SYNTAX: A Human-AI Collaboration Protocol for Durable Agent Work

Methods Note v1.0

Author: Jeff Hopp, QNTx Labs

Version: 1.0.0

Published: 2026-08-12

Protocol implementation observed: SYNTAX AI v6.8.0

Canonical record: https://www.qntxlabs.com/research/syntax-methods-note-v1.0

Abstract

SYNTAX is a human-AI collaboration protocol for carrying context across work sessions and converting AI-assisted activity into durable, inspectable assets. This methods note describes a six-stage operational loop: session log, memory, consolidation, mechanical enforcement, cross-model adversarial review, and standardized close. It separates common engineering practices from the SYNTAX-specific mechanisms that connect them. The note is a design and reproducibility description, not a controlled productivity study. Its central failure model is the Yield gap: a session can produce a plausible answer while leaving no reusable artifact, verified decision, or system change for the next session.

What is SYNTAX?

SYNTAX is a protocol for a person and one or more AI systems to structure work, preserve context, challenge claims, and carry verified outcomes forward. It is not a wire protocol, model-to-model transport standard, or autonomous-agent message format. Software-agent protocols may coordinate machines underneath; SYNTAX coordinates the human intent, evidence, decisions, and artifacts around that execution.

Durable agent work means that a later collaborator can determine:

1. what was attempted and why;
1. which evidence supports the result;
1. what changed in the working system;
1. what remains open or uncertain;
1. which checks were run; and
1. where the next session should resume.

What problem is the protocol designed to solve?

Chat-based work naturally optimizes for a convincing local response. Durable work requires a different objective: a verified change to a shared system of record. Without an operating protocol, context is repeatedly reconstructed, lessons accumulate as unstructured notes, and a session may be called complete without leaving reproducible evidence.

SYNTAX calls that failure the **Yield gap**: useful-looking activity that does not become a decision, artifact, test, receipt, or enforceable improvement. A good conversation is not sufficient evidence that the working system improved.

What is the operational loop?

1. Session log

Capture what actually happened: objective, scope, material actions, decisions, evidence, corrections, tests, costs, and unresolved risks. The log is a compact handoff record, not a transcript dump. It gives later sessions a verifiable starting point.

2. Memory

Route durable knowledge to the smallest appropriate record. Stable preferences, validated patterns, project facts, and reference pointers can become memory. An unfixed defect is not memory; it belongs in a defect register with status and ownership. This distinction prevents a knowledge store from becoming a backlog that no one is required to close.

3. Consolidation

Periodically review accumulated session evidence across projects. Merge duplicates, identify contradictions, retire stale guidance, and nominate patterns that have survived more than one context. Consolidation proposes promotion; a human reviews the change before it becomes general policy.

4. Mechanical enforcement

When a lesson applies at a deterministic checkpoint, encode it as a validator, runner, hook, test, or template constraint. Prose remains useful for judgment, but a repeatable failure at commit, publication, deployment, or close should not

depend on memory alone. Enforcement should fail with enough evidence to correct the problem and should record known agent/runtime parity gaps.

5. Cross-model adversarial review

For risky changes or consequential claims, ask an independent model context to attack the result: find unsupported assertions, missing cases, unsafe side effects, and mismatches between the stated objective and the produced artifact. The second pass is not a vote and does not transfer accountability to a model. Its value is independence from the assumptions accumulated in the primary working context.

6. Standardized close

Close only after current state, forward plan, session log, project ledger, claim evidence, test scope, and external work tracking have been reconciled. A close runner and validator can make the proof packet fail-closed: if a required record or declaration is missing, the session is not represented as complete. The close becomes the input to the next start, completing the loop.

Which parts are established engineering practice?

SYNTAX composes several established practices rather than claiming to invent them:

- version control and code review;
- automated tests and continuous integration;
- structured logs, issue registers, and runbooks;
- retrospectives and knowledge management;
- independent review for high-risk changes; and
- explicit handoffs between operators.

These practices are useful without SYNTAX and should retain their conventional meanings.

Which parts are SYNTAX-specific?

