

Prairie Simple Software

WONDER portfolio and system specification

Aaron Moyer

Founder and developer

25 October 2026 • Public review edition

Complex questions. Clear next steps.

Research and decision intelligence made understandable

Prairie Simple Software is the proposed business identity behind a founder-led software effort. Its WONDER ecosystem combines evidence handling, structured judgment, numerical simulation, evaluation and application interfaces. This portfolio explains what people can use these components for, how they work together and where their present limits lie.

The portfolio covers seven WONDER repositories, with Healthy Energy shown as a separate application of selected components. It is both an introduction for interested readers and a detailed system specification for developers and technical reviewers. [S1]

What this portfolio demonstrates

- Reusable software components for evidence-governed analysis, numerical comparison and traceable decisions.
- A modular approach spanning finance, engineering, model development and evaluation.
- Concrete component reuse in Healthy Energy, where evidence review and judgment receipts support a practical operating workflow alongside a separate numerical decision study.

The business name remains subject to formal clearance. The product name WONDER is also a working identity requiring its own commercial review. The public description is software for research and decision-making. The technical description is governed hybrid research and decision intelligence, with those terms defined in the following pages.

Public document. Private repository links, internal business economics and confidential application planning details are excluded. This document does not grant source access or represent an open-source release.

Reading guide: start with the opening pages for the brand and current evidence. Follow the repository map for the architecture, then choose a component chapter. The capability index and evidence catalogue provide a detailed reference path.

Portfolio contents

Choose a plain-language application chapter first, or follow the architecture into the detailed technical specification. The capability index provides a second, method-by-method navigation path.

Brand and positioning	3
Current source and release evidence	4
Architecture and repository map	6
Inputs outputs and responsibilities	7
State and failure behavior	8
Shared Core platform	10
Candidate construction and learning	15
Numerical simulation and multilevel methods	17
Synthetic research environments	21
Evaluation evidence	23
Engineering repair application	26
Financial application and numerical methods	28
Healthy Energy application case study	33
Learning and governance limits	35
Verification and roadmap	37
Public research section copy	39
Detailed capability index	41
Revision inventory and evidence	44

Prairie Simple and the product identity

Prairie Simple is a working identity for software that makes complex work easier to understand. Its purpose reaches beyond business planning to research, simulation, evidence review and domain-specific decision support. The strength of the approach is a result that a person can inspect, challenge and improve.

What the name promises

Simple describes the experience, not the mathematics. A user should be able to identify the inputs, see the evidence and understand the next step. Uncertainty and refusal remain visible. The technical chapters explain how these goals are supported, while measured usability remains a task for future user evaluation.

The brand structure

Prairie Simple Software is the proposed business name. Prairie Simple is the everyday brand, and WONDER is the ecosystem and product working name. Research and decision intelligence describes the broad purpose. Governed hybrid research and decision intelligence describes the combination of methods and controls.

Governed means implemented controls such as evidence checks, review requirements, permission boundaries and retained records. Hybrid means a combination of conventional rules, numerical methods, optional third-party AI and human judgment. Neither term represents certification or universal reliability.

Language that fits the audience

“It’s very simple when it’s Prairie Simple” is the founder’s expression of the intended experience. The concise brand line is “Complex questions. Clear next steps.” Both express an aspiration for clarity, while the software keeps assumptions, uncertainty and responsibility explicit.

Name readiness

The names Prairie Simple Software and WONDER are working commercial identities. Formal clearance and registration remain outstanding. The Canadian and Saskatchewan naming guidance is listed under public references. [P1–P2]

Who this is for

Entrepreneurs may use this approach to organize supplier evidence or test assumptions. Researchers may use numerical tools to compare models. Developers may embed a specific package in an application. Financial and engineering workflows demonstrate other uses, subject to their own scope and acceptance evidence.

Current evidence and release status

Evidence cutoff: 25 October 2026 at 14:30 UTC, or 8:30 a.m. in Saskatchewan. This snapshot separates merged source, draft development and live use. It is not a continuously updated service-status page.

Merged source and consumer pins

The seven main revision identities in the repository inventory remain unchanged at this check. Main CI records are passing within their recorded scopes. Financial's merge-head run completed lightweight checks only, while the merged pull-request head passed the broader platform suite. Healthy Energy is separately pinned to the reviewed Core main snapshot. Newer draft work does not update it.

Current supervisor candidate

At the 14:30 UTC check, Core's draft supervisor and clock-policy candidate 4c8e86b1 passed Ubuntu and Linux container checks. Windows offline verification passed, but the subsequent source-integrity step failed. The cause remains unresolved. Financial's companion draft candidate 72dbf68e passed Ubuntu full verification, PostgreSQL and container checks, while Windows remained in progress. Neither candidate had been merged, deployed or run as a live pilot.

Open qualification work

Local monotonic deadlines and cross-host ordering are different concerns. The draft work now includes a database-causal freshness contract and controlled clock-discrepancy qualification, but production adoption and live acceptance remain open. Linux production result admission still needs independently bound process-exit evidence. Historic timeout causes remain unknown. A separate immutable-read draft has a merge conflict despite older passing CI.

How to interpret the specification

A callable function is an implemented interface. A fixture exercises a declared scenario. A saved result shows what happened in one retained run. A successful CI job shows that a named set of checks passed at an exact revision. Current deployed behavior requires separate deployment and acceptance evidence. The chapters use these distinctions deliberately.

What this means for a reader

The portfolio shows substantial implemented software and recorded tests. The newest operational changes are still being qualified. A prospective user should evaluate a specific application and version, rather than assume the entire ecosystem is already one production-ready service.

A practical entry point

For a non-technical reader, start with Healthy Energy, Engineering or the Financial decision workspace. For integration planning, read the system contracts and the relevant repository chapter. The capability index locates the detailed methods, while the evidence catalogue identifies the supporting artifact families.

Portfolio scope and founder profile

Development model

I am the founder and developer behind WONDER. I direct its architecture and development from my laptop, using ChatGPT and connected research tools to investigate problems, assist with code and turn research into usable software. This is a one-founder development effort, not a staffed software organization.

My objective is to build software that can reason over evidence, quantify uncertainty and preserve an accountable record of how a conclusion was reached. The longer-term ambition includes OGI and artificial conscientiousness. Those terms describe a research direction. This portfolio does not claim demonstrated consciousness or general intelligence.

What the repository inventory contains

Group	Repositories	Portfolio role
Foundation	WONDER-Core WONDER-Monte-Carlo WONDER-Models WONDER-Evals WONDER-Lab WONDER-Engineering	Six complementary foundation repositories, reviewed individually in the following chapters.
Domain application	WONDER-Financial	An ecosystem-powered application with its own substantial quantitative research library and authenticated workspace.
Application case study	Healthy Energy	Concrete component reuse in a separate application, outside the seven-repository count.

All seven repositories in scope were private and nonarchived in the reviewed inventory. Repository count is not a count of deployed products, independent models or customers. Each repository has a distinct role and its own evidence of maturity. Names and revision identities are included to distinguish the components, without publishing private source links. [S1]

How to read the evidence

Implemented means an inspected code path or software contract exists. Recorded result means a saved run, report or check was reviewed. CI verified refers to a retrieved service result for an identified revision. Research direction identifies proposed or exploratory work. The detailed catalogue preserves the earlier source review and updates time-sensitive status. No tests or live application workflows were rerun for this edition.

How the repositories work together

WONDER assigns different responsibilities to different components. Core supplies decision, evidence and execution contracts. Monte Carlo supplies numerical estimates. Domain applications choose which methods and interfaces they admit. Lab, Models and Evals form a research chain in which producing a candidate and accepting its evidence remain separate responsibilities.

Relationship	Observed integration and boundary
Financial uses Core and Monte Carlo	Installed pinned numerical and judgment packages, native Decision Lens, and bounded forecast readers. Consumer versions remain independent of the latest upstream versions.
Engineering uses Core	Two pinned unmodified Core packages support the local repair workbench and its offline provenance checks.
Lab supplies Models	Synthetic observations and experiment receipts feed bounded candidate construction and estimation. This is an implemented research handoff.
Models supplies Core and Evals	Core can consume a versioned fitted candidate. Evals assesses frozen artifacts and their provenance. A candidate does not authorize its own admission.
Evals supplies released evidence	Development export binds an existing released result to Financial identities. This is not an Evals runtime import in Financial or a claim of protected native acceptance.
Monte Carlo supplies numerical contracts	Model, stream, path range and result identities belong to the numerical layer. Core orchestration owns jobs, attempts, budgets and execution permissions.
Healthy Energy reuses selected components	A separate application consumes pinned Core files and Monte Carlo for evidence review and launch-option analysis.

A decision lifecycle

A typical integration captures evidence and explicit assumptions, runs the appropriate model, evaluates alternatives and constraints, presents a reviewable judgment, and then applies a separate permission decision if an action is requested. Execution receipts and independently observed effects are retained separately. The actual boundary depends on the host application. A library export alone does not establish a deployed service.

The learning research lifecycle

Lab produces observations. Models fits or proposes an immutable candidate. Evals checks a frozen evaluation and its result lineage. Core may consume the candidate under a versioned policy. Rejected, incomplete and adverse outcomes remain visible. This chain is concretely exercised in synthetic routing and repair studies, while protected holdout custody and broad autonomous learning remain research work. [C1–C8, M1–M4, L1–L3, E1–E5]

System interfaces and responsibility boundaries

WONDER is a family of versioned components and bounded applications, not one continuously autonomous service. An application selects and pins the parts it uses. Its host supplies the deployment environment, identity and permissions. The following matrix is an architectural integration guide, not a claim that every row is a running service. [C1–C4, N1–N5, M1–M4, L1–L3, E1–E5, G1–G3, F1–F6]

Component	Inputs	Outputs and boundary
Core	Evidence, assumptions, options, constraints and policy	Judgments, dossiers and receipts. Authority and effects remain separate.
Monte Carlo	Mathematical model, canonical inputs, seed and resource budget	Estimates, uncertainty and replay-bound results. No real-world truth guarantee.
Lab	Versioned synthetic scenario and intervention	Observations and receipts, including negative and unresolved outcomes.
Models	Validated training records or repair evidence	Immutable candidate. Candidate creation does not admit or deploy it.
Evals	Frozen candidate, exact output and declared protocol	Scoped result and eligible evidence. Evaluation is not deployment permission.
Financial	Authenticated selections and explicit financial snapshots	Calculations, saved artifacts and retained forecast readers. Methods are admission-specific.
Engineering	One of three pinned repair cases and bounded edits	Tests, differences and local review history. It does not apply the patch.

A typical handoff

Evidence and assumptions enter a calculation. The result is checked against its exact request. A judgment compares options under explicit rules. A consequential action requires a separate authority decision. Execution and independently observed effects are recorded separately. A host may stop before execution and present only research or decision support.

State persistence and failure behavior

State belongs to a specific component or host. Pure library records do not imply a durable database, a tenant boundary or an authenticated source. Integrators must establish each of these separately. [C3–C4, C11, C17–C19, N9, G2–G7, F4–F6]

Data and identity

Versioned inputs, source fingerprints and exact request identities make a result traceable. A digest can identify unchanged bytes, but cannot prove that a supplier statement is true. Replay requires the relevant implementation, model, seed and inputs, rather than a seed alone. A record of permission is distinct from the effect that was later observed.

Persistence and process boundaries

Core's general artifact registry and Lab worlds are process-local. Specific Core adapters and Financial durable workflows use PostgreSQL. Engineering retains local SQLite history. Models and Evals development records use immutable file publication. Ordinary Monte Carlo has no database. Financial's session-only comparison, local jobs and authenticated durable worker are distinct paths.

Outcomes that should remain visible

- Missing, stale or contradictory evidence can block eligibility or produce abstention. Unknown must not be silently changed to passed.
- Timeout, cancellation and a process outcome that cannot be confirmed are different states. An uncertain external outcome must not be treated as permission for an automatic conflicting retry.
- A failed numerical check, nonfinite value or mismatched request/result identity requires rejection within the relevant contract. Nonconvergence must remain visible.
- A failed or inconclusive evaluation is retained. A candidate cannot establish its own evaluation independence or grant itself deployment authority.

