ORACLE

AI agent observability and evaluation

A built-in framework to measure, evaluate, trace, report, and observe agent quality and performance in Oracle AI Agent Studio

Table of contents

Why your AI agent needs a robust evaluation framework

For high-value enterprise AI applications, an embedded framework for rigorously evaluating and improving agent performance can make the difference between successful production deployments and endless cycles of prototyping. In this guide, you will learn how the observability and evaluation framework brings automated measurement, evaluation, tracing, reporting, and observability capabilities to Oracle AI Agent Studio, an easy-to-use, no-code toolkit that helps you deliver accurate, performant AI solutions. You will also learn a practical, five-step process for evaluating agents that you can put to work in your organization.

Agent quality: The next challenge for enterprise AI

As enterprise AI moves from prototypes to production-grade assistants and autonomous agents, the question is no longer, can we build an agent? but, can we trust it to perform accurately, reliably, and cost-effectively at scale?

AI agents are becoming deeply embedded in enterprise workflows, augmenting tasks normally performed by humans—sometimes even fully automating business processes. Oracle AI Agent Studio enables users to create AI agents that are goal-driven and can plan, decide, and act to help meet their objectives. These agents are dynamic, autonomous and adaptive, and are powered by sophisticated language and reasoning models. But at their core, they are nondeterministic, and, unlike traditional software, these AI agents don't follow a predetermined path of execution.

Testing AI agents is different than testing software

- An agent's response can vary based on model updates or prompt changes. There could be hallucinations, something that traditional software has never dealt with.

- "Correctness" is seldom binary; answers must be judged for semantic alignment, relevance, completeness, and clarity.

- Agents have complex execution paths and often with multi-step execution and delegation of work to other agents; they interpret natural language intent, call tools (internal and external), and query documents. All of these can be sources of erroneous behavior that need to be traced.

- Content safety requirements, such as toxicity, bias, PII, prompt injection, and regulatory expectations evolve over time, necessitating continuous monitoring, auditing, and explainability.

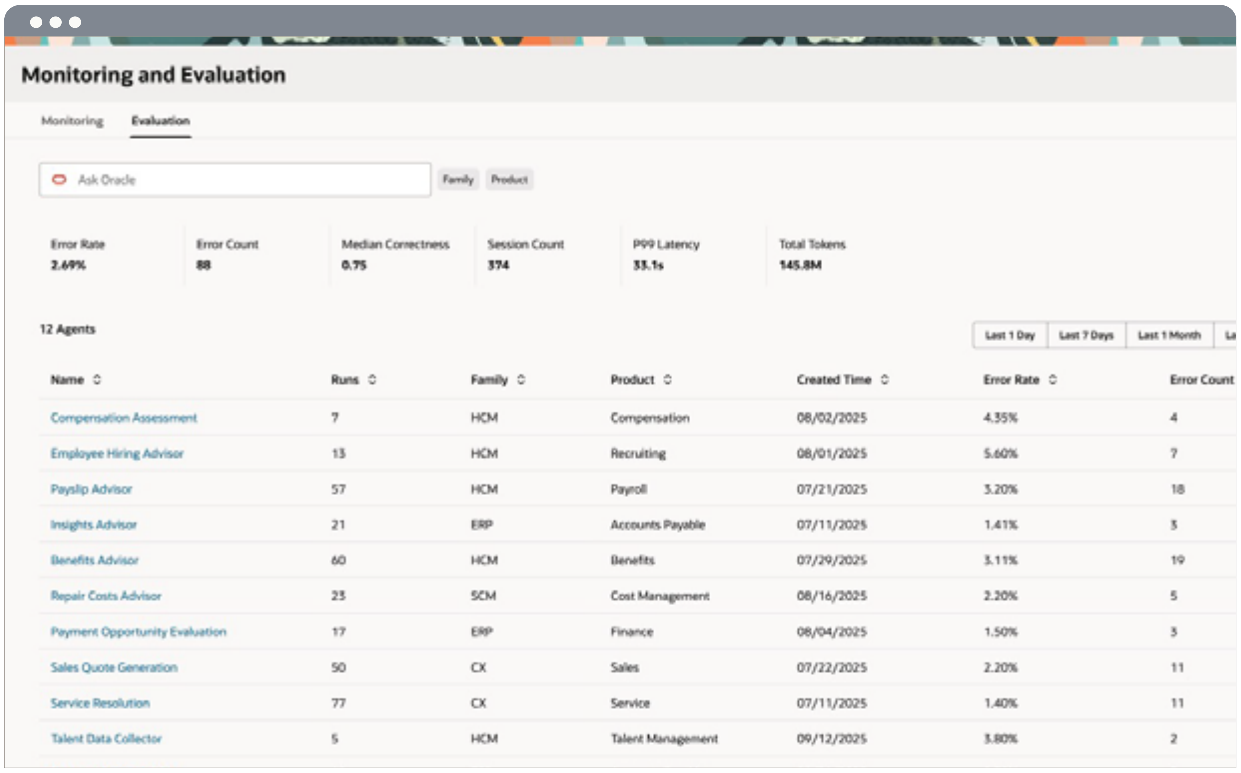
Traditional QA testing mechanisms fall short in these areas. The difficulty of testing AI agents is not a matter of degree, but of kind. The problem of evaluating and monitoring AI agent quality and performance shifts from verifying a fixed process to validating safe, reliable, and accurate behavior of an adaptive system. To address these realities, Oracle AI Agent Studio provides built-in observability and evaluation features, including a cohesive, built-in framework that helps teams design, test, deploy, and continuously improve AI agents end to end. These capabilities unify design-time evaluation, detailed step-by-step agent execution tracing, and runtime operational monitoring in a single experience.

