Nick Clark · Published April 25, 2026 · PDF

Personal-Model Personalization: A User's Own Corpus-Internalized Model on the Agent-Resident Execution Substrate

Cloud inference services train one shared model against aggregate data and keep no persistent representation of any individual user's body of work, forcing users to re-supply their own context as prompt text on every request, which is then discarded. This general application shows how a per-user model that internalizes a user's own corpus into its weights, kept current automatically as the user keeps working, can be built on the Agent-Resident Execution Substrate, disclosed in U.S. Provisional Application No. 64/070,239.

What This Application Specifies

This application concerns personal-model personalization: giving an individual user a model that reflects that user's own domain knowledge, terminology, structural conventions, and prior outputs, rather than a generic model consulted through repeated prompting. The mechanism is a personal corpus model, defined in the filed disclosure as a model artifact whose parameters are fine-tuned against a training corpus derived from artifacts the user authored, curated, or designated. The distinguishing property is where the personalization lives. The user's accumulated body of work is internalized in the parameter values themselves; inference behavior is determined by the model's weights, not by retrieval over a document store at query time.

The personal corpus model is one managed inference endpoint among others held in a local tool registry on the user's own device. It sits under a persistent semantic agent that owns a hardware-anchored identity field, a cognitive state field, an append-only lineage field, and a governance policy field. Every artifact the user authors through a text editor, code editor, content interface, or recording interface is registered in the lineage field with its content reference, modality, timestamp, scope identifier, and admissibility metadata. A corpus assembly module periodically derives a training set from those lineage records, selecting only artifacts admissible under the applicable corpus policy, filtering for modality, and applying declared redaction or anonymization. A fine-tuning module applies a parameter-efficient update to the model. A governed substitution module then promotes the updated artifact into the registry, replacing the prior one, and records the substitution in the lineage field. The updated model assists the user's next authoring session, and the artifacts produced under it feed the next cycle. The disclosure frames this as a closed loop.

Why It Matters

The domain problem is structural, not incidental. A shared cloud model is undifferentiated by user identity, so a professional whose value lies in an accumulated body of work has no way to make the model carry that work as an intrinsic property. The common workaround, retrieval-augmented generation, indexes the user's documents and injects fragments into the prompt at inference time. The filed disclosure is explicit that this does not internalize the corpus; the base model's intrinsic representation never improves, and output quality remains hostage to retrieval quality and chunk-boundary effects. User-initiated fine-tuning services offer weight-level personalization but decouple it from authoring: they require manual corpus curation, manual training initiation, and manual deployment, so continuous improvement from ongoing work is not provided.

Personal-model personalization on this substrate closes both gaps at once. Personalization is at the weight level, so the user's conventions are internalized rather than re-fetched. And it is continuous and automatic, driven by the authoring activity itself rather than by discrete curation events. For a user whose competitive edge is their own corpus, this is the difference between renting a generic capability and owning a model that has become an extension of their practice.

How It Composes With the Domain

The personalization loop maps directly onto how an individual actually works. Consider a practitioner who writes in a consistent style, reuses a specific vocabulary, and follows recurring structural patterns. As they author, each finished artifact enters the lineage field. The corpus assembly module later selects the admissible ones and the fine-tuning module updates the model, so the model progressively reflects the current body of work without the user ever running a training job.

The disclosure supports maintaining several personal corpus models at once, each fine-tuned against a distinct subset of the user's artifacts and optionally bound to a distinct named scope. A single agent identity can carry a professional scope, a personal scope, and a project-specific scope, each with its own corpus, its own admissibility policy, and its own lineage partition. One model can specialize in the user's prose in a professional scope, another in the user's source code in a project scope, and another in the user's designated publications. The dispatcher selects among them by input modality, task category, and the active scope, which the user can set explicitly or the agent can infer from input characteristics or the originating application. This scope structure keeps a work persona and a personal persona from bleeding into each other while remaining one continuous identity.

Two composition choices deserve emphasis for this domain. First, the personal corpus model can operate as a specialization layer over a frozen base model: the base supplies general language and reasoning, and the personal layer bends the output toward the

user's accumulated work. Second, and central to why the loop stays honest over time, the training signal is drawn from the lineage field's downstream-outcome references, not from the model's own prior text. Those references record real acceptance, real revision, execution success or failure, and integrity-signal feedback measuring whether an output matched the user's established structural conventions. The disclosure contrasts this with training on a model's own outputs, which is known to concentrate on the model's prior distribution across iterations, and explains that grounding the signal in real outcomes preserves the variance of real-world results across successive retraining events. In this domain, that means the model tracks how the user actually accepts and revises suggestions, not merely what it already tends to produce.

What This Enables

The most direct enablement is a model that a user genuinely owns, running locally, that improves as a byproduct of daily work. Because retraining is a governed lifecycle operation, the update runs in a staging area and is promoted only after policy validation; a failed or invalid update rolls back to the prior artifact with the cause recorded. The user's identity, cognitive state, and lineage are preserved across every substitution under the substrate's continuity guarantee, so the personal model can be replaced arbitrarily without disturbing the agent that owns it.