Security and acceptance limits

Worker termination is not hostile-code isolation. Declared evaluator separation is not protected holdout custody. An uncertain commit or termination retains UNKNOWN or COMMIT_UNCONFIRMED and requires reconciliation of the original identities. A cancellation request does not prove external effects stopped. Each application needs appropriate identity, permissions, data protection, recovery and deployment acceptance.

Understanding uncertainty in the results

A precise calculation can still rest on weak assumptions. WONDER's most useful result is often a clearer account of what is known, what was assumed and what needs another observation. Different uncertainty measures answer different questions. [C12–C15, M3–M4, N3–N4, E3–E4]

Simulation sampling error

This describes the effect of using a finite number of numerical draws under a fixed model. More draws may reduce this error without making the model or its business assumptions more realistic.

Parameter estimation

A fitted parameter comes from observed data under stated assumptions. The Route B example estimates one Bernoulli success probability from synthetic outcomes. It is real bounded estimation, not neural-model training or universal learning.

Discretization and estimator bias

A finite-level approximation can differ from the desired target even with many samples. Native fixed MLMC keeps sampling uncertainty separate from unknown discretization bias. An approximate normal interval is not a coverage certificate.

Source and input uncertainty

A missing quotation, expired observation or unauthenticated assertion can matter more than numerical precision. Hashing preserves captured content identity. It does not prove that the source is true or suitable for the decision.

Model and decision assumptions

Scenario ranges, utility, horizon and constraints shape the answer. A model may omit a regime or a relevant cost. A favourable score cannot override a hard constraint or create permission to act.

Evaluation uncertainty

Development-visible fixtures, dependence across cases and unprotected holdouts limit what a passing evaluation establishes. Negative and inconclusive results remain part of the evidence rather than disappearing into a positive average.

WONDER Core platform

WONDER-Core is the shared decision-assurance and controlled-execution platform. Twelve JavaScript packages divide reasoning, evidence, judgment, permission, action and observed outcome into inspectable interfaces. The reviewed workspace is version 0.13.0-rc.1.scientific.1 at b380d2441af85ab07790b326127ef901b6971c2b. [C1]

Twelve complementary packages

Package	Responsibility
action-runtime	Prepare exact effects, validate authorization, reserve permits, emit receipts and verify observed effects.
authority-engine	Evaluate default-deny policy, signed approval, attenuating delegation, freshness and revocation.
constitutional-kernel	Enforce prohibited operations and mutation zones, including self-admission and self-grading restrictions.
decision-lens	Render semantic DOM decision views with evidence, confidence, uncertainty and authority context.
decision-orchestrator	Sequence reasoning, judgment, preparation, authorization and execution with integrity checks.
decision-protocol	Bind artifact identities, lineage graphs, proposals, dossiers and prospective forecast contracts.
evidence-engine	Bind evidence, models and assumptions. Invalidate affected downstream work when inputs change.
judgment-engine	Evaluate constraints, viability, ranking, scenario criteria, value of information and numerical evidence.
model-assurance-engine	Provide scientific and exact numerical primitives, model registries, assurance cases and release gates.
organism-contracts	Define episodes, memory, candidates, threshold-based homeostasis, artifacts and recovery interfaces.
reasoning-gateway	Connect an injected LLM provider with bounded responses, deadlines, cancellation and untrusted-output receipts.
stochastic-computation-fabric	Plan and coordinate bounded computation, provider interfaces, artifact storage and recovery reconciliation.

What a developer can do with it

A developer can embed structured judgment in an application, register evidence and assumptions, compare options under different decision criteria, present native review views and retain verifiable decision artifacts. For a consequential workflow, the developer can separately prepare an action, check authority and record the observed result. Host integrations determine persistence and external permissions. Core does not confer those simply by being imported.

WONDER Core integrated capabilities

Quantitative judgment with explicit uncertainty

Judgment checks evidence sufficiency, contradictions, hard constraints, viability, weighted criteria, interval-aware rankings and abstention. Missing assessments cannot silently satisfy a hard constraint. Finite-state value of perfect information and expected sample information use explicit probabilities, utilities and observation models. Native MLMC bindings retain the finite-level target and sampling error rather than concealing unknown discretization bias. [C2]

A concrete decision workflow

An inspected end-to-end regression compares three infrastructure options using actual Monte Carlo calculations. Expected utility prefers fixed capacity, maximin selects elastic capacity and minimax regret selects hybrid capacity. The deliberate route runs nine simulations and the fast route runs three. A sealed local effect executes once, and replay verifies the retained bundle. Missing evidence, failed checks or an exhausted budget produce zero numerical or effect calls. Source invalidation blocks an effect even after an earlier authority decision allowed it. [C5]

This example demonstrates the interaction between numerical evidence, decision criteria and controlled execution. It is a synthetic local workflow, not a claim that production infrastructure is changed autonomously.

Action and evidence accountability

The orchestrator binds prepared actions and judgments to exact content and evidence snapshots. The action runtime can bind proposal, evidence, policy, approval, action and expiry in an opt-in sealed profile. It separates a provider's acceptance from independently observed effects. Compensation is a new consequential action with its own checks. Decision Protocol records lineage and rejects cycles or unresolved dependency edges. [C3–C4]

Research API and forecasting lifecycle

The authenticated Research API exposes three calculations: binary Brier loss, a fixed-design zero-event upper bound and deterministic subject partitioning. For example, a probability of 0.8 and an observed outcome of 1 produce Brier loss 0.04. Outputs explicitly declare that they are not persisted and do not authorize action. JWT, origin, body-size, deadline and concurrency controls bound the service.

The newer default-off UNRATE pilot preserves a policy-to-lock sequence covering source capture, input admission, generation and immutable forecast lock, with separate outcome admission, settlement and scoring commands. Its source contract binds an explicit baseline and exact producer identities. Late recovery is unsupported in this newer pilot version, distinct from older lifecycle paths. Controlled local rehearsal is not a live prospective forecast or independent time witness. [C6–C8]

Verification: merged main verification record passed. Previously inspected logs record 1,824 library tests and 35 runtime tests, plus 12 standalone consumers and overlapping integration groups. Counts are not additive. The supervisor repair is on a separate draft candidate and is not part of main. See Current evidence and release status for the dated candidate snapshot. [C9]

Core quantitative and scientific methods

Decision algorithms

Beyond weighted ranking, Judgment Engine implements interval-aware hard rules, maximin and minimax regret with controlling scenarios and ties. A probability-robustness routine searches bounded integer-mass total-variation ambiguity over a supplied scenario universe, returning exact weighted numerators and an adverse witness distribution. Finite EVPI and EVSI calculations assess whether another observation is worth its cost. DAG d-separation and backdoor checks evaluate a declared causal graph. Scientific-claim review binds the dataset, estimator and estimate while distinguishing association, prediction, intervention and counterfactual claims. [C12]

Calibration reference pricing and validation

Model Assurance contains independent normal-CDF and Black-Scholes reference calculations, analytic Greeks and implied-volatility inversion. Its calibration code generates deterministic starts, runs bounded pattern search and multistart optimization, computes finite-difference Jacobians and checks parameter identifiability. Flat Black-Scholes calibration reports residuals, weighted errors, RMSE, bounds, convergence and near-optimal non-identifiability against an immutable market snapshot.

Benchmark utilities represent tolerances, reference comparisons, convergence studies and champion–challenger comparisons. Financial diagnostics include strike-grid static-arbitrage and calendar-total-variance checks, Kupiec unconditional-coverage likelihood ratios and fractional empirical upper-tail expected shortfall. A diagnostic statistic remains distinct from a policy decision. [C13]

Forecast and numerical diagnostics

CORP/PAV binary calibration supports equal-row-weight in-sample recalibration and Brier decomposition into miscalibration, discrimination and uncertainty, with inclusion and exclusion records and tolerances. It describes the supplied sample rather than establishing prospective improvement.

Numerical primitives include log-product, log-sum-exp, Neumaier compensated summation, mergeable online moments, probability-boundary tolerance corrections and dimension-aware rates. Exact rational and monetary values support parsing, addition and exchange-rate conversion with floor, ceiling, half-up or half-even rounding. Probability-to-hazard and hazard-to-probability transforms support duration-aware risk calculations. [C13]

Time information and configuration qualification

Temporal values cover canonical instants, half-open intervals, approximate-time provenance and bitemporal applicability. Point-in-time views distinguish what was known then from what is known now, and report revision sensitivity. Information sets preserve true, false and unknown, along with assumption, inference and dispute states.

Configuration qualification separates fixed, validated-range and experimental parameters, checks typed schemas and cross-parameter constraints, and generates deterministic Cartesian or pairwise coverage plans. Actual-result coverage evidence, platform and input envelopes, and promotion evidence support admitted, review-required or rejected outcomes. The model-use kernel binds population, purpose, horizon, input contract, evidence profile and monitoring snapshot to an inference or deployment assessment. [C14]

Core computation and adaptive system contracts

Planning compute effort

The computation planner checks mathematical compatibility before cost. Analytic routes require a declared closed form. Current fingerprint-bound assurance controls estimator eligibility. Experimental estimators are not selected automatically. Eligible methods can be compared using pilot-predicted work. Resource scheduling adds value-of-computation ranking, integer batch allocation toward a sampling-variance target, pilot racing and fine/coarse coupling diagnostics. These routines use explicit error and cost models rather than inventing them. [C15]

Operational integration families

The Fabric includes compute descriptors, durable and PostgreSQL coordinators, numerical lineage and fencing, a Modal provider adapter, local and S3-compatible artifact storage, R2 configuration, recovery checkpoints, reconciliation, backup and restore contracts, cancellation, backpressure, graceful shutdown, schema migration and readiness checks. Their role is to let a host assemble and verify a bounded execution system. An adapter's presence does not mean the corresponding service is deployed.

Security and release families cover workload identity, credential-lifecycle records, principal quotas, idempotency and state-changing admission, structured audit events and hash chains, checkpoints, incident and fault evidence, supply-chain and lockfile provenance, offline toolchain checks, CI invariants, release bundles, requirement traceability and evidence-gate graphs. [C11]

Adaptation memory and declared evidence

Adaptive-system APIs represent memory admission, versioning, revocation and erasure plans. They also define forgetting classes, consolidation windows, threshold-based homeostasis, attention ranking, proposed value revisions and belief-change classifications. Representation challengers, evaluator separation and protected-challenge descriptors support controlled adaptation without claiming that these records operate as a general autonomous intelligence.

System-identification helpers create candidates and compare supplied baseline and held-out losses. Formal-verification helpers construct requests and proof or counterexample receipts. Evaluator-isolation helpers inspect declared scopes. None of those names alone means the package trains a latent-state model, runs a theorem prover or inspects real provider access.

Orchestration and provider fixtures

Automation supports trigger and context validation, in-memory schedules, cache identities and bounded parallel simulation jobs. Deliberation routes through FAST, DELIBERATE, INVESTIGATE, DEFER or STOP using stakes, reversibility, evidence, contradictions and budget. Parent and child attempt reservations track measured wall time and response bytes. Those measurements are not provider billing.

Organism provider fixtures actually submit and poll injected providers, verify output bytes, store and reread artifacts and quarantine mismatches. Recovery checks preserve terminal uniqueness, fencing, pending outbox, digests, append-only history, revocation boundaries and budget monotonicity. The fixture explicitly distinguishes this integration evidence from live-provider evidence. [C16]

Core validation and integration contracts

Strict validation and release records

The strict validator runs dependency-ordered checks with deadlines, cancellation and resource limits. Incomplete evidence yields an explicit inconclusive result. Advisory judgment can follow scientific checks, rather than overriding them. Immutable claim, commit and read interfaces preserve an UNKNOWN publication state after an uncertain commit, so a missing acknowledgment is not treated as permission to publish a second conflicting result. [C17]

Run manifests distinguish bitwise, numerical and statistical reproducibility. Logical tasks and moment-aggregation attempts support deterministic reduction. Release profiles, waivers, five assurance ledgers and conjunctive gates bind subject, scope and evidence. A waiver is retained as an explicit exception, not erased from the release record.