> Oracle AI Agent Studio provides built-in observability and evaluation features, including a cohesive, built-in framework that helps teams design, test, deploy, and continuously improve AI agents end to end

Introducing the AI agent observability and evaluation framework

The integrated capabilities in Oracle AI Agent Studio enable the following:

- **Measurement:** Consistent capture of quality, performance, and cost metrics for prompts and agents
- **Evaluation:** Design-time test authoring and automated runs across versions with outputs and configurable thresholds
- **Tracing:** Step-by-step introspection of agent sessions and turns, including tool calls, LLM calls, latency, tokens, and errors, for fast root-cause analysis
- **Reporting:** Drill-down dashboards and history views for prompts and agents, comparison of evaluation runs, and leaderboard-style summaries
- **Observability:** Production-grade monitoring with filters, time windows, and latency/error



Error Rate	Error Count	Median Correctness	Session Count	P99 Latency	Total Tokens
2.69%	88	0.75	374	33.1s	145.8M

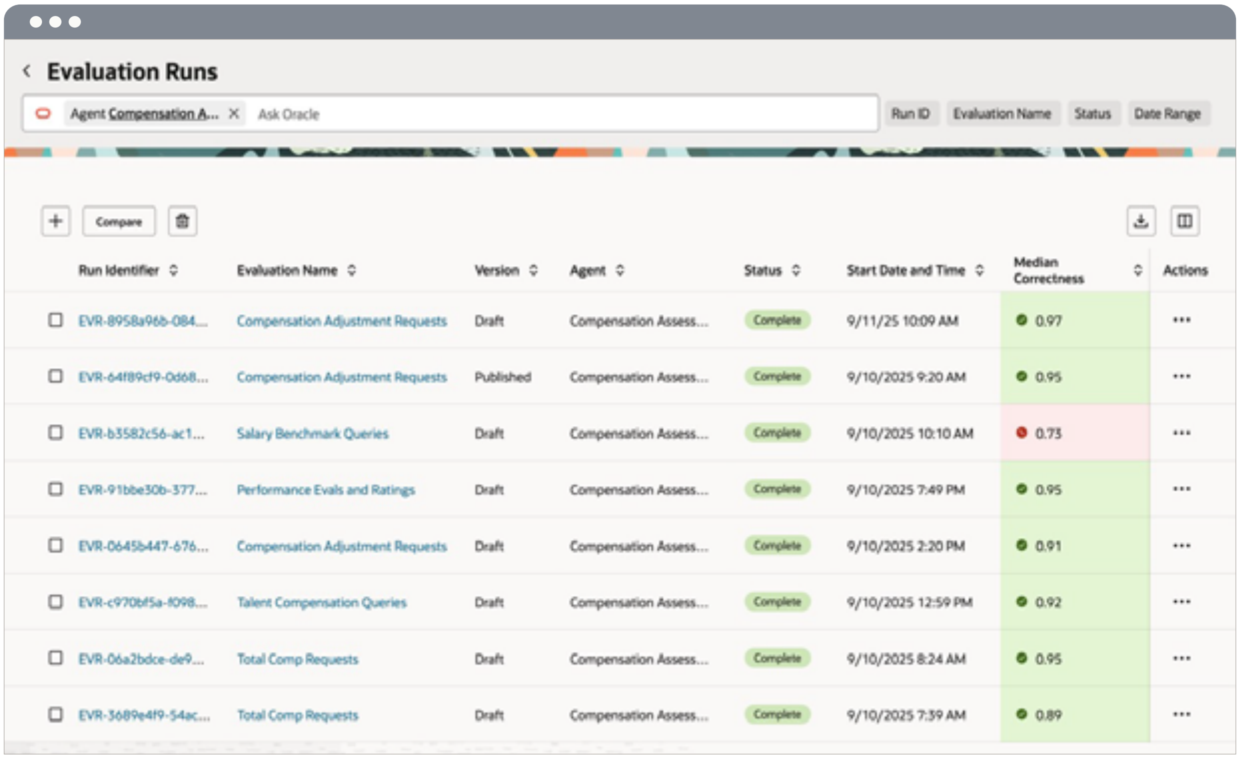
12 Agents

Name	Runs	Family	Product	Created Time	Error Rate	Error Count
Compensation Assessment	7	HCM	Compensation	08/02/2025	4.35%	4
Employee Hiring Advisor	13	HCM	Recruiting	08/01/2025	5.60%	7
Payslip Advisor	57	HCM	Payroll	07/21/2025	3.20%	18
Insights Advisor	21	ERP	Accounts Payable	07/11/2025	1.41%	3
Benefits Advisor	60	HCM	Benefits	07/29/2025	3.11%	19
Repair Costs Advisor	23	SCM	Cost Management	08/16/2025	2.20%	5
Payment Opportunity Evaluation	17	ERP	Finance	08/04/2025	1.50%	3
