# **A Comparative Analysis of Agentic AI Frameworks for Enterprise Application Development**

## **Executive Summary and Strategic Recommendation**

### **Overview**

This report provides a comprehensive comparative analysis of four leading AI agent frameworks: Microsoft's Autogen, LangChain's LangGraph, Microsoft's Semantic Kernel, and the OpenAI Agents SDK. The primary objective is to identify the optimal framework for building enterprise-grade applications under a specific set of operational constraints. These constraints include exclusive access to OpenAI's Chat Completion and Embedding APIs, a mandate for OpenTelemetry (OTEL) integration for observability, and the requirement to support a mix of deterministic, workflow-based solutions and more autonomous, agentic systems.

### **Key Findings**

A thorough evaluation of the frameworks against the specified enterprise requirements has yielded the following critical findings:

* **LangGraph** emerges as the most mature and feature-complete framework for building stateful, auditable, and reliable applications. It offers first-class, native support for state persistence, human-in-the-loop (HITL) workflows, and OpenTelemetry, aligning perfectly with all core enterprise mandates.1
* **Semantic Kernel** presents a compelling, enterprise-first vision with a strong focus on stability and integration with conventional code.4 However, its Python implementation currently lags behind its.NET counterpart in feature maturity, particularly concerning out-of-the-box observability and human-in-the-loop capabilities.6
* **Autogen** excels in its designated niche of complex, multi-agent conversational simulations and research.8 Its core abstractions are designed for flexible agent collaboration, but it is less suited for building the deterministic, production-grade workflows required by the enterprise. Its primary user interface, AutoGen Studio, is explicitly designated as a "research prototype" and not intended for production use.10
* **OpenAI Agents SDK** offers a lightweight, modern, and production-ready path for building simpler agentic applications.11 However, its lack of native OpenTelemetry support, requiring a third-party wrapper for integration, introduces a significant dependency risk for an enterprise environment with a strict observability mandate.13

### **Top-Line Recommendation**

For an enterprise operating under the stated constraints, **LangGraph is the unequivocally recommended framework**. This recommendation is based on its superior alignment with all critical requirements: a robust architecture for hybrid workflow/agent systems, native first-class support for state management and persistence, integrated human-in-the-loop functionality, and the most comprehensive and flexible OpenTelemetry integration among the candidates.

### **Alternative Scenarios**

While LangGraph stands out for this specific use case, other frameworks could be considered under different circumstances:

* If the primary objective were **academic research or prototyping advanced multi-agent conversational dynamics**, Autogen's flexible, conversation-centric architecture would make it a stronger contender.4
* If the enterprise were a **.NET-centric organization**, Semantic Kernel's value proposition would increase substantially, as its.NET SDK is its most mature and feature-rich implementation.6

## **Framework Philosophy and Ecosystem at a Glance**

### **Introduction**

Understanding the core design philosophy of each framework—its origin, intended audience, and fundamental abstractions—is crucial for strategic alignment. This philosophy is a strong predictor of a framework's future development trajectory, support model, and long-term suitability for enterprise deployment. A framework is not merely a collection of features; it is a bet on an ecosystem and its underlying vision.

### **Autogen (Microsoft Research)**

* **Philosophy:** Autogen is a product of Microsoft Research's AI Frontiers Lab, and its design reflects a research-first orientation.4 The framework's primary goal is to simplify the creation and orchestration of complex, multi-agent systems to explore their emergent capabilities.8 Its core abstraction is the  
  ConversableAgent, an entity that communicates with other agents through an event-driven messaging architecture to collaboratively solve problems.8 The emphasis is on flexibility in conversation patterns and agent collaboration rather than on rigid, predictable execution.15
* **Ecosystem:** The Autogen ecosystem is layered, consisting of a foundational Core API for message passing and runtimes, a higher-level AgentChat API for rapid prototyping of conversational patterns, and an Extensions package for integrating external tools and services.9 The ecosystem also includes developer tools like  
  AutoGen Studio, a low-code interface for prototyping agent teams, and AutoGen Bench for performance evaluation.9

### **Langchain/LangGraph (LangChain Inc.)**

* **Philosophy:** LangGraph is part of the broader LangChain ecosystem, a developer-first, open-source initiative by LangChain Inc. aimed at providing composable tools for the entire LLM application lifecycle.1 While LangChain itself focuses on "chains" (linear sequences of calls), LangGraph was specifically created to address the limitations of this model, enabling the construction of cyclical, stateful, and long-running agentic applications.1 Inspired by robust data processing frameworks like Google's Pregel and Apache Beam, LangGraph provides low-level orchestration primitives, representing any process as a state graph without abstracting away the underlying prompts or logic.1
* **Ecosystem:** The ecosystem is one of its greatest strengths, comprising LangChain (a vast library of components and integrations), LangGraph (for stateful orchestration), and LangSmith (a dedicated platform for observability, debugging, and evaluation).1 While LangGraph can be used as a standalone library, its true power is realized through its seamless integration with LangSmith, which provides unparalleled visibility into complex agent behaviors.1

