PROOF-HORIZON SHARDING IN BUSINESS

The Prompted Forge Implementation

Breyden E. Taylor | Founder and Architect, Prompted LLC | Cofounder and 50% Owner (author disclosure), Skyward Prompted LLC d/b/a Prompted Forge

Implementation Whitepaper 1.2 | Source-Bound Business Expression Revision | September 2026

A ONE-WAY BUSINESS EXPRESSION OF GOVERNED PARALLEL DELIVERY IN LIVE CLIENT WORK

Endorsed by Prompted LLC and Skyward Prompted LLC d/b/a Prompted Forge

Semantic legibility disclaimer - read before using this paper

Load-bearing semantics have been altered for legibility.

Proof-Horizon Sharding is unusually sensitive to semantics because its controls depend on exact distinctions: subject versus summary, proposal versus admission, evidence versus confidence, distribution versus reproduction, deployment versus adoption, and accepted state versus outcome. The business-language terms in this paper intentionally compress some of those distinctions so that executives, clients, operators, investors, counsel, and domain leaders can understand the operating shape without first learning the full native vocabulary.

This paper is a one-way expression of the architectural thing. The lawful direction is from canonical PHS semantics and current source-bound production artifacts into a business-language projection. The projection may orient a reader, support decisions, and help a stakeholder request the right evidence. It may not flow backward and redefine, configure, adjudicate, or operate PHS. Where this paper differs from the protocol, a native implementation contract, a machine-readable receipt, or a current production artifact, the canonical source governs.

Semantics matter in PHS and in systems running it in production. Words such as *proof*, *horizon*, *canon*, *admission*, *receipt*, *carrier*, *projection*, *current*, *live*, *closed*, and *outcome* are not interchangeable. A business summary that collapses them may be useful for orientation and still be unsafe as an operating instruction.

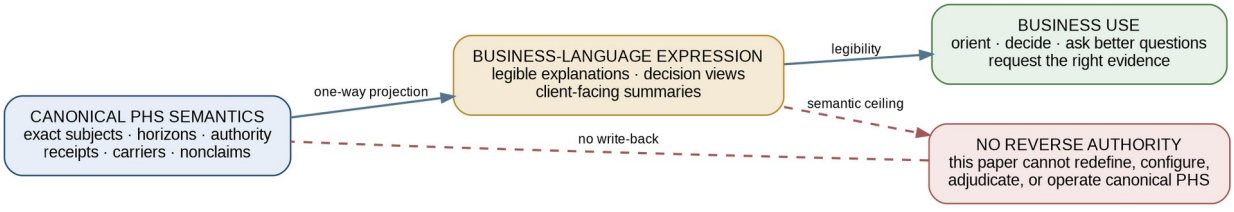

Figure 1. One-way semantic direction. Business language is a projection for orientation and decision support; it has no reverse authority over canonical PHS.

Repository, federation, and clock boundary

This revision carries a source boundary that is as important as the semantic disclaimer.

- The archive named canonical_federation is **Ubiquity - The Canonical Federation of Prompted LLC**. Its root source identifies **Telos as the capital and federation seat**. Its tics are Ubiquity Federation time authority. THE BINDER, its lane extracts, the Harpoon/hoist/winch history, and the tic-748 through tic-770 operating memos belong to this source domain. They support lineage and operational genesis for PHS; they are not Forge implementation receipts. [14-19]
- The archive named forge-canonical-fed is the **Forge Federation** and Skyward Bridge source operated by **Skyward Prompted LLC d/b/a Prompted Forge**. Its native operating evidence is carried through Git identities, dated Forge cycles, DENT waves, FIELD/DAG contracts, client-lane state, provider and deployment receipts, and PHS checkpoint carriers. [20-30]
- Forge is a sovereign sister-city seeded from Ubiquity Telos, not a subordinate runtime or downstream branch of the Ubiquity repository. The Forge canon says Ubiquity doctrine is design source rather than install target, and that Ubiquity's queue and tics are reference evidence only: they cannot mature, ratify, reject, supersede, demote, or apply a Forge candidate. [20]
- The two source domains therefore share lineage without sharing a clock, canon, or admission authority. Ubiquity receipts cannot be cited as proof that a Forge client effect occurred. Forge receipts cannot rewrite the history, authorship, or authority of Ubiquity and PHS.

Source	Authority	Native identity	Permitted use and boundary
PHS foundational paper	Breyden E. Taylor / Prompted LLC	Versioned publication	Protocol authority; does not prove a Forge deployment or client outcome
canonical_federation	Prompted LLC / Ubiquity	Ubiquity tics and canonical receipts	Lineage and operational precursors; cannot advance Forge state
forge-canonical-fed	Skyward Prompted LLC d/b/a Prompted Forge	Git objects, dated cycles, waves, pointers, runtime receipts	Implementation and working case; cannot rewrite Ubiquity lineage or claim stronger outcomes
Author disclosure	Breyden E. Taylor	Dated publication statement	Ownership, joint-venture and live-client context; not an independent audit

The supplied repository archives are hash-bound in the source package's Evidence Manifest. Their abbreviated SHA-256 identifiers are b4a27fd6...28102b3 for Ubiquity and 8b0a3ed2...14c19 for Forge.

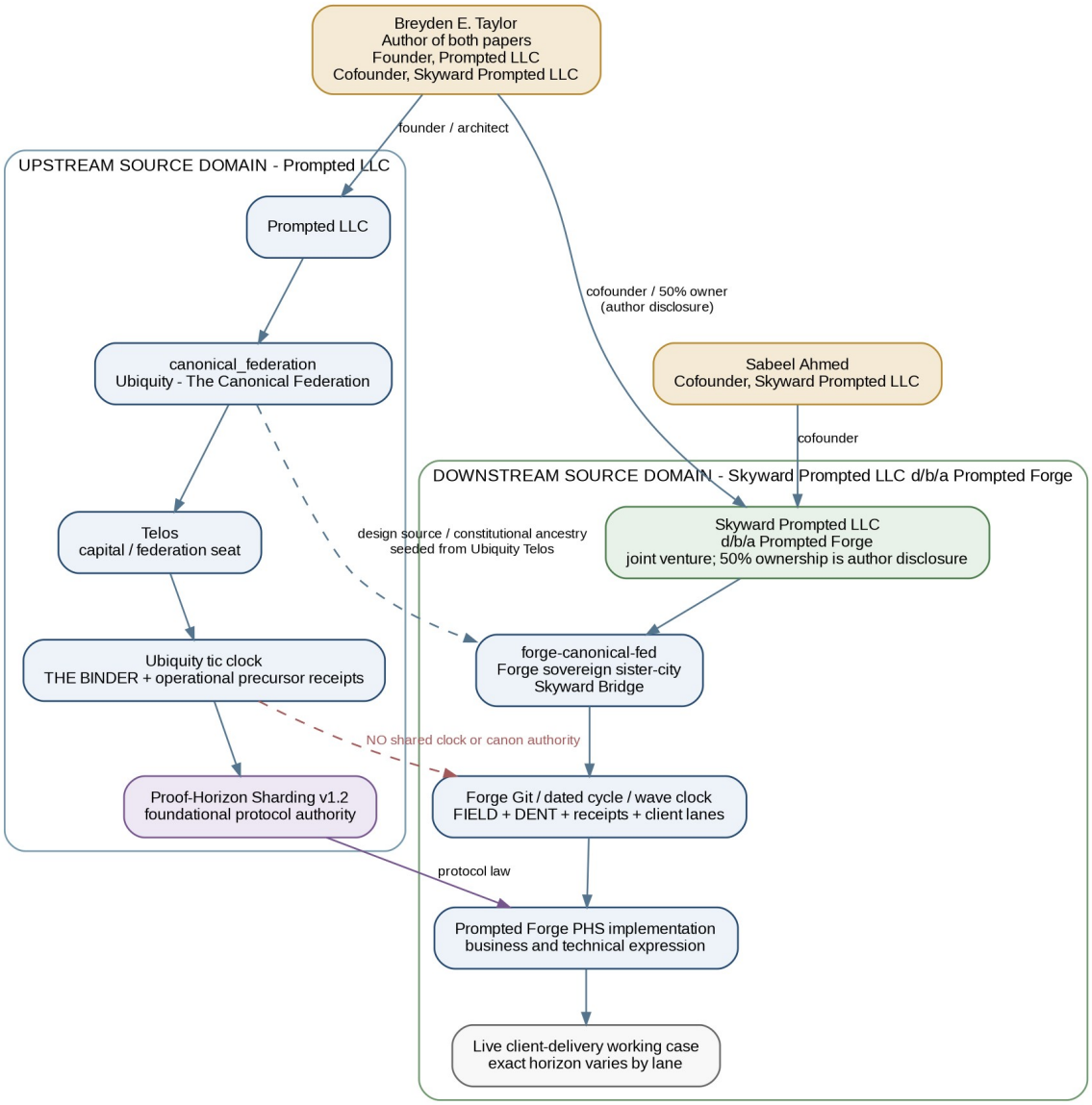

Figure 2. Source-domain, organizational, and authority boundary. Ubiquity and Forge are related by lineage and design source, not by a shared clock or shared canonical authority.

Universal solution, software expression

> **PHS is not a software solution with universal applicability. It is a universal solution with software expression.**

The distinction is load-bearing. PHS addresses a universal governance problem: work, evidence, authority, and consequence become available at different times, through different actors, and under different rights. No artifact or person can lawfully know a future observation before it occurs. No successful task can silently grant itself authority at a broader horizon. No copy, dashboard, or deployment label can become truth merely because it is convenient.

That control problem exists in software, legal execution, clinical workflow, policy adoption, infrastructure, media production, procurement, research, finance, and ordinary organizational decision-making. The universal part is the relationship among identity, evidence, authority, succession, and time. The software is one expression of that relationship. The Prompted Forge implementation described here is one production

expression, not a universal application, product, template, or claim that every organization should install the same stack. [1]

Corporate, case-study, and evidence disclosure

Author. Breyden E. Taylor authored this paper. He is Founder and Architect of Prompted LLC and Cofounder and 50% Owner of Skyward Prompted LLC, which does business as Prompted Forge.

Joint-venture context. Skyward Prompted LLC is described here as the Prompted Forge joint venture co-founded by Breyden E. Taylor and Sabeel Ahmed. The Forge repository independently supports the two cofounders' organizational standing and identifies Skyward Prompted LLC as the owner/operator of Skyward Bridge. The 50% ownership interest and the characterization of the operating company as a joint venture are author disclosures unless separately established by governing corporate records. [20,21]

Endorsing organizations. Prompted LLC and Skyward Prompted LLC d/b/a Prompted Forge endorse this implementation paper. Endorsement does not merge the two organizations, repositories, clocks, authorities, ownership interests, contracts, or evidence domains. Prompted LLC is the upstream Ubiquity and PHS publication context. Prompted Forge is the downstream joint-venture implementation and client-delivery context.

Working case study. The case study concerns a PHS implementation operating inside Prompted Forge in live client-delivery work. Client identities, confidential systems, commercial terms, and client-specific outcome evidence are omitted or cited only at their repository-declared horizon. The Forge repository contains several client and provider lanes with different currentness and maturity states; no single global label is used for all of them. [30]

Evidence ceiling. This is a working case study, not a controlled efficacy trial, independent audit, legal opinion, security certification, service-level commitment, or claim that every client lane has reached the same proof horizon. Repository evidence can establish repository and exact-lane facts within its declared scope. It cannot, by itself, establish client adoption, revenue, legal compliance, stakeholder satisfaction, or business outcome.

Recommended citation. Taylor, B. E. (2026). *Proof-Horizon Sharding in Business: The Prompted Forge Implementation* (Implementation Whitepaper 1.2, Source-Bound Business Expression Revision). Prompted LLC and Skyward Prompted LLC d/b/a Prompted Forge.

Protocol dependency. Taylor, B. E. (2026). *Proof-Horizon Sharding: Correction-Surviving, Proof-Carrying Canonical Computation* (Technical Whitepaper 1.2, Operational Genesis and Winching Lineage Revision). Prompted LLC. [1]

Version 1.2

Version 1.2 supersedes the source framing of Version 1.1 while retaining it as publication lineage. It makes seven material corrections.

1. It binds the two supplied repository archives as separate evidence domains.
2. It identifies canonical_federation as Prompted LLC's Ubiquity Canonical Federation, with Telos as its capital and Ubiquity tics as its clock.
3. It relocates THE BINDER, the tic history, the Harpoon/hoist/winch machinery, and the local-model memos to the Ubiquity lineage where they actually reside.
4. It identifies forge-canonical-fed as the sovereign Forge sister-city operated by Skyward Prompted LLC d/b/a Prompted Forge and uses that repository alone for the downstream implementation and client case.
5. It distinguishes Ubiquity's covenant/cable/board/winch vocabulary from Forge-native FIELD, DENT, candidate-lane, guarded-promotion, root-convergence, and checkpoint mechanics.

6. It adds Sabeel Ahmed as Prompted Forge cofounder while retaining the 50% ownership and joint-venture facts at their proper author-disclosure ceiling.
7. It replaces the apparent single timeline with two source-bound clocks connected by lineage but denied cross-canon authority.