Evidence capture and forecast custody

Exact text capture retains source-byte commitments and UTF-8 excerpt spans for source material up to 1 MiB. Cutoff and rights checks can accompany that captured identity. A valid byte span proves which text was used. Semantic support is separately reviewed.

Decision Protocol also provides preregistered decision trials, forecast revisions and distributions, calendar custody, official UNRATE release-schedule bindings, FRED monthly observations, prospective outcome resolution and PostgreSQL forecast locks. Outcome admission, settlement and late recovery have separate contracts. These are the identity and lifecycle mechanisms surrounding a forecast, rather than the statistical generator itself. [C18]

Authority and action integration

Authority supports Ed25519 delegation chains, attenuated permission and fingerprints. Review-package authentication derives evaluator and reviewer identities and checks separation-of-duty links rather than trusting a caller's favorable summary. In-memory and PostgreSQL execution permits, revocation epochs, provider adapters, effect receipts, verification deadlines and compensation dossiers extend the action boundary. Host evidence still determines who those identities represent in the real organization.

Reasoning and presentation adapters

Reasoning Gateway registers providers and checks their capabilities. Its bounded JSON-over-HTTP adapter supports retries, deadlines, cancellation, structured-output validation and source and usage receipts. Evidence-reviewed support constrains claims to a host-selected evidence universe. The provider performs inference. The gateway supplies no foundation-model weights.

Decision Lens normalizes multi-engine artifacts into an outcome, decision-health, boundary and action view, then renders semantic DOM with responsive and system-theme styling. Missing confidence remains "Not supplied" and numerical zero remains zero. The view does not silently recalibrate confidence or turn a recommendation into authorization. Constitutional Kernel supplies a common prohibited-operation and mutation-zone vocabulary for these integrations. It is a policy contract rather than an operating-system security boundary. [C19]

WONDER Models candidate lifecycle

WONDER-Models provides reproducible candidate construction, coding-repair lifecycle contracts and bounded learning from observed synthetic outcomes. Version 0.2.5-dev.retained.1 runs as dependency-free local ESM JavaScript. It keeps proposing, evaluating, admitting and deploying a candidate as distinct acts. [M1–M4]

What a developer can do with it

A developer can construct a dataset or training candidate with provenance and privacy classifications, bind parent model, code, data and evaluation plan, propose a successor patch, preserve the original parent and publish an immutable candidate for later evaluation. File-backed stores and CLIs support exact readback and reconstruction.

Coding-agent contracts enforce the sequence ORIENT, REPRODUCE, DIAGNOSE, PLAN, IMPLEMENT, VERIFY, REVIEW and PACKAGE. Tool receipts bind task, phase, inputs, outputs and exit codes. Mutation zones and review requirements constrain a successor proposal. These are usable workflow and receipt mechanisms, not an autonomous coding model embedded in the repository.

Repair and forecast candidates

The neutral retry path validates a complete Lab experiment and its exact boundary bytes. Its finite patch grammar can replace a retry policy with 1 to 32 attempts and 0 to 32 units of delay. A proposal may differ from the interventions already observed, but the record does not claim that untested proposal succeeded.

Frozen forecast challenger records support statistical baselines, direct LLM, retrieval LLM and fixed-pool candidates. They bind implementation, prompt, retrieval index, calibrator, training manifest, cohort and evaluation protocol. SFT, DPO, PEFT and continual-training labels are supported as lifecycle metadata. They do not themselves execute weight training or establish forecasting skill.

Actual estimation from experience

The Route B fitter counts validated synthetic completed-job outcomes and computes the Bernoulli maximum-likelihood estimate. A retained cohort of 256 observations contains 163 successes and 93 failures, yielding 0.63671875. Its conditional 95 percent Hoeffding interval is 0.5518374 to 0.7216001. The result is calculated from observations and can be recomputed, rather than accepting a caller-supplied probability. [M3]

The retained-experience fitter adds linked observation identity, context revision, budget and producer checks. It rejects hidden world parameters, evaluation answers, future or expired history and implementation mismatches. Its one-day age limit is an explicit freshness rule, not a detector of silent distribution shift.

Verification: current-main verification record passed on Ubuntu and Windows. Inspected Ubuntu logs show 102 passed tests. New-process refitting and installed-package use reproduced the recorded candidate. The original candidate's evaluation field remains UNRUN because later evaluation is retained separately. No neural model weights or general self-improvement are established. [M3–M5]

Models packaging and integration contracts

Model packaging and registry candidates

`modelPackage` binds a model and parent to the training manifest, code and tokenizer, with declared capabilities and limitations. `registryCandidate` creates an awaiting-independent-evaluation state and denies production serving. This lets an integrator package a proposed model without conflating its existence with approval. [M6]

Sandbox specifications and mock workloads

`sandboxPolicy` declares CPU, memory, output and tool limits. `sandboxCommand` allowlists read, patch, test, lint, typecheck, build, diff and text search while excluding network and shell expansion. `sandboxResult` preserves completion, timeout or failure and output digests. These functions construct command and result specifications. Actual command execution and operating-system isolation belong to the execution host.

The explicitly named coding-sandbox mock hashes a patch and captures supplied visible-test records. The consolidation mock freezes episode identities and produces a non-training-eligible candidate. They are useful integration fixtures for checking component boundaries, and do not run a compiler or learn model weights. [M7]

Provenance replay and rollback

Replay manifests preserve the exact order and duplicate structure of digest arrays. Provenance records connect model, data, training and evaluation identities and retain negative evidence. `rollbackPlan` preserves the parent identity and requires independent authorization instead of mutating production directly. These records make a proposed reversal reviewable in the same way as a proposed successor.

Provider receipts and repair compatibility

Provider receipts distinguish succeeded, failed and termination-uncertain outcomes. A live-evidence receipt refuses a fixture declaration but validates supplied metadata rather than independently contacting the provider. A host must establish the authenticity of execution evidence.

Mission-bound repair candidates, executable finite retry patches and neutral experiment-bound proposals coexist. “Executable repair” identifies a bounded patch artifact that a downstream runner and evaluator may consume. It does not enable a candidate to modify, admit or deploy itself. These compatibility families allow older retained studies to remain readable while newer forward-only experiments avoid circular evidence dependencies. [M8]

WONDER Monte Carlo numerical toolkit

WONDER-Monte-Carlo is a deterministic TypeScript and JavaScript numerical library for evaluating a caller-supplied mathematical model repeatedly and summarizing its output distribution. Version 1.0.4-mlmc.1 provides an ESM library, an mce command-line program and an optional bounded worker adapter, with zero external runtime dependencies. [N1–N3]

What a developer can do with it

A developer can define a model, supply explicit inputs and a seed, run a reproducible scenario study, inspect convergence and tail summaries, compare model revisions or distribute bounded path ranges through a host-controlled worker. Reliability, finance and cost comparison are plausible uses of the primitives, not built-in validated forecasts.

Sampling statistics and numerical reproducibility

Counter-addressed Philox4x32-10 streams bind seeds, path identities and versioned stream domains. Canonical aggregation order avoids making a result depend on worker arrival order. Exact replay also requires the engine, generator version, model and inputs. Repeating the seed alone is insufficient.

The distribution toolbox includes uniform, Bernoulli, categorical, normal, lognormal, exponential, gamma, beta, Poisson, Student-t, triangular, empirical-discrete and multivariate-normal sampling, with parameter validation and Cholesky preparation. Streaming summaries include mean, sample variance, standard error, optional skewness and kurtosis, quantiles, bounded retained paths and explicit convergence results.

Tail-risk helpers compute empirical upper-tail loss VaR and expected shortfall using fractional tail mass. These characterize supplied samples. They do not establish adequate extreme-event coverage or real-world predictive accuracy.

Variance reduction and execution control

Implemented methods include antithetic pairs, fixed or disjoint-pilot control variates, stratification, Latin-hypercube sampling, importance sampling and randomized Sobol sampling. Uncertainty calculations use the relevant independent unit, such as pair means or full-design replicate means. The current Sobol implementation uses a 32-bit digital shift with Philox sub-cell jitter and a 1,024-dimension direction table, not Owen scrambling.

Requests bound path count, dimensions, workers, batches, retained values, input size and time. Progress, cancellation and complete, cancelled, failed or nonconverged outcomes are explicit. Exceptions, nonfinite outputs and changing metric schemas fail closed. Synchronous cooperative execution cannot preempt an infinite loop. A hard stop depends on the terminable worker or process boundary.

Verification: exact-head verification record succeeded across eight jobs, including Node 22 and 24 compatibility on Ubuntu, Windows and macOS and full verification on Ubuntu and Windows. The numerical registry separates selected implementation tests from broad statistical validation and independent reproduction. [N6]

WONDER Monte Carlo multilevel methods

Generic multilevel estimation

Multilevel Monte Carlo estimates a finite-level expectation by combining a base-level estimate with coupled fine-minus-coarse corrections. The generic implementation supports fixed and adaptive allocation. The adaptive method uses a separate pilot to estimate correction variances and costs, then allocates fresh production samples within cumulative path, work and time limits. Variance is assembled from per-level sample-mean variances. [N3]

This is a numerical variance-reduction method. The ML in MLMC means multilevel, not machine learning. It does not train a predictor, and it is not quantum computation. Its usefulness depends on a defensible hierarchy whose correction behavior and costs make the method worthwhile.

A native verifiable integration contract

The newer native fixed-MLMC interface exposes request identity, execution, result verification and an installed source descriptor. It currently supports one built-in synthetic polynomial-uniform hierarchy, rather than arbitrary caller models. Fine and coarse evaluations share a draw within a correction, while levels use separate stream domains. [N4]

The result verifier checks the envelope, installed descriptor, constituent receipts, level streams, sample allocation, costs and aggregate mean, variance and uncertainty. Tests compare analytic expectations, independently replay simplified corrections, recompute two-pass sample variance, reject mutations and check cancellation, zero variance and safe-integer cost overflow. Sampling intervals are approximate normal 95 percent intervals. Discretization bias remains separately unknown.

Mathematics and orchestration remain separate

Monte Carlo owns the generator, seed commitment, model, stream, path range and numerical-result identity. Core owns job and attempt identity, fencing, idempotency, cumulative budgets, durable state, outbox and provider execution. A retry can repeat an identical mathematical range under a new fenced attempt without changing its path identity. [N5]

Replay manifests, path ranges and checkpoints support caller-owned resumption. The library does not itself supply durable storage or distributed job authority. The Financial replay contract records a source-level integration contract with additional execution evidence pending. It is not proof of a live distributed restart service.

Choosing the right estimator

Healthy Energy uses the reviewed 1.0.4-mlmc.1 package but deliberately selects ordinary Monte Carlo for its inexpensive algebraic cost models. Financial's directly pinned numerical dependency is 1.0.2. These distinctions preserve both the library's broader capability and the exact method and version actually used by each application.

Monte Carlo developer interfaces

Model authoring and execution

The `createMonteCarloEngine` function accepts an explicit seed, generator, resource limits, executor and timestamp policy. Its `runSync` method executes synchronously, `run` executes asynchronous batches and `close` releases executor resources. A model declares its identity, version, optional digest and dimension budget, then returns finite numeric metrics from `evaluate`. Inputs are canonical-copied and frozen. [N7]

Requests choose path count, batching, workers, estimator outputs, retained samples, progress, cancellation, stopping rules and variance-reduction configuration. A developer can compute the expected request identity before execution, verify the full returned result against it and persist the package, model and inputs alongside the result.

Statistics and stopping rules

Streaming accumulators return nullable values when a statistic is undefined. Exact retained-sample quantile helpers sit alongside a deterministic Greenwald–Khanna sketch with an explicit accuracy parameter. Optional bounded-mean confidence sequences use `hoeffding-union-v1` and require declared finite metric bounds and the stated IID assumptions. Ordinary asymptotic confidence intervals remain a different object.