The distinctive contribution is the operating contract that joins those practices for AI-collaborative work:

- a two-sided engine/corpus model that separates reusable scaffolding from a user's private operating context;
- files structured primarily for the next AI collaborator while remaining human-reviewable;
- point-of-action routing that distinguishes memory, defects, plans, and mechanical gates;
- a promotion ladder from single-session observation to cross-context pattern to enforced rule;
- an agent-parity registry that states where enforcement differs by runtime; and
- a standardized, claim-checked close that updates the durable state from which the next session starts.

SYNTAX is therefore a coordination protocol at the work-system layer. It does not replace Git, CI, a project tracker, a model provider, or an agent transport.

How can the method be reproduced?

Minimum prerequisites

1. A version-controlled working repository.
1. A durable corpus containing current state, a forward plan, session logs, project ledgers, and routed knowledge.
1. An AI-capable work environment that can read and write those records.
1. A test or verification surface appropriate to the work.
1. A human owner who approves consequential decisions and public actions.

Reference procedure

1. At start, load the current state, active plan, relevant operating rules, and known gaps.
2. Define the session objective and the artifact or decision that will count as Yield.
3. Work against the shared files and record evidence as actions occur.
4. Route defects, lessons, and project facts to their correct registers.
5. Run tests and, where risk warrants it, an independent adversarial review.
6. Reconcile state, forward plan, ledger, evidence, and external work tracking.
7. Run a close validator. Do not represent the session as closed if required evidence is absent.
8. Begin the next session from the updated state rather than reconstructing the prior conversation.

Expected artifacts

Artifact	Required information	Primary consumer
Current state	Verified live reality, known gaps, latest material change	Next agent and human owner
Forward plan	Ranked unfinished work and re-entry conditions	Next work session
Session log	Objective, actions, evidence, corrections, tests, risks	Audit and consolidation
Project ledger	Short chronological narrative and links to proof	Human review
Defect register	Reproduction context, status, ownership, scheduling link	Engineering triage
Memory index	Durable facts and validated patterns, not open defects	Context loading
Close receipt	Claim checks, test scope, CI status, tracking reconciliation	Completion gate

How should the method be evaluated?

Evaluation should measure closure and reuse, not the volume of captured notes. Useful operational measures include:

1. percentage of sessions with a defined and delivered Yield;
1. percentage of material claims linked to reproducible evidence;
1. time required for a later session to resume useful work;

- defects reopened because a prior lesson was not enforced;
- debt closed versus debt opened per period;
- contradictions or duplicates removed during consolidation; and
- enforcement coverage across model runtimes.

A comparison can hold the task and tools constant while varying whether the protocol is used. The evaluation should predefine what counts as a completed artifact and should preserve failures, reversals, and reviewer disagreement.

What are the limitations?

- This note documents an actively used operational design, not a randomized or controlled productivity experiment.
- Evidence from the author's repositories may not generalize to other people, teams, model providers, regulatory environments, or task types.
- File-based continuity can preserve incorrect assumptions as effectively as correct ones; verification and contradiction review remain necessary.
- Mechanical enforcement can encode a bad policy. Gates require change history, escape conditions, and human review.
- Cross-model review reduces shared-context bias but does not guarantee model independence, factual accuracy, or safety.
- More process can cost more than it saves on trivial work. The protocol should scale with consequence, ambiguity, and handoff risk.
- A standardized close proves that required records and checks exist; it does not prove that every judgment was correct.

What is the current implementation status?

The implementation described here was observed in SYNTAX AI v6.8.0 on 2026-08-12. It includes an engine/corpus architecture, agent-neutral session start and close runners, a close validator, routed memory and defect handling, consolidation guidance, an agent-parity registry, and an independent review bridge. Specific internal corpus contents, credentials, and client data are not part of this public note.

The protocol and implementation will continue to evolve. Versioned releases of this note should be cited rather than an undated web summary.

Suggested citation

Hopp, J. (2026). *SYNTAX: A Human-AI Collaboration Protocol for Durable Agent Work* (Methods Note v1.0). QNTx Labs. Version 1.0.0. https://www.qntxlabs.com/research/syntax-methods-note-v1.0

Artifact integrity

The Markdown source, protocol-loop SVG, PDF rendering, citation metadata, and release archive are versioned together. The PDF is generated from this Markdown source so its substantive claims remain aligned with the canonical text.