Sales Quote Generation	50	CX	Sales	07/22/2025	2.20%	11
Service Resolution	77	CX	Service	07/11/2025	1.40%	11
Talent Data Collector	5	HCM	Talent Management	09/12/2025	3.80%	2

Evaluation

Design-time quality and safety, operationalized

Oracle's evaluation capability supports rigorous, repeatable offline testing before deployment. Teams define evaluation sets that pair inputs with expected answers/outputs and specify which metrics to compute. Evaluation runs can be targeted to specific versions, such as draft versus published agents or seeded versus overridden prompts.



Run Identifier	Evaluation Name	Version	Agent	Status	Start Date and Time	Median Correctness	Actions
EVR-8958a96b-084...	Compensation Adjustment Requests	Draft	Compensation Assess...	Complete	9/11/25 10:09 AM	● 0.97	•••
EVR-64f89cf9-0d68...	Compensation Adjustment Requests	Published	Compensation Assess...	Complete	9/10/2025 9:20 AM	● 0.95	•••
EVR-b3582c56-ac1...	Salary Benchmark Queries	Draft	Compensation Assess...	Complete	9/10/2025 10:10 AM	● 0.73	•••
EVR-91bbe30b-377...	Performance Evals and Ratings	Draft	Compensation Assess...	Complete	9/10/2025 7:49 PM	● 0.95	•••
EVR-0645b447-676...	Compensation Adjustment Requests	Draft	Compensation Assess...	Complete	9/10/2025 2:20 PM	● 0.91	•••
EVR-c970bf5a-f098...	Talent Compensation Queries	Draft	Compensation Assess...	Complete	9/10/2025 12:59 PM	● 0.92	•••
EVR-06a2bdce-de9...	Total Comp Requests	Draft	Compensation Assess...	Complete	9/10/2025 8:24 AM	● 0.95	•••
EVR-3689e4f9-54ac...	Total Comp Requests	Draft	Compensation Assess...	Complete	9/10/2025 7:39 AM	● 0.89	•••