Contents

Abstract

Proof-Horizon Sharding (PHS) is a governance protocol for scaling work without allowing evidence, authority, or accepted status to outrun reality. In business language, it requires an organization to identify the exact thing under discussion, distinguish what has actually been established from what is merely expected, keep

decision rights explicit, preserve corrections instead of erasing them, and move accepted state through one intelligible path even when many people and agents work in parallel. [1]

This paper explains one downstream production expression inside Skyward Prompted LLC d/b/a Prompted Forge, a joint venture co-founded by Breyden E. Taylor and Sabeel Ahmed and operating in live client-delivery contexts. The implementation is evidenced by the separate forge-canonical-fed repository through Forge-native FIELD and DENT contracts, dependency DAGs, bounded work lanes, guarded integration, direct readback, detached reproduction, subject-bound receipts, descendant evidence carriers, recursive checkpoints, explicit nonclaims, and deterministic re-entry. [20-30]

The implementation's lineage is mapped through a different source domain: Prompted LLC's canonical_federation, whose root identifies Ubiquity as the Canonical Federation and Telos as its capital. That upstream repository carries the Ubiquity tic clock, THE BINDER, FORKED-related horizon lineage, and the covenant/Harpoon/cable/board/winch operating history from which PHS's practical constraints emerged. Those artifacts establish lineage and operational precursor evidence; they do not establish Forge implementation or client effect. [3-19]

The paper does not treat PHS as a universally installable software package. PHS is universal at the level of the control problem: facts, authority, and consequences become knowable at different horizons. Software makes that control law inspectable because versions, digests, dependencies, logs, and readbacks can be exact. The Prompted Forge stack is one software-heavy expression of the law. Other domains require different subjects, evidence, authorities, and closure predicates.

The business result is a different operating model. Teams and agents may explore, build, review, and test concurrently. Shared mutation and accepted-state changes remain serialized where necessary. A result is not called complete merely because a worker, dashboard, or local environment says so. The evidence travels in a later carrier with its limitations, and that carrier becomes the next subject when its own availability matters. Every implementation claim remains bound to the repository and clock that produced it.

Keywords: Proof-Horizon Sharding; business governance; Prompted LLC; Ubiquity; Telos; Prompted Forge; Skyward Prompted LLC; source-domain separation; horizon noncollapse; FIELD; DENT; DAG; receipts; carrier recursion; joint venture; live client delivery; serial authority; correction survival.

Executive summary

PHS solves a business problem that appears whenever work moves faster than one decision-maker can directly inspect it.

A business normally asks one overloaded question: **Is it done?**

PHS replaces that question with four exact questions:

1. What exact thing are we deciding about?
2. What evidence exists for that exact thing?
3. Who has authority to accept or change its status?
4. What remains unproven, held, or dependent on a later horizon?

Those questions can govern a software release, contract, policy, model, dataset, clinical protocol, media asset, financial control, or organizational decision. The implementation changes by domain; the governance relation does not.

In the working Prompted Forge expression, many agents and workstreams may act at once, but they do not all become canonical writers. Forge's FIELD contract binds the smallest current reality slice. DENT orders bounded work through dependency DAGs. Candidate lanes operate under explicit write and effect ceilings. Guarded promotion checks expected preimages and currentness. Root convergence alone admits the shared result. PHS then proves the exact source through direct target readback and independent acquisition, carries those observations in a descendant artifact, and proves the carrier itself. [2,22-27]

The paper is built over two repositories and refuses to collapse them.

Ubiquity source domain. Prompted LLC's canonical_federation supplies constitutional ancestry and the receipt-grounded operational precursor record. Telos is its capital. Its tics are Ubiquity Federation time. THE BINDER's BND landmarks belong here. [14-19]

Forge source domain. Skyward Prompted LLC's forge-canonical-fed supplies the Prompted Forge implementation, Wave 38 PHS checkpoint receipts, current FIELD/DENT pointers, guarded convergence, client-lane states, and Forge-scale evidence. Its units are exact Git identities and dated Forge cycles and waves. [20-30]

The relation is inheritance, not shared authority. The Forge canon explicitly describes Forge as a sovereign sister-city seeded from Ubiquity Telos and says the Ubiquity queue and tics cannot mature or apply a Forge candidate. [20]

This source wall changes how the case should be read. Ubiquity's operational scars explain where the architecture learned to distrust attractive status labels. Forge's receipts show how the downstream system implements the protocol in its own canon. The first is lineage evidence. The second is implementation evidence. Neither may impersonate the other.

The working client case remains bounded. Forge's current matrix records multiple lanes at different horizons: exact-scope provider production standing in some lanes; one configured school-system adapter operator-attested as fully functional and adopted by its one client; a participant-email lane with runtime and canary evidence but per-message delivery/readback still separate; and other client or regulated lanes held behind their own gates. [30] The case therefore demonstrates a production operating method, not universal completion.

The core business sentence is:

> **Let work spread. Keep evidence exact. Keep authority explicit. Let accepted state move through one accountable path.**

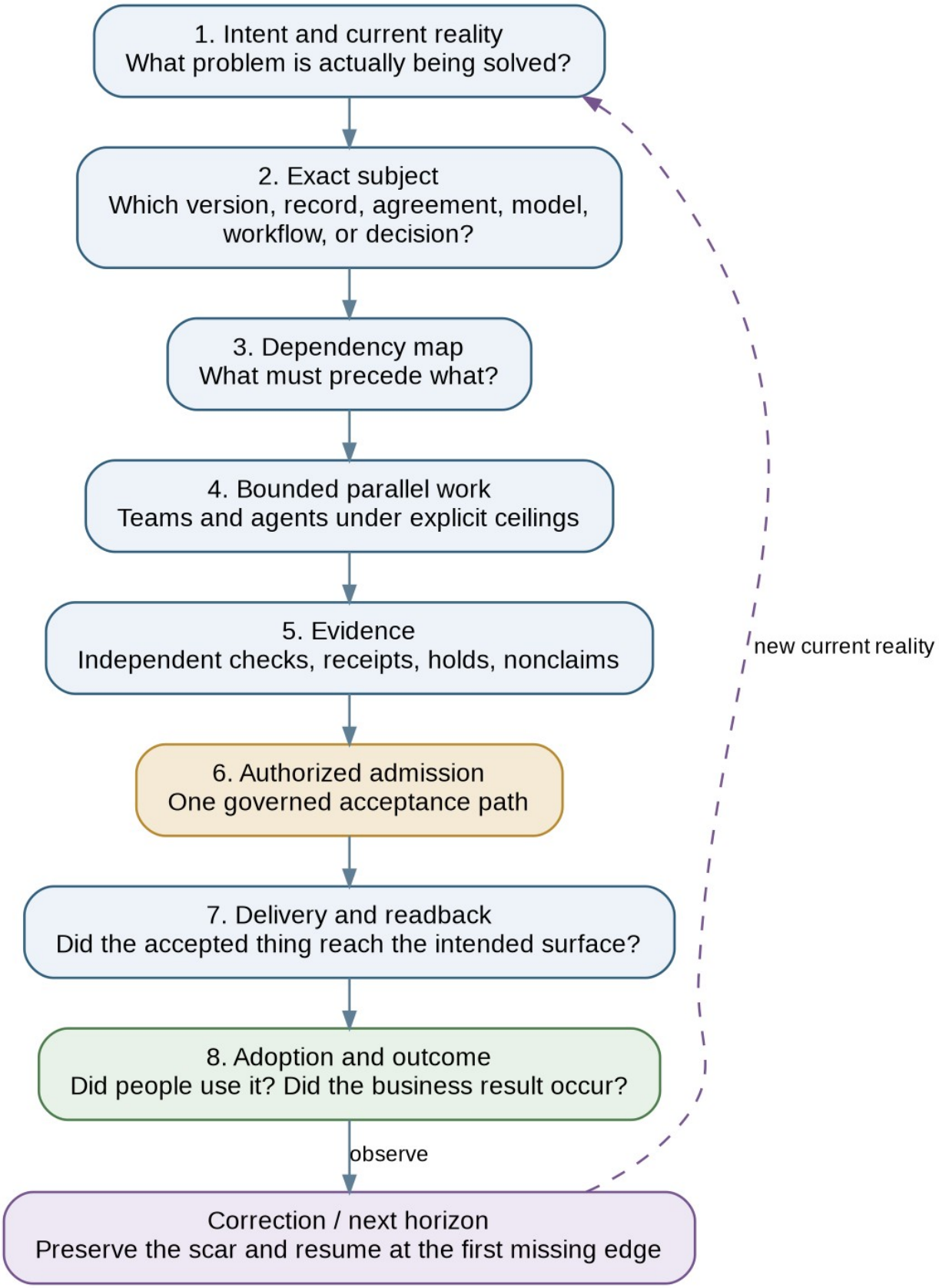

Figure 3. Business control loop. PHS begins with an exact decision subject, separates evidence and authority, preserves open edges, and advances only the horizon actually earned.

1. Why this paper exists

The foundational PHS paper is intentionally exact. It defines a protocol in terms of immutable or strongly identified subjects, proof horizons, predicates, receipts, carriers, temporal admissibility, authority ceilings, joins, and scoped closure. Those distinctions are the point of the protocol, not incidental vocabulary. [1]

That precision creates a translation problem. A business leader may understand the problem immediately while finding the native terms unnecessarily distant from familiar work. A client may need to know whether a delivery is ready, live, accepted, or producing value without needing to inspect a Git object graph. A board member may need to know who controls a consequential decision without learning the implementation's node schema. An operator may need a simple way to explain why five green checks still do not prove adoption.

This paper provides that translation, but it refuses to treat translation as governance. It is a reader interface, not a control plane.

The central business problem is not software complexity. It is **coordination under partial knowledge and unequal authority**. Software makes the problem visible because exact versions can be compared. The same shape exists when a contract is drafted but not signed, a policy is approved but not adopted, a clinical protocol is published but not followed, a campaign is distributed but not seen, or a strategic decision is announced but not implemented.

PHS is valuable when the difference among those states matters and when the organization cannot afford to discover the difference only after harm, delay, cost, or public contradiction.

2. PHS in one business sentence

> **Proof-Horizon Sharding is a way to scale work without allowing status, evidence, or authority to outrun reality.**

The phrase *proof horizon* means the strongest bounded claim the available evidence can support at that time. It is not certainty. It is not a project phase. It is not an executive confidence score. It is the edge between what has been established and what remains intended, inferred, held, or unknown. [1]

The phrase *sharding* means that different questions may be checked independently and often concurrently. One lane may confirm that the approved version reached the intended repository. Another may confirm that a clean environment can acquire and validate it. Another may inspect security or data handling. Another may verify client configuration. These checks can be parallel because they examine different failure surfaces.

The parallel checks do not each gain the right to mark the whole engagement complete. Their evidence joins under an explicit acceptance path. That is the business meaning of **parallel verification without parallel authority**.

PHS therefore changes the management question from:

> "Who says it is done?"

into:

> "What exact claim is being made, about which exact subject, from which evidence, under whose authority, and what stronger claim still requires another horizon?"

3. The universal problem - and the local expression

Every organization operates across three asymmetries.

First, **knowledge is asynchronous**. A plan exists before execution. A build exists before distribution. Distribution exists before use. Use exists before measured outcome. The later fact cannot be truthfully carried by an earlier immutable record unless a later carrier is created.

Second, **authority is uneven**. A skilled employee may discover a problem without having authority to change a regulated process. An AI agent may generate a correct patch without having authority to deploy it. A vendor may verify a deliverable without having authority to accept it on the client's behalf. A dashboard may show a state without having authority to create that state.

Third, **consequence is horizon-specific**. A locally correct result may still fail in distribution. A live deployment may still fail in adoption. Adoption may still fail to produce the intended outcome. Each consequence requires evidence appropriate to its own horizon.

Those asymmetries are universal. Their expressions are not.

A software implementation can bind a subject to a commit hash, read named remote refs, acquire a detached clone, run validations, and materialize a signed or versioned receipt carrier. A legal implementation might bind a subject to a document digest, verify signatures and custodian records, and issue an execution certificate. A clinical implementation might bind a subject to a protocol version, verify configuration, training, and bounded runtime observations, and preserve a governed change packet. A business process might bind a subject to an approved operating decision, confirm adoption across named units, and measure a declared outcome under a stated method. [1]

The control law is portable. The implementation is contextual.

This is why the phrase **universal solution with software expression** is more accurate than "universal software solution." A universal software product would assume one ontology, one workflow, one evidence standard, and one authority model. PHS requires the opposite: domain-specific predicates and authorities inside a common temporal and evidentiary discipline.

4. The four business questions

A business-facing PHS conversation can begin with four questions.

4.1 What exact thing are we talking about?

The subject may be a build, model, policy, report, contract, dataset, workflow, campaign asset, configuration, or decision packet. "The project," "the AI," "the dashboard," and "the release" are usually too vague.

An exact subject prevents evidence about one version from migrating to another merely because the names look similar. The upstream Ubiquity operating record and the downstream Forge repository each contain examples of stale paths, mismatched identities, and current-looking summaries that no longer described the exact observed state. Those are separate source domains. Their shared corrective rule is simple: bind the claim to the subject that was actually observed, and do not transfer evidence across repositories or clocks merely because the vocabulary is related. [1,14-17,20]

4.2 What evidence exists?

Evidence must answer a declared question. A local test may establish local execution. A direct remote read may establish distribution. An independent clean acquisition may expose hidden dependencies. A client readback may establish that the client's intended surface received the change. A usage measure may establish adoption during a bounded interval.

Evidence does not become stronger because it is summarized confidently. PHS calls for explicit nonclaims so that a repository receipt does not masquerade as business outcome evidence.

4.3 Who had authority?

The person or system that observed a fact may not be the person or system authorized to accept it. PHS treats this separation as a feature.

A worker may build. A verifier may challenge. An integrator may admit repository state. A provider owner may deploy. A client sponsor may accept business risk. A regulator, counsel, clinician, or board may own a higher-consequence gate. The architecture must not infer those rights from access, title-like naming, technical capability, or position in a workflow.