Fixed path count is the default. Absolute-MCSE and relative-MCSE stopping require a metric, tolerance, minimum count and explicit increasing checkpoints. A relative-scale floor can handle near-zero estimates. Convergence records distinguish a met rule from a maximum-path ceiling, runtime limit, cancellation or failure. [N8]

Variance reduction choices

Antithetic requests require an even count. Stratification declares strata per dimension and replicate count. Latin-hypercube and randomized-Sobol designs declare replicates and experimental acknowledgment. Control variates identify the target, control, known expectation and either a fixed coefficient or disjoint pilot. Importance sampling uses a named log-weight metric and standard or self-normalized estimation, with weight and effective-sample-size diagnostics.

Receipts path ownership and replay

Receipt APIs support deterministic identity, input and result digests and full request/result binding. Path-range APIs assert disjointness, compute a path-set digest, inspect coverage and bind a completed result to its range. Replay manifests preserve range ownership and completeness, reconcile runs and compare equivalent replay. Lightweight recovery and checkpoint validators compose with host-owned storage. Format-only validation explicitly returns `bindingVerified` false and must not substitute for the full verifier. [N9]

Random-stream interfaces expose path-indexed streams, named forks, open or closed uniforms, normal draws and dimension limits. The explicit random-seed helper is available when desired, but the engine and CLI never silently invent a seed. Retaining the chosen seed and algorithm identity is part of reproducibility.

Monte Carlo CLI workers and capacity

Command line tools

Command	Capability
mce run	Run a trusted default-export ESM model with canonical JSON input, explicit paths and seed, and optional output, progress, workers and statistics.
mce validate-model	Validate the model interface and hash exact entry bytes, reporting identity, dimensions and artifact size.
mce verify-receipt	Check receipt structure and optionally rebind supplied seed, model, input, configuration and result artifacts.
mce benchmark	Run the bundled domain-neutral numerical benchmark with selected paths, batches, workers and repeats.
mce version	Report the installed package version.

JSON results go to standard output or a new file. Progress and diagnostics go to standard error. SIGINT requests cancellation. Exit codes distinguish complete 0, cancelled 130, nonconverged 6 and failed 70. The CLI exposes the basic run interface. Advanced MLMC, confidence-sequence and variance-reduction capabilities are library APIs. [N10]

Worker execution

The optional worker package binds exact model entry bytes and offers a worker-pool executor plus engine adapter. It bounds workers, batch size, output metrics, startup, task and termination timeouts, diagnostic bytes and selected Node resource limits. Worker threads provide a terminable execution boundary, not a security sandbox: trusted model code may still access Node built-ins and the network.

Published admission ceilings

Resource	Default ceiling	Absolute override ceiling
Paths	10,000,000	100,000,000
Model dimensions	1,024	65,536
Output metrics	128	4,096
Runtime	60 seconds	24 hours
Workers	8	64
Paths per batch	1,000,000	10,000,000
Retained numeric values	10,000,000	100,000,000
Canonical input	16 MiB	256 MiB
Quantiles	32	1,024

Ordinary execution defaults to one worker and 4,096 paths per batch. The table shows admission limits, not measured throughput or a promise that every maximum is feasible simultaneously. Method-specific limits still apply, including Sobol's 1,024-dimension table. Retained values count paths multiplied by metrics. [N11]

WONDER Lab synthetic environments

WONDER-Lab produces reproducible synthetic worlds, experiment observations and replay receipts. Version 0.3.4-dev.retained.1 gives developers controlled environments in which to investigate system behavior before asking an evaluator to assess a candidate. It may consume versioned Core, Financial and Models artifacts, but does not import Evals. [L1–L3]

What a developer can do with it

A developer can construct a scenario, inject bounded faults, collect action and observation receipts, generate a model candidate, replay a run and preserve both successful and adverse interventions. The package supplies evidence to downstream reasoning and evaluation. It does not decide whether a candidate may deploy.

World families and systems behavior

Fourteen named families cover retry storms, queue saturation, cache stampedes, circuit-breaker and autoscaler oscillation, rate limiting, recirculation, memory and garbage-collection pressure, lock contention, database connection exhaustion, unstable feedback, self-confirmation, dampening and apparent-loop controls. These are simplified parametric development generators, not faithful replicas of fourteen production systems.

Event-time graphs distinguish temporal order from mechanism dependency. Strongly connected components and feedback-candidate extraction identify supported dependency loops without claiming that connectivity proves causal harm or gain. Fault schedules can represent latency, packet loss, stale cache, failed dependencies, retry amplification, lost capacity, sensor bias and action delay.

Routing worlds and numerical care

A seeded two-regime routing environment supports Route A, Route B and probes, with operation and latency budgets, digest-linked immutable receipts, idempotent repeated operations and conflict rejection. State transitions occur after observations, giving the controller an explicit filtering order. The retained-routing producer keeps hidden reliability parameters out of candidate-facing views and separates training and evaluation seed domains.

Shadow comparisons snapshot dense finite arrays before invoking a metric, reject mutation hooks and accumulate binary64 values using exact integer units before rounding the mean. This protects calculations from accessor side effects, overflow and cancellation. Held-out evaluation rejects explicit identity overlap and reports scale-aware RMSE without calling unverified isolation independent provenance.

Financial fixtures and evidence quality

Four deterministic portfolio worlds exercise stable gains, ranking reversal, missing terminal prices and downside outside planning support. Retry studies preserve improved, unchanged, worsened and unfinished outcomes. The general intervention helper transforms observations. It is not a rerun of the generating mechanism.

Verification: current-main verification record passed on Linux and Windows. Tests include independent Python arithmetic vectors, immutability, duplicate and conflict handling, hidden-parameter omission and budget limits. The process-local world store and development-visible fixtures do not establish protected holdouts or general cross-instance crash recovery. [L4]

Lab forecasting retry and workload APIs

Scenario interchange and real retry execution

Scenario and replay interfaces construct bounded scenario artifacts, project declared candidate views and bind scenario, intervention and outcome identities. The simulation fixture generates public-world observations and hashes serialized output. Internal truth helpers bind declared oracle material and remain outside the main candidate API. A candidate view is a data projection rather than a security boundary. [L5]

The retry-service simulator executes a bounded synthetic queue with attempts, capacity and outage, ready requests, completed, exhausted and timed-out requests, and completion rate. Baseline and intervention runs preserve improved, unchanged, worsened and unresolved results. The neutral v2 experiment validator rechecks that evidence. Because attempts and delay may change together and the capacity control is absent, the original experiment does not establish an isolated causal effect.

Forecast experiment accounting

`runSyntheticForecastTrial` processes supplied attempts for statistical-baseline, direct-LLM, retrieval-LLM and fixed-pool candidate kinds. It tracks forecasted, abstained, failed, cancelled, timed-out and not-run states, enforces reservation and closure accounting, retains incurred overruns and prevents duplicate accepted effects. Citation timing, coverage, unresolved rate and declared citation support are calculated from the supplied records. The fixture does not invoke the model or verify a provider bill. [L6]

Model and repair evidence construction

Dynamics candidates can name linear state-space, change-point, latent-state and nonlinear classes with explicit parameters and assumptions. Held-out helpers compare supplied predictions with observations. They provide useful candidate-management and discrepancy logic without claiming to fit all four model classes.

Repair evidence binds failure observations, competing explanations, opposing evidence and interventions for downstream Models and Evals use. New forward-only evidence avoids depending on a repair digest that does not yet exist, while the earlier constructor remains available for compatibility.

Operational fixtures and provider records

Fault schedules, transformed signals, adversarial actors with budgets, operational-failure scenarios and restart fixtures let developers test contract behavior under declared disturbances. They represent disruptions such as timeouts and restart. They do not cause packet loss or prove a real service recovered.

Provider receipts distinguish success, failure and uncertain termination. Live-evidence helpers bind nonfixture submitter declarations while explicitly retaining unverified authenticity. This preserves a place for real provider evidence without mistaking a timestamp or execution reference for an independent attestation.

WONDER Evals evaluation evidence

WONDER-Evals defines an evaluation boundary for candidate implementation, exact outputs, suite declarations and result identities. Version 0.4.12-dev.dt06-admission.1 includes repair and mission evaluation, routing and portfolio studies, numerical recalculation, custody and recovery protocols, and durable development feedback. [E1–E5]

What a developer can do with it

A developer can freeze a candidate and evaluation inputs, invoke a bounded installed evaluator, verify the full producer chain, retain a distinct result and release eligible evidence to a consumer. Candidate success claims are checked against observed artifacts rather than accepted as the evaluation itself.

Provenance and durable feedback

Repair evaluation recomputes Lab evidence and checks that Models identifies the same failure and intervention observations. Evaluator evidence distinguishes verified, unverified and contaminated status. Malformed, empty, nonfinite or mismatched numerical inputs are rejected. Separate identities help establish material separation, but are not automatically proof of independence.

The feedback ledger records reservation and exposure, terminal outcome and retirement, manual release and consumption. Immutable stage identities, predecessor links, atomic no-overwrite publication and exact readback address duplicate processing and lost acknowledgments. PASS, FAIL and INCONCLUSIVE remain scientific outcomes. Incomplete operational evidence does not silently become accepted training feedback.

Recorded evaluation examples

A fitted synthetic Route B study retained all 512 assigned outcomes and fresh-process readback. Its Brier criterion passed, but the pre-evaluation analysis showed that every complete binary cohort would pass that threshold. The demonstrated result is reproducible provenance and lifecycle integration, not a discriminating predictive win.

A later exact-rational utility diagnostic compared the actual policy with known-optimum and deliberately poor controls. It met its finite-model retained-advantage criterion at 40- and 60-tick roots while remaining slightly worse than the known-model planner. Expected-job gaps to that planner were approximately 0.000913 and 0.005354 respectively. [E3]

Retained-experience studies preserve adverse results as well: 88 paired comparisons and 176 actual Core executions in synthetic worlds produced 893 jobs with experience against 924 without it. Stable cases improved by 1.75 jobs on average, drift cases were adverse and equally weighted no-information cases averaged zero. This is research into when adaptation helps, not a universal learning claim. [E4]

Application handoffs and maturity

Financial export checks an already released and consumed result against the full tenant and application binding, preserving its status without requiring Financial to import Evals at runtime. Native and protected acceptance, learning eligibility and activation authority remain false for that development export.

Verification: current-main verification record passed on Linux and Windows. Protected holdout custody and hostile-code runtime isolation remain unestablished. The issue ledger retains unresolved historical aggregate lifecycle failures rather than erasing them because later runs passed. [E5–E6]

Evals lifecycle and application contracts

A complete result chain

The evaluation protocol connects a Lab-produced observation or mission, a Models candidate, an exact candidate output and a declared suite to a distinct evaluator result. Revalidation checks the full producer chain. In a repair study, the evaluator verifies that the proposal names the same failure and observed interventions, then emits a new mission-bound receipt rather than trusting the candidate's own score. [E1–E2]

Neutral retry diagnostics use bounded standard input and output. Routing and portfolio evaluators bind frozen inputs, candidate identities and declared controls. Numerical recalculation rejects empty, malformed or nonfinite values. Material-separation checks require unique, nonempty identities on both sides while preserving the difference between identity nonoverlap and independent provenance.

Feedback states and durable publication

Lifecycle stage	Purpose
Reservation and exposure	Record what evaluation material was assigned and exposed before accepting feedback.
Terminal outcome or retirement	Preserve PASS, FAIL or INCONCLUSIVE, or identify a terminally incomplete attempt.
Manual release	Make the eligibility decision explicit rather than inferring it from a successful process exit.
Consumption	Bind accepted downstream use to the specific released evidence.
Exact readback and reconciliation	Distinguish a committed immutable record from an acknowledgment that was lost.

A recorded repair to the development lifecycle preserved completed episode receipts under worker interruption, retained orphan receipts and kept incomplete attempts incomplete. Atomic hard-link publication became the explicit success point. Tests distinguish stopping before publication from losing the acknowledgment after publication. Power-loss durability and operating-system-enforced deadlines remain separate questions.