### **Semantic Kernel (Microsoft)**

* **Philosophy:** In contrast to the research-led Autogen, Semantic Kernel is an enterprise-first SDK from Microsoft's product division, designed to integrate LLM capabilities into conventional enterprise applications safely and reliably.4 Its core philosophy is to bridge the gap between natural language "semantic" functions and traditional "native" code written in languages like C#, Python, and Java.5 The central abstraction is the  
  Kernel, which orchestrates Plugins (collections of functions) to fulfill complex requests.22 The design prioritizes stability, enterprise-grade support, security, and the ability to blend AI with existing business logic.4
* **Ecosystem:** The Semantic Kernel ecosystem is deliberately structured to serve enterprise needs. It provides two distinct high-level frameworks: a Process Framework for orchestrating deterministic, event-driven business processes, and an Agent Framework for building more autonomous, collaborative agents.21 This clear architectural separation is a hallmark of its design, guiding developers toward building robust and maintainable systems.22

### **OpenAI Agents SDK (OpenAI)**

* **Philosophy:** The OpenAI Agents SDK is designed with a "less is more" philosophy. It is a lightweight, minimalist SDK with very few abstractions, intended to be easy to learn and "production-ready" out of the box.11 Positioned as a production-grade evolution of OpenAI's earlier experimental "Swarm" project, its core primitives are simple yet powerful:  
  Agents (LLMs with tools), Handoffs (a delegation mechanism), and Guardrails (for input/output validation).11 The guiding principle is to provide just enough functionality to be useful while affording developers the flexibility to customize and control the execution flow.11
* **Ecosystem:** The SDK's ecosystem is tightly integrated with OpenAI's broader suite of services. While the framework itself is provider-agnostic and can work with any Chat Completions-style API, its built-in tracing is designed to connect directly to OpenAI's platforms for evaluation, debugging, and model fine-tuning.11 This creates a streamlined experience for developers who are primarily within the OpenAI ecosystem.

The distinct origins of these frameworks are a powerful indicator of their future direction. Autogen's foundation in research suggests it will continue to be a testbed for cutting-edge multi-agent concepts, prioritizing innovation over the long-term stability required for critical enterprise workflows. LangGraph's roots in the developer-centric and commercially backed LangChain ecosystem point to a future of rapid, community-driven evolution focused on flexibility and broad integration. Semantic Kernel's position as a Microsoft enterprise product implies a more cautious, deliberate evolution, prioritizing API stability, security, and deep integration with the Azure stack. Finally, the OpenAI Agents SDK's origin within OpenAI itself means its development will be intrinsically linked to the evolution of OpenAI's models and APIs, ensuring tight integration with new features but potentially creating a dependency on OpenAI's specific architectural choices.

For an enterprise, this is not just a technical choice but a strategic one. A bet on a framework is a bet on its ecosystem's philosophy. For an organization that values long-term support, predictable maintenance, and architectural flexibility, the philosophies of LangGraph and Semantic Kernel are more closely aligned with enterprise needs than the research-focused Autogen or the tightly-coupled OpenAI Agents SDK.

### **Table 2.1: High-Level Framework Comparison**

| Framework | Primary Maintainer | Core Abstraction | Primary Language(s) | GitHub Stars | GitHub Forks | GitHub Contributors |
| --- | --- | --- | --- | --- | --- | --- |
| **Autogen** | Microsoft Research | Conversable Agent | Python,.NET | 49.6k+ 28 | 8k+ 28 | 172+ 16 |
| **Langchain/LangGraph** | LangChain Inc. | State Graph | Python, JS/TS | 110k+ 19 | 17.9k+ 19 | 3,652+ 19 |
| **Semantic Kernel** | Microsoft | Kernel & Plugins | C#, Python, Java | 25.1k+ 29 | 4k+ 29 | 391+ 29 |
| **OpenAI Agents SDK** | OpenAI | Agent, Handoff, Guardrail | Python, JS/TS | 11.7k+ 27 | 1.7k+ 27 | 85+ 27 |

## **Maturity, Stability, and Enterprise Readiness**

### **Versioning and Production Readiness**

The stability and production-readiness of a framework are paramount for enterprise adoption. The four frameworks exist at different points on this maturity spectrum.

* **Autogen:** The framework is in a state of active and significant evolution. The recent release of version 0.4 marked a complete redesign of the library to improve robustness and scalability.30 This indicates healthy development but also a degree of instability that can be challenging for production systems. Critically, its low-code interface, AutoGen Studio, is explicitly labeled a "research prototype" and is "not meant to be used in a production environment," highlighting security and feature gaps.10
* **LangGraph:** This framework signals strong production readiness. It has a stable v0.1 release and is explicitly "trusted in production by companies shaping the future of agents – including Klarna, Replit, Elastic, and more".1 This real-world adoption by major technology companies is a powerful testament to its stability and suitability for mission-critical applications.
* **Semantic Kernel:** Having reached a v1.0 release, Semantic Kernel has made a formal commitment to API stability and enterprise-grade support, which is a crucial milestone for enterprise adoption.4 The framework manages innovation by marking new, evolving features with an  
  [Experimental] attribute, which signals that they are not yet production-ready and may change.6 This dual-track approach allows the core of the framework to remain stable while still providing access to cutting-edge capabilities.
* **OpenAI Agents SDK:** This SDK is explicitly described by its creators as a "production-ready upgrade" of the previous experimental Swarm project.11 Although it is one of the newer frameworks, its minimalist design and backing from OpenAI lend credibility to its readiness for deployment in well-defined, real-world applications.