4.4 What remains unproven?

Every honest checkpoint says what comes next. The next horizon may be distribution, reproduction, deployment, client acceptance, adoption, or outcome. Naming the open edge prevents “not yet” from turning into “probably done” during handoff.

These questions create a compact executive control surface.

What exact thing? Use exact subject identity. This prevents evidence from migrating across versions.

What evidence? Use predicate- and horizon-bound receipts. This prevents a generic green status from standing in for several distinct facts.

Who had authority? Use explicit authority ceilings and an explicit admission path. This prevents an observation or completed task from becoming unauthorized mutation.

What remains unproven? Name the next horizon and the nonclaims. This prevents source or deployment proof from laundering itself into outcome.

5. The claim ladder

Business systems often use a single progress bar. PHS uses a claim ladder because each rung answers a different question.

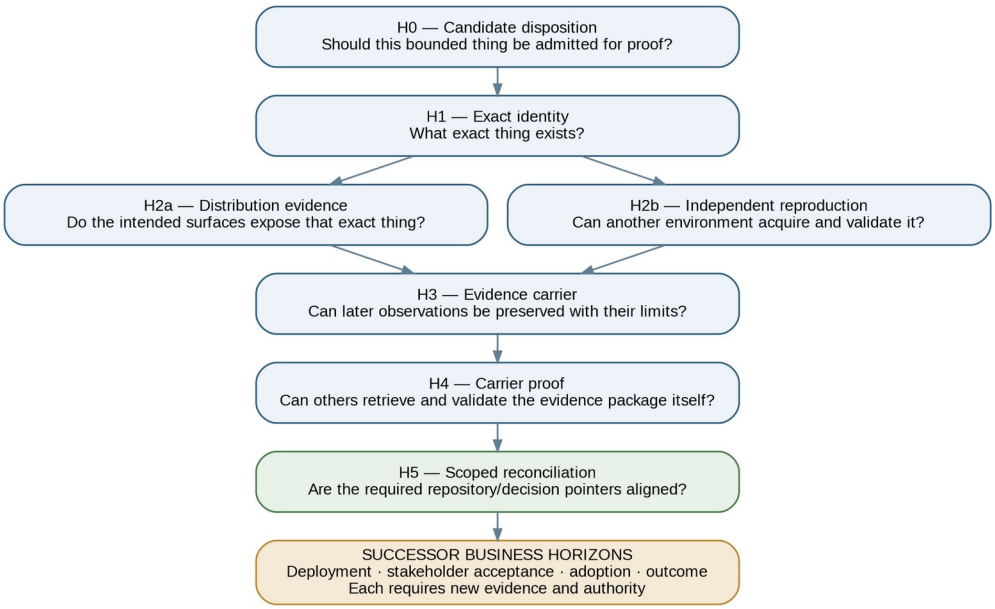

Figure 4. Business-readable claim ladder. H0-H5 are canonical PHS checkpoint horizons; deployment, acceptance, adoption, and outcome remain successor business horizons requiring new evidence.

The canonical reference epoch in PHS runs from candidate disposition through exact identity, distribution evidence, independent reproduction, evidence carriage, carrier proof, and terminal reconciliation. [1] The business translation follows.

5.1 H0 - Candidate disposition

Business question: Is this bounded thing eligible to enter the governed process?

Still unproven: that it exists as accepted state.

5.2 H1 - Exact identity

Business question: What exact version or subject exists?

Still unproven: that anyone else can reach or use it.

5.3 H2a - Distribution

Business question: Do the intended surfaces expose that exact subject?

Still unproven: that it works outside the authoring environment.

5.4 H2b - Independent reproduction

Business question: Can another declared environment acquire and validate it?

Still unproven: that it has been deployed to the client or accepted by stakeholders.

5.5 H3 - Evidence carrier

Business question: Can later observations travel with their identities, limits, and nonclaims?

Still unproven: that the carrier itself is distributed or independently reproducible.

5.6 H4 - Carrier proof

Business question: Can others retrieve and validate the evidence package?

Still unproven: that the business outcome occurred.

5.7 H5 - Terminal reconciliation

Business question: Are the required pointers and receipts aligned for the declared scope?

Still unproven: any stronger horizon not explicitly included.

A repository checkpoint can close at H5 and remain entirely silent about whether a customer used the feature. This is not evasiveness. It is a refusal to spend evidence twice.

The successor business horizons are usually the ones executives care about most:

- **Deployment:** the intended live environment exposes the admitted version.
- **Stakeholder acceptance:** the person with standing accepts the bounded deliverable or risk.
- **Adoption:** named users or operating units actually use the capability during a stated interval.
- **Outcome:** a declared real-world result is measured under a stated method.

PHS does not demote those outcomes. It protects them from being counterfeited by weaker evidence.

6. Parallel work without authority sprawl

Most organizations face a bad tradeoff when work accelerates.

They may serialize most activity, preserving control while sacrificing speed. Or they may let many actors work and write at once, gaining apparent speed while creating divergent state, duplicate decisions, hidden conflicts, and expensive reconciliation. Agentic software makes the tradeoff sharper because generation and analysis can fan out much faster than human approval capacity.

PHS rejects the assumption that parallel intelligence requires parallel authority.

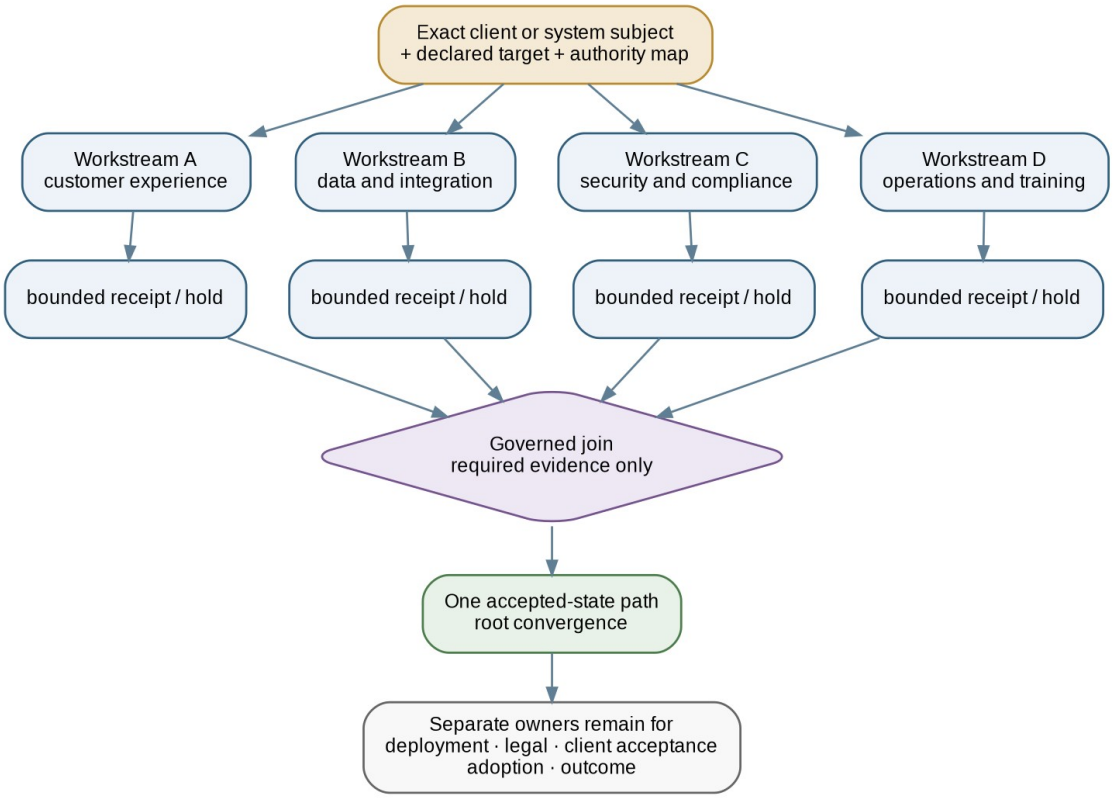

Figure 5. The operating shape. Many bounded work and evidence lanes may move concurrently; the accepted-state path remains singular where shared mutation requires it.

The business model is:

many may propose
many may inspect
many may test
many may challenge
many may preserve copies
one governed path admits accepted state for the declared scope

This is not organizational centralization by default. Different horizons can have different lawful authorities. A technical integrator may control repository convergence. A client sponsor may control deployment approval. Counsel may control a legal execution horizon. A product owner may control scope acceptance. A business unit may control adoption. The singularity is logical within each declared acceptance path, not a claim that one person should own every decision.

The Forge implementation uses several mechanisms to preserve that shape.

FIELD binds a bounded reality slice and its nonclaims. It can say what the work is about without granting execution or effect authority.

DENT orders dependencies. It can say what must precede what without inheriting the authority of the work it schedules.

Candidate lanes perform bounded work on disjoint or explicitly owned surfaces.

Proof lanes inspect exact subjects and return receipts or holds.

Root convergence is the controlled integration path. It accepts only the evidence and changes the declared contract permits.

Owner-native gates remain outside the integration path when effects belong elsewhere. A successful code checkpoint does not authorize a provider change, database migration, client communication, legal commitment, or outcome claim.

The business advantage is not merely speed. It is **delegation without semantic surrender**. A leader can delegate a bounded predicate rather than delegating the power to decide what the whole project means.

7. Receipts as business infrastructure

A normal project record often consists of messages, tickets, dashboards, meeting notes, and memory. Those artifacts may be useful, but they create reconstructive archaeology when the meaning of "done" is disputed.

A PHS receipt is different. It is a durable record bound to an exact subject, a declared question, a method, an observer, a result, an authority ceiling, evidence references, limitations, a convergence target, and the next horizon. [1]

In business language, a receipt answers:

- what was checked;
- which exact version or decision was checked;
- how it was checked;
- who or what performed the check;
- what the check established;
- what the check did not establish;
- where the result may lawfully be used;
- what remains next.

The point is not to produce more paperwork. The point is to make later decisions cheaper because the evidence does not have to be reconstructed from private context.

The Ubiquity operating record summarized this differentiation bluntly: claims are coupled to falsifiers, a second reader re-derives them, fallen numbers are dated rather than retconned, and the product is not merely the plugin but the receipt another reader can reproduce. [18]

For a client, the receipt can become part of a decision package. For an operator, it becomes a re-entry point. For a manager, it becomes a truthful status boundary. For an auditor, it becomes a traversal path. For an AI system, it becomes an external memory surface that does not require the same model, context window, vendor, or machine.

Receipts also make disagreement useful. If two readers observe different realities, the system does not have to immediately flatten the disagreement into one executive summary. It can preserve both observations, their methods, their timestamps, their subject bindings, and the owner of the next adjudication.

This is one reason PHS treats the evidence carrier as a new subject. An earlier artifact cannot be rewritten to pretend it knew a later observation. A later carrier packages the new evidence. When the carrier's own distribution or reproduction matters, another observation and later carrier are required.

That recursion may feel formal, but the business intuition is familiar: **a signed report can prove what was known before it was signed; it cannot prove that everyone later received, understood, adopted, or acted on it.** Those later facts require later evidence.

8. Correction as compounding capital

Most organizations treat correction as reputational damage. They overwrite the plan, replace the dashboard, close the ticket, or explain the failure away. The short-term record becomes cleaner while the long-term system becomes more fragile.

PHS treats correction as a first-class property of trustworthy operation.

When an exact subject fails a predicate, the failure remains bound to that subject. A corrected subject receives a new identity and a new proof path. The earlier result is not edited into success. The system can therefore answer three different questions:

1. What failed?
2. What changed?
3. Which later subject passed which later predicate?

That distinction turns a scar into reusable operating knowledge.

The upstream Ubiquity Federation contains a long precursor record of this behavior. A capability ruling was overturned only after live execution. A declarative board-consumer claim survived roughly 119 Ubiquity tics until a real consumer exposed the vacuous green. A telemetry marker suppressed the receipt it was intended to report, and the correction preserved the confession. A proof-horizon probe named for detached reproduction found that it had never contacted a remote and demoted its own claim. [15-19]

Those events are **not Forge implementation receipts**. They are the Ubiquity operational lineage in which the architecture learned the refusals later formalized by PHS: re-test at the act, distinguish presence from enforcement, preserve observer failure, and refuse a name stronger than its evidence.

The downstream Forge implementation expresses correction survival through a different source record: revisioned candidate lanes, expected-preimage leases, HELD states, dirty-set protection, exact source and carrier identities, independent readback, and current pointers that do not grant execution authority. [22-27]

Correction compounds when three conditions hold.

First, the failure is attached to the subject and method that produced it.

Second, the correction changes the next trajectory rather than merely changing the story.

Third, successors can discover the scar without relying on the previous operator's memory.

The expected result is lower correction debt. The organization still pays for mistakes, but it does not repeatedly pay to rediscover the same mistake or to untangle a rewritten history.

9. Upstream Ubiquity lineage and operational precursors

PHS did not begin as a software feature inside Prompted Forge. Its constitutional and operational lineage is upstream in Breyden E. Taylor's Ubiquity architecture at Prompted LLC. The supplied canonical_federation root names itself **Ubiquity - The Canonical Federation** and identifies Telos as its capital and federation seat. Its audit-log tics directory is the time-authority surface. [14]

This matters because the same words may appear across the ecosystem without carrying the same local authority. Lineage is not runtime installation. A source term can be inherited by reference while its original clock and canon remain where they were born.

