

STUDY · AI AGENTS

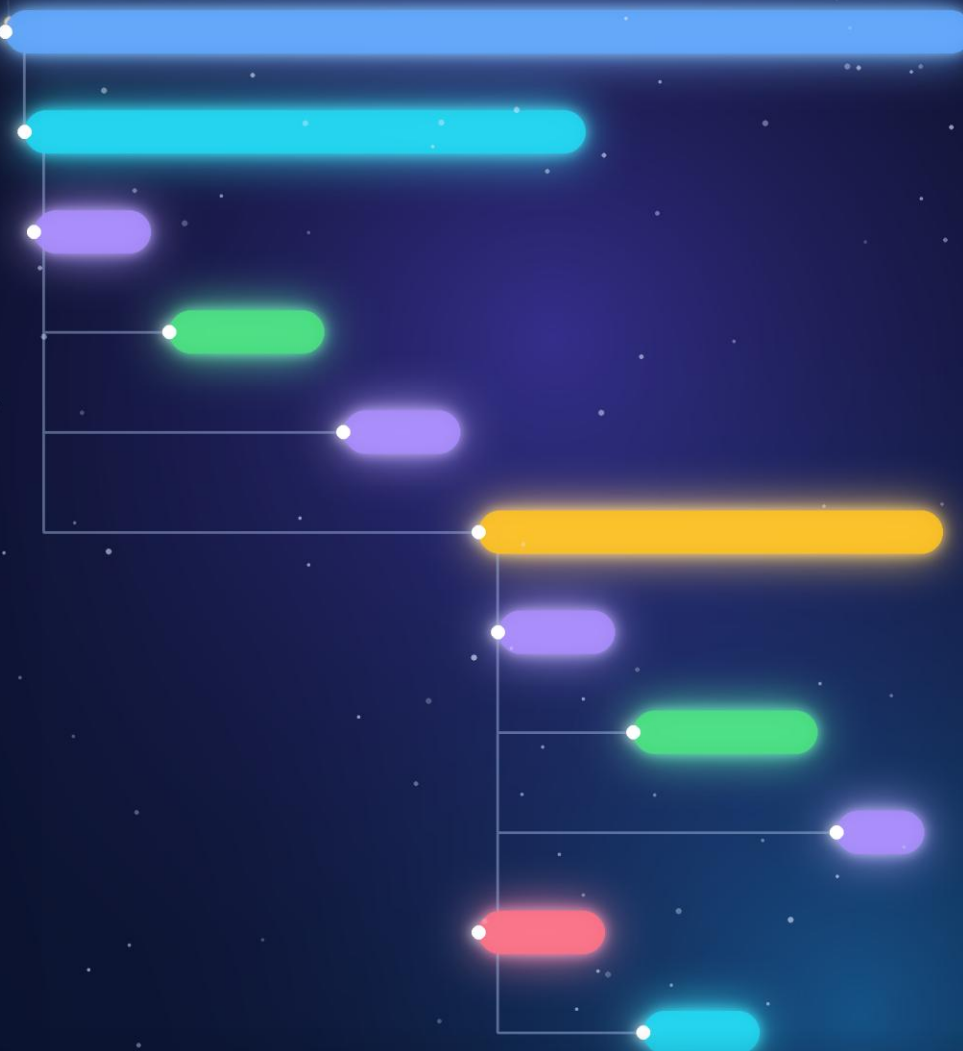
AI Agent Execution Tracing

Fundamentals, state of the art and a comparison of 13 platforms

Aridio Silva @aridiosilva

Independent Researcher · ORCID 0009-0008-2411-6995

October 2026





Summary: what the study concludes

- **Traceable, but no stable standard.** All major platforms already let you trace every step of an agent; none offers a stable standard.
- **A trace is not proof of success.** It shows what the agent did, not whether the task worked: a separate evaluation is needed.
- **Maturity in progress.** Several features are in beta or preview; the OTel GenAI conventions are still in Development.
- **Privacy varies.** Prompts and tools are captured by default on some platforms and not on others.

13

platforms compared: 5 vendors and 7 independents

47

references with validation [A], [B] or [C]

Oct 10, 2026

survey date; versions read from PyPI and npm

1

Why trace AI agents

- **What the operator needs to know:** whether the task was completed, where it failed or deviated, how much time and cost it consumed, and which steps it ran.
- **Non-deterministic behavior:** agents decide at run time which tools to call and whom to delegate to.
- **Linear logs are not enough:** the answer is to adapt distributed tracing to the agent loop.
- **Who provides the record:** the layer that runs the agent (SDK, framework or platform), not the model alone.

