

DeepSeek Harness: Plugin Models and Agent-Resident Substrates

DeepSeek Harness, as publicly described, treats models, tools, session state, sandboxes, and the agent loop itself as plugins over a kernel. That is a clean and defensible way to build a modular agent runtime, and the industry is converging on it. U.S. Provisional Application No. 64/070,239 arranges the same parts differently. In the disclosed embodiments the semantic agent is the persistent execution substrate of the device, and inference endpoints are managed components registered in a governed tool registry beneath it. Identity, cognitive state, and lineage are held by the agent and, in an embodiment, preserved across installation, retraining, replacement, archival, and removal of any subordinate component. The question the filed architecture answers explicitly is which layer holds authority.

When the Model Is Just Another Component

Agent frameworks have spent two years learning to stop hard-coding the model. The lesson took, and the current generation of runtimes treats the inference provider as a swappable dependency behind an interface. That is good engineering. It also leaves a question the interface itself does not settle: after the swap, what is still the same thing?

For a runtime that only routes requests, the answer matters little, because nothing was accumulating. The moment a runtime starts holding things worth keeping, the question becomes structural. If a user's accumulated behavioral disposition, outcome history, and governance rules live inside the model component, replacing that component discards them. If they live in a session record, they are as durable as the session. An architecture designed to enable component churn has to place those things somewhere the churn does not reach. The filing discussed below places them in the agent and states the conditions under which they are preserved.

What DeepSeek Harness Describes

DeepSeek Harness v0.1 was released as an open-source developer preview on August 13, 2026, under the MIT license. Its stated central claim, as publicly described, is direct: "Everything is a plugin." Public materials describe models, tools, session state, filesystems, sandboxes, the agent loop, orchestration, and the web experience all as plugins. Applying the idea to the loop and to session state, not only to the model, is the part worth noting.

It is built on Cordis, a plugin kernel, and the harness uses Cordis v4, as publicly described. Putting a general kernel underneath an agent runtime is a real architectural commitment.

Observability is the part most worth studying. Public materials describe an append-only session log recording system prompts, reasoning, tool calls and results, subagent scheduling, and context injections. Public materials further state that resume, fork, search, and replay "operate on the same event stream." That last property is easy to underestimate. Where a debugging view and an execution view are kept separately, they can drift apart; deriving all four operations from one record keeps them consistent by construction. A Trajectory view is described for inspecting records by source, which fits the same design: if many components write to one stream, a useful question when reading it back is which component produced a given record.

Everything above is what the public materials say. The comparison that follows is drawn against the filed architecture, not against anything unstated about the product.

The Substrate Described in the Filing

Provisional Application No. 64/070,239 describes an agent-resident execution substrate with a governed inference tool registry and lineage-derived personal corpus model training. The difference from a peer arrangement of components is stated at the level of layering. In the disclosed embodiments, the semantic agent (100) is instantiated on the substrate device as a persistent computational entity and maintained across the lifetime of the device under a continuity guarantee enforced by the substrate runtime. Inference endpoints, knowledge ingestion modules, and related computational assets are maintained as governed managed components subordinate to the agent.

The agent carries four persistent fields. A persistent identity field (102) initialized at first instantiation, comprising one or more identifier values, an initialization timestamp, and continuity metadata sufficient to verify continuity of the agent across substrate restart events. A cognitive state field (104) of persistent cognitive domain fields, each tracking a distinct property of the agent's behavioral disposition, normative alignment, execution readiness, and capability awareness. A lineage field (106), an append-only sequence of records covering dispatched inference requests, their outcomes, integrity-signal feedback, lifecycle operations applied to endpoints, ingestion events, governance policy updates, counterparty encounters, scope mutations, resource governance events, and substrate-runtime updates. And a governance policy field (108) of cryptographically signed, versioned, machine-evaluable policy objects.