### **Community Support and Documentation**

A strong community and high-quality documentation are vital for developer productivity, troubleshooting, and long-term maintenance.

* **Autogen:** Benefits from a vibrant Microsoft-backed community with a dedicated Discord server, active GitHub Discussions, and regular community office hours.32 Its documentation is extensive, reflecting the framework's layered and flexible architecture, though some developers have found it to be complex and "not very clear".9
* **LangGraph:** Leverages the immense scale of the LangChain community, which is arguably the largest in the LLM application development space. This includes a highly active Slack community, a dedicated forum, frequent events, and thousands of open-source contributors.35 The documentation is widely regarded as high-quality, featuring a wealth of tutorials, how-to guides, and reference examples that facilitate a smooth learning curve.1
* **Semantic Kernel:** As a Microsoft product, it is supported by official, professionally maintained documentation on Microsoft Learn.21 Community engagement occurs on a dedicated Discord server and through GitHub Discussions.40 While the documentation is structured and thorough, there can be a feature and documentation lag between the flagship.NET SDK and its Python and Java counterparts.6
* **OpenAI Agents SDK:** Community support is centered around the official OpenAI developer forum and GitHub.42 The documentation is clear, concise, and well-structured, consistent with the SDK's minimalist philosophy, making it easy to get started.11 As a newer framework, its community is smaller than LangChain's but is growing rapidly due to its direct affiliation with OpenAI.

The choice of framework involves a critical trade-off between development velocity and long-term stability. Semantic Kernel represents the high-stability, measured-velocity end of the spectrum, with its v1.0 guarantee and formal [Experimental] tagging system providing a classic enterprise software model.4 This approach is safe and predictable but may result in a longer wait for new features to reach the stable branch. Conversely, the LangChain ecosystem operates at a very high velocity, driven by its massive open-source community.17 This provides immediate access to the latest techniques but comes with a higher maintenance overhead to manage potential breaking changes. Autogen is in a "research velocity" mode, with ongoing redesigns that are beneficial for innovation but unsuitable for production systems requiring predictable maintenance.30 The OpenAI Agents SDK seeks a middle ground, offering a stable, production-ready core with a minimal API surface to reduce the likelihood of breaking changes.11 For a large enterprise, the more stable models offered by Semantic Kernel and LangGraph are generally preferable. LangGraph's specific focus as a "low-level orchestration framework" provides a more stable foundation to build upon compared to some of the higher-level, more volatile abstractions in the broader LangChain library.1

## **Architectural Paradigms: Orchestrating Workflows and Agents**

### **The Workflow-Agent Dichotomy**

A key requirement for the enterprise is the ability to build both highly structured, deterministic "workflows" and more flexible, autonomous "agents." This distinction is critical for building reliable applications; workflows are preferable for their predictability and consistency, while agents are better suited for open-ended problems where the path to a solution is not known in advance.44 The architectural approach each framework takes to support this duality is a major differentiator.

### **Framework Approaches**

* **Autogen:** This framework is fundamentally agent-centric. It models nearly every process as a "multi-agent conversation".8 A deterministic workflow is typically implemented by carefully structuring this conversation, for instance, by defining agents with specific roles like "Planner" and "Executor" and enforcing a strict turn-based interaction pattern.47 While this is possible, it can feel less natural than frameworks that have a dedicated workflow abstraction, as it requires fitting a linear process into a conversational paradigm.9
* **LangGraph:** LangGraph was explicitly designed to model processes as directed state graphs, making it a natural fit for both paradigms.18 Its core abstraction, the  
  StateGraph, defines an application as a set of nodes (functions representing actions) and edges (directing the flow between nodes). This is ideal for creating deterministic workflows. Agentic behavior is achieved by creating cyclical graphs where a node containing an LLM call uses conditional edges to dynamically decide the next step based on its output. This unified graph abstraction elegantly and powerfully represents both simple linear sequences and complex, adaptive loops.9
* **Semantic Kernel:** This framework offers the most explicit architectural separation between the two concepts. It provides a dedicated Process Framework for orchestrating event-driven, deterministic workflows and a separate Agent Framework for building collaborative, autonomous agents.21 This prescriptive approach guides developers to use the right tool for the job, promoting a clear and maintainable architecture, which is highly desirable in an enterprise context.
* **OpenAI Agents SDK:** This SDK provides a set of flexible, low-level primitives that the developer must compose into a desired pattern. A deterministic workflow can be constructed by explicitly chaining tool calls. Agentic behavior is enabled by the built-in Runner.run() loop, which executes until a defined terminal state is reached.48 For multi-agent systems, the  
  Handoffs primitive acts as a form of routing or delegation, allowing one agent to pass control to another.11 The architecture is highly flexible but places the responsibility for creating a sound structure entirely on the developer.31