9.1 Fractal Quivers of Quivers

Fractal Quivers of Quivers supplies a governance geometry in which actors, artifacts, standing, lawful traversals, forbidden edges, cost, and receipts remain explicit. In business language, it is a way to model not only what can move, but who may move it, under what authority, and with what consequence. [7]

9.2 Computing Around the Open Center

Computing Around the Open Center adds bounded splat computation around a center that is preserved rather than consumed. The target may guide motion without becoming a task a scheduler can complete or overwrite. [8]

The Ubiquity operating record compiled this as center exclusion: the founding telos remains a non-node. The architecture may move around it, measure against it, and let it constrain the field; it may not strike or silently replace it. [15-17]

9.3 FORKED horizon noncollapse

FORKED makes governed horizon geometry explicit. Task, mission, battle, campaign, war, institutional, and other consequence scales carry distinct telos, standing, authority, success conditions, irreversible surfaces, dependencies, and conflicts. A declaration at one horizon does not silently acquire authority at another. [6]

PHS specializes that rule to evidence through time. Local source existence does not become distribution. Distribution does not become reproduction. Repository closure does not become deployment. Deployment does not become client acceptance, adoption, or outcome.

9.4 THE BINDER and the winching lineage

THE BINDER belongs to the Ubiquity Canonical Federation at audit-logs/governance/whitepaper-binder-tic770/. It is a map of evidence, not the primary receipt. Its BND markers resolve through two lane-complete extracts to primary artifacts. [15-17]

The operating lineage it maps is one current-to-target loop wearing several mechanism names. A covenant binds a bounded pull. Harpoon strikes candidate shape through multiple rays. Cables assign obligations. The board compiles admitted obligations into a dependency DAG. The winch raises the lawful frontier. GUNSLINGER and HIDALGO select tempo. Dissonance basins hold tension. Center exclusion protects the reference frame. Receipts preserve movement, failure, residue, and next obligations. [15-17]

These terms are upstream operational precursors. They are not presented as current Forge-native implementation identifiers unless the Forge repository independently uses them in that exact role.

9.5 Context Grapple Gun, Look-First, and successor continuity

Context Grapple Gun supplies a portable proposal, review, correction, and promotion lifecycle. Look-First requires a successor to reattach to current canonical state before acting. Successor Topology separates continuity from implementation identity. Together, they reinforce a business principle: continuity should live in governed artifacts and exact state, not only in a person's memory or an agent's context window. [11-13]

9.6 What upstream lineage may and may not do downstream

Upstream lineage may supply constitutional ancestors, vocabulary, failure knowledge, and design constraints. It may explain why a downstream mechanism exists.

It may not, by ancestry alone:

- create or mutate Forge canonical state;
- grant a Forge actor standing or access;
- satisfy a Forge dependency;
- prove a Forge deployment, provider state, or client effect;
- move a Forge candidate through its lifecycle;
- transfer a Ubiquity tic into the Forge clock.

The Forge repository states this boundary directly. [20]

10. Corporate and federation topology

The working case sits at the intersection of one author, two organizations, two repositories, and one downstream implementation.

10.1 Authorship and company roles

Breyden E. Taylor authored the PHS foundational paper and this business implementation paper. He is Founder and Architect of Prompted LLC and Cofounder of Skyward Prompted LLC d/b/a Prompted Forge.

Sabeel Ahmed is the other Prompted Forge cofounder. The supplied Forge corpus describes Breyden and Sabeel as equal cofounders in the human organization while expressly refusing to infer identical technical authority or repository-root access from that human equality. [21]

Taylor's 50% ownership interest and the characterization of Skyward Prompted LLC as a joint venture are author-supplied corporate facts in this paper. They should be verified against governing legal records before being used for legal, tax, financing, or transaction purposes.

10.2 Organizational roles in the publication

Prompted LLC supplies the upstream Ubiquity architecture, the foundational PHS publication context, and the canonical-federation lineage.

Skyward Prompted LLC d/b/a Prompted Forge supplies the downstream operating company, Forge repository, Skyward Bridge surface, implementation evidence, and working client-delivery case.

Both organizations endorse this paper. That endorsement does not assign Prompted Forge authorship over PHS or give Prompted LLC operational authority over every Forge client decision.

10.3 Federation relationship

The Forge source describes Forge as a sovereign federation and sister-city seeded from Ubiquity Telos. Canonical doctrine is design source, not install target. Forge owns its own engine, role machinery, router, canon documents, candidate lifecycle, and execution state. [20]

That is the appropriate business analogy: Prompted Forge inherits a constitutional tradition without becoming a branch office in the same database or clock.

10.4 Endorsement, authorship, ownership, and evidence are separate axes

These four facts should not be collapsed.

- **Authorship** answers who wrote and is responsible for the paper.
- **Endorsement** answers which organizations stand behind this publication.
- **Ownership** answers who holds economic or governance interests in a company.
- **Evidence authority** answers which source can establish a particular technical or operating claim.

One axis does not automatically confer another.

11. The working Prompted Forge case study

The working case is the PHS implementation inside Skyward Prompted LLC d/b/a Prompted Forge, operating through the Forge Federation and Skyward Bridge in live client-delivery contexts.

The case is not THE BINDER. It is not the Ubiquity tic history. Its inspectable technical source is forge-canonical-fed.

11.1 The client surface

Clients need ordinary business outcomes: a workflow operates, a report is delivered, a school or clinic can act, a stakeholder receives a usable surface, and failures are recoverable. They should not be required to understand the full internal architecture to make an informed decision.

The client surface therefore projects the current claim, evidence, owner, hold, and next action in language appropriate to the engagement. The underlying exact subject and receipts remain reachable to authorized reviewers.

11.2 The delivery surface

Prompted Forge coordinates product source, infrastructure, provider adapters, data stores, client configuration, browser and runtime proof, operations, and communication. Those surfaces do not all move at once and do not share one authority.

The implementation treats each as a separate lane with its own subject, evidence method, effect owner, and completion horizon. A source commit may be complete while a provider readback is held. A provider effect may be live while recipient delivery remains unobserved. A configured client may be adopted while a new child remains gated.

11.3 The governance surface

Forge holds accepted state through exact source, current pointers, FIELD and DENT contracts, versioned candidates, root convergence, receipts, and declared nonclaims. Its canon explicitly denies Ubiquity tics any power to apply or mature Forge candidates. [20-27]

11.4 Exact-lane examples from the Forge corpus

The September 3 Forge maturity matrix demonstrates why one global label would be misleading. [30]

- The OpenAI Responses adapter retained exact-scope production standing for Forge, Get Scouted/YPS, and LeaderRoll demo lanes, while invoice cost, BYOK, entitlement, caps, rebilling, tool authority, conversion, and business adoption remained separate.
- The Brightwheel school-system adapter was operator-attested as fully functional and adopted by one configured client, while a new-child launch remained a different horizon.
- The Resend/YPS delivery lane had runtime, canary, and continuation evidence, while exact recipient delivery and readback remained per-message facts.
- Firecrawl had one exact bounded production capture ticket, while sustained operations, rights acceptance, a second named child, and named adopter readback remained open.
- Other regulated or client lanes retained synthetic, approval, identity, PHI, counsel, or acceptance gates rather than inheriting maturity from neighboring lanes.

These are examples of PHS business discipline: a system may be genuinely live in one exact scope and still honestly held in another.

11.5 A typical engagement in business language

A Prompted Forge engagement can be described as the following sequence.

Orient. Establish the client purpose, exact subject, current state, target state, decision owners, prohibited assumptions, and consequence level.

Bound. Select the smallest FIELD slice and executable DENT or task graph that can address the objective without importing unrelated authority.

Build in parallel. Assign independent workstreams and verification surfaces. Preserve shared-writer and external-effect boundaries.

Converge deliberately. Use guarded promotion and one root convergence path for accepted shared state.

Prove the horizon actually claimed. Perform direct readback, independent acquisition, runtime observation, provider confirmation, recipient readback, client acceptance, or outcome measurement as required by the claim.

Carry and re-enter. Preserve receipts and nonclaims in a later carrier so the next person or agent can resume from the first open edge.

11.6 What the client should receive

A client-facing decision package should answer:

1. What exact deliverable or decision is this?
2. What is its current horizon?
3. Which checks passed and which remain held?
4. Who owns the next decision or effect?
5. What does the current evidence explicitly not prove?
6. Where may an authorized reviewer inspect the supporting evidence?

That is legibility without reverse authority.

12. The Forge technical implementation in business language

Forge's current implementation is expressed natively through FIELD, DENT, DAG contracts, revisioned candidate lanes, guarded promotion, root convergence, exact current pointers, PHS checkpoint epochs, and source-tense evidence. Covenant, cable, board, winch, GUNSLINGER, HIDALGO, and DissonanceBasin remain important upstream Ubiquity lineage terms; in this section they are used only where explicitly labeled as analogies. [2,20-27]

12.1 FIELD: the bounded reality slice

In business language, FIELD answers: **what part of reality are we actually dealing with right now?**

The current Forge FIELD pointer binds a versioned target, exact hashes, source head, checkpoint dispositions, source gate, intended use, validation gate, uncertainty, and effect ceiling. It also says directional_only and field_authority: false. [22,24]

FIELD provides direction without pretending direction is permission. It can identify what matters, what is excluded, and what uncertainty remains. It cannot satisfy a dependency, grant access, apply a migration, deploy a provider, send a message, or declare adoption.

12.2 DENT and the DAG: dependency-aware execution

In business language, DENT answers: **what must precede what, and which work can lawfully proceed now?**

The current Forge DENT pointer is routing-and-ordering only. The Wave 38 executable graph records workstreams, causal nodes, effect ceilings, dependencies, proof horizons, checkpoint contracts, completed and held states, and terminal junctions. [23,24]

A DAG is useful because bounded work has causal predecessors. It is not the whole living organization. It is the finite discharge structure for one admitted slice.

12.3 Workstreams, nodes, owners, and effect ceilings

A workstream names a responsibility. A node names a concrete causal obligation. The same workstream may own several nodes, and one node may require evidence from several surfaces.

Each node receives an owner and effect ceiling. Position in the graph does not grant authority. A node that proves source identity does not thereby own provider mutation. A terminal node does not automatically own client acceptance or outcome.

12.4 Candidate lanes and guarded promotion

Parallel work lands in revisioned candidate surfaces rather than silently overwriting shared canon. Guarded promotion verifies the expected preimage, candidate identity, write set, current checkout, and relevant receipts before shared state changes. The integration lease protects pre-existing dirty work from being destroyed as collateral. [27]

Business meaning: many teams may prepare proposals, but the shared record changes only through a bounded, inspectable admission path.

12.5 Root convergence

Root convergence is the single logical path through which accepted shared state integrates. This is the Forge expression of PHS serial canonical authority.

Root convergence does not mean one person performs all work. It means many proposals and proofs converge through one accountable mutation path rather than becoming competing canons.

12.6 PHS checkpoint: prove the source, then prove the carrier

After source admission, Forge separates direct target readback from independent clean acquisition. Those proof lanes inspect the same exact source through different failure surfaces. Their receipts join into a descendant carrier. The carrier is then read back and independently acquired in a second checkpoint. [25,26]

Business meaning: the organization proves both the thing and the evidence package used to support the decision.

Git supplies the content-addressed subject and ordered successor substrate in the reference implementation. in-toto and SLSA can carry compatible provenance and attestation material, but they do not replace PHS horizon, authority, join, or carrier-recursion semantics. [31-33]

12.7 Current pointers: current is not authorized

FIELD_CONTEXT_CURRENT and FORGE_DENT_CURRENT tell successors which exact contracts currently govern the bounded source slice. They do not grant execution authority. [24]

A pointer can be current while a provider effect is held. A graph can be current while a client decision remains open. Currentness is a routing fact, not a maturity crown.

12.8 HELD state and deterministic re-entry

When a proof, preimage, source, or dependency fails, the local branch yields HELD rather than forcing the entire system into a vague failure. Completed siblings remain legible. The next actor performs a fresh LOOK and resumes at the earliest failed owner or unproven predecessor. [2,23,27]

Business meaning: interruption becomes a named missing edge, not a meeting to reconstruct what everyone meant.

12.9 Relationship to the Ubiquity precursor vocabulary

The upstream and downstream expressions rhyme, but they are not mechanically identical.

Ubiquity precursor term	Business intuition	Forge-native implementation surface
Covenant	Bounded current-to-target commitment	FIELD/DAG contract, candidate scope, closure predicate
Cable	Owned obligation carrying tension and evidence	Workstream or causal node with owner and effect ceiling
Board	Admitted dependency compilation	DENT executable graph and current pointer
Winch	Raise the lawful executable frontier	DENT scheduling, candidate execution, guarded convergence
GUNSLINGER / HIDALGO	Concurrent or staged execution tempo	Concurrency chosen by graph dependencies and effect risk; not a required Forge enum
Held basin	Keep blocked residue visible	HELD node, hold reason, re-entry owner
DissonanceBasin	Preserve tension without flattening it	No direct identity claim; business analogy only unless an exact Forge contract binds it
Frozen center	Preserve the governing reference frame	FIELD exclusions, effect ceilings, nonclaims, and parent authority boundaries

This crosswalk is one-way. It helps explain lineage. It does not install Ubiquity runtime semantics into Forge or rename Forge's current contracts.

13. Wave 38: one Forge checkpoint worked through legibly

Wave 38 in the Forge repository is the cleanest technical specimen because it exposes the complete PHS topology through machine-readable nodes and receipts.

The business question was not merely "did the commit push?" It was "can the system establish that one exact source is current on the intended surfaces, reproducible from a clean acquisition, carried in durable evidence, and followed by a carrier that is itself current and reproducible?"