Financial and protected receipt interfaces

The DT06 importer verifies a pinned native-reader and oracle closure, binds the request, action and cleanup evidence, and passes through an installed Core strict validator and immutable store. Its offline modes label simulated observations rather than claiming native acceptance.

Financial export permits only already released and consumed evaluation evidence with a matching tenant, application binding and frozen candidate. Installed CLI tests check unchanged ledgers, repeat output identity and rejection of unreleased results, wrong tenants, substituted packages, missing cleanup and unsupported multiple-evaluation scope. This is a concrete artifact handoff, not a runtime requirement that every Financial process imports Evals. [E5]

Evals scientific and diagnostic catalogue

Predeclared suites and causal repair scoring

Suite and submission APIs bind cases, metrics, thresholds, CPU budget, candidate implementation, output and result. Numerical aggregation uses exact binary-unit accumulation before final rounding. Thesis-kill criteria and suite-rotation manifests record a proposed evaluation lifecycle without themselves scheduling hidden evaluation or terminating a process. [E7]

RCRE scoring covers trigger, amplifier, enabler, propagator and missing-inhibitor identification, feedback classification, intervention quality, trajectory improvement and source mapping. It includes a false-causal-attribution penalty and a budget flag. Resource measurements are supplied through a constrained result channel, not observed independently by the score helper.

The evaluator-owned retry path actually runs admitted declarative patches against synthetic cases covering failure mechanisms, misleading temporal correlation, benign feedback, transfer and regression noninterference. Neutral experiment-bound evaluation uses five source-visible cases, checks exact experiment, candidate, repository and patch identities, compares completion rate and attempt efficiency, and caps simulation steps. Latency is reported but not scored. Cancellation is checked before execution. [E8]

Forecast cohort and portfolio evaluation

Forecast protocol APIs create visibly unconfigured drafts, freeze configured protocols and intake bounded idempotent artifacts. The current cohort evaluator supports FRED monthly unemployment, tracks resolved, pending, ambiguous, cancelled and excluded cases, verifies cluster membership and checks exact empirical-CRPS identities supplied through Core replay. It does not add an uncertainty bound or permit model promotion without independent prospective evidence.

The finite portfolio oracle independently calculates two assets plus cash in CAD/USD using BigInt arithmetic, fixed and basis-point fees and ties-to-even rounding. It checks feasibility, terminal wealth and P&L and compares serialized Financial outputs without importing the candidate's runtime code. Terminal-only prices do not provide pathwise drawdown. [E9]

Evidence diversity and operational assessment

Evidence-channel summaries accommodate deterministic verifiers, formal checkers, statistical benchmarks, model judges, cross-language oracles, human review, simulation truth and future outcomes. They count channels and principals while preserving unresolved status. The labels do not mean all those services are connected.

Custody, contamination, isolation and restore helpers validate declarations and scope. Operational assessment categories cover timeout, duplicate completion, lost results, restart, transaction rollback, outbox redelivery, corrupt or missing artifacts, provider outage, budget rejection, quarantine and restore. These records make failure evidence comparable without claiming that every named cloud service has undergone live fault injection. Explicit mocks remain mocks. [E10]

WONDER Engineering repair workbench

WONDER-Engineering is a local application with a chat-style browser interface, shared CLI and SQLite history. Version 0.1.0 turns a bounded software-repair investigation into a sequence of interventions, measured checks, retained counterexamples and explicit local review. Its interface binds to loopback on port 4319. [G1–G4]

What a user can do with it

The user can inspect one of three admitted repair cases, run a candidate, compare results, replay evidence, propose a correction, review it and export the record. CLI commands include inspect, run, replay, propose, history, show, open, export, review, cancel and reset. Review records a local decision. It never applies, pushes, merges or deploys the repair.

Measured repair progression

Case	Recorded comparison
Clamp	Baseline, rejected candidate and repair behavior demonstrate the bounded workflow.
Theta	A known dividend-sign defect passes 20 of 32 baseline checks. Either one-sided correction passes 24. Correcting both signs passes all 32.
Observer	The baseline passes 9 of 13 checks. An overbroad edit passes 10, a split-clock edit passes 12 and the final correction passes all 13.

The split-clock regression remains part of the evidence. Candidate grammars permit only predefined edits to exact source pins, not arbitrary code, shell commands, paths or replacement of the test oracle. Minimum-edit statements therefore apply to the enumerated candidate set, rather than all possible repairs.

Durable review and controlled execution

SQLite atomic commits retain original revisions. Restart restores evidence and review state without rerunning the code or granting approval. Source, oracle, policy, implementation or environment changes make earlier evidence stale and unreviewable. Interrupted owner states become UNKNOWN. Reset changes selection rather than deleting evidence.

Fixed reviewed sources run in fresh temporary directories with scrubbed environments, no shell, process timeouts and bounded output and heap. Limits include at most four test processes and sixteen seconds for a whole run. The HTTP service checks loopback host and origin, cross-site requests, session token, strict JSON and UTF-8, body size and security headers. These are controls for a bounded trusted-source workflow, not a hostile-code sandbox.

Two unmodified pinned Core packages supply reasoning-gateway and model-assurance functionality. Vendored archives and per-file provenance enable offline installation and verification. The current application does not automatically connect to an LLM or run arbitrary repair tasks.

Verification: exact-head verification record passed on Linux and Windows. Regression coverage includes denied grammars, stale approvals, interrupted work, replay and tampering. Historical local Edge evidence covers ten browser groups. It was not rerun for this report. Storage is local plaintext with bounded capacity, not a synchronized multi-user service. [G5]

Engineering inspection and ownership interfaces

Grounded checkout inspection

`inspectCheckout` uses bounded Git operations to inspect the application's fixed checkout and admitted source definitions. Optional locks and filesystem monitoring are disabled for the inspection. The caller cannot select an arbitrary repository, command or shell. This makes the intake reproducible and tied to the cases the workbench can actually evaluate. [G6]

Workspace state and exact evidence binding

`EngineeringWorkspace` tracks bounded candidate attempts, staged events, cancellation, review, export and explicit FAILED or UNKNOWN outcomes. A provider-adapter abstraction is present, while the supplied gateway is deterministic fixture mode. `bindingFor` and `assessBinding` connect source, criteria, implementation and environment. `verifyExport` checks semantic content rather than accepting a recomputed outer hash alone.

Replay and ordinary reopening

`replayProjection` and `replayArtifact` reexecute an admitted bounded case and compare the new projection with the saved artifact. Opening history or restarting the application is different: it restores retained state without automatically rerunning code. This lets a user inspect an old decision without changing the evidence it was based on.

Concurrent durable ownership

`EngineeringHistory` and `DurableEngineeringWorkspace` coordinate selected runs, owned execution, revision snapshots, expected-digest review, cancellation, abandonment and durable history. If persistence or ownership inspection is uncertain, the application returns a retry or error state instead of granting a second execution owner.

Process ownership uses Linux boot and start identity or Windows process-start identity, rather than trusting a PID alone. Inspection distinguishes unavailable, absent and observed states and is bounded with timeout diagnostics. These controls support local coordination across restarts and concurrent requests. They do not establish hostile-process containment. [G7]

Local history has explicit capacity limits: 50 runs, 24 revisions per run, 64 KiB per artifact and a 128 MiB database plus possible rollback-journal overhead. Capacity refusal preserves records rather than deleting evidence. Package verification checks archive and installed integrity, syntax, consumer declarations, formatting and regressions. It does not claim full static type checking of all JavaScript.

Interface parity

The loopback API exposes session, state, history, run, cancel, reset, open, review and export operations. The CLI shares durable history and adds inspect, propose and replay. Browser smoke evidence includes keyboard and mobile-emulation paths, while offline verification checks application behavior and packaged Core boundaries. This gives a developer both a review interface and an automation-friendly local command surface without creating a deployment path.

WONDER Financial numerical laboratory

WONDER-Financial is the ecosystem-powered domain application that combines WONDER numerical and decision components with its own substantial quantitative research library and authenticated workspace. Its local numerical surface is substantially broader than the set of methods admitted to the hosted interface. The reviewed package is `@wonder/financial-research-core 0.64.0-rc.1`, at main revision `8ecc424bcc071e6fda05764530bb629f6358b004`. [F1–F3] This is financial research and scenario analysis, not personalized investment advice, brokerage or order execution, tax reporting or regulatory certification.

Pricing calibration and risk

Capability family	Implemented local research surface
Option pricing	Black-Scholes and CRR/binomial. Geometric and arithmetic Asian methods. GBM barriers. American least-squares Monte Carlo. Analytic/Fourier and Monte Carlo Heston.
Volatility and calibration	Historical volatility. Flat Black-Scholes and Heston calibration. SVI slices and surfaces. Static-arbitrage diagnostics. Bid/ask quality and repair. Constrained SSVI and monotone theta curves.
Rates and fixed cash flows	Deposit and par-swap curves. Regular schedules and calendar conventions. Fixed-rate cash flows. OIS/RFR compounding. Projection curves and multicurve swaps. Bucketed rate risk.
Portfolio risk	Historical and Gaussian VaR and expected shortfall. Correlated GBM. Linear and nonlinear revaluation. Joint spot, volatility, rate and FX factors. Yield-curve PCA. Calibration uncertainty.
Backtesting and performance	Causal single- and multi-instrument engines. Point-in-time universe representation. Execution, fee and volume constraints. Time-weighted and money-weighted returns.

These capabilities are grounded in the public exports and the local server's actual imports and routes, rather than inferred from a list of planned screens. The package pins `@wonder/monte-carlo-engine 1.0.2` and `@wonder/judgment-engine 2.0.1-rc.1`. A newer upstream Core or Monte Carlo revision does not silently change those installed contracts.

What a developer can do with it

A developer can construct a dated market or portfolio state, choose an explicit pricing or risk method, run a seeded numerical study, inspect diagnostics and preserve the inputs used. The laboratory supports comparing numerical methods and investigating where a method fails, including Greek estimation and convergence studies. FX tools check conversion networks and covered-interest-parity forward consistency.

The repository retains limitations alongside successful results. Adaptive Heston QE multilevel Monte Carlo and randomized-unbiased feasibility remain blocked by their research evidence. Tests preserve nonmonotone bias-envelope and seed-overlap rejection behavior. This makes the library useful for method development without presenting every experiment as an approved production method. [F9]

WONDER Financial decision workspace

A user interface backed by canonical producers

The workspace lets users import reviewed settled-equity trade and quote CSV files, inspect diagnostics, build positions and cash, value holdings with dated supplied marks and compare fixed plus or minus 10 percent spot scenarios. Rates screens build deposit and ordered par-swap curves, generate schedules and discount dated flows. Hosted pricing is bounded to European Black-Scholes and seeded European Monte Carlo. The risk form computes historical VaR and expected shortfall from explicit same-horizon P&L observations. [F2–F4]

Lifecycle accounting retains opening state, events, closing state, cash postings and replayable evidence. It supports identified-lot partial closes, rational stock splits, same-currency cash dividends and compensating reversals. BigInt-backed decimal arithmetic separates fees, financing, rounding and projection differences in dated P&L. Unsupported or incomplete lifecycles are rejected rather than filled in by inference.

Saved work and decision planning

The authenticated platform service and tenant-bound worker provide immutable artifacts, idempotent saves, history and reopen, bounded enqueue and replay, cancellation, eligible dead-letter redrive, and lease and fencing controls. Session Results keeps up to ten responses in browser-tab memory. This is distinct from durable saved artifacts.

Decision Trials turns two-asset CAD/USD assumptions, two finite terminal-price scenarios, weights, fees, a loss constraint and a horizon into a reviewable comparison of hold, cash and adjusted allocation. A separately confirmed freeze binds the exact inputs, planner result and action. Native Core Decision Lens shows the actual advisor state. An offline verifier checks retained plan and trial bytes and semantic identity. A frozen plan remains planning only. It does not establish trade execution or an observed outcome. [F5]

Forecast accountability and contextual data

Default-off, tenant-scoped readers expose retained forecast locks, captured and admitted outcomes, settlement, policy, provenance and exact rational score evidence. The supported pilot baseline is unchanged-last-value UNRATE. Read-only export preserves a validated loaded snapshot, not a new database read, fresh membership check or guaranteed disk save. Synthetic restore and restart checks compare exact reader output. These readers do not create a forecast or begin scoring merely when opened. [F6]

Public-data tools retrieve specified FRED series, dates and vintages. Issuer tools retain SEC facts, submissions and history with replay. Demo quotes remain fixture-backed. These functions support evidence inspection, not a streaming market feed or autonomous trading agent.