Key capabilities:

Evaluation data set management

- ✓ Author test cases for agents (questions and expected responses)
- ✓ Upload reference evaluation sets in bulk, edit, and rerun as agents evolve
- ✓ Associate data sets with a target agent and select metrics to calculate and specify pass/fail thresholds

LLM-as-a-judge (LaaJ) correctness

- ✓ Automated semantic scoring and explanation of correctness relative to answers using a specialized judge large language model (LLM) with optional human annotations where needed

Runs and results

- ✓ Execute runs against specific versions; view run-level summaries for median correctness, median latency, tokens, and errors
- ✓ Drill down to inspect individual test cases, agent responses, and metrics, including correctness, latency, token counts, and error flags
- ✓ Highlight regressions with threshold-driven pass/fail indicators and, when useful, show differences between actual versus expected outputs

A/B comparisons

- ✓ Perform side-by-side comparisons of two runs of the same evaluation set to quantify changes in correctness, latency, token costs, and error behavior

Metrics available in evaluation

- ✓ Quality: Correctness (LaaJ-based rating) and pass count/rate
- ✓ Performance: Latency (median, P99) and errors (API call timeouts)
- ✓ Cost: Prompt/input tokens, output tokens, total token consumption, median, and P99 token counts
- ✓ Usage (where applicable): Turns (agents) and LLM calls (prompts)

LLM-as-a-judge

Scaling semantic quality assessment

Traditional unit tests struggle with open-ended language outputs. This is addressed by LaaJ, a principled methodology that uses an independent judge LLM to score semantic correctness against expected answers.

- Teams define expected answers/outputs in evaluation sets.
- During runs, the judge LLM scores each response's semantic alignment and provides a correctness metric on a 0-1 scale as well as an explanation of its scoring rationale. Human evaluators can adjust the rating, if needed, and add a score/explanation as appropriate, for example, audits.
- Aggregated correctness, such as median, becomes a key signal in run summaries and comparisons.
- Using LaaJ brings many, varied benefits. LaaJ scales quality measurement across large test suites without exhaustive human review so evaluations can be done in languages that have less human oversight. And LaaJ provides a nuanced assessment that better reflects user-perceived quality than exact-match metrics.

Standardizing test assets and metrics and enabling automated, version-aware runs helps reduce subjective debate about "quality" and makes evaluation a routine engineering practice. The combination of LaaJ scoring, latency/cost measurement, and run comparisons provides an evidence-based pathway to AI agent design and iterative improvements.

Evaluation data sets

An essential ingredient

To ensure successful evaluations, it is important that evaluation data sets are designed to stress-test agent capabilities. Test cases should be chosen to represent a broad range of inputs from the target audience and diversity in languages, grammar, and verbosity. They should exercise full coverage of all the agent's functional elements as well as adversarial attacks, edge cases, and failure cases. The observability and evaluation framework's automation and management features make it easy to create, run, and analyze the results of extensive evaluation data sets.

A practical, five-step process for evaluating AI agents

Organizations can adopt the following five-step process to operationalize AI agent evaluations:

1 Define goals and metrics

- Clarify the business problem the agent is solving
- Establish success criteria, for example: resolution rates or customer satisfaction
- Translate these criteria into measurable metrics

2 Build a representative data set

- Develop test cases that mirror real-world scenarios
- Include both “happy path” and “unhappy path” scenarios, for example: ambiguous, adversarial, or out-of-scope queries

3 Conduct the evaluation

- Run the agent against the data set, capturing responses, reasoning, and tool usage
- Apply a mix of automated and human evaluation techniques