Why it matters: Without tracing, you cannot debug where the agent failed, measure latency and cost per step, or audit what was done.

QP1

How do trace and span apply to the execution of one or more agents?

QP2

How to answer completion, error, deviation, duration and steps?

QP3

Which platforms offer this, with what maturity and gaps?

Trace, span and session

- **Span:** a traced segment of execution; it records a specific step of the agent's work.
- **Trace:** the complete journey of an operation, made up of spans.
- **What a span carries:** start and end (duration), parent span (hierarchy) and, where supported, status and events.
- **Session:** groups several traces from the same conversation or interaction.

Why it matters: individual spans let you measure time, tokens and cost for each step and understand the agent's decision flow.

Most common span types

Model call

time, tokens and, depending on the platform, the prompt sent

Tool use

which tool, the parameters and the result

Handoff

transfer of control to another agent

Three nuances when reading a trace

1

A trace is not always the whole task

- OpenAI Agents SDK: the trace is an end-to-end operation; group_id groups runs
- AgentCore: the trace is a single request-response cycle; the session groups several traces

2

A handoff is not always a span type

- OpenAI has a dedicated handoff span
- In OTel, the subagent appears as another nested invoke_agent
- Microsoft added a span for agent-to-agent communication

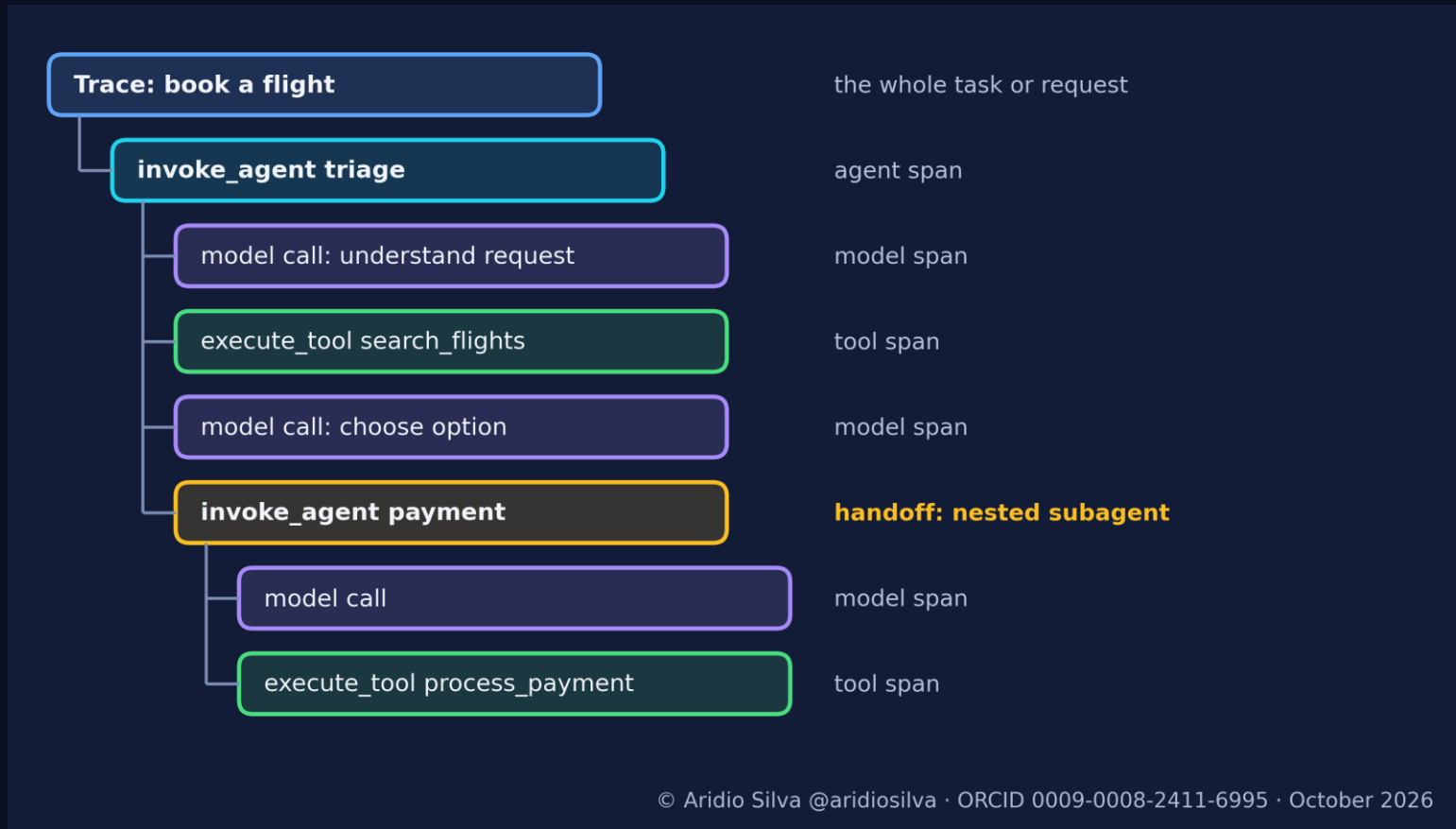
3

There are more spans than model, tool and handoff

- There are guardrail and turn spans
- The Claude Agent SDK has a child span for the time a tool waited for user approval

Why it matters: each platform models trace, session and handoff differently; check the data model before comparing numbers.

Span tree: booking a flight



- **A tree, not a list:** each step is a span nested inside the trace.
- **In OTEL**, the handoff appears as a nested subagent (invoke_agent).
- **In the OpenAI Agents SDK**, the handoff has its own span.

Why it matters: lets you debug where it failed and measure latency and cost per step.

Figure: illustrative example from the paper.

Six observability questions

Question	Signal to watch
Did it complete the task?	Final run status and, separately, verification of the result (tests, schema, evaluator or human review)
Were there errors or deviations?	Spans with errors, tool failures, exceeded limits, denied permissions and triggered guardrails
When did it finish and how long did it take?	Run and per-span duration; tokens and cost