Verification: full PR-head verification record passed before the corresponding Financial change was merged. The merge-head verification record passed lightweight integrity and control-contract checks, while full verification, Windows, PostgreSQL and container jobs were skipped. These are different evidence scopes. Historical hosted acceptance covers a bounded saved-workspace flow, not current deployment of every newer feature. [F7–F8]

Financial market data and accounting

Instrument identity and provider operations

Financial's foundation represents stable instrument identifiers, exchange, currency, contract multiplier, tick size, time zone and calendar, plus underlying, expiration, strike and exercise style where applicable. InstrumentMaster rejects duplicate identifiers. VersionedInstrumentMaster preserves revision history, increasing effective timestamps and deactivation without changing identity. The vocabulary covers equities, ETFs, indices, FX, commodities, futures, options, bonds, crypto and cash. Pricing and accounting support remain method-specific. [F10]

Quote and bar normalization, staleness thresholds, provenance-aware cache keys and freshness checks allow research to distinguish eligible inputs from old or malformed observations. Provider wrappers add bounded retry, fallback provenance, caching and metrics. The shipped demonstration provider remains a fixture, while real-provider use requires the appropriate integration and data rights.

Issuer research with retained sources

The stock research workflow combines explicit instrument and entity bindings, observed quotes and retained SEC scalar fundamentals. It selects fiscal periods, reports ambiguity and exclusions, retains filing links and source history, and replays the exact stock artifact.

Research free cash flow is derived only when operating cash flow and nonnegative cash capital expenditure satisfy the required fiscal duration, currency, accession, date, form and complete-import conditions. Mixed-period derivations are refused. The resulting stock artifact explicitly leaves forecasting unsupported and fair value unestimated instead of inventing an investment recommendation. [F11]

Corporate actions and retained lifecycle accounting

Reusable historical helpers normalize splits and dividends, split-adjust OHLC and volume, construct dividend-inclusive total-return indices and adjust position quantity, cost or distributions. These tools are separate from the newer exact-decimal lifecycle record.

The retained lifecycle binds an opening anchor, validated dated events, closing state, marks and replay evidence. Supported events include bounded partial close against an identified lot, exact rational stock split, same-currency cash dividend and compensating reversal of specified dividend or external-financing postings. Dated P&L separates opening holdings from new lots, trade cash from external financing, and fees and settlement rounding from floating-point projection differences.

This allows a user or developer to explain where a change in portfolio value came from and which source events support it. Unsupported currencies, incomplete endpoints, collateral and undisclosed events are explicit refusal cases. The record remains research accounting rather than tax-lot or certified investment-performance accounting. [F4, F11]

Practical workflow

Capture the source and dated market state, resolve instrument identity, inspect input diagnostics, apply explicit events, calculate the result and retain the exact artifact. Replaying that artifact can then distinguish a changed assumption from a changed implementation or an actual new market observation.

Financial calibration and uncertainty research

Options and rates

The library provides arbitrage price bounds and bounded-bisection implied volatility, analytic and finite-difference Greeks, binomial and Asian, barrier and American least-squares methods. Curve tools include flat, dividend, discount and projection curves, deposit–swap and OIS construction, fixed bonds, zero-coupon values, spot-start and multicurve swaps, RFR conventions and business calendars. The local Models, Pricing and Rates forms expose research workflows in addition to callable APIs. [F12]

Volatility surfaces and quote repair

The volatility workflow includes deterministic bounded multistart raw-SVI fitting, maturity-slice construction, finite-grid call-bound, vertical-spread, convexity and calendar checks, quote quality, and bid/ask-weighted preparation. Constrained quote and surface repair preserves residuals and adjustments so the researcher can inspect what changed.

SSVI support includes total variance, a power-law phi function, sufficient-condition diagnostics, monotone theta curves and serialized surface identity. A finite-grid check has a finite scope, and any stated SSVI guarantee remains tied to its conditions and calibrated domain. The value is a traceable calibration and diagnostic workflow, not an unexplained arbitrage-free label. [F13]

Factors correlation and curve uncertainty

Historical dependence estimators include sample correlation, fixed shrinkage, automatic OAS and EWMA. Yield-curve PCA returns covariance, eigenvalues and loadings, retained variance, factor definitions and a scenario template. These can drive shared correlated GBM scenarios, linear or nonlinear portfolio revaluation and joint spot, volatility, rate and FX studies. FX tools check direct, inverse and triangulated conversions and covered-interest-parity consistency.

Calibration uncertainty uses IID and moving-block residual bootstraps, parameter and correlation intervals and boundary and failure diagnostics. PCA uncertainty additionally tracks eigenvalue, loading and drift intervals, sign alignment, eigengaps, loading cosine similarity and retained-subspace distance.

Nested parameter and curve-risk envelopes distinguish a fixed-calibration baseline, between-calibration variation and repeated inner Monte Carlo variation. They retain draw, scenario and risk lineage and guard the total path count. This decomposition lets a researcher see how much output variation arises from fitting versus simulation, within the chosen model. [F14]

What a researcher can do with it

Inspect a quote grid, diagnose inconsistencies, fit a bounded parameter set, record residuals, generate scenarios and compare portfolio sensitivity under calibration uncertainty. These methods support conditional numerical research. Structural-model error, unmodeled regimes and regulatory capital approval require different evidence.

Financial strategies and advanced estimators

Causal backtests and execution assumptions

Single- and multi-instrument backtests expose bounded historical views to the strategy, with explicit next-open, slippage, fees and execution assumptions. Volume-limited fills, per-contract minimum fees and dated universe memberships make trading constraints and survivorship status visible. Moving-average research can therefore be evaluated without giving the strategy unrestricted future observations. The versioned route also requires adjustment provenance. [F15]

Performance and exposure analytics

Return utilities calculate simple, logarithmic, period and total returns. Performance summaries include compounded cumulative and annualized return, annualized volatility, Sharpe and Sortino ratios, maximum drawdown, Calmar ratio and hit rate. Undefined ratios remain null. Fill analytics report turnover, realized-event hit rate, gross gains and losses and profit factor.

Time-weighted returns declare start or end timing for cash flows. Money-weighted returns use a bracketed internal-rate-of-return solve with convergence and residual reporting. Exposure summaries distinguish gross, net, long and short and group by asset, currency or sector. Derivative risk exposure must be supplied explicitly rather than inferred from an option's premium. [F15]

Multilevel and stochastic volatility research

Continuous arithmetic-average Asian GBM MLMC reports estimate, sampling interval, estimated bias and RMSE, convergence diagnostics, levelwise work and sample counts, and termination reasons. Job and integrity-bound checkpoint helpers support interrupted research. Experimental continuous up-and-out call and European or Asian Delta and Vega methods retain their restricted scope.

Heston full-truncation MLMC has a screened parameter envelope and an explicit high-risk override. Ordinary Heston QE is distinguished from QE-M and QE-MLMC. Additional library research examines coupling strategies, transition quantiles and CDFs, adaptive-quadrature cross-checks, branch mismatch, Rao–Blackwellization, replicated weak-bias identification, information allocation, held-out cancellation, simultaneous tail envelopes and randomized-unbiased feasibility. A justified refusal to promote a method is retained as a research result. [F16]

Three distinct application layers

The loopback application has Portfolio workspace, Advanced pricing, Models, Rates, Market, Accounting tools, Risk, Portfolio MC, Stress, Backtest and System tabs. Its versioned runtime contract, computation identities, structured logging, metrics, process-local asynchronous jobs and numerical checkpoint files support local research.

The authenticated platform separately supplies tenancy, durable worker execution, Decision Trials, native Lens and forecast readers. The reusable numerical library can be consumed without either interface. This division gives Financial substantial breadth while keeping each user interface's admitted methods and operating guarantees explicit. [F3, F17]

Healthy Energy as a working application

Healthy Energy is the clearest inspected example of WONDER components being reused in a separate business application. Its value as a portfolio case is the complete workflow: research becomes attributed evidence, evidence supports structured review, and accepted inputs feed business decisions and numerical comparisons. [HE1–HE5]

Business workflow and operating software

The application uses SQLite to persist operational records. Its reviewed release includes source capture, reviewed supplier quotes, ingredient and inventory lots, quarantine, recipe costs, recorded sales and expenses, and profit calculations. An owner-authenticated hosted mode sits alongside a loopback-only local mode. Revision controls, idempotency, append-only validation history and backup and recovery tools help preserve the operating record.

The application also organizes business-plan documents, research records and validation items with versioned inputs and download integrity checks. Those records support review but do not establish product safety, customer demand or approved commercial returns. Confidential formulations, supplier details and internal business economics are outside this public software case study.

WONDER components used in the application

The source contains 32 vendored MIT-licensed Core files pinned to revision b380d2441af85ab07790b326127ef901b6971c2b. They provide an evidence registry, a judgment evaluator and a receipt verifier. A reasoning gateway supports selected supplier-text extraction with optional AI. Manual workflows remain available.

The decision path checks captured-text hashes, active source status, timestamps, quote identity and units, human review and expiry. Scoped ingredient constraints are evaluated separately from the weighted score. A high score cannot override a hard constraint, and invalid or stale sources can block eligibility. Human attestations are kept as attestations rather than promoted into scientific facts.

AI within a bounded workflow

AI is used for selected text-processing assistance, under source-specific consent and bounded jobs. It does not replace the stock ledger or determine numerical truth. Earlier local-provider evidence is recorded in the project. This portfolio review did not make a fresh live provider call. The application demonstrates reuse of selected Core functions, not integration of the entire WONDER ecosystem.

Application evidence: current reviewed source is d30219ead3091df1c3ca6bd0575b24950b32e1d8. Retrieved PR-head CI passed workspace verification, container build and a missing-authentication refusal check. Additional Windows, browser and recovery results are author-reported. This review did not rerun tests or verify current hosted deployment acceptance. [HE1]

Healthy Energy numerical decision study

A separate launch-options study compares existing business alternatives under explicit assumptions. Its public value is the software method: equations, input definitions, paired sampling and comparison results can be retained for review. Confidential scenario economics are omitted here. The study is distinct from the operating app, which uses deterministic costing. [HE3–HE5]

Method and reproducibility

The study uses WONDER-Monte-Carlo revision 1b30c104e008003df98c7635fbaf1d47a2f63c4b, package version 1.0.4-mlmc.1. The selected method is ordinary Monte Carlo: four factors and 4,096 paired paths. Multilevel Monte Carlo was deliberately not used because these inexpensive algebraic cost models do not supply a defensible hierarchy of coarse and fine approximations.

The implementation checks independent accounting equations and a two-pass aggregation against the same random draws. This is a useful consistency check, not an independent random replication. The earlier 10,000-path study was recovered and preserved rather than rerun. All 16 corners of the four-factor input range were also evaluated.

Technical evidence retained by the study

Technical record	What it demonstrates
Four uncertain input factors	The simulation operates on an explicit, bounded scenario definition.
4,096 paired paths	Alternatives use common draws for a controlled model comparison.
All 16 range corners	The study also checks extreme combinations of the stated four-factor bounds.
Two-pass numerical check	Aggregation is checked against the same draws, not represented as independent random replication.
Retained earlier study	The prior study is preserved rather than silently rerun or overwritten.

The retained outputs describe conditional model scenarios rather than observed sales, realized profit or a measured distribution of customer demand. Monte Carlo sampling error describes the precision of simulation within the declared model. It does not measure whether the commercial assumptions are true. No confidential dollar outcomes are reproduced in this public edition.