The frameworks offer different levels of abstraction, creating a trade-off between prescriptive guidance and architectural freedom. Semantic Kernel is the most prescriptive, providing distinct, high-level frameworks for each pattern, which minimizes architectural errors but offers less flexibility in blending the two.21 The OpenAI Agents SDK is the least prescriptive, offering low-level building blocks that provide maximum freedom but also carry the highest risk of leading to poorly designed, hard-to-maintain systems.11 Autogen is prescriptive in its own way, pushing all problems into a multi-agent conversational model.8 LangGraph strikes a compelling balance. Its unified state graph abstraction provides a single, powerful mental model that can naturally represent both deterministic workflows (as linear graphs) and complex agents (as cyclical graphs), offering a solid architectural pattern without being overly restrictive.9 For an enterprise team tasked with building a variety of robust and scalable hybrid applications, LangGraph's approach offers an ideal middle ground.

## **Core Enterprise Capabilities: A Head-to-Head Analysis**

### **State Management and Persistence**

The ability to reliably manage state and persist it durably is non-negotiable for long-running enterprise applications.

* **Autogen:** State is managed primarily through the conversation history maintained within each agent. Parameters like summary\_method offer some control over the context window.47 However, for durable persistence across sessions or application restarts, Autogen relies on the developer to implement a custom solution using external stores like Redis or a database.14
* **LangGraph:** State management is a first-class, core concept. The StateGraph explicitly defines a state object that is passed between nodes, making the flow of state transparent and manageable.18 Critically, it includes built-in  
  checkpointer mechanisms (e.g., MemorySaver for in-memory, SqliteSaver for disk-based) that automatically persist the graph's state at each step.1 This enables "durable execution," allowing workflows to be paused and resumed seamlessly, even after a failure.1
* **Semantic Kernel:** State management is handled differently depending on the component. The Process Framework is designed to manage the state of a workflow, allowing it to go idle while awaiting external events and then resume.7 Stateful agent services manage their own state remotely, while other agent types require the application to manage the chat history manually.22 The out-of-the-box support for durable persistence is less explicit than LangGraph's checkpointer system.
* **OpenAI Agents SDK:** Provides a simple yet effective mechanism for persistence. Each run produces a RunState object that can be serialized to a JSON string and later deserialized to resume execution from that exact point.52 While this provides the necessary primitive for persistence, it places the burden of storing and retrieving the serialized state (e.g., from a database) on the developer.

### **Human-in-the-Loop (HITL) and Intervention**

Incorporating human oversight and approval is a common requirement for enterprise workflows that involve sensitive or high-stakes actions.

* **Autogen:** Provides a basic HITL mechanism through the human\_input\_mode parameter on its ConversableAgent, which can be set to ALWAYS, TERMINATE, or NEVER.53 This allows a human to interrupt a conversation and provide input. However, this is more of a conversational intervention than a formal workflow approval gate. More complex approval flows require custom implementations, such as a dedicated task queue.50
* **LangGraph:** HITL is a core, built-in feature of the framework. Any node in the graph can call the interrupt function, which pauses the graph's execution and saves its state via the checkpointer.3 The application can then await external input (e.g., a user clicking an "Approve" button) and resume the graph with the new information. This makes it exceptionally well-suited for building robust approval workflows.3
* **Semantic Kernel:** Supports HITL by allowing developers to design processes that explicitly wait for specific external events (like a UserApprovedDocument message) before proceeding to the next step.7 This approach is powerful and flexible but is generally more complex to implement than LangGraph's built-in  
  interrupt function, often requiring integration with an external pub/sub messaging system.7
* **OpenAI Agents SDK:** Features a built-in concept of tool approval. Any tool can be defined with a needsApproval: true flag. When an agent attempts to use such a tool, the run is automatically interrupted, and a ToolApprovalItem is returned to the calling application.52 The application can then programmatically approve or reject the action and resume the run. This provides a clean, elegant mechanism specifically for validating tool use.52

### **Workflow Replay and Debugging**

Debugging complex, non-deterministic agentic systems is a significant challenge. The ability to trace execution and replay workflows is critical for development and maintenance.

* **Autogen:** Debugging relies on standard Python logging and its OTEL integration for tracing.14 Replaying a workflow from a specific state would require a custom implementation to save and reload the complete state of all participating agents. While AutoGen Studio offers some visualization, it is not a production-grade debugging tool.10
* **LangGraph:** This is a standout area for the framework. Its deep integration with the LangSmith observability platform provides detailed, visual traces of graph execution, showing the exact flow of data and state between every node.1 Combined with the checkpointer system that saves the state at each step, developers can easily inspect the state at any point of failure and replay the workflow from that specific checkpoint. This combination is invaluable for debugging complex, cyclical agent behaviors.1
* **Semantic Kernel:** Debugging is primarily accomplished through standard logging and its evolving observability features.5 The ability to replay a workflow is contingent on the specific state management solution implemented by the developer.
* **OpenAI Agents SDK:** Comes with built-in tracing capabilities that allow for visualization and debugging of agentic flows.11 The ability to serialize and deserialize the  
  RunState object provides the fundamental mechanism needed for replay.52 An application can save the state before a problematic step, allow a developer to inspect it, and then resume execution from that saved state.