What did it do at each step?	Hierarchical trace: tool, arguments and result
Who asked and what was authorized?	End-user attributes and permission-decision events (audit and SIEM)
Was there a loop or waste?	Number of turns and calls per run

Done? A trace does not prove success

- **Final status:** tells how the run ended, not whether the goal was reached.
- **Claude Agent SDK:** the result carries subtype, is_error, turns, duration and cost (success, error_max_turns, error_during_execution; third-party source).
- **Content failures:** a wrong answer delivered with confidence often triggers no error status.
- **OpenAI:** trace grading scores traces to find errors at scale.
- **Deviations:** show up in permission events and guardrails; in Claude Code each decision is an event with its source.

● Importance

- Separate “finished without error” (trace) from “correct result” (evaluation)

● Caution

- A run without exceptions only proves the agent did not crash

Methodology and validation

Package registries

Stable version and release date read via API from PyPI and npm on Oct 10, 2026.

Official documentation

Prioritized whenever available, for each vendor and platform.

Secondary sources

Only where official coverage was missing; marked “third-party”. Competitor comparisons used with caution.

Validation marks in the bibliography

[A] page fully opened

[B] excerpt checked against the claim

[C] data obtained via registry API

- **Reassigned during validation:** claims moved to more precise sources (R17, R43 to R47); Logfire's evaluation became ①.
- **Limitations:** SaaS has no single version (the column shows the SDK); prices excluded; Weave, Opik and AgentOps not covered in depth; beta and GA reflect the query date.

OpenTelemetry GenAI: the open standard

- **Status Development:** the GenAI conventions are the candidate for a common language, but names can still change.
- **Jun 12, 2026:** in v1.42.0, they were moved to the semantic-conventions-genai repository.
- **Operations:** create_agent, invoke_agent and execute_tool; the agent span name is recommended to be invoke_agent + agent name.
- **Metrics:** run duration, number of tool calls and inference calls per run.
- **Who already adopts it:** Datadog accepts OTel GenAI 1.37+; MLflow ingests and exports; Google ADK implements it natively.

● Positive implication (+)

- Portability across vendors and backends

● Negative implication (–)

- Names can change between versions; Claude emits its own spans with the claude_code prefix.