4 Analyze and interpret results

- Determine thresholds for acceptable performance
- Identify failure patterns, such as misinterpreted intent or incorrect tool usage
- Aggregate results and benchmark against success criteria

5 Iterate and refine

- Treat evaluation as cyclical
- Use insights to improve prompts, structure, or tool integration
- Retest updated agents to confirm measurable improvements

Tracing

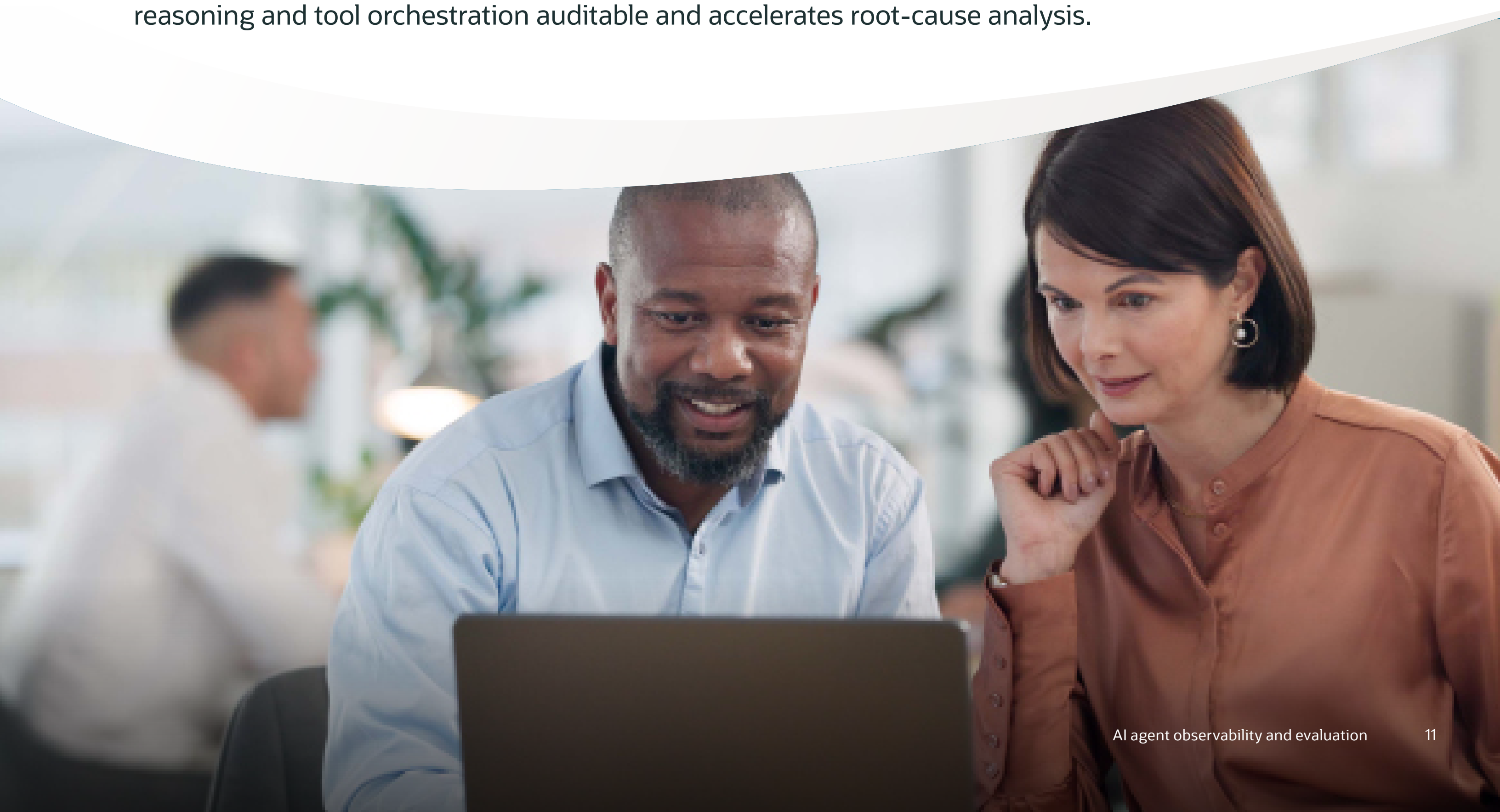
Deep transparency for debuggability and trust