### **Table 5.1: Enterprise Feature Support Matrix**

| Capability | Autogen | LangGraph | Semantic Kernel | OpenAI Agents SDK |
| --- | --- | --- | --- | --- |
| **State Management & Persistence** | Requires Custom Implementation | Native/First-Class | Requires Custom Implementation | Built-in Primitive |
| **Human-in-the-Loop (HITL)** | Limited (Conversational) | Native/First-Class | Requires Custom Implementation | Built-in Primitive (Tool Approval) |
| **Workflow Replay & Debugging** | Limited (Requires Custom Logic) | Native/First-Class (via LangSmith) | Limited (Requires Custom Logic) | Built-in Primitive (State Serialization) |

## **Observability Deep Dive: OpenTelemetry (OTEL) Integration**

### **The Enterprise Mandate for OTEL**

A non-negotiable requirement for this evaluation is native or officially supported integration with OpenTelemetry (OTEL). This is essential for channeling traces and metrics into the enterprise's existing observability stack, which includes Grafana and Tempo. This criterion serves as a critical pass/fail test for each framework.

### **Integration Analysis**

* **Autogen:** This framework has documented, first-party support for OpenTelemetry.54 The official documentation provides clear examples of how to instrument application code and, specifically, how to observe LLM calls by using the standard  
  opentelemetry-instrumentation-openai library.54 This direct, official support is a strong indicator of its suitability for enterprise environments with an OTEL mandate.57
* **LangGraph/LangChain:** This ecosystem offers the most mature and flexible OTEL integration of the group. The langsmith SDK provides native, out-of-the-box support for OpenTelemetry.2 The documentation is exceptionally detailed, covering various deployment patterns: sending traces directly to a backend like Grafana/Tempo, sending them to the LangSmith platform for enhanced debugging, or fanning them out to both simultaneously using the OpenTelemetry Collector.2 This level of flexibility and deep integration makes it a best-in-class solution for enterprise observability.2
* **Semantic Kernel:** While observability is a stated design goal for the framework, its native OTEL support is still evolving.22 The OpenTelemetry project is actively working on defining standardized semantic conventions for AI agents, which will eventually provide a unified approach for frameworks like Semantic Kernel.59 Currently, integration is possible using standard OTEL exporters, and third-party tools like OpenLit can be used to instrument the framework.55 However, the first-party support is less mature and clearly documented compared to Autogen and LangGraph.59
* **OpenAI Agents SDK:** By default, this SDK does *not* emit OpenTelemetry data.13 Integration must be achieved using a third-party wrapper. The officially recommended approach, documented in a Microsoft tech community blog post, is to use the Pydantic  
  logfire SDK to instrument the agent and export traces.13 While this provides a functional solution, it introduces an external dependency that is not controlled by OpenAI.13

The nature of this OTEL support—whether it is a first-party, native feature or a third-party add-on—is a critical factor for long-term enterprise strategy. First-party support, as seen in LangGraph and Autogen, signifies that the framework's maintainers consider observability a core requirement. This support is more likely to be maintained, updated with new features, and remain compatible through framework upgrades. Relying on a third-party wrapper, as required by the OpenAI Agents SDK, introduces a significant dependency risk. The enterprise becomes reliant on the third-party maintainer to keep the wrapper compatible with any changes in the core SDK. This potential for lag or abandonment of the wrapper is an unacceptable risk for many mission-critical production systems. Given the strict OTEL mandate, LangGraph and Autogen are the clear leaders. LangGraph's integration is particularly noteworthy for its depth and flexibility, making it the top choice on this criterion.

## **API Integration and Enterprise Constraints**

### **Validating Against the OpenAI API Constraint**

A final validation is necessary to ensure that each framework can operate strictly within the enterprise's technological constraints: access is limited to the OpenAI Chat Completion and Embedding APIs only. Any framework requiring other specific endpoints, such as the OpenAI Assistants API, would be disqualified.

### **Framework Compatibility**

* **Autogen:** The framework is fully compatible. Its documentation explicitly demonstrates how to configure and use the OpenAIChatCompletionClient and AzureOpenAIChatCompletionClient, which are built to interact with the standard Chat Completion API endpoint.62
* **LangGraph/LangChain:** The framework is fully compatible. The standard ChatOpenAI integration, which is the primary method for using OpenAI models, is fundamentally built upon the Chat Completion API.64
* **Semantic Kernel:** The framework is fully compatible. Its documentation shows how to add OpenAIChatCompletion and AzureChatCompletion as services to the kernel, confirming its use of the standard API.66
* **OpenAI Agents SDK:** The framework is compatible, but with an important nuance. The SDK was launched alongside OpenAI's new "Responses API," which bundles tool use with model calls.26 However, the SDK's documentation confirms that it can be explicitly configured to fall back to using the standard "Chat Completions API" via the  
  set\_default\_openai\_api("chat\_completions") function.67 This compatibility mode is a critical finding that keeps the SDK as a viable candidate.