Beneath the agent sits the managed inference tool registry (110). Each managed inference endpoint (112a, 112b, 112c) comprises a model artifact, an interface specification, and an associated governance scope, and the governance scope specifies the policy objects governing that endpoint's installation, retraining, replacement, archival, and removal. Endpoints of distinct types and distinct sizes may be co-resident,

subject to local memory and storage constraints. A per-endpoint capability declaration records, without limitation, modality, task category, input and output formats, a latency profile, a model version identifier, a publisher identifier, a resource envelope descriptor, a governance compatibility tag set, and a capability confidence value. It is signed by the endpoint's publisher and verified at installation and at each lifecycle operation affecting the endpoint.

The agent-to-tool dispatcher (120) routes each inference request on a decision conditioned on the current value of the cognitive state field, the input characteristics of the request, the policy objects applicable to the request, and the per-endpoint capability declaration. In an embodiment, the cognitive state field is consulted to determine an applicable confidence threshold, an applicable normative threshold, and an applicable capability threshold, and candidate endpoints whose declared confidence, normative compliance, or capability fall below the applicable thresholds are excluded from the routing decision. Among the remaining candidates, the selection function is configured to prefer endpoints with higher historical outcome quality recorded in the lineage field for requests of similar input characteristics. The routing decision is therefore described as a policy evaluation and a history lookup, not only an interface match.

The tool lifecycle controller (130) drives endpoints through a state machine running from an uninstalled state (200) through installed-inactive (210), active (220), retraining (230), updated-active (240), archived (250), and removed (260), with each transition governed by policy objects in the governance policy field and recorded in the lineage field. Retraining and substitution are described as performed in a staging area distinct from the active tool registry, with promotion of the updated model artifact only on successful completion and successful policy validation; on validation failure the substitution is rolled back and the failure is recorded in the lineage field together with the cause of failure.

Continuity is where the layering does its work. In an embodiment, the agent's identity, cognitive state, and lineage fields are preserved across lifecycle operations performed on managed inference endpoints, across addition or removal of an endpoint, across governance policy mutation, and across substrate-runtime updates, and are not modified by subordinate components except by appending to the lineage field under the continuity proof. Enforcement is described through structural separation, cryptographic continuity proofs over the lineage field, or a combination: the agent's persistent state is held in storage domains accessible to the agent runtime, while lifecycle operations write to the tool registry, to staging areas, and to the lineage field. Substrate-runtime updates are staged, policy-evaluated, and linked across versions by a continuity attestation comprising a cryptographic chaining of the agent's identity over the sequence of runtime versions. In a further embodiment, authority over one or more agent fields may instead be delegated under policy to designated subordinate components or to external authorities.

Same Category, Different Locus of Authority

Both architectures accept that components should be replaceable and that execution should leave a durable record. The convergence is real, and the append-only record is the clearest instance of it: one a session log described in public materials, the other a lineage field described in the filing.

Where they differ is in what the filed architecture makes explicit about authority. A plugin kernel, as publicly described, is a peer arrangement in which the kernel mediates and components including the model, the loop, and session state are uniform participants. The filed architecture is described as a hierarchy. In an embodiment, the semantic agent is the sole authority within the substrate over operations affecting its persistent identity, cognitive state, lineage, and governance policy fields, with subordinate components permitted to modify state within their own domains but not the agent's persistent fields except by appending to the lineage field under the continuity proof.

That difference propagates into what a record is used for. The disclosed lineage field is not only an execution record but a training signal: the corpus assembly module (410) derives a training corpus from lineage records admissible under the corpus policy object, filtering for modality compatibility and applying declared redaction rules; the fine-tuning module (420) applies a parameter-efficient operation configured to be feasible within the local compute envelope and completable within a policy-declared training window; and the governed substitution module (430) promotes the updated artifact into the registry, with the agent's identity, cognitive state, and lineage preserved across the substitution under the continuity guarantee. The disclosure distinguishes this signal from training on model self-output, because downstream-outcome references encode user acceptance, user revision, downstream execution success or failure, and integrity-signal feedback, which is information the model under training did not generate.

Uniform pluggability answers how components get composed and replaced. The substrate architecture answers what is held fixed while that happens, and under whose policy a replacement is admitted.