Anthropic (Claude)

Version: py 0.2.165 · npm 0.3.296 · CLI 2.1.296

State: Traces in beta; 0.x SDK

● Features

- Agent SDK: spans per interaction, model and tool, metrics and events; nested subagents
- W3C context propagated between your app and the agent
- Managed Agents: tracing in the Console and session, span and agent events via API

● Weaknesses and limits (–)

- Traces in beta: span names may change
- Export failures are silent by default
- Hook spans need extra beta config; Managed Agents needs a beta header

● Strengths (+)

- Open OTLP, no mandatory backend
- Content is not recorded by default, only the structure
- Permission events per end user feed a SIEM

● What it does not do

- The SDK does not generate its own telemetry or have an analysis UI
- No native evaluators over traces in the documents consulted

OpenAI (Agents SDK)

Version: py 0.23.1 · JS 0.20.0

State: Tracing on by default; 0.x SDK

● Features

- Tracing on by default: run, task, turn, agent, generation, tool, guardrail, handoff and audio
- group_id links traces of the same conversation
- Dashboard with trace grading and evals; about 30 external processors

● Strengths (+)

- Zero configuration and rich hierarchy for handoffs and guardrails
- Free tracing even with other vendors' models

● Weaknesses and limits (–)

- Unavailable under Zero Data Retention
- Inputs and outputs captured by default
- Redaction in a separate processor does not stop the default exporter if it fails; batched export

● What it does not do

- Own native format: the bridge to OTel depends on third-party instrumentors

Google (ADK and Agent Engine)

Version: py 2.11.0 · JS 2.2.1

State: Major 2.x; native telemetry

● Features

- ADK implements the OTel GenAI conventions: agent as root span, children for LLM and tools
- Context propagated across processes
- Agent Engine exports to Cloud Trace; Python, Go and Kotlin

● Strengths (+)

- Native OTel
- Managed backend with a free tier

● Weaknesses and limits (–)

- Cloud Logging drops entries that exceed the maximum log size

● What it does not do

- Cloud Trace is a trace viewer
- Continuous evaluation and agent-specific dashboards did not appear in the documents consulted

Microsoft (Foundry)

Version: py 1.21.0 (azure-ai-projects 2.8.0)

State: GA: prompt and hosted · preview: workflows and external

● Features

- Tracing GA for prompt and hosted agents; preview for workflows and external agents
- Storage in Application Insights
- Multi-agent OTel conventions (with Outshift, from Cisco); Agent Framework emits `invoke_agent` spans

● Strengths (+)

- The most advanced in multi-agent semantics
- Integrated with Azure Monitor

● Weaknesses and limits (–)

- Cost depends on Application Insights volume and retention
- Requires the Log Analytics Reader role
- Status differs across pages: the hosted-agent quickstart still says preview

● What it does not do

- Outside Azure, you must export via OTel to another backend

6.5

AWS (Bedrock AgentCore)

Version: py 1.24.1

State: Evolving (unified log on Jul 23, 2026)

● Features

- Session, trace and span; traces with tools (input, output and time) and errors
- Since Jul 23, 2026: telemetry in one log group per agent, with per-agent IAM and CMK
- Default for agents created since Jul 20, 2026

● Strengths (+)

- Data stays in the customer's account, with native AWS governance
- Also covers agents hosted outside AgentCore

● Weaknesses and limits (–)

- Agent spans require ADOT instrumentation; by default only memory emits spans
- Requires enabling CloudWatch Transaction Search

● What it does not do

- No built-in quality evaluation in the documents consulted

Version: py 0.14.7 · JS 0.10.11

State: Commercial SaaS; 0.x SDKs

● Features

- Runs are spans, in projects; threads via metadata; analysis assistant (Polly)
- Ingests OTel; distributed tracing, sampling and sensitive-data prevention
- Cost per token, dashboards and online evaluation

● Positive implication (+)

- Online evaluation and dashboards in the same product
- 400-day retention on SaaS

● Weaknesses and limits (–)

- Run limit per trace: excess runs are rejected
- OpenAI Agents SDK integration is Python only

● License and hosting

- SaaS with EU and APAC regions
- Self-host restricted to the enterprise plan (per third parties)

Version: py 4.17.0 · JS @langfuse/tracing 5.13.1

State: Platform v4 (OTLP-first)

● Features

- Nested observations, sessions, agent graph and human annotation queues
- v4 prioritizes OTLP ingestion; runs on ClickHouse

● Positive implication (+)

- MIT core with self-host on any plan
- Built-in evaluation with human annotation

● Weaknesses and limits (–)

- Open source removes the license, not the cost of running it (third-party)
- Third-party reports of breaking changes migrating the SDK from v3 to v4

● License and hosting

- Runs in the cloud or self-hosted, with an MIT-licensed core

Arize Phoenix and AX

Version: Phoenix 20.20.0

State: Frequent releases; ELv2 license

● Features

- OTEL + OpenInference: span kinds LLM, Tool, Agent and Retriever
- Evals, datasets, experiments and prompt management
- Instrumentors for OpenAI Agents, LangChain and DSPy