Explicit local comparison updates

The current merged source supports immutable reviewed input versions, comparison previews, explicitly requested bounded execution and separate human admission. It allows 32, 64, 256, 1,024 or 4,096 draws. Opening or refreshing the app never starts a simulation. Hosted operation refuses this workflow's create, run, cancel and admit actions. The frozen historical study remains unchanged. Unknown execution outcomes fence later runs rather than silently retrying. [HE6]

Learning reasoning and governance

WONDER contains several distinct forms of intelligence-related software. Keeping them separate makes its capabilities clearer: numerical estimation, language-model inference, decision analysis, experience reuse and permission control answer different questions.

Capability	What the portfolio establishes	What it does not establish
Numerical simulation	Seeded Monte Carlo and multilevel estimators with explicit targets, uncertainty and replay identities.	A trained ML model or real-world forecasting accuracy.
Parameter learning	Models fits a Route B probability from observed synthetic outcomes and preserves the training record.	Neural weight training or general reasoning ability.
Experience reuse	Context- and age-bound candidate reuse with actual Core consumers and separate evaluation.	Reliable benefit under arbitrary distribution shift.
Language-model assistance	Core provides an injected-provider gateway. Healthy Energy uses bounded optional extraction.	An autonomous general-purpose LLM product inside every repository.
Decision reasoning	Explicit utility, constraints, viability, scenario criteria, value of information and abstention.	Truth of a source merely because it is hashed or scored.
Governance	Typed preparation, approvals, policy checks, lineage and observed-effect contracts.	Consciousness, self-created authority or blanket permission to act.

The demonstrated learning loop

Lab supplies observations. Models produces a reproducible fit. Core consumes a pinned parameter under a versioned policy. Evals evaluates an explicitly frozen claim. The original 163/256 fit and the later utility results remain separate artifacts. This is a real bounded experiment in learning and adaptation, not merely a collection of training-manifest labels.

The retained-experience study is especially useful for development because it preserves where adaptation failed. Its stable, drift and no-information cases expose different behavior instead of averaging them into a universal improvement claim. The next technical question is whether context-change detection and selective reuse can improve outcomes under a prospectively frozen evaluation.

A longer term research direction

OGI and artificial conscientiousness remain founder-defined research ambitions. The current contribution toward those ambitions is concrete software for memory identity, evidence handling, controlled adaptation and explicit responsibility boundaries. This report makes no claim of demonstrated consciousness or general intelligence.

Implementation stack and engineering practices

Layer	Observed technologies and patterns
Foundation code	JavaScript ESM with public declarations across Core, Models, Lab and Evals. TypeScript and JavaScript in Monte Carlo.
User interfaces	Browser modules and semantic DOM. Core Decision Lens. Financial authenticated workspace. Engineering loopback chat-style UI.
Service interfaces	Node HTTP, bounded JSON contracts, installed package APIs and CLIs. Injected providers rather than embedded credentials.
Numerical correctness	Seeded stream identities, exact rationals and monetary decimals, independent arithmetic references, explicit tolerance and uncertainty contracts.
Persistence	SQLite for Engineering and Healthy Energy. PostgreSQL-backed platform and rehearsal paths. Immutable file-backed development artifacts.
Identity and authorization	JWT and host/origin controls in the bounded Core service. Membership-derived tenant scope in Financial. Explicit action and policy bindings.
Packaging and provenance	Pinned package archives, vendored files, source integrity manifests, exact readback, public consumer checks and reproducible installed-package tests.
Continuous integration	Cross-platform GitHub Actions. Offline verification and installed consumers. Separate numerical, runtime, browser and container evidence.

Toolchain and version discipline

Inspected current foundation workflows use Node 24.21.0 and npm 12.2.0. Full verification in relevant repositories pins Python 3.14.8. Some stable CI job names retain an older Node version for branch-protection compatibility. The actual workflow and executable version check are the authoritative evidence. Monte Carlo additionally verifies Node 22 and 24 compatibility.

Consumer pins are part of the architecture. Financial's directly declared Monte Carlo dependency is 1.0.2 even though the numerical repository's inspected head is 1.0.4-mlmc.1. Healthy Energy explicitly pins its own source and numerical package. Treating every upstream main branch as an automatic application upgrade would erase useful reproducibility information.

Scope of the review

This portfolio retains a detailed review of repository metadata, API exports, selected implementation and regression files, CI records and saved execution reports. The current edition refreshes repository and release status and checks the application boundaries. It is not an exhaustive line-by-line code, security, licensing or scientific audit. Source code and operational records remain private.

Verification and maturity matrix

The matrix summarizes the strongest inspected evidence at the review date. A successful test run is evidence for its tested scope, not a general production certification. Counts from overlapping suites are deliberately not added together.

Repository	Implemented delivery form	Inspected verification	Current maturity
Core	12 packages. Bounded Research API. Controlled local workflows	Merged-main CI passed. Separate draft supervisor candidate has its own dated status. Installed consumers tested on the identified revisions.	Tested modular platform. Bounded API and synthetic integrations. Current-head deployment not verified.
Monte Carlo	Offline library, CLI and bounded worker adapter	Main verification record. 8 successful compatibility/full-verification jobs	Implemented numerical toolkit. Selected analytic and replay checks. Broader independent statistical validation pending.
Models	Candidate APIs, fitters, stores and CLIs	Main verification record. 102 passed tests in inspected Ubuntu logs	Implemented bounded estimation and provenance. No trained neural weights established.
Lab	Synthetic worlds, experiment APIs and receipts	Main verification record. Independent arithmetic references	Implemented development environments. Not protected holdouts or faithful production simulators.
Evals	Installed evaluators and durable development evidence	Main verification record. Recorded routing and feedback studies	Implemented evaluation contracts. Protected custody and hostile-code isolation remain open.
Engineering	Loopback UI, CLI and SQLite history	Main verification record. Retained browser evidence	Working local bounded repair workbench. No automatic patch deployment.
Financial	Quantitative library. Authenticated application and worker	Full PR-head verification record passed. Merge-head verification record ran lightweight checks only.	Broad source-qualified application. Bounded historical hosted acceptance. Newer paths not live-rechecked.

Evidence that remains intentionally separate

Core's local pilot rehearsal is separate from a live forecast. Financial's historical acceptance is separate from current release acceptance. Healthy Energy's merged source and reported tests are separate from a freshly verified hosted deployment. Draft Core branches are separate from main and from the pinned Core copy used by Healthy Energy. A newer upstream branch does not automatically change an application.

Prioritized development roadmap

The following priorities are recommendations based on the reviewed capabilities, not claims of work already completed or commitments to dates. They favor stronger demonstrations and easier adoption of the existing software before expanding its stated scope.

First establish repeatable product demonstrations

Package a small set of replayable demonstrations around the strongest completed workflows: Core's three-option decision example, Engineering's observer repair, Models' reproducible fit, Financial's saved-workspace and planning flows, and Healthy Energy's launch comparison. Each demonstration should identify exact source, inputs, expected output and the boundary between local and hosted evidence.

Then qualify integrated release candidates

Resolve version drift explicitly and record which packages each application consumes. Verify a proposed release against its exact source, deployment identity, authenticated user journey and recovery path. Complete the current supervisor and source-integrity qualification before relying on the candidate. Keep cross-host clock ordering and evidence-acceptance prerequisites visible rather than relaxing the gates to obtain a passing result.

Strengthen scientific evaluation

Freeze evaluation criteria that can meaningfully distinguish a good model or policy from a weak control. Preserve negative and inconclusive results. Develop protected custody and appropriate runtime isolation where the evaluation needs adversarial independence. For retained experience, test context change and selective reuse under frozen stable, drift and no-information conditions rather than assuming recent data is relevant.

Broaden numerical validation where it changes use

Prioritize independent replication and statistical qualification of the exact Monte Carlo package bytes used by applications. Retain separate evidence for random generators, variance reduction, numerical stability and domain calibration. Expand native MLMC beyond its synthetic hierarchy only when a target model provides a defensible level structure, bias treatment and economic reason to use it.

Make adoption easier without weakening controls

Publish concise integration guides for evidence registration, judgment review, installed numerical use and result replay. Add application-specific examples with explicit input schemas and expected refusal states. Preserve deterministic operational ledgers in new applications while introducing advisory AI or numerical analysis through bounded interfaces.

Healthy Energy research section text

This section provides a concise case-study presentation suitable for the Healthy Energy app or a public introduction. It describes software development and use, not scientific validation of a food product.

How WONDER supported this app

Healthy Energy Sask shows how selected components from the WONDER software ecosystem can support a practical business workflow. Developed under Aaron Moyer's direction, the app combines source review, costing and operating records in one workspace. Prairie Simple Software is the proposed business identity for this software work.

WONDER components help organize captured evidence, produce reviewable advisory judgments and route optional AI-assisted extraction. The application adds the interface and records needed for the Healthy Energy workflow. This is component reuse, rather than a claim that WONDER autonomously created or operates the business.

For example, a supplier quotation can be saved with its source. AI can prepare a draft of relevant values and supporting text. A person still checks ingredient identity, units, price and missing information. A research lead does not automatically become an approved quote or eligible stock.

The app keeps scenarios separate from entered sales and expenses. A retained numerical study shows reproducible comparison under stated assumptions. A later local workflow allows explicitly requested bounded comparisons and separate human admission. Opening the app never starts a simulation, and hosted operation does not execute that local workflow. Assumptions remain distinct from observed demand or proven profit.

A wider purpose

The intended wider use is to help entrepreneurs and other users organize evidence, compare choices and decide what to test next. Business development is one application of a broader research and decision-intelligence system. The same architecture can support other domain tools when their methods and safeguards are qualified for the intended use.

People remain responsible

Aaron directs the project, using AI-assisted development and research tools. The app's optional language-model features use third-party providers. Outputs require review. This software case study does not establish product safety, clinical benefit, environmental performance or approval for sale. Those questions need their own evidence and appropriate professional review.

Prairie Simple Software. Complex questions. Clear next steps.

Practical uses and responsible adoption

The portfolio offers reusable technical building blocks, not a promise that a new business can be launched automatically. The most useful next application is one with a clearly defined question, known inputs and a result that a person can verify.

An entrepreneur example

A prospective owner wants to compare supplier options. First, capture the dated quotations and check the unit and landed-cost assumptions. Then compare a small set of scenarios, identify the missing facts and decide whether another quotation or a limited test is worthwhile. Retain the result and later record what happened. This is an illustrative workflow, not a claim of measured savings or a currently packaged universal startup service.

A researcher example

A developer supplies a mathematical model with explicit assumptions and a reproducible seed. The numerical engine estimates the output distribution and records convergence. The developer compares it with an analytic reference or independent implementation where available. A narrow passed test should be reported as that test, rather than as universal validation of the method.

An integration example

An application team may need only evidence registration and a judgment view. It can select those packages without adopting every domain application. Before release, the team should check the exact dependency versions, permitted data flows, refusal states and recovery behavior. New upstream work should be admitted explicitly rather than silently changing old results.

How to judge the experience

Simple needs observable evidence. Test whether a reader can find a source, understand the assumptions, recognize an unknown and explain why a result was refused. Compare task completion and correction needs with a declared baseline before making claims of speed, accuracy or usability. Keep failures and inconclusive findings in the record.

Next development priorities

Complete current platform qualification, preserve numerical and evidence boundaries, and publish a small set of clearly scoped demonstrations. Any prospective pilot, live deployment or new domain claim needs its own authorization and acceptance evidence. This document is a public specification and portfolio, not authorization to perform those actions.

Capability index for Core and Models

This index groups capability families. Algorithm means executable numerical or control logic. Contract means validation or record interfaces. Workflow means a composed path. Research means a bounded experimental method or diagnostic. A family may span several types. Page references identify the main treatment.

