Observability and evals with Radium

Observability and eval tools work by instrumenting your SDK client, not by talking to a specific provider. Radium serves an OpenAI-compatible Chat Completions endpoint and an Anthropic-compatible Messages endpoint, so a tool that wraps the OpenAI or Anthropic client library keeps capturing requests and responses the same way once that client points at Radium. Nothing about how you trace, log, or score generations has to change.

This page covers what carries over unmodified, and the one category of feature that can silently stop working: anything that depends on the tool recognising the model name.

What changes, what doesn't

What you change

- The base URL your client points at
- The model string

What stays the same

- Your tracing or eval library and its setup
- Your instrumentation code
- The shape of the request and response your tool reads from

If your observability setup wraps the client instance itself, LangSmith's `wrap_openai`, Braintrust's `wrap_openai` or `wrap_anthropic`, Weave's autopatching, or an OpenTelemetry GenAI instrumentation, it intercepts calls at the SDK layer and has no dependency on which base URL the client was constructed with.

Python, wrapping the OpenAI client

```
from openai import OpenAI
from langsmith.wrappers import wrap_openai

client = wrap_openai(OpenAI(
    api_key="YOUR_RADIUM_API_KEY",
    base_url="https://api.radium.cloud/v1",
```

```
))

response = client.chat.completions.create(
    model="hal-1.0",
    messages=[{"role": "user", "content": "Hello, Radium!"}],
)
```

The wrapper never inspects the base URL. It sees the same method call and the same response object it would see against OpenAI directly.

Where it can break: model-name lookups

Some platforms maintain their own catalog of known models to drive features like automatic cost calculation, context-window warnings, or model-specific capability flags. `hal-1.0`, `clarke-1.0`, and `tycho-1.0` will not be in that catalog for most third-party tools. Tracing and logging still work, the request and response are captured either way, but any feature that depends on recognising the model by name can fail quietly: a cost column that reads zero, a context-limit warning that never fires, a capability badge that shows the wrong thing.

> **Check this before you trust a dashboard** If a tool's cost or token-limit features rely on a model catalog, confirm it has an "unknown model" fallback that still uses the token counts on the response, rather than dropping the row.

Usage data on the wire

Every response, streaming or not, carries the same usage shape your eval tool already reads:

```
"usage": {
  "prompt_tokens": 1234,
  "completion_tokens": 321,
  "total_tokens": 1555
}
```

This is the field most cost-tracking and eval scoring reads from directly, independent of any model catalog. If a platform's built-in cost column comes up empty for a Radium model, the token counts underneath it are still there to compute cost yourself against your own rate table.

Tool calls in traces

Tool-calling data shows up in the captured trace the same way it would against OpenAI or Anthropic, `tool_calls` on the OpenAI-compatible path, `tool_use` and `tool_result` content blocks on the Anthropic-compatible path, since your eval tool reads these from the response body, not from anything Radium-specific. See tool calling for the full shape.

Streaming

Streaming responses follow the event format of whichever compatibility layer you're using. A tracing tool that already assembles streamed tool-call deltas for OpenAI or Anthropic assembles them the same way here.

Before you rely on it

Run your team's actual eval suite and observability setup against Radium once, end to end, before trusting a dashboard built around OpenAI or Anthropic responses to reflect Radium data accurately. Tracing and token counts are the parts most likely to be exactly right by default. Cost columns and model-specific UI are the parts most likely to need a second look.

Next

- API quickstart, for the request and response shapes referenced above
- Tool calling, for the full `tool_use` and `tool_result` contract

Created 2026-08-23 22:10:01 UTC by Admin
Updated 2026-08-23 22:11:42 UTC by Admin