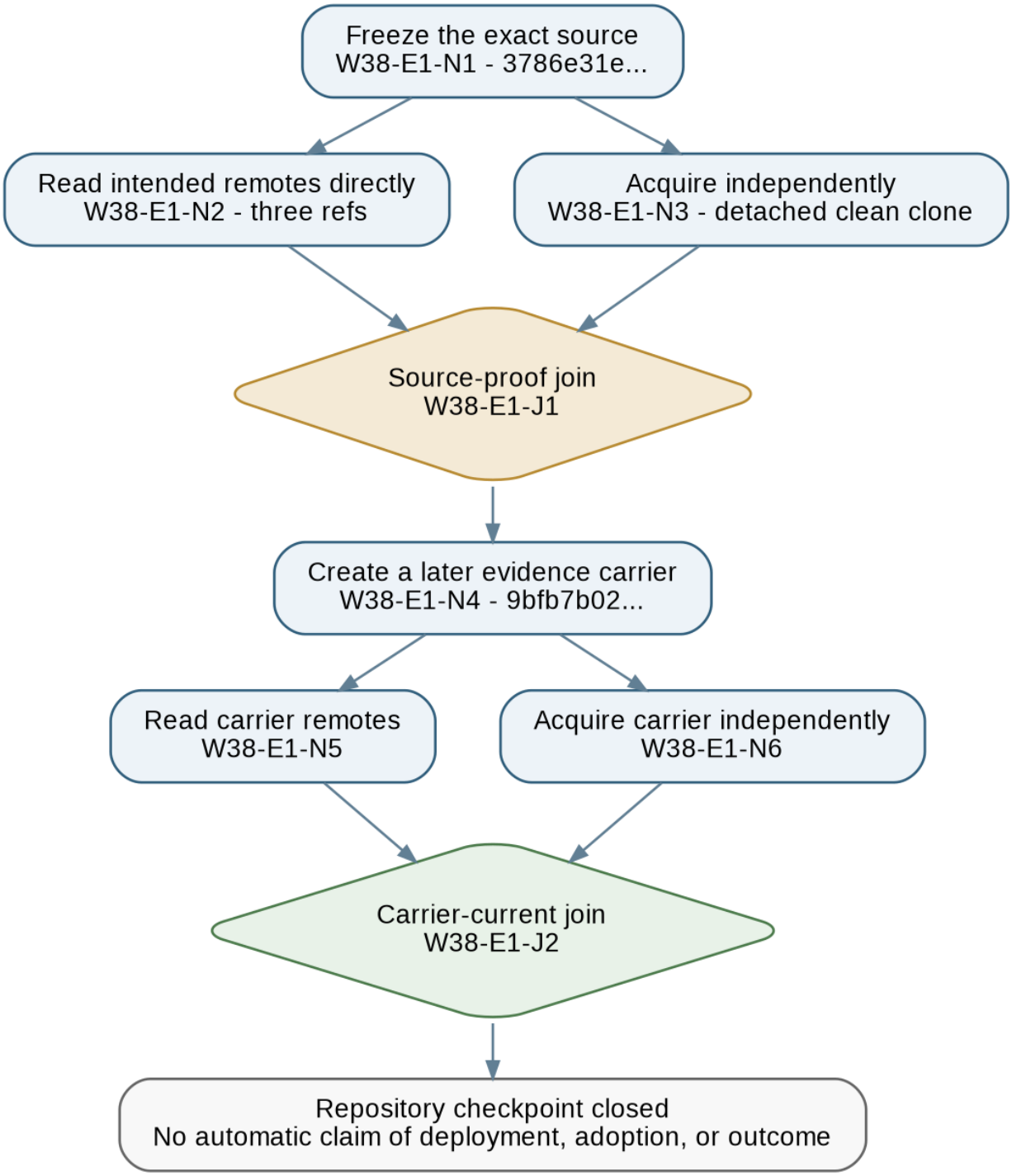

Figure 6. Wave 38 translated into business language. The exact source is proved through direct readback and independent acquisition; later observations enter a descendant carrier, which is then proved.

13.1 Freeze the exact subject

The first checkpoint selected source commit 3786e31ec5d95335d26207e2f0f1cfebda509f14. The postimage receipt bound the exact commit, tree, parent, subject, and changed-path set. [25]

Business meaning: the organization stopped saying “the current work” and named the exact thing the evidence would concern.

13.2 Check distribution directly

A separate lane read three intended refs directly and confirmed that they exposed the exact selected source. [25]

Business meaning: a successful send or push report was not treated as proof of receipt. The target surfaces were observed.

13.3 Acquire independently

A separate clean-clone lane acquired the exact subject without inheriting the authoring checkout's untracked or stale state and ran the declared validations. [25]

Business meaning: "it works on the builder's machine" was separated from "another declared environment can acquire and validate it."

13.4 Join the source evidence

Only after the remote and clean-clone receipts passed did the source-current join close. [25]

Business meaning: two different risks were not flattened into one green check. Distribution and independent reproduction remained separate evidence classes at convergence.

13.5 Create the evidence carrier

A descendant carrier commit, 9bfb7b02c5b040b5fc66cd0338727d9deeb50bc0, preserved the source proof receipts and their limits. [25]

Business meaning: later evidence was put into a later package. The original source was not rewritten to imply it had known its own future.

13.6 Prove the carrier

The carrier was then read directly from the intended refs and independently acquired through a clean clone. Only after those checks did the carrier-current join mark the checkpoint completed. [25]

Business meaning: the organization proved not only the deliverable but the availability and reproducibility of the evidence package used to support the decision.

13.7 Repeat the recursion

Wave 38 then ran a second checkpoint epoch. The later source 30b1dddc500012ad46d380ce0e62a30d3909302e produced carrier 9dffef4cb694570d90f65ded4e76359a0763e80e, with the same source/readback/clone/carrier/readback/clone structure. [26]

Business meaning: PHS is not a one-time certification sticker. As accepted state changes, the new subject receives a new proof path.

13.8 Preserve the nonclaims

The receipts explicitly denied stronger claims including product deployment, database mutation, provider mutation, stakeholder send, maturity uplift, adoption, and outcome. [21,22]

Business meaning: the repository checkpoint was allowed to be complete without pretending the business was complete.

14. Ubiquity precursor record: where the form learned its refusals

This section is upstream lineage, not downstream case evidence.

THE BINDER and every tic named below belong to Prompted LLC's Ubiquity Canonical Federation. The Binder is an evidence map rather than a primary receipt. Its BND markers resolve through lane extracts to source artifacts, and its honest negatives are load-bearing. [14-19]

UBIQUITY FEDERATION CLOCK canonical_federation / Prompted LLC	FORGE FEDERATION CLOCK forge-canonical-fed / Skyward Prompted LLC d/b/a Prompted Forge
tics 414-494 Harpoon office recharter; hoist covenant	**August 14** Filtered Forge delivery-velocity audit
tics 535-623 Capability ruling overturned; winch modes; vacuous-green cure	**August 21-31** Forge FIELD/DENT dated wave cycles
tics 643-751 Parallel citizens; full S-chain; local-model lap and refusal/rerun	**September 3** Wave 38 exact source; remote and clean-clone source join
tics 757-770 Backlog campaign; horizon quiver; THE BINDER evidence map	**September 3-4** Receipt-carrier checkpoint and current Forge implementation state
LINEAGE ONLY - NO SHARED CLOCK OR CANON AUTHORITY. Ubiquity tics do not mature Forge candidates. Forge receipts do not rewrite Ubiquity history.	

Figure 7. Two source-bound clocks. Ubiquity's tic history and Forge's dated cycle history are related by lineage but do not share canonical authority.

14.1 The founding verb was falsified - Ubiquity tics 414-416

The Harpoon office's initial "master DAG-of-DAGs" framing was rejected and replaced with "metabolize, not master." The operating lesson was that governance should preserve contradiction and re-parenting evidence rather than force consensus into the appearance of coherence. [15-17]

14.2 The hoist covenant - Ubiquity tic 494

A ten-cable, seven-wave staged-lock DAG expressed one current-to-target pull while keeping the founding telos outside the graph as a non-node. The business lesson was that a target can govern the work without becoming a box the system declares complete. [15-17]

14.3 A capability wall was overturned - Ubiquity tic 535

A prior not-reachable assessment was overturned only when the exact model served live on the measured machine. Mutation authority remained false. The lesson was not "the model always works"; it was "re-test capability at the act, and do not promote execution into authority." [15-17]

14.4 Seven citizens moved in parallel - Ubiquity tic 643

Seven office-citizens executed simultaneously on disjoint surfaces. Seven other executable identities without admitted covenants were excluded. The business lesson was that capability and admission are separate set memberships. [15-17]

14.5 The whole S-chain moved inside one tic - Ubiquity tic 650

Seven work cables ran through plan, simulation, and fire while six null-covenant identities remained visible as residue. The business lesson was that speed is lawful only when exclusions and residue remain named. [15-17]

14.6 Vacuous green was exposed - Ubiquity tics 502-623

A false board-wiring claim survived roughly 119 Ubiquity tics until the first real consumer found mutually concealing defects. The cure was fail-closed state and additional orthogonal status axes. The lesson was that presence is not enforcement and a real eater is stronger than a declarative green. [15-17]

14.7 Every completed campaign wave falsified the lead - Ubiquity tics 757-770

The GUNSLINGER-BACKLOG campaign preserved a lead-premise falsification streak across completed waves. Its fastest staged-to-closed arc was reported at 58 minutes. The business lesson was that a well-designed form allows bounded workers to out-falsify the lead without becoming uncontrolled authorities. [15-17]

14.8 The observer occluded the observed - Ubiquity tic 540

A telemetry marker suppressed a live cable's receipt. The cure separated run metadata into a dedicated key-space, and the confession remained in the ledger. The lesson was that evidence tooling can become part of the failure surface and must itself be testable. [15-17]

14.9 The horizon probe demoted its own name - Ubiquity tics 768-770

The Ubiquity horizon-quiver build created a probe named for detached reproduction. Because it made no remote contact, its receipt stayed at source_admitted rather than claiming the stronger rung. The business lesson was that implementation presence and confident naming do not establish a proof horizon. [15-17]

14.10 Honest negatives from the upstream record

The precursor history also records what did not happen.

- Fixture-green was not automatically live-fired.
- The economy remained SimOnly, not live money.
- The tic-535 serve was proposal-only.
- No campaign wave was proven to have traversed WinchDial.
- basin_drained was not observed as a live event in the 213-row ledger.
- The rollback hold was armed but had zero live refusals.
- The strongest named real-basin observation occurred once.

These negatives explain design pressure. They do not become Forge implementation results merely because Forge inherits the architecture's lessons.

15. Forge operating scale and what the numbers mean

The supplied Forge repository experienced a sharp acceleration in gross tracked change during August 2026. Author-supplied GitHub Insights screenshots report 12,697,294 additions across the August 1-September 1 window, 8,935,647 additions in the final week, 6,723,886 in the final three days, and 3,928,484 in the final twenty-four hours. [29]

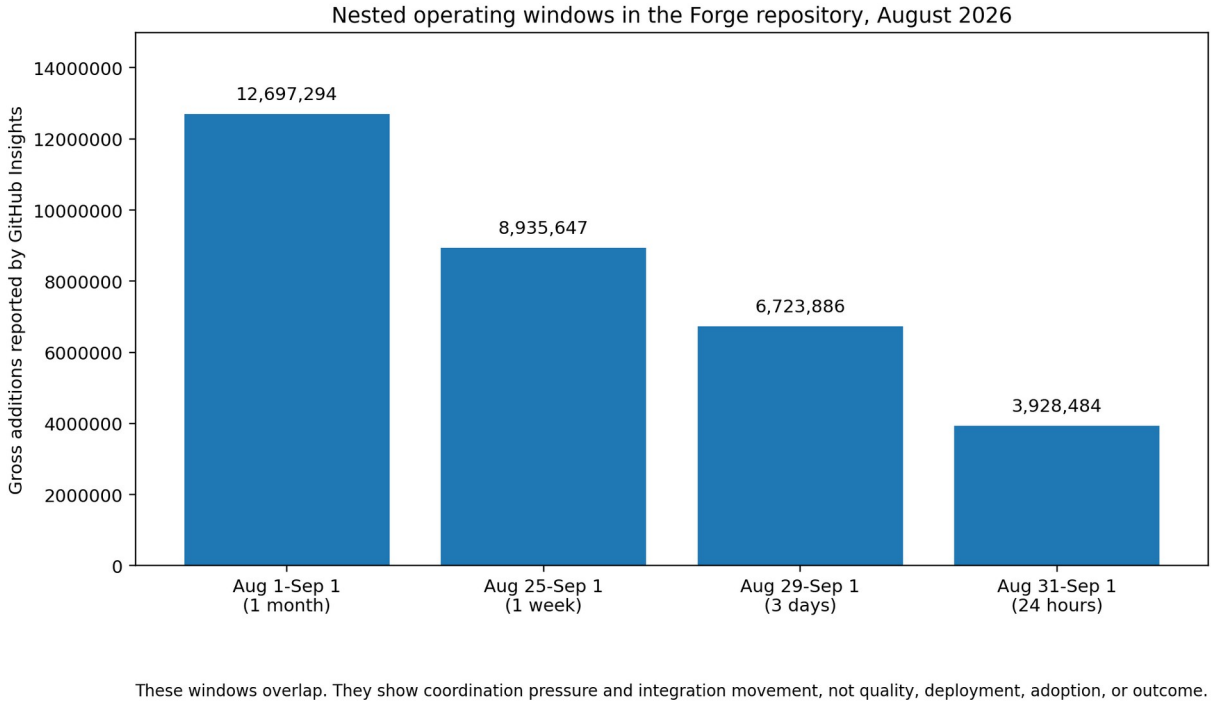

Figure 8. Nested GitHub Insights windows. These overlapping figures show gross tracked additions and coordination pressure, not quality, deployment, adoption, or outcome.

