Nick Clark · Published April 25, 2026 · PDF

Apple Intelligence (on-device foundation models, Private Cloud Compute) vs a persistent agent-resident substrate: who owns identity, lineage, and the model?

Apple Intelligence pairs on-device foundation models with Private Cloud Compute to run generative features privately, escalating to Apple-operated servers only when a request exceeds on-device capacity. The open architectural question is not whether inference runs locally, but whether the personal device carries a persistent, auditable agent identity that owns its models, records an append-only history of every inference, and internalizes the user's own body of work into model weights. That is the axis addressed by the Agent-Resident Execution Substrate, disclosed in U.S. Provisional Application No. 64/070,239.

What Apple Intelligence (on-device foundation models, Private Cloud Compute) Does

Apple Intelligence is Apple's system-level generative AI capability integrated across its operating systems. Its architecture is built around two publicly described tiers. First, a compact foundation model runs directly on the device, handling many language and image tasks locally so that the associated data does not leave the hardware. Second,

when a request exceeds what the on-device model can handle, Apple Intelligence can escalate to Private Cloud Compute, a server environment Apple operates on its own silicon.

Private Cloud Compute is a serious and well-executed piece of privacy engineering. Apple has publicly committed that data sent to Private Cloud Compute is used only to fulfill the request and is not retained, that the servers run software whose images are made available for independent inspection, and that a device will send data only to a server whose software has been cryptographically verified against a published measurement. This is a meaningfully stronger posture than typical multi-tenant cloud inference, and it deserves credit as such. For a large consumer base, Apple Intelligence delivers capable on-device inference with a credible, verifiable privacy story and deep operating-system integration.

The design goal of Apple Intelligence is a privacy-preserving inference service woven into consumer features. It is optimized to answer a request well and then move on. That framing is the right one for a mass-market assistant, and it is also what distinguishes it from the architecture discussed below.

The Architectural Axis

The axis here is statefulness and ownership at the personal device, not privacy of a single request. Apple Intelligence, whether it runs on-device or on Private Cloud Compute, is fundamentally a request-and-response inference facility. Each request is served against a model shared across all users of that model version. The model is Apple's, refreshed on Apple's schedule; the device is a client of it. There is no persistent, device-resident entity that treats the local models as assets it owns, that keeps a durable and verifiable record of what it inferred and how those inferences turned out, or that folds the individual user's accumulated work back into model weights over time.

None of that is a defect in Apple Intelligence. A consumer assistant does not need a hardware-anchored agent identity or an append-only audit ledger to autocomplete an email. But there is a distinct class of use, including regulated authoring, professional drafting, and long-horizon autonomous operation, where the durable state, the auditability, and the ownership of the model are precisely the point. That is the class the disclosed architecture targets.

How the Disclosed Approach Differs

The Agent-Resident Execution Substrate inverts the client-of-a-service relationship. As disclosed in U.S. Provisional Application No. 64/070,239, a semantic agent persists as the execution substrate of the device, and inference endpoints are managed assets subordinate to that agent rather than an external service the device calls.

Four structural differences follow.

A persistent, continuity-verified identity. The agent maintains a persistent identity field that may be hardware-anchored, and its identity, cognitive state, and lineage are preserved across power cycles, application restarts, and, critically, across replacement or retraining of every model it uses. The spec discloses a continuity guarantee under which no lifecycle operation on a subordinate model alters the agent's persistent fields except by appending to its lineage. The device is not a stateless caller; it hosts a continuous entity whose identity survives arbitrary substitution of the underlying models.

An append-only lineage of every inference. The agent records each dispatched inference request, its output, and its downstream outcome as an append-only lineage record chained to its predecessor under a continuity proof, such that the complete operational history is deterministically reconstructible and any alteration is detectable. This is a durable, verifiable record of what the device inferred and how it turned out, held on the

device. A privacy-preserving request-response service, by design, does not maintain such a per-user ledger; the point of Private Cloud Compute is that the request leaves no trace on the server.