● Positive implication (+)

- Open package with built-in evaluation
- AX adds a proprietary datastore and online evaluations (third-party)

● Weaknesses and limits (–)

- Elastic 2.0 license restricts offering the software as a managed service (third-party)
- The TypeScript evals library is in alpha

● License and hosting

- Phoenix: source-available under Elastic 2.0
- AX: managed service

Datadog (Agent Observability)

Version: ddtrace 4.15.6

State: SaaS

● Features

- Accepts OTel GenAI 1.37 or later
- Span links feed an Execution Graph for multi-agent pipelines
- Auto-instrumentation for OpenAI Agents SDK, LangGraph and CrewAI

● Positive implication (+)

- Shows how one span's output feeds another in multi-agent pipelines
- Tracks error rate, latency, cost and agent decisions

● Weaknesses and limits (–)

- Links to spans of another trace are stored but not drawn
- Product not supported on gov sites

● License and hosting

- Offered only as SaaS

7.5

Braintrust

Version: py 0.45.0 · JS 3.37.2

State: SaaS and hybrid self-host

● Features

- Offline and online evaluation over real logs
- Integrations with Claude Agent SDK, OpenAI Agents and Google ADK
- Data masking

● Positive implication (+)

- Evaluation and tracing in the same workflow
- Integrates with several agent SDKs

● Weaknesses and limits (–)

- Hybrid self-host: the docs describe telemetry sent back to the vendor's control plane
- Cost grows with scoring usage (third-party)

● License and hosting

- Hosting: SaaS or hybrid, with the data plane in the customer's environment

7.6

MLflow Tracing

Version: 3.17.0

State: Open source

● Features

- OTel-compatible: ingests and exports GenAI conventions
- 30+ integrations via autolog
- Lean production SDK (mlflow-tracing)

● Positive implication (+)

- Portability: OTel end to end
- One call enables tracing for each supported framework

● Weaknesses and limits (–)

- You must call autolog explicitly for each integration

● License and hosting

- Hosting: open source or managed service

Pydantic Logfire

Version: 5.1.1

State: Commercial product

● Features

- OTel-based layer with Python and JavaScript SDKs
- Integration with Pydantic AI and an AI gateway

● Positive implication (+)

- Any application with OTel instrumentation can send data

● Weaknesses and limits (–)

- Evals and prompt management appear only in search-mirrored docs, without official confirmation (🕒 mark)
- Commercial product

● License and hosting

- Hosting: SaaS; self-host only on the Enterprise plan

Versions and maturity (1/2)

Platform	Current version	Released	Maturity state
Claude Agent SDK / Claude Code	py 0.2.165 · npm 0.3.296 · CLI 2.1.296	Oct 8–9, 2026	Traces and telemetry in beta; 0.x numbering
Claude Managed Agents	API, no package	—	Beta (header managed-agents-2026-04-01)
OpenAI Agents SDK	py 0.23.1 · JS 0.20.0	Oct 2 and Oct 8, 2026	Tracing on by default; 0.x numbering
Google ADK	py 2.11.0 · JS 2.2.1	Oct 2 and Oct 6, 2026	Major 2.x; native telemetry
Microsoft Agent Framework	py 1.21.0 (azure-ai-projects 2.8.0)	Oct 8 and Oct 5, 2026	GA for prompt and hosted agents; preview for workflows and external
AWS Bedrock AgentCore	py 1.24.1	Oct 7, 2026	Evolving (unified log per agent on Jul 23, 2026)
LangSmith	py 0.14.7 · JS 0.10.11	Oct 9, 2026	Commercial SaaS; 0.x SDKs

Version = latest stable on Oct 10, 2026. A 0.x number means the SDK is pre-1.0.

Versions and maturity (2/2)

Platform	Current version	Released	Maturity state
Langfuse	py 4.17.0 · JS @langfuse/tracing 5.13.1	Oct 5 and Oct 7, 2026	Platform v4 (OTLP-first)
Arize Phoenix	20.20.0	Oct 9, 2026	Frequent releases; ELv2 license
Datadog	ddtrace 4.15.6	Oct 8, 2026	SaaS
Braintrust	py 0.45.0 · JS 3.37.2	Oct 7 and Oct 9, 2026	SaaS and hybrid
MLflow Tracing	3.17.0	Oct 7, 2026	Open source
Pydantic Logfire	5.1.1	Sep 25, 2026	Commercial product
OpenTelemetry (Python SDK)	1.45.1	Oct 6, 2026	GenAI conventions in Development

SaaS has no single version: the column shows the SDK.

