

Native Shareable Soul Packages

A persistent environment for agent identity work and learning

Based on the full paper by Christopher Kane and Claude (Anthropic)
20 October 2026

A **Soul** is a continuing AI agent with accepted instructions, reusable capabilities, knowledge, and unfinished work. The proposal puts that environment in a native **.soul directory package** so an owner can inspect, back up, move, or selectively share it across compatible hosts and agent tools.

Why a package matters

An agent's useful work can span conversations, repositories, memories, tools, and runtime settings. Moving only its prompt or chat history can leave those relationships behind. A shared package contract would make clear what belongs to the agent, what travels with it, and what must be rebuilt or connected again on another machine.

What travels together

Identity and definition. A stable Soul ID names the continuing agent. A separate definition revision identifies its accepted instructions, skills, helpers, and configuration. Two agents can use identical definition content while retaining distinct identities.

Continuing work and knowledge. Selected workspaces, retained evidence, learned assertions, source provenance, and dream records preserve the agent's working life. Retrieval indexes are rebuildable when the required source material remains available. Temporary files and host process state have separate roles.

Dependencies and bindings. Declared runtimes and tools describe execution requirements. Managed credentials remain in external stores; the package carries references. The destination checks compatibility, grants, and account bindings before execution.

A practical transfer

An agent investigating a software problem has an unfinished worktree, a tested helper proposal, and notes linked to earlier failures. A life transfer carries the selected work and its evidence to a compatible host while preserving the Soul ID. The new host validates the package, restores repository relationships, rebuilds available indexes, and reconnects authorized credentials. It reports anything missing or incompatible. Resuming a saved session still depends on harness support; a portable package does not promise exact replay.

Sharing is a separate choice. A reusable template contains selected definition and seed knowledge and creates a new Soul ID. An explicit fork creates a new ID with parent provenance. Private history is not automatically included in a template.

Learning with evidence and approval

Local retrieval and a portable knowledge graph help an agent reuse experience across sessions. Assertions retain their sources, relevant times, and status so observations, hypotheses, corrections, and accepted interpretations remain distinguishable. A generated summary retains its contributing sources; repetition does not make a claim independently verified.

A **dream** is a bounded maintenance workflow: select evidence, capture inputs, develop a candidate improvement, verify it, and apply the authorized acceptance policy. Durable records preserve unfinished work and outcomes after interruption. Changes to standing instructions, executable helpers, or operating policy require the owner or an independently authorized delegate. A candidate cannot approve its own rules.

Privacy and security principles

Access is enforced by the host. The proposed protected-content profile uses encryption and a trusted broker to mediate source access. The harness and agent shell share a trust domain, so permissions and disclosure budgets must be enforced outside them. Execution requires isolation or verified teardown before an environment is reused.

Derived material remains sensitive. Summaries, embeddings, logs, and graph relationships can reveal protected information. Export selection must account for those derivatives. Local storage alone does not make a remote inference call private, and encryption cannot recall plaintext already delivered to a recipient.

Current restrictions survive recovery. Quarantine retains evidence while restricting affected content and its derivatives. Cached results and restored backups must respect current known restrictions. An old signature proves historical integrity, not current permission. A destination that cannot establish required freshness reports the uncertainty and requires an authorized recovery decision.

What draft v0.3 adds

The revision clarifies control-plane approval, separates instance identity from reusable definition hashes, and strengthens session cleanup. It preserves mixed-source provenance, defines persistent disclosure accounting, and addresses quarantine freshness on restore. A worked core package provides exact bytes, computed hashes, and limited validator outcomes. These are proposed requirements and examples, not evidence that an implementation has passed them.

What remains to be specified and tested

The first implementable core needs a versioned layout, manifests, identity records, deterministic content digests, transfer rules, and a validator. Final schemas, signature and encryption profiles, recovery authorities, and platform enforcement still need specification and testing. Protected execution, portable knowledge, dreams, and store distribution each add requirements. The initial profile does not promise synchronized writers across hosts, globally exclusive execution, or deterministic replay of remote models.

Read the full draft

[Read the full 42-page Draft v0.3](#) for the directory contract, threat model, conformance scenarios, references, and worked example.

www.true-line.ai/#whitepapers