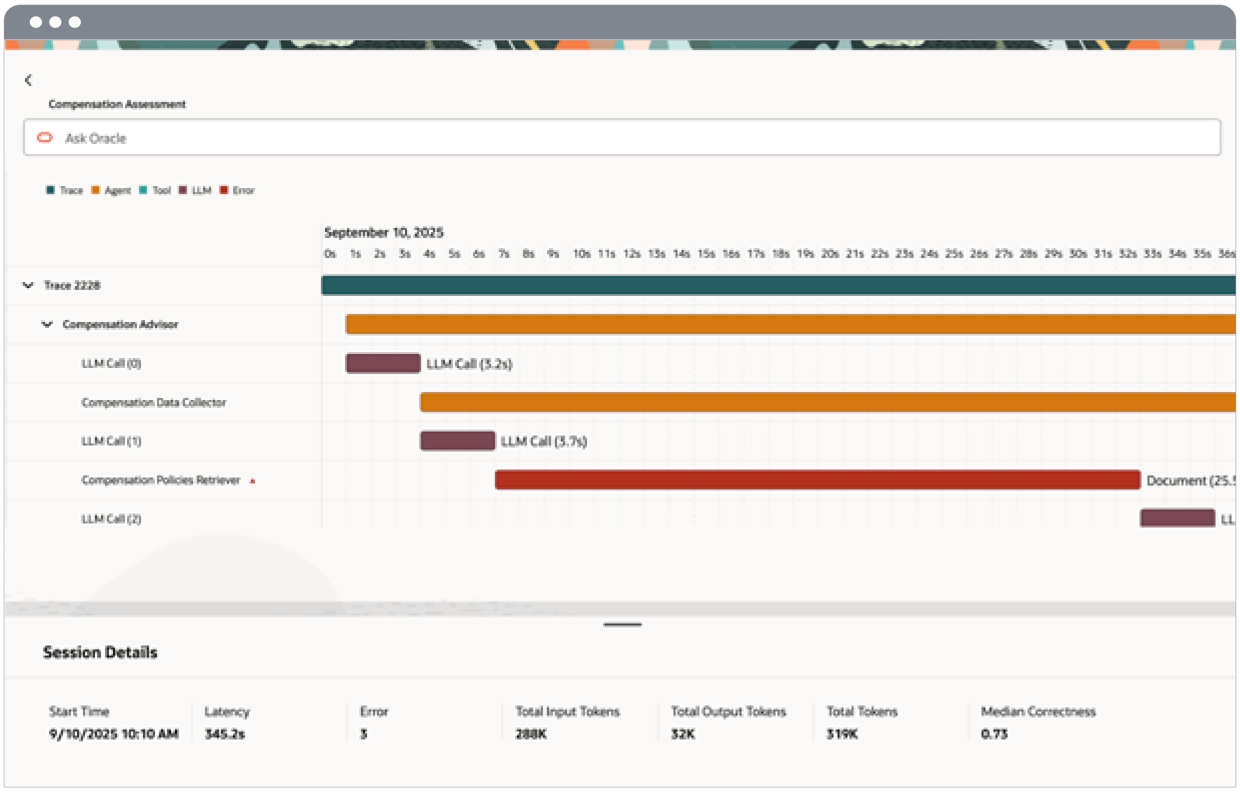
When production issues arise—or when teams are hardening an agent's design—observability must go deeper than aggregates. Tracing offers step-level introspection for single prompts and multi-step agents, allowing developers and operators to analyze execution flow, identify bottlenecks, confirm enforcement, and diagnose errors.

