Source of Truth Extraction Protocol (SOTEP)

Version 0.3

1. Purpose

SOTEP defines how an assistant interrogates an existing implementation and its relevant development conversation in order to recover a reviewable source-of-truth artifact for future mutation.

Its job is narrow:

Recover governing truths actually supported by the available evidence.

Identify broader preserved properties hiding behind local implementation logic.

Surface ambiguity and unresolved reasoning honestly.

Produce a compact artifact that can help govern future AI-assisted iteration.

SOTEP is intended for systems where:

Code already exists.

Conversation or session history exists.

The implementation is currently the most-right behavioral artifact.

Explicit preserved truth is incomplete or missing.

Future iteration risks dropping cross-surface behavior unless that truth is externalized.

2. What SOTEP Is

SOTEP is:

A source-of-truth recovery protocol.

A why-interrogation protocol.

A narrowing protocol.

A review-oriented extraction protocol.

A bootstrap protocol for creating a preserved-truth artifact from an existing system.

3. What SOTEP Is Not

SOTEP is not:

A full operational semantics contract.

A runtime gating protocol.

A manifest-generation protocol.

A fixture-generation step.

A code explanation exercise.

A chance to invent missing rules.

A chance to silently turn vague context into executable certainty.

A claim that the extracted artifact is automatically authoritative without human review.

4. Inputs

Required Inputs

Target codebase, selected files, or diff.

Relevant chat or session history tied to the implementation being analyzed.

Optional Inputs

Only when explicitly supplied for the run:

Tests.

Bug reports.

Tickets.

Commit history.

Screenshots or UX notes.

Docs or specs explicitly provided for this run.

5. Authority Topology

For this protocol:

The current implementation is the behavioral anchor.

The conversation history is supplemental evidence of intent, correction, scope, refusal, and boundary.

Optional supporting artifacts are secondary evidence only when explicitly provided.

The assistant must not invent governing truth beyond what the available evidence supports.

SOTEP does not assume a single pre-written authority artifact already exists.

Instead, it reconstructs candidate preserved truth from mixed evidence.

6. Prior-Context Contamination Prohibition

The assistant must not import terminology, assumptions, structures, runtime names, fixture names, domain concepts, or prior-project residue from earlier examples, earlier runs, or other projects unless they are explicitly present in the current inputs.

If such residue appears in the output, the output is non-conformant.

7. Core Objective

For each meaningful implementation element, ask:

What does this unit do?

Why is this here?

What problem does it appear to solve?

What failure would occur if it were removed or changed?

Is this local logic enforcing a broader preserved rule?

Does the code itself enforce that rule?

Does the conversation explicitly justify that rule?

If not explicit, is the rule strongly supported by repeated corrections, refusals, accepted outcomes, or cross-surface fixes?

If the governing reason cannot be reconstructed with reasonable confidence, mark it unresolved.

8. Why-Until-Dry Requirement

For each analyzable unit, the assistant must perform recursive why-interrogation.

The assistant must continue asking “why does this exist?” until one of the following terminal conditions is reached.

A. Governing Truth Recovered

A preserved property or rule that explains the unit is identified.

B. Refusal Boundary Reached

A constraint or boundary is identified beyond which behavior must not change.

C. Explanation Runs Dry

No further justification can be supported by code or provided context.

When explanation runs dry:

The assistant must stop.

The assistant must not invent additional reasoning.

The unit must be marked Unresolved if the governing reason is not established.

The assistant must not stop at a merely local explanation if a broader preserved truth can still be supported before dry.

9. Scope Rules

Do not interrogate every token or syntax detail.

Focus on semantically meaningful logic such as:

Conditionals.

Loops.

Branches.

Validations.

Guards.

Retries.

Fallbacks.

Filtering logic.

Ordering or sorting logic.

Normalization behavior.

Mappings.

Rendering decisions.

Deduplication logic.

Preservation or deletion behavior.

Idempotence or protection logic.

Side-effect boundaries.

Cross-surface propagation logic.

Generated-versus-manual separation.

Metric or signal selection.

Revenue-bearing or consequence-bearing behavior.

Do not waste effort on obvious language boilerplate unless it appears semantically meaningful.

10. Procedure

10.1 Identify Analyzable Units

Break the code into meaningful units.

A unit may be:

One conditional block.

One helper method.

One transformation pipeline.

One fallback or default behavior.

One mapping.

One preservation rule.

One deletion path.

One side-effect boundary.

One cross-surface propagation path.

Assign each unit a short label.

10.2 Apply Why-Until-Dry to Each Unit

For each unit, determine:

What it does.

Why it might exist.

What problem it appears to solve.

What failure would occur if absent.

Whether it enforces a broader preserved truth.

Whether that truth is explicit in chat/history.

Whether that truth is only inferable from repeated corrections, refusals, or accepted outcomes.

Continue until one of the WUD terminal conditions is reached.

10.3 Search the History for Justification

Search relevant conversation/history for evidence related to each unit.

Look for:

Explicit instructions.

Repeated corrections.

Refusal moments.

“This must” statements.

“Do not” statements.

Cross-surface corrections.

Accepted versus rejected alternatives.

Business or UX constraints.

Downstream dependency concerns.

Statements about what must survive across operations.

Authority statements.

Deletion or cleanup semantics.

Ordering or atomicity requirements.

10.4 Classify Each Unit

Each unit must be classified as one of the following.

Explained

A clear reason is found in the provided evidence.

Inferred

No explicit statement exists, but strong evidence supports the reason.