Coexistence and Boundaries

The filed architecture does not require any particular runtime above it. What it describes is a place to put the durable layer: inference endpoints are registered with signed capability declarations and governance scopes, and every lifecycle operation on them is recorded in the lineage field as a deterministic event, such that the sequence of endpoint registrations across the device's lifetime is reproducible from the lineage. Parallel operating contexts are handled by named scope records in the cognitive state field, each comprising a scope identifier, a scope-local corpus policy object, a scope-local tool subset reference, a scope-local lineage partition designation, and one or more scope-local cognitive state values, all maintained under a single persistent agent identity.

The disclosed architecture has boundaries worth stating plainly. The filing does not address developer ergonomics, packaging, or the extension API surface that makes a kernel pleasant to build against. Retraining is bounded by the local compute envelope of the substrate device and by a policy-declared training window, so model improvement under the disclosure is paced by the device. Outcomes are conditioned on declared bounds: policy validation of an updated artifact can fail, and the disclosure handles that by rollback and a recorded failure cause rather than by asserting the failure cannot occur. In an embodiment, a continuity break is detectable as a discrepancy between a computed continuity hash and a stored prior reference value, and triggers escalation under the governance policy field. Detection and escalation are the described behavior.

Disclosure Scope

This article is a technical description of subject matter disclosed in U.S. Provisional Application No. 64/070,239, "Agent-Resident Execution Substrate with Governed Inference Tool Registry and Lineage-Derived Personal Corpus Model Training." Reference numerals and mechanism names are those used in the filing. Embodiments are described as disclosed; nothing here characterizes the scope of any claim, and the application is pending.

References to DeepSeek Harness are to public materials and are used for comparison only; no relationship, endorsement, or infringement is asserted.

Agent-Resident Execution **Substrate** (</agent-resident-execution-substrate>)

[All 49 steps → \(/inventive-steps\)](#)

Persistent execution environment carried by the agent, not the host — identity, state, and lineage across power cycles, devices, and upgrades.

Provisional application

PRIMARY TECHNICAL DISCLOSURE

- [Agent-Resident Execution Substrate, Articles \(/articles/agent-resident-execution-substrate\)](/articles/agent-resident-execution-substrate)

SECONDARY TECHNICAL

- [Persistent Semantic Agent \(/articles/agent-resident-execution-substrate/persistent-semantic-agent\)](/articles/agent-resident-execution-substrate/persistent-semantic-agent)
- [Managed Inference Tool Registry \(/articles/agent-resident-execution-substrate/managed-inference-tool-registry\)](/articles/agent-resident-execution-substrate/managed-inference-tool-registry)
- [Agent-to-Tool Dispatcher \(/articles/agent-resident-execution-substrate/agent-to-tool-dispatcher\)](/articles/agent-resident-execution-substrate/agent-to-tool-dispatcher)
- [Lineage-Derived Training Signal \(/articles/agent-resident-execution-substrate/lineage-derived-training-signal\)](/articles/agent-resident-execution-substrate/lineage-derived-training-signal)
- [Identity Preservation Across Upgrades \(/articles/agent-resident-execution-substrate/identity-preservation-across-upgrades\)](/articles/agent-resident-execution-substrate/identity-preservation-across-upgrades)
- [Cognitive State-Conditioned Dispatch \(/articles/agent-resident-execution-substrate/cognitive-state-conditioned-dispatch\)](/articles/agent-resident-execution-substrate/cognitive-state-conditioned-dispatch)
- [Governed Tool Lifecycle \(/articles/agent-resident-execution-substrate/governed-tool-lifecycle\)](/articles/agent-resident-execution-substrate/governed-tool-lifecycle)
- [Continuity-Proof Lineage \(/articles/agent-resident-execution-substrate/continuity-proof-lineage\)](/articles/agent-resident-execution-substrate/continuity-proof-lineage)
- [Substrate Runtime Continuity \(/articles/agent-resident-execution-substrate/substrate-runtime-continuity\)](/articles/agent-resident-execution-substrate/substrate-runtime-continuity)
- [Personal Corpus Model Training \(/articles/agent-resident-execution-substrate/personal-corpus-model-training\)](/articles/agent-resident-execution-substrate/personal-corpus-model-training)
- [Heterogeneous Inference Endpoints \(/articles/agent-resident-execution-substrate/heterogeneous-inference-endpoints\)](/articles/agent-resident-execution-substrate/heterogeneous-inference-endpoints)
- [Atomic Lifecycle Substitution \(/articles/agent-resident-execution-substrate/atomic-lifecycle-substitution\)](/articles/agent-resident-execution-substrate/atomic-lifecycle-substitution)
- [Integrity Signal Feedback \(/articles/agent-resident-execution-substrate/integrity-signal-feedback\)](/articles/agent-resident-execution-substrate/integrity-signal-feedback)
- [Hardware-Bound Identity \(/articles/agent-resident-execution-substrate/hardware-bound-identity\)](/articles/agent-resident-execution-substrate/hardware-bound-identity)
- [Cognitive State Append-Only Invariant \(/articles/agent-resident-execution-substrate/cognitive-state-append-only-invariant\)](/articles/agent-resident-execution-substrate/cognitive-state-append-only-invariant)
- [Counterparty Identity Records \(/articles/agent-resident-execution-substrate/counterparty-identity-records\)](/articles/agent-resident-execution-substrate/counterparty-identity-records)
- [Privacy Egress-Controlled Disclosure \(/articles/agent-resident-execution-substrate/privacy-egress-controlled-disclosure\)](/articles/agent-resident-execution-substrate/privacy-egress-controlled-disclosure)
- [Federated Cross-Device Agent Identity \(/articles/agent-resident-execution-substrate/federated-cross-device-agent-identity\)](/articles/agent-resident-execution-substrate/federated-cross-device-agent-identity)

