, not authority.
Store raw logs, recordings, credentials, private notification data, or large artifacts in the ledger	Reject	Privacy, sync, audit, and backup risk. Store redacted metadata and source pointers only.
Make service inventory status auto-updated from agent assertions	Reject	Existing inventory correctly differentiates documented status from live verification. Only verified target readback plus review may update durable service records.
Build a new dashboard before proving behavior	Defer	Herdr and existing surfaces may be sufficient. A dashboard must earn its operational cost through proven pilot use.

13. Final Review Checklist

Before approving any implementation phase, confirm:

- 1 ☐ The existing Markdown brain, factory records, repository rules, and platform/service inventories remain authoritative as defined here.
- 1 ☐ The new component is a small local overlay, not a second personal OS platform.
- 1 ☐ Every task has `nextAction`, DoD, risk, allowed/forbidden effects, evidence, and review gate before it can be claimed.
- 1 ☐ Claims are exclusive, leased, heartbeat-backed, revocable on policy change, and auditable.
- 1 ☐ Agents submit evidence; they do not self-authorize high-impact effects or self-accept their own high-risk work.
- 1 ☐ Context packets are bounded, cited, privacy-filtered, and explicitly label unavailable/partial retrieval.
- 1 ☐ Apple Reminders and messaging adapters are idempotent delivery-only mirrors.
- 1 ☐ No legacy brain data is bulk-migrated, rewritten, or made dependent on a new service.
- 1 ☐ Retrieval remains keyword/metadata-first until a cited benchmark justifies a hybrid layer.

- 1 [] Every later network/service/deployment step has its own explicit human approval, backup/rollback plan, and target readback verification.

Source Notes

- 1 Local sources inspected: brain/rules.md, brain/index.md, areas/platform.md, areas/operations/services_inventory.md, areas/emi_things_to_do.md, projects/FACTORY_OPERATIONS.md, projects/factory/README.md, projects/para_brain_mcp_server.md, projects/paseo_software_factory_working_model_2026-08-17.md, projects/paseo_centralized_agents_2026.md, and projects/macOS-notification-watcher-poc.md.
- 1 Hermes configuration inspected read-only: agent, memory, approvals, delegation, checkpoints, and curator sections.
- 1 External inspiration inspected as untrusted public reference: <https://github.com/lawyer112/personal-os-wiki>, its README, docs/ARCHITECTURE.md, docs/AGENT_GUIDE.md, docs/WHY_NOT_LONG_TERM_MEMORY.md, docs/VECTOR_MEMORY_DECISION.md, docs/MAC_AGENT_ADAPTER.md, and personal-os-app/docs/HERMES_API.md.
- 1 This document proposes no code, configuration, service, repository, task, cron, reminder, or message changes. It is a plan only.