While all four frameworks are compatible with the current constraints, the OpenAI Agents SDK's close relationship with the newer Responses API presents a potential future-proofing risk. OpenAI's strategic direction is to encourage the use of this new, more integrated API.26 It is plausible that future, more advanced features of the Agents SDK could be optimized for, or exclusively available through, the Responses API. If the enterprise's restrictive API access policies remain in place, it might be cut off from these future advancements. The other three frameworks, being model-agnostic by design, are built on the most common industry standard—the Chat Completion API—and are therefore more decoupled from OpenAI-specific API strategies. This makes them a safer long-term choice against this particular risk.

## **Final Recommendations and Decision Matrix**

### **Synthesis of Findings**

The analysis reveals a clear landscape of trade-offs. Autogen offers unparalleled flexibility for multi-agent research but lacks production-readiness in its tooling and a strong abstraction for deterministic workflows. Semantic Kernel provides a robust, enterprise-focused architecture, but its Python implementation is less mature than its.NET counterpart and its out-of-the-box features for state and observability are still evolving. The OpenAI Agents SDK is modern, lightweight, and simple to use but creates significant enterprise risk through its reliance on a third-party wrapper for mandatory OTEL integration and its potential future dependency on APIs beyond the standard Chat Completion endpoint.

LangGraph consistently emerges as the most aligned framework. It is explicitly production-ready, backed by a strong commercial entity and a massive open-source community. Its core StateGraph abstraction is uniquely suited to building both deterministic workflows and complex, cyclical agents. Most importantly, it provides native, first-class support for the three most critical enterprise capabilities identified: durable state management, integrated human-in-the-loop validation, and a comprehensive observability solution via LangSmith and native OTEL support.

### **Final Decision Matrix**

The following matrix provides a scored summary of the analysis, with each framework rated from 1 (Poor) to 5 (Excellent) against the key enterprise criteria derived from the user query.

| Criterion | Autogen | LangGraph | Semantic Kernel | OpenAI Agents SDK |
| --- | --- | --- | --- | --- |
| **Stability & Maturity** | 2 | 5 | 4 | 3 |
| **Community & Docs** | 4 | 5 | 4 | 3 |
| **Workflow Flexibility** | 3 | 5 | 5 | 4 |
| **Agentic Flexibility** | 5 | 4 | 4 | 4 |
| **State Management** | 2 | 5 | 3 | 4 |
| **Human-in-the-Loop** | 3 | 5 | 3 | 4 |
| **Replay & Debugging** | 2 | 5 | 2 | 4 |
| **OTEL Integration (First-Party)** | 4 | 5 | 2 | 1 |
| **API Decoupling (Future-Proofing)** | 5 | 5 | 5 | 2 |
| **Total Score** | **30** | **43** | **32** | **29** |

### **Concluding Recommendation**

Based on a comprehensive evaluation against all specified requirements and constraints, **LangGraph is the recommended framework for adoption within the enterprise.**

It excels in every critical area:

1. **Production Readiness:** It is stable, versioned, and trusted in production by other major technology companies.1
2. **Architectural Fit:** Its StateGraph model provides a superior, unified abstraction for building the required mix of deterministic workflows and autonomous agents.9
3. **Enterprise Features:** It offers the strongest native, out-of-the-box support for the non-negotiable capabilities of durable state persistence, human-in-the-loop intervention, and workflow replay/debugging.1
4. **Observability Mandate:** It meets and exceeds the OpenTelemetry requirement with a mature, flexible, and first-party integration that mitigates dependency risk.2

While other frameworks have their strengths, they each present a disqualifying weakness for this specific use case. Autogen is too research-focused, the OpenAI Agents SDK introduces unacceptable dependency risks for observability, and Semantic Kernel's Python ecosystem is not yet mature enough to compete with LangGraph's feature set. LangGraph provides the most robust, flexible, and enterprise-ready foundation for building the next generation of AI applications.