APPLICATIONS · GENERAL

- [Personal AI Agents That Survive Device Loss: One Continuous Identity and a Private Corpus Across Every Device \(/articles/agent-resident-execution-substrate/personal-cross-device-agents\)](/articles/agent-resident-execution-substrate/personal-cross-device-agents)
- [Enterprise Agent Fleets: Stable Agent Identity and Governed Tool Access Across Model Upgrades and Infrastructure Migration \(/articles/agent-resident-execution-substrate/enterprise-agent-fleets\)](/articles/agent-resident-execution-substrate/enterprise-agent-fleets)
- [Audit-Grade Agent Identity for Regulated Finance and Healthcare: Continuity-Proof Lineage Across the Agent Lifecycle \(/articles/agent-resident-execution-substrate/regulated-industry-agents\)](/articles/agent-resident-execution-substrate/regulated-industry-agents)
- [Edge and On-Device Agents: Hardware-Bound Identity Across Heterogeneous Inference Endpoints \(/articles/agent-resident-execution-substrate/edge-and-on-device-agents\)](/articles/agent-resident-execution-substrate/edge-and-on-device-agents)
- [Agent-to-Agent Commerce With Counterparty Identity Records and Egress-Controlled Disclosure \(/articles/agent-resident-execution-substrate/agent-to-agent-commerce\)](/articles/agent-resident-execution-substrate/agent-to-agent-commerce)
- [Governed Tool Lifecycles for Managed Inference-Provider Ecosystems: A Substrate Approach to Owning, Routing, and Retiring AI Tools \(/articles/agent-resident-execution-substrate/managed-to-tool-ecosystems\)](/articles/agent-resident-execution-substrate/managed-to-tool-ecosystems)
- [Proving Unbroken Continuity in Long-Lived Autonomous Systems Across Substrate Migration and Atomic Model Substitution \(/articles/agent-resident-execution-substrate/long-lived-autonomous-systems\)](/articles/agent-resident-execution-substrate/long-lived-autonomous-systems)
- [Personal-Model Personalization: A User's Own Corpus-Internalized Model on the Agent-Resident Execution Substrate \(/articles/agent-resident-execution-substrate/personal-model-personalization\)](/articles/agent-resident-execution-substrate/personal-model-personalization)
- [On-Device Agent Identity for Robots and Autonomous Vehicles: An Auditable Substrate for Embodied Physical-World Agents \(/articles/agent-resident-execution-substrate/embodied-physical-world-agents\)](/articles/agent-resident-execution-substrate/embodied-physical-world-agents)
- [When the Agent Rents Everything It Depends On \(/articles/agent-resident-execution-substrate/agent-resident-execution-substrate-stakes\)](/articles/agent-resident-execution-substrate/agent-resident-execution-substrate-stakes)