Feature matrix (1/2)

Platform	Steps (LLM and tool)	Multi-agent	OTel	Sessions and threads	Evaluation on traces	Hosting
Claude Agent SDK	✓ (beta)	✓ subagents	✓ OTLP	✓ session.id	—	Backend of your choice
Claude Managed Agents	✓ (Console)	—	—	✓	—	Managed
OpenAI Agents SDK	✓	✓ handoffs	⚠ via third parties	✓ group_id	✓ trace grading	OpenAI dashboard
Google ADK and Agent Engine	✓	⚠	✓ native	—	—	Cloud Trace
Microsoft Foundry and Agent Framework	✓	✓ own semantics	✓	—	⚠	Application Insights
AWS AgentCore	✓ (with ADOT)	✓	✓ ADOT	✓	—	CloudWatch
LangSmith	✓	✓	✓	✓ threads	✓	SaaS; enterprise

✓ confirmed in the documentation · ⚠ partial or conditional · — not found in the documents consulted (does not prove absence)

Feature matrix (2/2)

Platform	Steps (LLM and tool)	Multi-agent	OTel	Sessions and threads	Evaluation on traces	Hosting
Langfuse	✓	✓ graph	✓	✓	✓	SaaS; MIT
Phoenix and AX	✓	✓	✓	—	✓	ELv2; SaaS
Datadog	✓	✓ graph	✓	—	🟡 experiments	SaaS
Braintrust	✓	✓	✓	—	✓ online and offline	SaaS; hybrid
MLflow	✓	✓	✓	—	—	OSS or managed
Logfire	✓	✓	✓	—	🟡	SaaS; enterprise

✓ confirmed in the documentation · 🟡 partial or conditional · — not found in the documents consulted (does not prove absence)

Gaps and risks

● Incomplete convergence

Everyone is moving to OTel, but with dialects: Claude (claude_code.*), OpenAI (own format), Google and Microsoft (GenAI), AWS (ADOT).

● Uneven maturity

Claude traces, Foundry workflow tracing and the GenAI conventions are in beta, preview or Development: plan for migration.

● Privacy and content

The Claude Agent SDK does not record content by default; the OpenAI Agents SDK records inputs and outputs and does not work under ZDR.

● Silent failures

On at least one platform, an export failure produces no visible error: a missing trace may be an export failure.

● A trace is not correctness

Evaluation on traces: OpenAI, LangSmith, Langfuse, Phoenix and Braintrust (Logfire, partial). The others require integrating an evaluator.

Why it matters

Long-term decisions should plan for migration of span names and attributes and define, before production, what the trace may contain.

9

Best practices

1

Instrument with OTel whenever possible: to keep portability across vendors and backends.

2

Define in writing what will be captured: content, arguments and results, and who may read them.

3

Monitor the telemetry export itself: with an alert when span volume drops.

4

Keep two metrics apart: “finished without error” (trace signal) and “correct result” (evaluation signal).

5

Group traces by session or conversation: on long tasks or tasks with several interactions.

We are midway through maturation: preliminary solutions, many in beta or preview

- **Feasible today:** tracing every agent step works on all platforms analyzed.
- **Beta:** Claude Agent SDK traces and Managed Agents.
- **Preview:** Foundry workflows and external agents.
- **Development:** OTel GenAI conventions. 0.x SDKs: Claude Agent SDK, OpenAI Agents SDK, LangSmith and Braintrust.
- **Migration cost:** span names, attributes and behaviors can change between versions.

QP1

The trace-and-span model of distributed tracing applies directly, with naming variations.

QP2

Completion, errors, duration and steps come from the trace; result correctness requires independent evaluation.

QP3

A broad ecosystem, but in transition: beta, preview and conventions still in Development.

Glossary (1/5)

● ADOT

AWS Distro for OpenTelemetry: AWS's OTel distribution for instrumenting agents and sending traces to CloudWatch.

● AI agent

A system that uses a language model to decide at run time which tools to call and how to complete a task.

● Autolog

Enabling automatic instrumentation with a single call (an MLflow term).

● Beta, preview, GA, Development

Maturity stages: beta and preview still change; GA is general availability; Development is the early stage of the OTel conventions.

● CMK

Customer-managed key: an encryption key managed by the customer.

● Context propagation

Passing the trace ID across processes so spans from different components form a single trace (W3C Trace Context).

● Distributed tracing

A technique for recording a request's path across several components.