A governed model lifecycle owned by the device. Models live in a managed inference tool registry, and installation, retraining, replacement, archival, and removal are governed lifecycle operations evaluated against cryptographically signed policy objects and recorded in the lineage. The device, under the user's governance policy, decides which models exist and when they change, rather than receiving model updates on the vendor's release cadence.

A weight-internalized personal corpus model. The registry can hold a personal corpus model whose parameters are fine-tuned, using parameter-efficient methods within the device's local compute envelope, against artifacts the user authored or designated. The spec is explicit that the user's body of work is internalized in the weights rather than consulted as external context at inference time, and that the model is progressively updated as the user produces more work through a closed loop of authoring, lineage recording, governed corpus assembly, and governed substitution. This is a different mechanism from prompting a shared model with per-request context.

The substrate also defines a privacy invariant enforced independent of network connectivity: lineage, model parameters, corpus, and counterparty records are not transmitted off-device except under an explicit, signed disclosure policy, enforceable through a substrate-runtime egress filter, with every off-device disclosure itself recorded as an auditable lineage event. When local capability or capacity is insufficient, a governed cloud-burst forwarding subsystem may forward a request to a remote endpoint, but only after policy, disclosure, and cost tests, and the forwarding is logged as a disclosure event subject to the same invariant.

Where They Fit Together

These are complementary, not mutually exclusive. Apple Intelligence and Private Cloud Compute are a capable, verifiable inference tier. The disclosed substrate is an ownership-and-accountability layer for a persistent personal agent. In principle the two compose: a strong external inference tier is exactly the kind of high-capability, capacity-elastic remote endpoint that the substrate's cloud-burst forwarding subsystem is designed to reach when a local model lacks capability or capacity, with the forwarding governed by policy and recorded as an auditable disclosure event.

Put plainly, Apple Intelligence answers the question "can this request be served privately and well?" The substrate answers a different question: "does this device carry a durable, auditable agent that owns its models and its history?" A deployment could use the substrate for persistent identity, lineage, and a personal corpus model while forwarding capability-limited requests to a hardened external tier under its own disclosure policy. Choosing between them is only necessary when a use case genuinely needs one axis and not the other.

Boundary Conditions

Honesty requires several qualifications. U.S. Provisional Application No. 64/070,239 is an early-stage disclosure describing an architecture and its embodiments; a provisional is not an issued patent, its claims are not yet examined, and the descriptions here trace to what the specification discloses rather than to a shipped, benchmarked product. On-device fine-tuning of a personal corpus model is bounded by the device's real memory, storage, compute, and power envelope, which is why the spec discloses parameter-efficient methods, policy-declared training windows, and scheduling into low-activity periods; a persistent lineage and multiple resident models also consume storage and management overhead that a stateless client avoids.

On the other side, the characterization of Apple Intelligence and Private Cloud Compute above is limited to genuine, widely-published, architecture-level facts as of this writing. Apple's specific model sizes, escalation heuristics, and feature roadmap evolve, and any detail can be superseded by later Apple releases. Nothing here should be read as attributing a defect or a security weakness to Apple Intelligence or Private Cloud Compute; the distinction drawn is one of architectural purpose, where a privacy-preserving request-response service and a persistent, self-owning agent substrate are simply built to do different jobs.

Disclosure Scope

The architectural mechanisms attributed to the disclosed approach, including the persistent hardware-anchorable agent identity, the append-only lineage under continuity proofs, the governed managed-inference tool registry, the weight-internalized personal corpus model, the privacy invariant with its egress filter, and governed cloud-burst forwarding, are described in and trace to U.S. Provisional Application No. 64/070,239. References to Apple Intelligence, on-device foundation models, and Private Cloud Compute are provided solely as external market and technical context to situate that disclosure; they are not claims of the filing, they describe products and standards operated by their respective owner, and nothing in this article asserts or implies any defect, deficiency, or wrongdoing on the part of that competitor. Any comparison is limited to architectural purpose and scope, and forward-looking descriptions of either side may be superseded by later development.

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)