APPLICATIONS · SPECIFIC

- [LangGraph Platform \(LangChain\) vs an agent-resident execution substrate: orchestration-graph state versus a portable, hardware-anchored agent runtime \(/articles/agent-resident-execution-substrate/langgraph-platform\)](/articles/agent-resident-execution-substrate/langgraph-platform)
- [OpenAI AgentKit and the Assistants/Responses API vs agent-carried, hardware-anchored identity with governed tool lifecycle \(/articles/agent-resident-execution-substrate/openai-agentkit\)](/articles/agent-resident-execution-substrate/openai-agentkit)
- [Microsoft Copilot Studio vs an agent-resident execution substrate: platform-hosted agent authoring versus portable, device-resident agent identity and continuity \(/articles/agent-resident-execution-substrate/microsoft-copilot-studio\)](/articles/agent-resident-execution-substrate/microsoft-copilot-studio)

- [Google Vertex AI Agent Engine \(managed runtime for deploying and scaling agents, with sessions/memory\) vs an agent-carried, continuity-proofed identity substrate \(/articles/agent-resident-execution-substrate/google-vertex-agent-engine\)](#).
- [AWS Bedrock AgentCore \(runtime, memory, identity, and gateway services for deploying agents at scale\) vs an agent-resident execution substrate: where does the agent identity actually live? \(/articles/agent-resident-execution-substrate/aws-bedrock-agentcore\)](#).
- [Letta \(formerly MemGPT\) vs an append-only cognitive-state substrate: what a memory-management framework does not provide \(/articles/agent-resident-execution-substrate/letta-memgpt\)](#)
- [Cognition's Devin, an autonomous AI software-engineering agent vs a portable, continuity-proofed agent-resident runtime \(/articles/agent-resident-execution-substrate/cognition-devin\)](#).
- [Cloudflare Agents \(Durable Objects\) vs an agent-resident execution substrate: portable hardware-bound identity and continuity-proof lineage \(/articles/agent-resident-execution-substrate/cloudflare-agents\)](#).
- [Ollama alternative: from local model runner to a governed agent-resident substrate \(/articles/agent-resident-execution-substrate/ollama\)](#)
- [Apple Intelligence \(on-device foundation models, Private Cloud Compute\) vs a persistent agent-resident substrate: who owns identity, lineage, and the model? \(/articles/agent-resident-execution-substrate/apple-intelligence\)](#).
- [**DeepSeek Harness: Plugin Models and Agent-Resident Substrates \(/articles/agent-resident-execution-substrate/deepseek-harness\)**](#)

HOW-TO GUIDES

- [How to Build a Personal AI That Runs on Your Own Device \(/articles/guides/personal-ai-on-your-own-device\)](#)
- [How to Build an Offline-First Personal AI That Syncs When Connected \(/articles/guides/offline-first-personal-ai-that-syncs\)](#).
- [How to Fine-Tune a Personal Model on Your Own Writing Without Sending It to the Cloud \(/articles/guides/fine-tune-personal-model-without-cloud\)](#)
- [How to Give an On-Device AI Agent Tools It Can Use Safely \(/articles/guides/governed-tools-for-an-on-device-agent\)](#).

TERMINOLOGY

- [agent-resident execution substrate \(/glossary#agent-resident-execution-substrate\)](#)
- [continuity attestation \(/glossary#continuity-attestation\)](#).
- [continuity hash \(/glossary#continuity-hash\)](#).
- [lineage-derived training corpus \(/glossary#lineage-derived-training-corpus\)](#).

- [personal corpus model \(/glossary#personal-corpus-model\)](/glossary#personal-corpus-model).

[Agent-Resident Execution Substrate overview → \(/agent-resident-execution-substrate\)](/agent-resident-execution-substrate).