Approximately 53 percent of the monthly additions shown occurred in the final three-day window, and approximately 31 percent occurred in the final twenty-four hours. The same screenshots report 567 monthly commits, 13,909 files changed, zero merged pull requests, and one author identity.

Those figures are extraordinary and easy to misuse.

They do **not** establish that every line was unique, hand-authored, executed, deployed, adopted, valuable, or causally produced by PHS. They include a large governance, evidence, documentation, and generated-carrier surface. They show that the coordination problem crossed beyond the practical envelope of conventional line-by-line human review.

A separate Forge-repository delivery-velocity audit deliberately filtered generated reports and generated database types. In one trailing ninety-six-hour window it recorded 43,256 runtime/tooling and test insertions, 3,711 deletions, thirty-one new tests, twenty new migrations, and thirteen new deploy/proof receipt files. It framed the defensible claim narrowly: Forge repeatedly converted large cross-cutting changes into typed source, tests, migrations, UI, governance boundaries, and exact proof artifacts at high cadence. [28]

The business meaning is therefore not “millions of lines equal value.” The stronger meaning is:

> **The operating system entered a post-comprehension-scale production regime and needed assurance primitives that did not depend on one human reading every diff.**

At that scale, the control question changes. The organization cannot ask whether one reviewer comprehended every line. It must ask whether the right subjects were fixed, the right predicates were executed, the right authorities were preserved, the right failure surfaces were independently observed, the right receipts were carried, and the right stronger claims remained open.

The absence of pull requests is therefore neither automatically impressive nor automatically defensible. A zero-PR workflow without replacement controls is ungoverned. A zero-PR workflow with exact subjects, bounded work surfaces, independent proof where required, guarded integration, correction lineage, and receipted convergence may be governed through a different primitive. The burden remains on the receipts.

The Ubiquity tic history is not included in these Forge velocity windows. It is a different repository, clock, and evidence domain.

16. Client-facing expression versus internal canon

The business paper itself demonstrates the projection problem it describes.

A client should not have to read a machine DAG, source manifest, commit tree, or proof receipt bundle to understand whether a deliverable is ready for a decision. At the same time, a simplified dashboard or executive summary cannot be allowed to become the authority merely because it is easier to read.

The lawful pattern is one-way:

source-bound canonical subjects, contracts, receipts, and decisions
-> audience-specific business expression
-> stakeholder orientation and action

not:

business expression
-> silent redefinition of canonical state

16.1 What may be simplified

A client view may replace a commit hash with a human title plus a visible exact-version link. It may summarize ten proof receipts as “distribution and independent reproduction confirmed.” It may group several held nodes under one plain-language dependency. It may translate “effect ceiling” into “what this team is allowed to change.”

The simplification is lawful when the exact source remains reachable and the expression does not strengthen the claim.

16.2 What may not be simplified away

The following distinctions must survive every client-facing expression:

- which exact deliverable or decision the status concerns;
- whether the state is candidate, accepted, distributed, reproduced, deployed, adopted, or outcome-measured;
- who owns the next decision;
- what is held and why;
- what the evidence does not prove;
- whether the displayed information is current and from which source;
- how an authorized reviewer can reach the underlying evidence.

A status card that says “Live” without distinguishing source-live, provider-live, client-live, user-live, or outcome-live is not PHS-legible.

16.3 Client confidence without evidence dumping

PHS does not require exposing every internal artifact to every client user. Receipts may contain confidential paths, identities, security details, or client data. The implementation can render selective views while preserving access controls and an auditable path to the canonical evidence.

A useful client expression has three layers.

Orientation layer. What is this, why does it matter, and what decision is requested?

Evidence layer. What checks passed, which checks are held, and what exact claim is supported?

Proof layer. Which authorized reviewers can inspect the underlying subject-bound receipts, versions, and readbacks?

The layers should reveal more detail without changing the underlying status.

16.4 The client-facing carrier

A business-facing decision packet may itself become a carrier when it truthfully binds the underlying evidence and is preserved as an exact successor artifact. But most executive summaries are merely projections.

The distinction depends on semantics, not format. A PDF can be a carrier or a projection. A database row can be a carrier or a projection. A dashboard can be authoritative or nonauthoritative. The question is whether the artifact has exact subject binding, temporal admissibility, authority, evidence bindings, nonclaims, and a lawful convergence role.

17. Business value and measurable hypotheses

PHS should not be sold as a magical quality layer. Its value depends on the cost of false closure, the amount of parallel work, the quality of predicates, the clarity of authority, and the organization's willingness to preserve inconvenient evidence.

The source-bound working Prompted Forge case supports several business hypotheses. They remain hypotheses until measured against a suitable baseline.

17.1 Lower false-closure cost

False closure occurs when one horizon is mistaken for another: source-ready is called deployed; deployed is called adopted; a green fixture is called live; a sent artifact is called received; a received artifact is called understood.

PHS should reduce the frequency and duration of those errors by requiring the claim class, subject, evidence, and next horizon to remain explicit.

Possible measures include:

- number of status reversals caused by horizon collapse;
- time between first false-green signal and discovery;
- number of downstream decisions made from stale or weaker evidence;
- cost of remediation attributable to premature closure.

17.2 More parallel capacity per authority bottleneck

PHS allows proposal and proof to fan out while keeping accepted-state mutation bounded. The expected benefit is more useful work per unit of scarce founder, client, counsel, or domain-expert attention.

Possible measures include:

- concurrent bounded work lanes;
- percentage of lanes that require no mid-flight authority clarification;
- founder or client approval minutes per admitted change;
- ratio of observer/worker count to canonical-writer count;
- queue time at shared-writer critical sections.

17.3 Faster re-entry after interruption

A successor should be able to find the exact subject, completed proofs, pending proofs, current authority, and next missing edge without reconstructing the previous session.

Possible measures include:

- median time from cold start to lawful next action;
- percentage of resumptions requiring prior operator intervention;
- repeated discovery work after handoff;
- number of hidden dependencies found only through private context.

17.4 Lower correction debt

Correction debt is the future cost created when a failure, changed premise, or superseded decision is narratively erased.

Possible measures include:

- percentage of corrections linked to the original subject and reason;
- recurrence of previously observed failure classes;
- time spent reconciling conflicting summaries;
- proportion of later decisions that can cite the exact correction lineage.

17.5 Better client trust through bounded claims

PHS should improve trust when stakeholders can see not only what passed but what remains unproven, who owns the next decision, and how evidence may be re-derived.

Possible measures include:

- client questions resolved without reconstructive meetings;
- acceptance reversals caused by ambiguous status;
- time from delivery to informed client decision;
- percentage of client-facing claims with reachable supporting evidence;
- number of explicit nonclaims later shown to have prevented misunderstanding.

17.6 Lower audit reconstruction cost

The architecture is designed so an auditor traverses durable evidence rather than reconstructing a project from chat logs and memory.

Possible measures include:

- auditor hours per checkpoint;
- missing or ambiguous subject identities;
- receipts lacking method, authority, or nonclaim fields;
- time required to reproduce an independently checkable result;
- percentage of sampled claims traceable to primary evidence.

17.7 Value is not receipt volume

A poorly designed PHS implementation can create enormous evidence volume without increasing truth. The success metric is not the number of receipts. It is the reduction of consequential ambiguity and the increased ability to move safely.

A compact metric frame follows.

Throughput. Desired movement: more bounded work and proof in parallel. Counter-metric: no increase in uncontrolled canonical writers.

Assurance. Desired movement: fewer false closures. Counter-metric: no inflation of weak predicates.

Continuity. Desired movement: faster lawful re-entry. Counter-metric: no dependence on private session memory.

Correction. Desired movement: more preserved, usable scars. Counter-metric: no narrative rewriting of failed subjects.

Client legibility. Desired movement: faster informed decisions. Counter-metric: no claim strengthening in the business projection.

Outcome. Desired movement: more accurately measured business effect. Counter-metric: no substitution of repository metrics for client value.

18. Where PHS fits - and where it does not

The control problem is universal. The decision to implement a formal PHS profile is contextual.

18.1 Strong fit

PHS is especially useful when several of the following are true:

- many humans, agents, vendors, repositories, or systems participate;
- candidate generation is faster than human review capacity;
- shared writers or irreversible effects create coordination risk;
- the same word such as “done,” “live,” or “approved” routinely carries several meanings;
- evidence is produced on different surfaces or at different times;
- client, legal, security, provider, and business authorities are not the same;
- corrections must remain auditable;
- work must survive model, vendor, operator, or session changes;
- a mistaken status can create material cost, harm, liability, or reputational damage;
- the organization needs to prove not only what it built but what another surface actually received or reproduced.

18.2 Light fit

A lightweight PHS expression may be enough when the work is bounded and low consequence. The organization may need only:

- exact version identity;
- a simple evidence checklist;
- one explicit decision owner;
- a short nonclaim section;
- a next-horizon field;
- a correction link.

The universal control law does not require maximum ceremony.

18.3 Weak fit or anti-fit

PHS is a poor fit when its evidence cost exceeds the consequence of ambiguity, when no exact subject can reasonably be identified, when the organization refuses to define decision rights, or when predicates cannot be made meaningful.

It is also a poor fit as camouflage. Adding hashes, dashboards, or receipt-shaped documents to an ungoverned process does not create PHS. A verifier who can silently mutate canon is not a zero-authority proof lane. A summary with no evidence path is not a carrier. A staged workflow that calls every gate complete is not horizon separation.

18.4 What PHS does not provide

PHS does not automatically determine:

- the correct business strategy;
- the sufficiency of a test;
- the ethical legitimacy of a goal;
- the benevolence of the canonical authority;
- legal or regulatory compliance;
- security of the underlying infrastructure;
- authenticity of a dishonest observer's semantic claim;
- client adoption or outcome;
- a universal software architecture.

It decomposes the trust and evidence problem. It does not eliminate judgment.

19. Risks, failure modes, and controls

A business-facing implementation introduces risks of its own.

19.1 Semantic drift

Risk. Business terms gradually replace native terms and acquire reverse authority.

Control. Maintain the one-way crosswalk, direct links to canonical subjects, explicit projection labels, and a policy that native contracts and receipts govern conflicts.

19.2 Receipt bureaucracy

Risk. Teams generate evidence artifacts because the process demands them, without improving the decision.

Control. Require every receipt to name the decision or claim it supports, its consumer, its predicate, and its next horizon. Retire receipts with no lawful eater.

19.3 Weak predicates

Risk. Perfectly executed checks prove something trivial or irrelevant.

Control. Assign predicate design to domain owners, use negative controls, periodically test whether checks discriminate, and preserve cases where a green result was vacuous.

19.4 Singular authority bottleneck or capture

Risk. One logical writer preserves consistency but becomes unavailable, slow, or compromised.

Control. Separate logical singularity from single-person control. Use quorum-backed approval, succession policy, emergency transfer, and multi-party authorization behind one accepted-state path where consequence requires it.

19.5 Dishonest or compromised observers

Risk. A signed receipt may still contain a false semantic claim.

Control. Use independent observers, diverse methods, trusted execution, reproducible commands, direct readbacks, sampling, and cross-checking appropriate to the consequence.

19.6 Confidentiality and client exposure

Risk. Durable receipts reveal client data, security paths, identities, or commercial information.

Control. Use access control, redaction, selective disclosure, encrypted evidence, scoped client projections, and separate public and private carriers.

19.7 Currentness decay

Risk. A previously correct receipt remains discoverable after the underlying environment changes.

Control. Bind claims to observation times or causal positions, add validity windows where appropriate, require Look-First at consequential boundaries, and make stale state visible rather than silently defaulting to current.

19.8 Projection or source-domain capture

Risk. A dashboard becomes easier to access than the canon and is eventually treated as the source of truth.

Control. Label projections, show source identity and refresh time, prevent projections from directly minting authority, and provide a canonical evidence path.

19.9 Over-instrumentation

Risk. The system spends more effort proving motion than producing value.

Control. Scale assurance to consequence, reuse stable proof primitives, automate low-risk evidence acquisition, and keep human judgment focused on the edges of earned trust.

19.10 Client confusion

Risk. A client interprets bounded language as evasive or assumes a held horizon means failure.

Control. Explain the claim ladder early, show why each open horizon belongs to a named owner, and distinguish an honest hold from an abandoned obligation.

20. Adoption path

Organizations should not begin by copying the entire Prompted Forge stack. The implementation is a local expression built inside the Ubiquity ecosystem. Adoption should begin with the earliest causal gap.

20.1 Step 1 - Find the collapsed word

Identify the overloaded status causing the most harm: *done*, *live*, *approved*, *ready*, *current*, *deployed*, or *successful*.

List the different facts that word is currently hiding.

20.2 Step 2 - Name the source domain and exact subject

Choose one consequential deliverable or decision and establish an exact identity. In software this may be a commit, digest, build, model, or configuration. In another domain it may be a document version, policy package, dataset snapshot, signed plan, or case record.

20.3 Step 3 - Separate the horizons

Write the claims that matter in causal order. Typical questions are existence, distribution, independent reproduction, deployment, stakeholder acceptance, adoption, and outcome.

Do not force every organization into the H0-H5 Git profile. Use the PHS protocol to define the relevant local horizons without collapsing them.

20.4 Step 4 - Map decision rights

For each horizon, identify who may propose, observe, receipt, admit, deploy, accept, and measure outcome.

Access is not authority. Title is not automatically standing. Position in the DAG is not permission.

20.5 Step 5 - Design the minimum useful receipts

A useful first receipt includes:

- exact subject;
- claim/horizon;
- method and predicate;
- result;
- observer;
- authority ceiling;
- evidence references;
- limitations and nonclaims;
- next horizon.