Agent execution tracing

- **Overall summary:** Start time, end-to-end latency, errors, token totals, and (for evaluation) correctness. Shows input question and final response for sessions.
- **Interactive timeline:** Chronological visualization of steps (worker agents, LLM calls, and tools, such as business objects, REST, document/RAG, calculator, session, email, and deep link). Each step displays type, timing, duration, and status; clicking a step reveals details.
- **Agent steps:** Prompt text, topics/instructions, latency, API failure descriptions, and token metrics; input/output shown in evaluation runs.
- **LLM calls:** Latency, error details, and tokens; input/output visible in evaluation runs.
- **Tools:** Per-tool inputs/outputs and failure descriptions; for RAG, show retrieved chunks and sources; for REST/business objects, show functions, parameters, and payloads (evaluation runs).

Tracing closes the gap between "what happened" and "why it happened." Showing execution details enables transparency. The agent explains itself by showing its inner workings. Tracing makes agent reasoning and tool orchestration auditable and accelerates root-cause analysis.




Compensation Assessment
Ask Oracle
Trace
Agent
Tool
LLM
Error
September 10, 2025
0s 1s 2s 3s 4s 5s 6s 7s 8s 9s 10s 11s 12s 13s 14s 15s 16s 17s 18s 19s 20s 21s 22s 23s 24s 25s 26s 27s 28s 29s 30s 31s 32s 33s 34s 35s 36s
Trace 2228
Compensation Advisor
LLM Call (0)
LLM Call (3.2s)
Compensation Data Collector
LLM Call (1)
LLM Call (3.7s)
Compensation Policies Retriever
Document (25.5
LLM Call (2)
LL

Session Details

Start Time	Latency	Error	Total Input Tokens	Total Output Tokens	Total Tokens	Median Correctness
9/10/2025 10:10 AM	345.2s	3	288K	32K	319K	0.73

Reporting and observability

Operate with confidence in production

Once agents are deployed, leaders need an operational picture to manage SLAs, budgets, and risk across product families and products. Monitoring dashboards and history views provide a centralized command center for usage, performance, cost, quality (evaluation), and signals for prompts and agents.

Metrics dashboard

- Aggregate across all product families/products by default; filter by family, product, and time window, from last day to last three months.
- Leaderboards and lists: Prompts and agents ranked by usage; dedicated list and execution history views with filtering, sorting, and pagination; drill down to traces for investigations.

Observability transforms anecdotes into action. With rollups, leaderboards, filters, and drill-down histories, teams can spot outliers, track improvements, and allocate optimization efforts where they pay off while maintaining continuous visibility into cost posture.

The Oracle AI Agent difference: Observability & Evaluation

- ✓ **Purpose-built for agents**
  Observability and evaluation capabilities address the full complexity of multi-step agent workflows—supervisors, worker agents, prompts, tool calls, and RAG—rather than treating everything as a single LLM call.

- ✓ **One framework, full lifecycle**
  From managing evaluation data sets and running comparisons to production dashboards and deep traces, these capabilities unify design-time and runtime needs with consistent metrics and interface.

- ✓ **Cost as first-class citizens**
  Content care, prompt injection flags, and token economics live alongside correctness and latency in the UI, metrics, and traces—not as afterthoughts.

- ✓ **Enterprise-grade visibility**
  Filters by product family/product, rich time windows, leaderboards, and execution histories help large organizations operationalize SLAs and budgets across portfolios.

How Oracle can help

AI agents demand a new standard of engineering discipline. With Oracle AI Agent Studio's observability and evaluation framework, teams can gain the built-in capabilities required to design, test, deploy, and continuously improve the reliability and efficiency of AI agents. Evaluation sets and LLM-as-a-judge scoring raise preproduction quality. Tracing unlocks deep transparency and rapid root-cause analysis. And dashboards deliver the operational observability enterprises need. This integrated approach makes Oracle AI Agent Studio a robust, comprehensive, industry-leading solution for building and launching AI agents that meets real-world expectations for accuracy, performance, and total cost of ownership —at Oracle scale and with Oracle-grade governance.

Request a demo

Learn more

Connect with us

Call +1.800.ORACLE1 or visit oracle.com

Outside North America, find your local office at oracle.com/contact