Repository and capability family	Type	Main section	Page
Core judgment constraints ranking and finite information value	Algorithm	WONDER Core integrated capabilities	11
Core robustness causal graph checks calibration and numerical diagnostics	Algorithm	Core quantitative and scientific methods	12
Core model use gates information time and configuration coverage	Algorithm / contract	Core quantitative and scientific methods	12
Core estimator planning resources pilots and coupling	Algorithm	Core computation and adaptive system contracts	13
Core provider storage recovery security and adaptive governance	Contract / workflow	Core computation and adaptive system contracts	13
Core strict validation release records and reproducibility	Workflow / contract	Core validation and integration contracts	14
Core exact evidence forecast custody authority and reasoning adapters	Contract / workflow	Core validation and integration contracts	14
Core twelve package responsibilities	Package map	WONDER Core platform	10
Models candidate coding and forecast lifecycle	Contract	WONDER Models candidate lifecycle	15
Models observed outcome fitting and retained experience	Algorithm / research	WONDER Models candidate lifecycle	15
Models packaging registry sandbox and mock workloads	Contract / fixture	Models packaging and integration contracts	16
Models provenance rollback receipts and repair compatibility	Contract	Models packaging and integration contracts	16

Capability index for numerical and Lab tools

This index groups capability families. Algorithm means executable numerical or control logic. Contract means validation or record interfaces. Workflow means a composed path. Research means a bounded experimental method or diagnostic. A family may span several types. Page references identify the main treatment.

Repository and capability family	Type	Main section	Page
Monte Carlo distributions streams moments and tail risk	Algorithm	WONDER Monte Carlo numerical toolkit	17
Monte Carlo generic fixed adaptive and native multilevel estimation	Algorithm / research	WONDER Monte Carlo multilevel methods	18
Monte Carlo model authoring statistics sketches and stopping	Algorithm / API	Monte Carlo developer interfaces	19
Monte Carlo variance reduction receipts path ranges and replay	Algorithm / contract	Monte Carlo developer interfaces	19
Monte Carlo CLI worker execution and admission limits	Workflow / API	Monte Carlo CLI workers and capacity	20
Lab graphs world families routing and numerical discrepancy	Algorithm / fixture	WONDER Lab synthetic environments	21
Lab actual retry execution and repair evidence	Workflow / contract	Lab forecasting retry and workload APIs	22
Lab synthetic forecast accounting and system identification records	Algorithm / contract	Lab forecasting retry and workload APIs	22
Lab scenario oracle operational and provider fixtures	Contract / fixture	Lab forecasting retry and workload APIs	22

Capability index for evaluation and applications

This index groups capability families. Algorithm means executable numerical or control logic. Contract means validation or record interfaces. Workflow means a composed path. Research means a bounded experimental method or diagnostic. A family may span several types. Page references identify the main treatment.

Repository and capability family	Type	Main section	Page
Evals scoring provenance recorded utility and adaptation studies	Algorithm / research	WONDER Evals evaluation evidence	23
Evals durable feedback publication and Financial export	Workflow / contract	Evals lifecycle and application contracts	24
Evals repair execution scientific forecast protocols and portfolio oracle	Algorithm / workflow	Evals scientific and diagnostic catalogue	25
Evals diversity custody and operational failure records	Contract	Evals scientific and diagnostic catalogue	25
Engineering repair cases review and controlled runner	Workflow	WONDER Engineering repair workbench	26
Engineering intake exact binding replay and durable process ownership	Workflow / contract	Engineering inspection and ownership interfaces	27
Financial pricing rates calibration risk and strategy library	Algorithm	WONDER Financial numerical laboratory	28
Financial authenticated portfolio saved work trials and forecast readers	Workflow	WONDER Financial decision workspace	29
Financial instrument identity issuer sources corporate actions and P and L	Algorithm / workflow	Financial market data and accounting	30
Financial surfaces factor dependence PCA and calibration uncertainty	Algorithm / research	Financial calibration and uncertainty research	31
Financial backtesting performance MLMC and stochastic volatility	Algorithm / research	Financial strategies and advanced estimators	32
Healthy Energy operating software and Core reuse	Application workflow	Healthy Energy as a working application	33
Healthy Energy paired numerical launch comparison	Recorded study	Healthy Energy numerical decision study	34

Repository revision inventory

Snapshot date: 25 October 2026. The seven repository identities below remain the reviewed main revisions. All were private, nonarchived and used main as the default branch at the current check. A source revision identifies software, not a deployed product or a completed release acceptance.

Repository	Inspected package version	Full main revision
WONDER-Core	0.13.0-rc.1.scientific.1	b380d2441af85ab07790b326127ef901b6971c2b
WONDER-Monte-Carlo	1.0.4-mlmc.1	1b30c104e008003df98c7635fbaf1d47a2f63c4b
WONDER-Models	0.2.5-dev.retained.1	68c4b98d4a82670311f1969fde4a6caeecf052b9
WONDER-Lab	0.3.4-dev.retained.1	1251f5c7295d9d799e397162028370fcc482da5b
WONDER-Evals	0.4.12-dev.dt06-admission.1	f8b68fc54606d124c1843c9277689b7743862ec8
WONDER-Engineering	0.1.0	d06f9b2975814c07a90e0660714914c5a5727332
WONDER-Financial	0.64.0-rc.1	8ecc424bcc071e6fda05764530bb629f6358b004

Application pin and test distinctions

Financial's direct numerical package is Monte Carlo 1.0.2, and its judgment package is 2.0.1-rc.1. The broader platform has separately versioned adapters. Healthy Energy's reviewed PR 3 merge is d30219ead3091df1c3ca6bd0575b24950b32e1d8, with Core b380d2441af85ab07790b326127ef901b6971c2b and Monte Carlo 1b30c104e008003df98c7635fbaf1d47a2f63c4b. These consumer identities are distinct from the general source inventory.

Financial full verification belongs to its merged pull-request source head 6c2ee9da5aad29a8392ca1efa6b7c8b7f94ad46. The merge-head workflow completed only the lightweight checks described earlier. Core's draft immutable-read and supervisor work remains outside main. Candidate results should be read against the exact dated status rather than assumed to update this inventory.

No CI job was launched, code changed or deployment triggered for this portfolio review. Source inspection, retrieved CI records and historical execution reports are identified separately throughout.

Evidence catalogue for Core and Models

The short codes identify the implementation or verification artifacts underlying this portfolio. These are sanitized descriptions of private engineering evidence. Their inclusion is not a public source-code release, an independent audit or a grant of access. Public naming references follow the catalogue. Private links and operational locations have been removed.

S1 Core repository | Financial repository

Founder and research-goal statements are based on Aaron Moyer's own project descriptions. Repository and implementation claims are tied to the inspected revision inventory. Earlier detailed source-review findings are retained unless a current status note explicitly updates them.

C1 Workspace manifest | Core overview

C2 Decision analysis | Judgment engine

C3 Orchestrator | Decision Protocol

C4 Authority | Action runtime

C5 Executed decision workflow regression

C6 Research API dispatcher | Research calculations

C7 Fitted routing consumer | Retained experience

C8 Prospective pilot contract | Merged rehearsal record

C9 Merged-main verification. Separate draft supervisor and immutable-read candidates are covered by the dated status section.

C10 Model Assurance exports | Assurance program

C11 Fabric public API | Organism contracts

C12 Probability robustness | Full Judgment API

C13 Calibration | Risk validation | Forecast diagnostics | Numeric primitives

C14 Domain primitives | Point in time | Configuration assurance | Use specific model gate

C15 Estimator planner | Scheduler | Pilot racing | Coupling diagnostics

C16 Automation | Deliberation | Provider fixture | Recovery

C17 Strict validation | Run replay | Release profiles

C18 Exact text capture | Forecast protocol API

C19 Delegation | Reasoning gateway | Decision Lens | Constitutional Kernel

Models sources

M1 Manifest | Public API | Training lifecycle

M2 Coding workflow | Retry proposal | Forecast challenger

M3 Actual fitter | Recorded fit | Execution artifact

M4 Retained experience fitter | Fitting regression tests

M5 Exact main verification record

M6 Model package and registry

M7 Sandbox specifications | Mock workloads

M8 Provenance | Rollback plan | Provider receipts | Executable repair contracts

Evidence catalogue for numerical and research tools

- N1** Manifest | Specification
- N2** Distributions | Tail risk
- N3** Variance reduction | Generic MLMC
- N4** Native fixed MLMC contract | Implementation | Focused tests
- N5** Core execution contract | Financial replay contract
- N6** Exact main verification record | Validation registry
- N7** Engine API | Random streams
- N8** Statistics | Quantiles | Stopping rules
- N9** Receipt APIs | Path ownership and replay | Format only checks
- N10** CLI | Worker API | Worker limits and security
- N11** Exact admission limits

Lab sources

- L1** Overview | Public API | World families
- L2** Procedural numerics | Routing contract
- L3** Retained routing | Routing regressions
- L4** Exact main verification record
- L5** Retry simulator | System identification | Graph analysis
- L6** Synthetic forecast accounting | Declared oracle helper

Evals sources

- E1** Overview | Isolation scope
- E2** Feedback contract | Ledger implementation
- E3** Route B acceptance | Utility result
- E4** Retained experience result
- E5** DT06 integration | Financial export | Export tests
- E6** Exact main verification record | Retained lifecycle issue
- E7** General evaluator | RCRE scoring
- E8** Neutral retry executor
- E9** Forecast protocol | Independent portfolio oracle
- E10** Evidence diversity | Operational assessment

Evidence catalogue for Engineering and applications

- G1** Engineering overview
- G2** Controlled runner | History
- G3** HTTP server
- G4** Observer regression
- G5** Exact main verification record | Recorded verification
- G6** Checkout intake | Workspace
- G7** Durable ownership | Process identity

Financial sources

- F1** Application overview | Numerical manifest
- F2** Public numerical exports
- F3** Executable local routes | Current authenticated markup
- F4** Risk workspace | Exact dated P and L
- F5** Planning inspector | Planning export
- F6** Score reader | UNRATE pilot boundary | Forecast export | Restore qualification
- F7** Financial merged pull-request full verification and merge-head lightweight checks, with separate evidence scopes.
- F8** Historical bounded hosted acceptance
- F9** Numerical limits | Bias envelope regression
- F10** Instrument domain | Market data | Provider resilience
- F11** Stock intelligence | Corporate action helpers | Lifecycle
- F12** Black Scholes | Greeks | Curves | Schedules
- F13** Quote quality | Surface repair | SSVI
- F14** Portfolio Monte Carlo | Factor calibration | Dependence | Yield curve PCA
- F15** Backtest engine | Multiple instruments | Execution | Performance | Portfolio analytics
- F16** Detailed API and research contract catalogue
- F17** Local and hosted UI inventory

Healthy Energy sources

- HE1** Reviewed merged PR 3
- HE2** Actual Core integration
- HE3** Launch study implementation
- HE4** Saved conditional decision report
- HE5** Numerical engine pin
- HE6** Versioned local comparison inputs, explicit bounded execution, separate admission and uncertain-outcome fencing. Hosted execution refusal checked in source.

Public references and document use

These public references explain the business naming process. They do not independently validate WONDER. The implementation-specific evidence catalogue identifies the private records used for the portfolio. The public file contains no private repository links or confidential Healthy Energy economics.

P1 Government of Canada business name guidance

Explains web, corporate-name and trademark checks and the limits of provincial trade-name registration. Retrieved 25 October 2026. [Read source](#)

P2 Saskatchewan Registries name reservation process

Describes the name application, review and reservation process. Retrieved 25 October 2026. [Read source](#)

Reading the evidence fairly

Technical source descriptions refer to specific versions and retained artifacts. Counts are not combined across overlapping suites. A component may be implemented without being exposed in a public application. A recorded result may be reproducible without proving external validity. Current legal clearance, customer adoption and commercial availability should not be inferred from this portfolio.

Third party software and AI

The project uses third-party development tools, language-model services and software dependencies. Their roles should be credited separately from the founder's direction, application design and integration work. Package-specific licenses and attribution requirements continue to apply. This public description is not a comprehensive license audit or a claim that all underlying algorithms and dependencies originated with the founder.

Document status

Public review edition prepared 25 October 2026. Prairie Simple Software and WONDER remain working commercial names subject to clearance. This document does not publish source code, grant repository access, authorize a deployment or approve a food, financial or other regulated product.