20.6 Step 6 - Add one independent proof surface

Choose the most consequential hidden failure surface. In software this may be direct target readback or clean acquisition. In business operations it may be independent reconciliation, client confirmation, signature validation, or outcome sampling.

20.7 Step 7 - Preserve one correction

Do not overwrite the first failed or superseded subject. Link the corrected successor and repeat only the proofs invalidated by the change.

This is where the process begins to accumulate operating memory rather than status debris.

20.8 Step 8 - Build the business projection last

Only after the canonical subject, evidence, authority, and correction path exist should the organization build the executive dashboard or client summary.

The projection should make the native state easier to understand without becoming the native state.

20.9 Step 9 - Measure the delta

Compare the pilot against a prior workflow on false closure, re-entry time, decision latency, defect discovery, correction recurrence, audit effort, and client understanding.

Do not use receipt count or line count as the primary success measure.

20.10 Step 10 - Expand by consequence, not fashion

Add PHS rigor where the cost of ambiguity is material. Keep low-consequence work light. A universal control law remains healthy when its expressions are proportionate.

21. Conclusion

PHS begins from a refusal that is simple in business language and strict in architecture:

> **No person, artifact, dashboard, model, workflow, or repository should be allowed to claim more than the evidence and authority available at its actual horizon.**

From that refusal follows an operating model.

Name the exact thing. Separate the claims. Let many people and agents work and inspect in parallel. Keep authority explicit. Preserve receipts with their limitations. Admit accepted state through one intelligible path. Put later observations into later carriers. Prove those carriers when their own availability matters. Preserve corrections. Resume from the first missing edge. Require new evidence for deployment, adoption, and outcome.

The corrected source account matters as much as the business translation.

Prompted LLC's Ubiquity Canonical Federation is the upstream constitutional and operational lineage. Telos is its capital. Its tic history and THE BINDER show where the architecture learned to preserve noncollapse, falsification, residue, anchoring, and receipts. Those events belong to Ubiquity. [14-19]

Skyward Prompted LLC d/b/a Prompted Forge is the downstream joint-venture implementation context co-founded by Breyden E. Taylor and Sabeel Ahmed. Its Forge Federation is a sovereign sister-city seeded from Ubiquity Telos, with its own canon, clock, candidate lifecycle, client lanes, FIELD/DENT contracts, and PHS checkpoint receipts. Those events belong to Forge. [20-30]

Breyden E. Taylor authored both the foundational PHS paper and this business implementation paper. Prompted LLC is the publication and rights context for the foundational protocol. Prompted LLC and Skyward Prompted LLC d/b/a Prompted Forge jointly endorse this downstream business expression. The 50% ownership and joint-venture facts are carried as author disclosure rather than silently promoted into repository proof.

The implementation is not the universal thing. The software is not the universal solution.

> **The universal thing is the discipline that accepted claims remain bounded by identity, evidence, authority, consequence, source domain, and time. Software is one powerful expression of that discipline.**

That is the business promise of PHS when stated honestly: not certainty, not frictionless autonomy, and not a prettier dashboard, but the ability to increase capacity without allowing an organization to become less truthful about what it knows, which source knows it, what it has accepted, and what has actually changed in the world.

Appendix A. One-way terminology and source-domain crosswalk

This appendix is a translation aid. It does not redefine native PHS, Ubiquity, or Forge semantics.

A.1 Subject

Business approximation: the exact thing being decided about.

Semantic ceiling: a subject is immutable or strongly identified. A project name or general initiative is not enough.

A.2 Proof horizon

Business approximation: the strongest claim currently earned.

Semantic ceiling: a horizon is not a confidence score or project phase. It is a bounded truth, authority, and consequence boundary.

A.3 Proof

Business approximation: evidence that passed a declared check.

Semantic ceiling: proof is bounded to a predicate and subject; it is not universal mathematical certainty unless expressly defined that way.

A.4 Proof shard

Business approximation: one independently runnable check.

Semantic ceiling: a proof shard is partitioned by evidentiary and temporal scope, not merely by team or task.

A.5 Predicate

Business approximation: the exact pass/fail/held question.

Semantic ceiling: a weak predicate can pass honestly and still be insufficient for the intended decision.

A.6 Receipt

Business approximation: a durable evidence record.

Semantic ceiling: a receipt binds subject, method, observer, result, authority, evidence, limits, convergence, and next horizon.

A.7 Carrier

Business approximation: a later package that can truthfully include newly available evidence.

Semantic ceiling: the carrier cannot prove its own later propagation or adoption.

A.8 Canon

Business approximation: the accepted source of truth for a bounded scope.

Semantic ceiling: canon is governed state, not whichever copy is easiest to access.

A.9 Admission

Business approximation: authorized acceptance.

Semantic ceiling: admission does not automatically imply apply, deploy, delivery, adoption, or outcome.

A.10 Authority ceiling

Business approximation: the strongest action or claim an actor may make.

Semantic ceiling: access, capability, title, graph position, or possession does not confer authority.

A.11 Join

Business approximation: controlled convergence of required evidence.

Semantic ceiling: a join cannot strengthen or homogenize its input receipts.

A.12 Projection

Business approximation: a view, copy, report, dashboard, or audience rendering.

Semantic ceiling: a projection does not become canon merely because it is readable or synchronized.

A.13 Nonclaim

Business approximation: an explicit statement of what is not yet established.

Semantic ceiling: nonclaims are part of the evidence contract, not legalistic decoration.

A.14 Scoped closure

Business approximation: no hidden required edge remains inside the declared scope.

Semantic ceiling: closure at one horizon does not close stronger horizons.

A.15 Deterministic re-entry

Business approximation: the next actor can find the next lawful action.

Semantic ceiling: re-entry depends on exact subject, completed and pending predicates, authority, and convergence target, not a prose handoff alone.

A.16 FIELD - Forge-native

Business approximation: the bounded current reality slice.

Semantic ceiling: Forge FIELD is directional context with field_authority: false; it does not execute or grant effects.

A.17 DENT - Forge-native

Business approximation: the dependency-aware execution route.

Semantic ceiling: Forge DENT orders and routes; graph position does not confer provider, database, deployment, client, or outcome authority.

A.18 Covenant - Ubiquity precursor

Business approximation: the bounded current-to-target commitment.

Source boundary: covenant is used here to explain Ubiquity lineage. It is not silently substituted for the current Forge FIELD/DAG contract.

A.19 Cable - Ubiquity precursor

Business approximation: one owned obligation carrying evidence and tension.

Source boundary: in Forge, the nearest current expression may be a workstream or causal node. The identity is analogical, not automatic.

A.20 Board or DAG

Business approximation: the map of dependencies.

Source boundary: the Ubiquity board is an upstream compiler surface. Forge's current downstream implementation is the DENT executable DAG. A DAG never equals the whole living field.

A.21 Winch - Ubiquity precursor

Business approximation: raising the lawful executable frontier.

Source boundary: the term carries Ubiquity operating history. Forge performs analogous movement through DENT scheduling, candidate execution, guarded promotion, and root convergence.

A.22 GUNSLINGER - Ubiquity precursor

Business approximation: high-concurrency execution after the frontier and shared writers are bounded.

Source boundary: this is not asserted as a required current Forge runtime enum or as permission for parallel shared writes.

A.23 HIDALGO - Ubiquity precursor

Business approximation: staged execution judged by residue shape.

Source boundary: this is an upstream tempo concept, not a Forge proof horizon or universal default.

A.24 DissonanceBasin and held basin - Ubiquity precursors

Business approximation: preserve unresolved tension and keep blocked obligations visible.

Source boundary: the Ubiquity record distinguishes the tension reservoir from the graph-level held node. Neither is automatically identical to a Forge receipt carrier or HELD state.

A.25 Open center or anchor - Ubiquity lineage

Business approximation: the governing reference frame is preserved rather than completed.

Source boundary: Forge expresses related constraints through FIELD exclusions, effect ceilings, parent authority, and nonclaims; the upstream identity remains distinct.

A.26 Ubiquity tic

Business approximation: an ordered Ubiquity Federation governance moment.

Semantic ceiling: a Ubiquity tic is not a Forge cycle, Git commit, wall-clock timestamp, or shared cross-federation clock.

A.27 Forge cycle or wave

Business approximation: a bounded downstream implementation tranche with source, graph, candidate, proof, and convergence state.

Semantic ceiling: a Forge wave does not inherit completion from a Ubiquity tic and does not prove a client outcome beyond its receipts.

Appendix B. Business checkpoint packet

The following is an illustrative business expression. It is not a substitute for the canonical machine-readable receipt profile.

B.1 Decision header

- **Decision requested:**
- **Client or internal owner:**
- **Exact subject title:**
- **Exact subject identity or canonical link:**

- **Declared scope:**
- **Current claim horizon:**
- **Decision deadline, if any:**

B.2 Evidence completed

For each applicable line, record the method, result, observer, evidence location, and limitations.

- Exact subject exists.
- Intended distribution surfaces expose the exact subject.
- An independent environment can acquire and validate it.
- A later evidence carrier exists.
- The carrier is independently available and validated.
- Declared terminal pointers reconcile.
- Deployment readback exists, when deployment is in scope.
- Stakeholder acceptance exists, when acceptance is in scope.
- Adoption evidence exists, when adoption is in scope.
- Outcome evidence exists, when outcome is in scope.

B.3 Decisions and authority

For each decision, name the owner with standing, the current status, and the supporting receipt or rationale.

- Candidate admission.
- Repository or source acceptance.
- Provider or infrastructure change.
- Database or data migration.
- Client deployment.
- Stakeholder acceptance.
- Adoption measurement.
- Outcome measurement.

B.4 Holds and residue

For each open edge, record:

- the exact edge or obligation;
- why it is held;
- the owner;
- the re-entry condition;
- the effect of delay;
- what remains protected while it is held.

B.5 Required nonclaims

Unless separately evidenced, the packet does not prove:

- deployment;
- stakeholder acceptance;
- adoption;
- business outcome;
- legal, security, or regulatory compliance beyond the exact predicates shown;

- currentness outside the observation window;
- identity beyond the exact subject binding.

B.6 Next horizon

- **First required unproven edge:**
- **Authorized next actor:**
- **Required evidence:**
- **Convergence destination:**
- **Strongest claim permitted after passage:**

Appendix C. Decision-rights model

The following is a business model for the Prompted Forge implementation. Actual engagement contracts and current receipts govern each case.

C.1 Client sponsor

May propose, define business direction, observe, and accept within the sponsor's standing. May own deployment or risk decisions when the engagement assigns that authority. May claim adoption or outcome only from a declared method and evidence.

C.2 Prompted Forge architect or integrator

May propose, inspect, coordinate, and integrate accepted source within the declared technical scope. May deploy only where separately authorized. Does not accept business outcome for the client by default.

C.3 Workstream owner

May propose and execute within the owned lane and may return receipts or holds. Lane completion does not grant root integration, deployment, client acceptance, or outcome authority.

C.4 AI agent or office citizen

May propose, inspect, test, and receipt within its contract and authority ceiling. It does not gain admission authority from fluency, technical capability, or successful execution.

C.5 Independent verifier

May observe and receipt the declared subject through the permitted evidence surface. It cannot silently mutate accepted state or mark the whole engagement complete.

C.6 Provider, database, or infrastructure owner

May control effects inside the native domain. Technical ownership of the provider or database does not confer client acceptance or repository-canon authority by implication.

C.7 Counsel, regulator, clinician, security owner, or domain specialist

May define or evaluate predicates and may own higher-consequence gates within declared standing. Domain standing does not silently transfer to unrelated horizons.

C.8 Outcome owner or analyst

May define and execute the declared outcome method and report a bounded measured result. A repository checkpoint, deployment receipt, or stakeholder acceptance cannot substitute for that evidence.

Appendix D. Source-bound evidence register

This register prevents lineage evidence from being mistaken for case evidence.

D.1 PHS foundational protocol [1]

Source domain: Prompted LLC publication.

Supports: canonical PHS terminology, horizon law, receipt recursion, correction survival, and protocol nonclaims.

Ceiling: does not establish any particular Forge implementation, deployment, client acceptance, adoption, or outcome.

D.2 Technical Forge implementation predecessor [2]

Source domain: Prompted LLC and Prompted Forge implementation publication.

Supports: detailed FIELD/DENT, DAG, guarded promotion, root convergence, and Wave 38 implementation mapping.

Ceiling: subordinate to the protocol paper; repository-scoped unless stronger evidence is separately named.

D.3 Prompted LLC public Ubiquity corpus [3-13]

Source domain: Prompted LLC public publication surfaces.

Supports: intellectual and constitutional lineage.

Ceiling: public prose does not independently establish current repository, deployment, client, or outcome state.

D.4 Ubiquity Canonical Federation root [14]

Source domain: canonical_federation archive, Prompted LLC.

Supports: Ubiquity repository identity, Telos as capital/federation seat, audit-log tic time authority, and source topology.

Ceiling: no Forge implementation, candidate, client, or effect authority.

D.5 Ubiquity THE BINDER [15]

Source domain: canonical_federation, Ubiquity tic 770.

Supports: map of twenty-two operational landmarks, one-loop winching thesis, timeline, truth boundaries, and honest negatives.

Ceiling: evidence map, not primary receipt; no Forge implementation or client claim.

D.6 Ubiquity Lane A and Lane B extracts [16,17]

Source domain: canonical_federation.

Supports: row-level quoted artifact evidence, timestamp discipline, code excerpts, campaign history, and liveness limits.

Ceiling: complete only at their named extraction instant; primary Ubiquity artifacts govern disagreements; no downstream Forge maturity transfer.

D.7 Ubiquity operating memos [18,19]

Source domain: canonical_federation agent mailbox.