● Elv2

Elastic License 2.0: source-available, broad use, but no offering the software as a managed service.

● Evaluation (eval)

Measuring the quality of an agent's result, with scores from code, an evaluator model or people.

● Event

A structured record of an occurrence (prompt, API error, permission decision), distinct from metrics and spans.

● Guardrail

A check that validates or blocks an agent's inputs or outputs.

● Handoff

Transfer of control from one agent to another.

● Hook

Code run at a defined point in the agent loop, such as before a tool is used.

● Instrumentation

Code or a library that generates telemetry, manually or automatically.

● Metric

A numeric measure aggregated over time (tokens, cost, latency).

● MIT

A permissive open-source license.

● Observability

The ability to understand a system's internal state from the telemetry it emits.

● Observability backend

A system that receives, stores and lets you query traces, metrics and logs.

● OpenInference

A layer of conventions on top of OTel, created by Arize, with span kinds such as LLM, Tool, Agent and Retriever.

● OpenTelemetry (OTel)

An open standard for generating and exporting telemetry: traces, metrics and logs.

● OTLP

The OpenTelemetry protocol for transporting telemetry.

● Sampling

Recording only a fraction of traces to reduce volume and cost.

● Self-host

Hosting the software on the user's own infrastructure.

● Semantic conventions

Standardized names and attributes for spans, metrics and events (e.g., `gen_ai.agent.name`).

● Semantic failure

A content error: the agent finishes without an exception but delivers a wrong answer.

● Session (or thread)

A grouping of several traces from the same conversation or interaction.

● SIEM

A security event management platform, used for auditing.

● Span

A traced segment of execution: a step with start, end, parent, status and attributes.

● Span link

A link between spans with no parent-child relation (e.g., one's output feeds another's input).

● Subagent

An agent invoked by another agent during execution.

● Tool use

An agent calling an external resource: API, database, search, code execution.

● Trace

The complete record of an operation, made up of spans.

● Trace grading

Scoring agent traces to find errors and regressions (an OpenAI term).

● ZDR

Zero Data Retention: the vendor does not retain request data; in the OpenAI Agents SDK, tracing does not work under ZDR.

Bibliography (1/7)

R1

Anthropic. Observability with OpenTelemetry (Agent SDK)<https://code.claude.com/docs/en/agent-sdk/observability>

[A]

R2

Anthropic. Monitoring (Claude Code)<https://code.claude.com/docs/en/monitoring-usage?m=1>

[A]

R3

Anthropic. Managed Agents: observability<https://platform.claude.com/docs/en/managed-agents/observability>

[B]

R4

OpenAI. Tracing (Agents SDK)<https://openai.github.io/openai-agents-python/tracing/>

[A]

R5

OpenAI. Trace grading<https://developers.openai.com/api/docs/guides/trace-grading>

[B]

R6

OpenAI. Evaluate agent workflows<https://developers.openai.com/api/docs/guides/agent-evals>

[B]

R7

Google. Agent activity traces (ADK)<https://adk.dev/observability/traces/>

[B]

Bibliography (2/7)

R8

Google. Observability for agents (ADK)<https://adk.dev/observability/>

[B]

R9

Google Cloud. Trace an agent (Agent Engine)<https://cloud.google.com/agent-builder/agent-engine/manage/tracing>

[B]

R10

Google Cloud. Observability for AI agent developers<https://docs.cloud.google.com/stackdriver/docs/instrumentation/ai-agent-overview>

[B]

R11

Microsoft. Trace agent concept (Foundry)<https://learn.microsoft.com/en-us/azure/foundry/observability/concepts/trace-agent-concept>

[B]

R12

Microsoft. Set up tracing in Foundry<https://learn.microsoft.com/en-my/Azure/foundry/observability/how-to/trace-agent-setup>

[B]

R13

Microsoft. Quickstart: tracing a hosted agent<https://learn.microsoft.com/en-my/Azure/foundry/observability/quickstarts/quickstart-tracing-hosted-agent>

[B]

R14

Microsoft. Enable observability for agents (Agent Framework)<https://learn.microsoft.com/el-gr/agent-framework/tutorials/agents/enable-observability>

[B]

Bibliography (3/7)

R15

AWS. Observability for agentic resources in AgentCore<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-telemetry.html>

[A]

R16

AWS. Add observability to AgentCore resources<https://docs.aws.amazon.com/bedrock-agentcore/latest/devguide/observability-configure.md>

[B]