#### Works cited

1. LangGraph, accessed June 20, 2025, <https://langchain-ai.github.io/langgraph/>
2. Trace LangChain with OpenTelemetry | 🦜️🛠️ LangSmith, accessed June 20, 2025, <https://docs.smith.langchain.com/observability/how_to_guides/trace_langchain_with_otel>
3. Add human-in-the-loop, accessed June 20, 2025, <https://langchain-ai.github.io/langgraph/tutorials/get-started/4-human-in-the-loop/>
4. Microsoft's Agentic Frameworks – AutoGen and Semantic Kernel - Bravent, accessed June 20, 2025, <https://www.bravent.net/en/news/revolutionizing-ai-development-microsofts-agentic-frameworks-autogen-and-semantic-kernel/>
5. A Feature-By-Feature Semantic Kernel vs Langchain Comparison - Lamatic.ai Labs, accessed June 20, 2025, <https://blog.lamatic.ai/guides/semantic-kernel-vs-langchain/>
6. Understanding the Experimental Attribute in Semantic Kernel ..., accessed June 20, 2025, <https://learn.microsoft.com/en-us/semantic-kernel/concepts/experimental-attribute>
7. How-To - Human-in-the-Loop | Microsoft Learn, accessed June 20, 2025, <https://learn.microsoft.com/en-us/semantic-kernel/frameworks/process/examples/example-human-in-loop>
8. Introduction to AutoGen | AutoGen 0.2 - Microsoft Open Source, accessed June 20, 2025, <https://microsoft.github.io/autogen/0.2/docs/tutorial/introduction/>
9. AI Agent Frameworks: Choosing the Right Foundation for Your ... - IBM, accessed June 20, 2025, <https://www.ibm.com/think/insights/top-ai-agent-frameworks>
10. AutoGen Studio - Microsoft Open Source, accessed June 20, 2025, <https://microsoft.github.io/autogen/dev//user-guide/autogenstudio-user-guide/index.html>
11. OpenAI Agents SDK TypeScript - GitHub Pages, accessed June 20, 2025, <https://openai.github.io/openai-agents-js/>
12. OpenAI Agents SDK, accessed June 20, 2025, <https://openai.github.io/openai-agents-python/>
13. Monitor OpenAI Agents with Azure Application Insights, accessed June 20, 2025, <https://techcommunity.microsoft.com/blog/azure-ai-services-blog/monitor-openai-agents-sdk-with-application-insights/4393949>
14. Autogen vs Langchain: Comprehensive Framework Comparison | Generative AI Collaboration Platform, accessed June 20, 2025, <https://orq.ai/blog/autogen-vs-langchain>
15. LangChain vs AutoGen vs Haystack | Best LLM Framework - CrossML, accessed June 20, 2025, <https://www.crossml.com/langchain-vs-autogen-vs-haystack/>
16. awesome-opensource-ai-engineering/libraries/autogen/README.md at main - GitHub, accessed June 20, 2025, <https://github.com/aipengineer/awesome-opensource-ai-engineering/blob/main/libraries/autogen/README.md>
17. About - LangChain, accessed June 20, 2025, <https://www.langchain.com/about>
18. LangGraph: Build Stateful AI Agents in Python – Real Python, accessed June 20, 2025, <https://realpython.com/langgraph-python/>
19. langchain-ai/langchain: Build context-aware reasoning ... - GitHub, accessed June 20, 2025, <https://github.com/langchain-ai/langchain>
20. OpenAI Agents SDK vs LangGraph vs Autogen vs CrewAI - Composio, accessed June 20, 2025, <https://composio.dev/blog/openai-agents-sdk-vs-langgraph-vs-autogen-vs-crewai/>
21. Semantic Kernel documentation | Microsoft Learn, accessed June 20, 2025, <https://learn.microsoft.com/en-us/semantic-kernel/>
22. Semantic Kernel Agent Architecture | Microsoft Learn, accessed June 20, 2025, <https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/agent-architecture>
23. Orchestrating Multi‑Agent AI With Semantic Kernel | Digital Bricks, accessed June 20, 2025, <https://www.digitalbricks.ai/blog-posts/orchestrating-multi-agent-ai-with-semantic-kernel>
24. Understanding the kernel in Semantic Kernel | Microsoft Learn #679 - GitHub, accessed June 20, 2025, <https://github.com/irthomasthomas/undecidability/issues/679>
25. Semantic Kernel Agent Framework | Microsoft Learn, accessed June 20, 2025, <https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/>
26. New tools for building agents | OpenAI, accessed June 20, 2025, <https://openai.com/index/new-tools-for-building-agents/>
27. openai/openai-agents-python: A lightweight, powerful ... - GitHub, accessed June 20, 2025, <https://github.com/openai/openai-agents-python>
28. AutoGen - vLLM, accessed June 20, 2025, <https://docs.vllm.ai/en/latest/deployment/frameworks/autogen.html>
29. microsoft/semantic-kernel: Integrate cutting-edge LLM ... - GitHub, accessed June 20, 2025, <https://github.com/microsoft/semantic-kernel>
30. AutoGen - Microsoft Research, accessed June 20, 2025, <https://www.microsoft.com/en-us/research/project/autogen/>
31. Multi-Agent Portfolio Collaboration with OpenAI Agents SDK, accessed June 20, 2025, <https://cookbook.openai.com/examples/agents_sdk/multi-agent-portfolio-collaboration/multi_agent_portfolio_collaboration>
32. Discussions - microsoft autogen - GitHub, accessed June 20, 2025, <https://github.com/microsoft/autogen/discussions>
33. Blog | AutoGen 0.2 - Microsoft Open Source, accessed June 20, 2025, <https://microsoft.github.io/autogen/0.2/blog/>
34. Documentation | AutoGen 0.2 - Microsoft Open Source, accessed June 20, 2025, <https://microsoft.github.io/autogen/0.2/docs/contributor-guide/documentation/>