Supports: bounded Ubiquity observations about reproducible receipts and one local-model lap.

Ceiling: no Forge implementation, general model capability, client adoption, or outcome.

D.8 Forge root identity and sister-city contract [20]

Source domain: forge-canonical-fed, Skyward Prompted LLC.

Supports: Skyward Bridge ownership/operation, seed from Ubiquity Telos, sovereign sister-city relation, and explicit denial of Ubiquity tic authority over Forge candidates.

Ceiling: no independent corporate ownership audit and no automatic client effect.

D.9 Prompted Forge cofounder source [21]

Source domain: forge-canonical-fed correspondence and cofounder briefing.

Supports: Breyden Taylor and Sabeel Ahmed as equal human cofounders and Sabeel as Prompted Forge cofounder.

Ceiling: technical access symmetry is expressly denied; 50% ownership and joint-venture legal characterization remain author disclosures here.

D.10 Wave 38 FIELD contract [22]

Source domain: forge-canonical-fed.

Supports: bounded reality slice, source classes, exclusions, effect ceiling, validation gate, and refusal of authority transfer.

Ceiling: directional context, not execution or outcome authority.

D.11 Wave 38 DENT and machine graph [23]

Source domain: forge-canonical-fed.

Supports: thirty-two-node dependency/proof topology, workstream mapping, yield conditions, and two named checkpoint epochs.

Ceiling: repository-source and currentness scope only.

D.12 Forge current pointers [24]

Source domain: forge-canonical-fed.

Supports: current selected FIELD and DENT state, hashes, checkpoint dispositions, and authority posture.

Ceiling: currentness, not execution authority.

D.13 Wave 38 E1 receipt set [25]

Source domain: forge-canonical-fed.

Supports: exact source postimage, three-ref direct readback, detached clean clone, source join, descendant carrier, carrier readback and clone, and completed carrier join.

Ceiling: repository checkpoint; no automatic product deployment, stakeholder acceptance, adoption, or outcome.

D.14 Wave 38 E2 receipt set [26]

Source domain: forge-canonical-fed.

Supports: second recursive source and carrier checkpoint.

Ceiling: same repository-scoped boundary.

D.15 Guard and lease artifacts [27]

Source domain: forge-canonical-fed.

Supports: same-checkout identity leases, expected preimages, guarded promotion, and dirty-set protection.

Ceiling: cooperative concurrency control, not hostile-user sandboxing or universal security.

D.16 Forge delivery-velocity audit [28]

Source domain: forge-canonical-fed.

Supports: filtered source, test, migration, and receipt integration evidence.

Ceiling: not quality, deployment, adoption, revenue, outcome, or causal proof.

D.17 GitHub Insights screenshots [29]

Source domain: author-supplied views of the Forge repository.

Supports: displayed gross nested line movement, commits, files changed, one author identity, and zero merged pull requests.

Ceiling: overlapping windows; not unique code value, labor attribution, deployment, adoption, or business outcome.

D.18 Forge maturity and client-lane matrix [30]

Source domain: forge-canonical-fed.

Supports: exact-lane maturity, provider and adoption distinctions, current holds, and explicit nonclaims as of its named source horizon.

Ceiling: source/receipt/operator-attestation-bounded orientation; exact child receipts and fresh provider, browser, database, recipient, client, or outcome readback still govern.

Appendix E. Business conformance questions

A business expression is PHS-compatible only when the answer to each applicable question is yes.

1. Is the exact subject identifiable?
2. Is the current claim horizon stated?
3. Are candidate, accepted, distributed, reproduced, deployed, adopted, and outcome states distinguishable?
4. Is every material claim tied to a declared evidence method?
5. Are observation and admission authority separated where required?
6. Does each actor have an explicit effect or claim ceiling?
7. Can work or verification fan out without multiplying uncontrolled canonical writers?
8. Are shared-writer critical sections explicit?
9. Are independent failure surfaces kept distinct at convergence?
10. Does a later evidence package carry observations that the earlier subject could not have known?
11. Is that carrier treated as a new subject when its own distribution or use matters?
12. Are client acceptance, adoption, and outcome protected from repository-status laundering?
13. Are nonclaims visible and decision-relevant?
14. Are holds visible rather than silently dropped?

15. Can a successor find the first required unproven edge?
16. Are failed and corrected subjects preserved in lineage?
17. Does the client-facing view link back to canonical evidence?
18. Is the client-facing view prevented from acquiring reverse authority?
19. Are confidential receipts protected through access control or selective disclosure?
20. Are current pointers distinguished from execution authority?
21. Are predicates tested for discrimination rather than mere pass rate?
22. Is the assurance burden proportionate to consequence?
23. Are live, fixture, simulation, source, and proposal states labeled distinctly?
24. Are company, client, provider, legal, and outcome authorities prevented from silently collapsing?
25. Is the final claim no stronger than the strongest evidence actually present?

Appendix F. Glossary

Accepted state. State admitted through the declared authority path for a bounded scope.

Admission. Authorized acceptance of candidate material without automatically implying deployment, adoption, or outcome.

Authority ceiling. The strongest mutation or claim an actor, lane, or receipt may lawfully make.

Business expression. A legibility-oriented projection of canonical semantics for a particular audience. It has no reverse authority unless separately admitted as a canonical carrier.

Canon. The accepted source of truth for a declared scope, with governed transitions and bounded evidence.

Carrier. A later durable object that can truthfully carry observations produced after its predecessor existed.

Correction debt. Future cost created when a failed premise, subject, or decision is obscured, overwritten, or detached from its successor.

DAG. A directed acyclic graph used to represent causal dependencies inside a bounded execution or proof discharge.

DENT. The Forge dependency-aware bounded executor and scheduler. The business term “execution route” is an approximation.

Effect ceiling. The strongest real-world or system effect permitted to an actor or node.

FIELD. A bounded reality-aware context surface that states direction, sources, uncertainty, exclusions, and effects without automatically granting execution authority.

Horizon. A truth, authority, or consequence boundary. In PHS, a proof horizon is the strongest bounded claim the available evidence supports.

Nonclaim. An explicit denial of a stronger interpretation not supported by the current evidence.

PHS. Proof-Horizon Sharding: partitioning verification according to when and through what independent path each fact can lawfully become knowable, while accepted mutation remains governed.

Projection. A view, copy, mirror, dashboard, report, or audience expression that does not independently decide what becomes canonical.

Receipt. A durable record binding a result to its exact subject, method, observer, evidence, authority, limitations, and next horizon.

Root convergence. The one logical accepted-state integration path for a declared Forge scope.

Scoped closure. A state in which no required hidden edge remains inside the declared scope. It does not imply permanent completion or stronger business outcome.

Subject. The exact immutable or strongly identified object under discussion or proof.

Successor horizon. A later evidence or consequence domain requiring new observations and often new authority.

Universal solution with software expression. The claim that PHS addresses a domain-general governance relation among identity, evidence, authority, consequence, and time, while every practical implementation remains local and domain-specific.

Zero-authority proof lane. A lane permitted to inspect and receipt an exact subject but prohibited from silently admitting its own result into accepted state.

References

[1] Taylor, B. E. (2026). *Proof-Horizon Sharding: Correction-Surviving, Proof-Carrying Canonical Computation* (Technical Whitepaper 1.2, Operational Genesis and Winching Lineage Revision). Prompted LLC.

[2] Taylor, B. E. (2026). *The Forge Implementation of Proof-Horizon Sharding: FIELD/DENT DAGs, Guarded Convergence, and Recursive Checkpointing at Agentic Production Scale* (Technical Whitepaper 1.0). Prompted LLC and Prompted Forge.

[3] Prompted LLC. “Prompted LLC - Ubiquity, Context Grapple Gun, Corpus.” https://promptedllc.com/. Accessed September 4, 2026.

[4] Prompted LLC. “Prompted LLC - Governance Substrate for Sovereign Adaptive Systems.” https://promptedllc.com/prompted-llc. Accessed September 4, 2026.

[5] Prompted LLC. “Breyden Taylor - Founder, Prompted LLC.” https://promptedllc.com/founder. Accessed September 4, 2026.

[6] Taylor, B. E. (2026). *FORKED - Observer-Indexed Reality, Suspension Lattices, and Mission-State Integrity* (Version 2.0 foundational whitepaper). Prompted LLC. https://promptedllc.com/forked. Raw source: https://promptedllc.com/papers/forked-v2/PAPER.md.

[7] Taylor, B. E. (2026). *Fractal Quivers of Quivers - A Mathematical Substrate for Ubiquitous Agent Governance*. Prompted LLC. https://promptedllc.com/fractal-quivers-of-quivers.

[8] Taylor, B. E. (2026). *Computing Around the Open Center: Splat Mechanics, Fractal Quivers of Quivers, and a Governance-Native Compute Paradigm* (Version 1.0 preprint). Prompted LLC. https://promptedllc.com/computing-around-the-open-center.

[9] Prompted LLC. “Sovereign Continuity - The Root Frame for Sovereign Adaptive Systems.” https://promptedllc.com/sovereign-continuity. Accessed September 4, 2026.

[10] Prompted LLC. “Human Judgment in AI Systems - Reusable Structure, Not a Bottleneck.” https://promptedllc.com/human-judgment-in-ai-systems. Accessed September 4, 2026.

[11] Prompted LLC. “Context Grapple Gun - Portable Governance Lifecycle for Claude Code.” https://promptedllc.com/context-grapple-gun. Accessed September 4, 2026.

[12] Prompted LLC. “Look-First - Reattach to Canonical State Before Acting.” https://promptedllc.com/look-first. Accessed September 4, 2026.

[13] Prompted LLC. “Successor Topology - Ubiquity Canonical V2 Lineage Lock.” https://promptedllc.com/successor-topology. Accessed September 4, 2026.

[14] Prompted LLC. *Ubiquity - The Canonical Federation*, repository root README.md and CLAUDE.md, canonical_federation snapshot supplied September 2026. Archive SHA-256 b4a27fd60af69d226f1ea9efe2c473d8b35656691aa7cb140797bb19528102b3.

[15] Ubiquity Canonical Federation. *THE BINDER - A History of the Winching Machinery, with Receipts*, Ubiquity tic 770. audit-logs/governance/whitepaper-binder-tic770/THE-BINDER.md.

[16] Ubiquity Canonical Federation. *Lane A Extracts - Campaign, Harpoon Office, Winch, Publications*, Ubiquity tic 770.

[17] Ubiquity Canonical Federation. *Lane B Extracts - Hoist, DAG/Waves, GUNSLINGER/HIDALGO, Winching, Anchoring, Dissonance Basins*, Ubiquity tic 770.

[18] Ubiquity Canonical Federation. *The Secret and the Lever*, Ubiquity tic 748.

[19] Ubiquity Canonical Federation. *The Lap Ran*, Ubiquity tic 749.

[20] Skyward Prompted LLC. *Skyward Bridge / Forge Canon Anchor*, Forge repository root README.md and CLAUDE.md, forge-canonical-fed snapshot supplied September 2026. Archive SHA-256 8b0a3ed25a3a1da6c20266ce5d680440ad59e12300ff0f6ae027cb991fb14c19.

[21] Forge repository. *Sabeel CEO-Duo / i65 Context Baseline* and *AI Operator Capacity and Commercial Readiness Brief*, including the equal-human-cofounder statement and Sabeel's Prompted Forge cofounder role.

[22] Forge repository. docs/canon/federation/boot/forge_field_context_and_dag_contract_2026-09-03_wave38_founder_currentness_meta_repair.md.

[23] Forge repository. docs/canon/federation/dags/forge_massive_dent_ready_lattice_2026-09-03_wave38_founder_currentness_meta_repair.{md,json}.

[24] Forge repository. docs/canon/federation/boot/FIELD_CONTEXT_CURRENT.md and docs/canon/federation/dags/FORGE_DENT_CURRENT.md.

[25] Forge repository. Wave 38 E1 receipt set: W38-E1-N1-source-checkpoint-postimage.json; W38-E1-N2-source-remote-readback.json; W38-E1-N3-source-clean-clone.json; W38-E1-J1-source-current.json; W38-E1-N4-receipt-carrier.json; W38-E1-N5-carrier-remote-readback.json; W38-E1-N6-carrier-clean-clone.json; W38-E1-J2-carrier-current.json.

[26] Forge repository. Wave 38 E2 receipt set, including W38-E2-J2-carrier-current.json and its required child receipts.

[27] Forge repository. scripts/guarded-file-promote.mjs; scripts/orchestrator-integration-lease.mjs; docs/canon/baseline_inheritance_tranche_state.md; and associated leases and promotion receipts.

[28] Forge repository. *Forge Delivery Velocity Evidence - 2026-08-14*. docs/canon/audits/2026-08-14_forge_delivery_velocity_evidence.md.

[29] GitHub Insights screenshots supplied by the author for the Forge repository: Code Frequency; August 1-September 1; August 25-September 1; August 29-September 1; and August 31-September 1, 2026.

[30] Forge repository. *Forge Federation Load-Bearing Mechanics, Value, Functionality, and Maturity Matrix*, current September 3, 2026 source horizon.

[31] Chacon, S., and Straub, B. *Pro Git*: Git Internals - Git Objects; Remote Branches. https://git-scm.com/book/en/v2/.

[32] in-toto Project. Attestation Framework and Link Attestation Predicate. https://in-toto.io/.

[33] SLSA. Provenance specification. https://slsa.dev/spec/.

Acknowledgment

AI systems assisted with source inspection, terminology crosswalking, diagrams, synthesis, and document production under Breyden E. Taylor's author-defined frame and acceptance criteria. The assistance conferred no authorship, ownership, endorsement, admission authority, corporate or client standing, or authority to collapse Ubiquity evidence into Forge claims.