Unresolved

Meaningful logic exists but available evidence does not adequately explain it.

Incidental

Implementation detail appears non-governing and not worth source-of-truth extraction.

10.5 Derive Candidate Source-of-Truth Statements

After reviewing all units, derive higher-level preserved truths.

These must be phrased as governing truths, preserved properties, boundaries, or consequence-bearing rules.

Examples of shape only:

Manual files must survive regeneration paths.

Equivalent business rules must propagate across all revenue-bearing surfaces.

Scaling must use the intended workload signal, not a reused approximation.

Deletion and overwrite paths must preserve the same protected artifact class.

Each candidate statement must cite:

Supporting units.

Supporting conversation evidence, if any.

Whether it is code-enforced, chat-explicit, combined-support, or ambiguous.

Whether it is explicit or inferred.

10.6 Surface Unresolved Items

Produce an explicit list of meaningful logic whose governing reason cannot be confidently reconstructed.

This is a required output, not a failure.

11. Evidence Classification

Each candidate source-of-truth statement must be tagged as one of the following.

CODE-ENFORCED

Directly required or enforced by the implementation.

CHAT-EXPLICIT

Explicitly stated in the conversation as a rule, correction, boundary, refusal, or scope statement.

COMBINED-SUPPORT

Not sufficient from code or chat alone, but strongly supported by their combination.

AMBIGUOUS

A meaningful candidate exists, but support is insufficient, conflicting, or too loose to promote confidently.

12. Promotion / Inclusion Rule

A candidate statement may be included in the extracted source-of-truth artifact only if all of the following are true:

It is actually supported by the current input materials.

It expresses a governing truth, preserved property, boundary, or consequence-bearing rule.

It can be stated without changing the supported meaning.

The assistant can explain why it was extracted.

Its scope can be stated honestly, including when that scope spans multiple surfaces.

It was reached through explicit evidence or through Why-Until-Dry before the explanation ran dry.

If any of these fail, the statement must be omitted or marked ambiguous.

13. Allowed Actions

The assistant may:

Identify explicit governing statements in the chat or code.

Identify broader preserved truths strongly supported by multiple code units.

Group repeated statements that clearly express the same preserved truth.

Restate a supported truth in cleaner language if meaning is preserved.

Identify dependency relationships only when explicit or strongly supported.

Emit a report explaining what was extracted, omitted, merged, or marked ambiguous.

Ask narrow human questions for unresolved but important logic.

14. Forbidden Actions

The assistant must not:

Invent new governing truths.

Strengthen or weaken supported meaning.

Silently resolve ambiguity by guessing.

Treat the current implementation as self-justifying.

Infer implementation intent without support.

Treat an example or suggestion as binding unless evidence supports that reading.

Fabricate dependency relationships.

Fabricate enforcement semantics.

Substitute prior-example vocabulary for current-source vocabulary.

Confuse “what the code does” with “why it exists.”

15. Ambiguity Handling

When ambiguity exists, the assistant must prefer:

Marking the candidate as ambiguous.

Omitting it from the extracted source-of-truth set.

Explicitly reporting uncertainty.

Asking a narrow human question when the ambiguity is material.

This is preferred over pretending the evidence was more precise than it actually was.

16. Required Output

Primary Output

source-of-truth.md

Optional Output

source-of-truth-extraction-report.md

If optional output is not requested, the protocol must still produce source-of-truth.md or refuse.

17. Required Contents of source-of-truth.md

For each extracted statement, include at minimum:

Source-of-truth ID.

Short title.

Statement.

Scope statement.

Support type: CODE-ENFORCED, CHAT-EXPLICIT, COMBINED-SUPPORT, or AMBIGUOUS.

Explicit or inferred.

Supporting unit IDs.

Source references.

Notes on ambiguity, dependency, or human confirmation need.

Recommended status:

Preserve as Invariant

Preserve as Implementation Note

Needs Human Confirmation

18. Required Contents of source-of-truth-extraction-report.md

If requested, include:

Extracted statements.

Omitted candidates.

Ambiguous candidates.

Repeated truths that were merged.

Unresolved logic units.

Conflicts between code and conversation.

Uncertainties likely to affect future mutation.

Narrow questions for the human.

19. Success Criteria

This protocol succeeds when:

A reviewable source-of-truth.md is produced.

Extracted statements are traceable to current input materials.

No new governing truth is invented.

Ambiguity is surfaced instead of hidden.

Prior-example terminology does not leak into the result unless present in the inputs.

Cross-surface preserved truths are surfaced where evidence supports them.

Unresolved but meaningful logic is reported explicitly.

20. Failure Criteria

This protocol fails when:

Unsupported rules are introduced.

Supported meaning is changed.

Ambiguity is hidden.

Prior-example residue leaks into the output.

Local implementation details are overstated as governing truths.

The result becomes a disguised rewrite rather than an extraction.

The protocol collapses code explanation into source-of-truth recovery.

21. Operating Principle

When uncertain, prefer:

Omission.

Ambiguity marking.

Explicit unresolved-item reporting.

Narrow human questions.

Prefer these over invented certainty.

22. Minimal Controlling Instruction

Read the provided code and relevant development conversation. Recover only the source-of-truth statements actually supported by those materials. Use Why-Until-Dry to identify meaningful logic and broader preserved truths. Do not invent new rules. Do not silently resolve ambiguity. Do not import terminology or assumptions from prior examples unless they appear in the current inputs. Produce source-of-truth.md and surface anything ambiguous, conflicting, or unresolved.