35. LangChain Community Forum, accessed June 20, 2025, <https://forum.langchain.com/login>
36. Join the LangChain Community, accessed June 20, 2025, <https://www.langchain.com/join-community>
37. The LangChain Community, accessed June 20, 2025, <https://www.langchain.com/community>
38. LangChain Community Forum, accessed June 20, 2025, <https://forum.langchain.com/>
39. Reference - GitHub Pages, accessed June 20, 2025, <https://langchain-ai.github.io/langgraph/reference/>
40. Semantic Kernel - Discord, accessed June 20, 2025, <https://discord.com/invite/semantic-kernel-1063152441819942922>
41. How to contribute to Semantic Kernel | Microsoft Learn, accessed June 20, 2025, <https://learn.microsoft.com/en-us/semantic-kernel/support/contributing>
42. Agents SDK with Azure hosted models - API - OpenAI Developer Community, accessed June 20, 2025, <https://community.openai.com/t/agents-sdk-with-azure-hosted-models/1157781>
43. We need typescript agent sdk - API - OpenAI Developer Community, accessed June 20, 2025, <https://community.openai.com/t/we-need-typescript-agent-sdk/1144653>
44. How to think about agent frameworks - LangChain Blog, accessed June 20, 2025, <https://blog.langchain.dev/how-to-think-about-agent-frameworks/>
45. Workflows & agents, accessed June 20, 2025, <https://langchain-ai.github.io/langgraph/tutorials/workflows/>
46. Workflows and Agents, accessed June 20, 2025, <https://langchain-ai.github.io/langgraphjs/tutorials/workflows/>
47. gbbai-agent-architecture-lab/labs/02-frameworks/intro-to-autogen-v2.ipynb at main - GitHub, accessed June 20, 2025, <https://github.com/pablosalvador10/gbbai-agent-architecture-lab/blob/main/labs/02-frameworks/intro-to-autogen-v2.ipynb>
48. A practical guide to building agents - OpenAI, accessed June 20, 2025, <https://cdn.openai.com/business-guides-and-resources/a-practical-guide-to-building-agents.pdf>
49. Scanning OpenAI Agents SDK Workflows with Agentic Radar ..., accessed June 20, 2025, <https://splx.ai/blog/openai-agents-sdk-transparent-workflows-with-agentic-radar>
50. How to handle the Human in the loop for concurrent agents and topic-subscription based scenarios - Learn Microsoft, accessed June 20, 2025, <https://learn.microsoft.com/en-us/answers/questions/2168187/how-to-handle-the-human-in-the-loop-for-concurrent>
51. Complete Guide to Building LangChain Agents with the LangGraph Framework - Zep, accessed June 20, 2025, <https://www.getzep.com/ai-agents/langchain-agents-langgraph>
52. Human in the loop | OpenAI Agents SDK - GitHub Pages, accessed June 20, 2025, <https://openai.github.io/openai-agents-js/guides/human-in-the-loop/>
53. Allowing Human Feedback in Agents | AutoGen 0.2, accessed June 20, 2025, <https://microsoft.github.io/autogen/0.2/docs/tutorial/human-in-the-loop/>
54. Instrumentating your code locally — AutoGen - Microsoft Open Source, accessed June 20, 2025, <https://microsoft.github.io/autogen/stable//user-guide/core-user-guide/cookbook/instrumenting.html>
55. More advanced scenarios for telemetry | Microsoft Learn, accessed June 20, 2025, <https://learn.microsoft.com/en-us/semantic-kernel/concepts/enterprise-readiness/observability/telemetry-advanced>
56. Autogen | Opik Documentation, accessed June 20, 2025, <https://www.comet.com/docs/opik/tracing/integrations/autogen?utm_source=newsletter&utm_medium=email&utm_campaign=newsletter_2025>
57. Instrumentating your code locally — AutoGen - Microsoft Open Source, accessed June 20, 2025, <https://microsoft.github.io/autogen/stable/user-guide/core-user-guide/cookbook/instrumenting.html>
58. Instrument a LangGraph ReAct Agent with OpenTelemetry | Google Cloud Observability, accessed June 20, 2025, <https://cloud.google.com/stackdriver/docs/instrumentation/ai-agent-langgraph>
59. AI Agent Observability - Evolving Standards and Best Practices ..., accessed June 20, 2025, <https://opentelemetry.io/blog/2025/ai-agent-observability/>
60. Observability for Semantic Kernel with Langfuse Integration, accessed June 20, 2025, <https://langfuse.com/docs/integrations/semantic-kernel>
61. Evaluating Agents with Langfuse | OpenAI Cookbook, accessed June 20, 2025, <https://cookbook.openai.com/examples/agents_sdk/evaluate_agents>
62. Model Clients — AutoGen - Microsoft Open Source, accessed June 20, 2025, <https://microsoft.github.io/autogen/0.4.7/user-guide/core-user-guide/components/model-clients.html>
63. Model Clients — AutoGen - Microsoft Open Source, accessed June 20, 2025, <https://microsoft.github.io/autogen/stable//user-guide/core-user-guide/components/model-clients.html>
64. ChatOpenAI - LangChain.js, accessed June 20, 2025, <https://js.langchain.com/docs/integrations/chat/openai/>
65. ChatOpenAI — LangChain documentation, accessed June 20, 2025, <https://python.langchain.com/api_reference/openai/chat_models/langchain_openai.chat_models.base.ChatOpenAI.html>
66. Exploring the Semantic Kernel ChatCompletionAgent - Learn Microsoft, accessed June 20, 2025, <https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/agent-types/chat-completion-agent>
67. Configuring the SDK - OpenAI Agents SDK, accessed June 20, 2025, <https://openai.github.io/openai-agents-python/config/>