R17

AWS. AgentCore: unified observability in a single log group (Jul 23, 2026)<https://aws.amazon.com/about-aws/whats-new/2026/07/amazon-bedrock-agentcore-unified-observability-single-log-group/>

[B]

R18

OpenTelemetry. Semantic conventions for GenAI agent spans<https://opentelemetry.io/docs/specs/semconv/gen-ai/gen-ai-agent-spans>

[B]

R19

Dash0 (third-party). OpenTelemetry GenAI semantic conventions explained<https://www.dash0.com/knowledge/opentelemetry-genai-semantic-conventions-explained>

[B]

R20

DEV Community (third-party). OTel GenAI conventions are not stable yet<https://dev.to/azena-ai/opentelemetrys-genai-semantic-conventions-are-not-stable-yet-heres-what-actually-shipped-in-2026-3mke>

[B]

R21

LangChain. Observability concepts (LangSmith)<https://docs.langchain.com/langsmith/observability-concepts.md>

[B]

Bibliography (4/7)

R22

LangChain. Trace with OpenTelemetry (LangSmith)<https://docs.langchain.com/langsmith/trace-with-opentelemetry/index.html>

[B]

R23

LangChain. Observability how-to guides (LangSmith)https://docs.smith.langchain.com/how_to_guides/tracing

[B]

R24

Langfuse. Engineering clarifications<https://langfuse.com/resources/engineering/clarifications.md>

[B]

R25

Langfuse. Data model<https://langfuse.com/docs/observability/data-model.md>

[B]

R26

Arize. Phoenix (repository)<https://github.com/Arize-ai/phoenix>

[B]

R27

Atlan (third-party). What is Arize<https://atlan.com/know/ai-agent/ai-agent-observability/what-is-arize/>

[B]

R28

Datadog. OpenTelemetry instrumentationhttps://docs.datadoghq.com/llm_observability/instrument/otel_instrumentation/

[B]

Bibliography (5/7)

R29

Datadog. Agent monitoringhttps://docs.datadoghq.com/llm_observability/monitoring/agent_monitoring/

[B]

R30

Braintrust. Self-hosting<https://braintrust.dev/docs/guides/self-hosting>

[B]

R31

Braintrust. Release notes<https://www.braintrust.dev/docs/release-notes>

[B]

R32

Braintrust. Glossary<https://www.braintrust.dev/docs/reference/glossary.md>

[B]

R33

MLflow. Tracing for LLM and agent observability<https://mlflow.org/docs/latest/genai/tracing/index.html>

[B]

R34

Databricks. Automatic tracing and integrations<https://docs.databricks.com/gcp/en/mlflow3/genai/tracing/automatic-tracing>

[B]

R35

Pydantic. Logfire: AI observability<https://pydantic.dev/logfire.md>

[B]

Bibliography (6/7)

R36

Inference.net (third-party). Claude Agent SDK tracing and evaluation in production<https://inference.net/content/claude-agent-sdk-tracing/>

[B]

R37

ClickHouse (third-party). What is AI agent observability<https://clickhouse.com/resources/engineering/what-is-ai-agent-observability>

[B]

R38

BenchLM (third-party). Best LLM observability tools<https://benchlm.ai/blog/posts/best-llm-observability-tools>

[B]

R39

Jannik Reinhard (third-party). Microsoft Foundry observability<https://jannikreinhard.com/microsoft-foundry-observability/>

[B]

R40

PyPI and npm registries, queried via API on Oct 10, 2026 (versions and dates)<https://pypi.org> · <https://registry.npmjs.org>

[C]

R41

Latitude (third-party). LLM observability tools compared<https://latitude.so/blog/llm-observability-tools-compared>

[B]

R42

RFP.wiki (third-party). Braintrust<https://www.rfp.wiki/vendors/braintrust>

[B]

Bibliography (7/7)

R43

Pydantic. Logfire FAQ<https://logfire.pydantic.dev/docs/faq>

[B]

R44

Microsoft. Tracing in Microsoft Foundry (earlier page; mentions Outshift)<https://learn.microsoft.com/azure/ai-foundry/agents/concepts/tracing>

[B]

R45

Langfuse. AI SDK C++ integration (v4 data model, OTLP-first)<https://langfuse.com/integrations/frameworks/ai-sdk-cpp.md>

[B]

R46

Inference.net (third-party). Arize Phoenix alternatives for agent observability<https://inference.net/content/arize-phoenix-alternatives/>

[B]

R47

Pydantic. Pydantic AI: Logfire<https://ai.pydantic.dev/logfire>

[B]