Personalization is also auditable and portable in a way ordinary fine-tuning is not. Every artifact admitted to a corpus, every retraining event, and every substitution is a deterministic lineage record, so the exact composition of what shaped the model is reconstructible. Across a user's own devices, the disclosure describes a federation layer that exchanges lineage records and governed model updates under a federation policy, without a centralized authority and without treating shared weights or a cloud account as the unit of coordination, so a single user's personalization can move with them across a laptop and a workstation as one federated identity.

Privacy is the enabling foundation for treating a personal model as a private asset. The disclosure specifies a privacy invariant: lineage records, model artifacts, training corpora, personal corpus model parameters, scope-local context, and counterparty records are not transmitted off the device except under an explicit disclosure policy object that names a recipient, a permitted scope, an authorization attestation, a retention requirement, and a revocation mechanism. Enforcement mechanisms named in the disclosure include a runtime egress filter over outbound traffic, per-component isolation, release of transmission keys only after signed disclosure preconditions, and hardware-anchored attestation that the runtime has not been tampered with. When local capability or capacity falls short, a cloud-burst forwarding subsystem can selectively forward a request to a remote endpoint, but only after a capability, capacity, disclosure, and cost admissibility test, and any forwarded payload is itself evaluated and recorded as an off-device disclosure event, with a confidential-execution mode that decrypts the payload only inside the remote endpoint's trusted execution environment. Personalization therefore stays private by default, and any exception is governed and logged.

Boundary Conditions

Personal-model personalization here is a weight-level mechanism whose quality tracks the artifacts a user actually produces; the disclosure describes the architecture, not any particular level of output quality, and this application makes no performance claim. A user with a thin or inconsistent body of work has less signal to internalize, and the value grows with the corpus. The substrate targets a bounded local compute envelope, so the fine-tuning is parameter-efficient and must fit within a policy-declared training window; very large base models or aggressive full-parameter retraining are constrained by device resources and the resource governance subsystem's budgets, schedules, and quiescence rules. A policy-declared lower bound on retraining frequency keeps the model from drifting too far from current work, but personalization is incremental, not instantaneous. Off-device forwarding and multi-device federation are optional and

strictly governed; where a disclosure policy does not permit it, forwarding is denied and recorded, which is the intended behavior rather than a limitation to design around. Whether any specific personalization outcome complies with a given jurisdiction's data-protection or sector rules is an external legal question this application does not resolve.

Disclosure Scope

The technology described here, the personal corpus model, its lineage-derived corpus assembly and parameter-efficient fine-tuning, the governed substitution and continuity guarantee, scope partitioning, the append-only lineage field, the privacy invariant, and cloud-burst forwarding, is disclosed in U.S. Provisional Application No. 64/070,239, and every statement above about what the invention does traces to that disclosure. The domain framing, including references to how individual professionals accumulate a body of work, how retrieval-augmented and manual fine-tuning approaches are used in practice, and general references to trusted execution environments and data-protection regimes, is provided as external context to illustrate an enabling implementation and does not form part of the disclosed invention. Nothing here should be read to expand the disclosure beyond its terms or to characterize any external product, service, or regulatory regime as covered by it.

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

All 40 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)

SECONDARY TECHNICAL

- Persistent Semantic Agent (/articles/agent-resident-execution-substrate/persistent-semantic-agent)
- 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)
- 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)
- Cognitive State-Conditioned Dispatch (/articles/agent-resident-execution-substrate/cognitive-state-conditioned-dispatch)
- Governed Tool Lifecycle (/articles/agent-resident-execution-substrate/governed-tool-lifecycle)
- Continuity-Proof Lineage (/articles/agent-resident-execution-substrate/continuity-proof-lineage)
- Substrate Runtime Continuity (/articles/agent-resident-execution-substrate/substrate-runtime-continuity)
- Personal Corpus Model Training (/articles/agent-resident-execution-substrate/personal-corpus-model-training)
- Heterogeneous Inference Endpoints (/articles/agent-resident-execution-substrate/heterogeneous-inference-endpoints)
- Atomic Lifecycle Substitution (/articles/agent-resident-execution-substrate/atomic-lifecycle-substitution)
- Integrity Signal Feedback (/articles/agent-resident-execution-substrate/integrity-signal-feedback)
- 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)
- Counterparty Identity Records (/articles/agent-resident-execution-substrate/counterparty-identity-records)
- 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)

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)

- Enterprise Agent Fleets: Stable Agent Identity and Governed Tool Access Across Model Upgrades and Infrastructure Migration (/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)
- Edge and On-Device Agents: Hardware-Bound Identity Across Heterogeneous Inference Endpoints (/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)
- Governed Tool Lifecycles for Managed Inference-Provider Ecosystems: A Substrate Approach to Owning, Routing, and Retiring AI Tools (/articles/agent-resident-execution-substrate/managed-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)
- **Personal-Model Personalization: A User's Own Corpus-Internalized Model on the Agent-Resident Execution Substrate (/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)

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)
- OpenAI AgentKit and the Assistants/Responses API vs agent-carried, hardware-anchored identity with governed tool lifecycle (/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)
- 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)

Agent-Resident Execution Substrate overview → (/agent-resident-execution-substrate)