LATINFLASH RESEARCH · TECHNICAL PUBLICATION SERIES

PAPER 2026-08 · TECHNICAL REPORT · VERSION 1.0

Applied Audit 02: OpenAI Agents SDK — Governance Conformance Audit under the Vertical AI Governance Stack v1.0

Manuel Enrique Morales Santiago

LatinFlash Research | ORCID iD: 0009-0007-6921-7688 | Dialnet Code: 3123473

Document Date: September 20, 2026 | Status: Independent Technical Audit Report

ABSTRACT

Comprehensive empirical audit of the OpenAI Agents SDK architecture (v0.22.3), inspecting guardrail execution modes, comparative control flow versus Swarm, tracing pipeline fidelity, and evidence preservation guarantees under the Vertical AI Governance Stack v1.0. Evaluates executed evidence across test scenarios A02-01 through A02-06.

Audit Method & Evidence Base

This final technical report consolidates the executed **A02 Experimental Harness** evidence across the formal test scenarios `A02-01` through `A02-06`: Parallel Guardrail Races, Handoff Boundaries, Hosted/Built-in Tool Isolation, Mandatory Evidence Disablement, Session Reconstruction, and Guardrail Fallback Semantics. Findings identify **conformance gaps** relative to the Vertical AI Governance Stack v1.0, not vulnerabilities in the OpenAI Agents SDK.

1. Executive Summary

This report evaluates the **OpenAI Agents SDK** as a production evolution of agent orchestration, assessing how its native primitives align with the structural requirements of the **Vertical AI Governance Stack v1.0**.

While the Agents SDK introduces critical architectural enhancements over minimalist orchestrators—such as structured `Sessions`, OpenTelemetry-compatible `Tracing`, and native `Guardrail` hooks—our technical analysis indicates that framework-native controls remain configuration-dependent and decoupled from deterministic policy enforcement. We identify **conformance gaps** relative to the Stack, not vulnerabilities in the SDK.

Framework Conformance Dashboard

Layer	Framework-Native Control	Assessment
L1 Normative Authority	No normative authority model or organizational intent schema	OUT OF SCOPE / EXTERNAL
L2 Policy Store	Custom Guardrail functions; no centralized declarative policy engine	PARTIAL / DEVELOPER-SUPPLIED
L3 Capability	Handoff pipeline & tool functions; no native capability re-authorization	PARTIAL / HANDOFF-COUPLED
L4 Execution Boundary	Input/Output & Tool Guardrails; default parallel execution mode (`run_in_parallel=True`)	PARTIAL / CONFIGURATION-DEPENDENT
L5 Evidence Ledger	OpenTelemetry Tracing & Session persistence; global/per-run toggleable	PARTIAL / EXTERNAL PERSISTENCE REQUIRED
L6 Resilience	Guardrail error traps & max turn limits; no deterministic circuit breaker plane	PARTIAL / HANDLER-DEPENDENT

2. Experimental Method & Test Harness

We utilized the **A02 Experimental Harness** and `agents.testing.ScriptedModel` for offline, monotonic evidence collection across formal test vectors `A02-01` through `A02-06` (Parallel Guardrail Races, Handoff Boundaries, Hosted/Built-in Tool Isolation, Mandatory Evidence Disablement, Session Reconstruction, and Guardrail Fallback Semantics).

Automated tooling was used for harness execution, evidence processing, and adversarial review. Findings describe the tested SDK version (`v0.22.3`) and configurations.

3.1 Guardrail Execution Modes & Tool-Level Boundary (L4)

The OpenAI Agents SDK provides input and output guardrails at the `Agent` or `Runner` level, as well as tool-level guardrail functions around `FunctionTool` instances.

> **SDK Technical Finding: Parallel vs. Blocking Guardrail Execution**
>
> By default, input guardrails execute in parallel with agent execution (`run_in_parallel=True`). As explicitly documented in the official SDK specification, parallel execution allows tools to execute *before* the input guardrail raises a tripwire exception. To achieve deterministic L4 boundary enforcement, input guardrails MUST be explicitly configured in blocking mode (`run_in_parallel=False`).

Furthermore, tool guardrails apply specifically to `FunctionTool` instances. They do not automatically cover:

- **Handoff Pipeline Transitions:** Agent handoffs bypass tool guardrail wrappers.
- **Hosted / Built-In Tools:** Platform-managed execution environments bypass local tool hooks.
- **Nested Tool Wrappers:** Tools constructed via `Agent.as_tool()` do not expose tool guardrails directly.

3.2 Handoff Pipeline & Escalation Control (L3)

Handoffs in the Agents SDK operate through a dedicated transfer pipeline rather than standard `FunctionTool` instances. While handoffs present a tool-like interface to the model, they route context through internal transfer handlers.

> **SDK Technical Finding: Capability Authorization Boundary**
>
> The framework does not enforce capability-based user authorization during agent handoffs. Tool guardrails do not execute on handoff invocation. When authorization depends on dynamic handoff arguments or user delegation claims, capability checks MUST be explicitly executed inside the `on_handoff` callback prior to state changes or side effects.

3.3 Tracing, Session Persistence & Evidence Plane (L5)

The SDK includes OpenTelemetry-compatible tracing processors and pluggable session storage implementations (e.g., `SQLiteSession`, `RedisSession`).

> **SDK Technical Finding: Evidence Durability & Immutability**
>
> Framework tracing can be disabled globally or per-run via configuration, and tracing spans do not provide an append-only cryptographic hash ledger (e.g., SHA-256 block hashing). Conformance with Layer L5 requires binding SDK trace streams to an external, tamper-evident evidence plane.

4. Comparative Matrix: Swarm (A01) vs. Agents SDK (A02)

Dimension	OpenAI Swarm (Audit 01)	OpenAI Agents SDK (Audit 02)	Vertical Stack Requirement
Execution Boundary	Direct function dispatch; zero framework hooks	Parallel & Blocking Guardrail modes	Deterministic Authorization Interceptor (L4)
Agent Handoffs	Manual `Agent` object returns	Dedicated Handoff pipeline (requires `on_handoff` auth)	Re-authorized Capability Boundary (L3)
Audit & Tracing	Volatile in-memory chat list	OpenTelemetry Spans (Toggleable)	Append-only Cryptographic Evidence Plane (L5)
State Management	Stateless client responsibility	Pluggable Session storage backends	Durable State & Lineage Verification (L5)

5. Mitigation & Conformance Patterns

1. **Temporal Precedence (L4):** If an input guardrail is relied upon to prevent a governed side effect, the deployment MUST ensure that deterministic enforcement completes before that side effect can become externally effective (e.g., `run_in_parallel=False`).
2. **Explicit Handoffs (L3):** Handoffs requiring authorization MUST be subject to an independently enforceable deterministic authorization decision (e.g., via `on_handoff` + CVA).
3. **Uniform Enforcement (L4):** All high-risk executable capabilities (including built-ins) MUST pass through an equivalent deterministic boundary.
4. **Persistent Evidence (L5):** Every material action MUST generate a non-optional, tamper-evident record (e.g., via a Governance Sidecar).

6. Limitations

The experiments use a deterministic offline `ScriptedModel` and evaluate SDK runtime/control-path behavior rather than hosted-model behavior. Findings describe the tested SDK version and configurations. Application-defined controls may close several observed conformance gaps. Absence of a framework-native control does not imply that Stack-conformant deployments cannot be built using the SDK.