May 2026

OpenTelemetry GenAI in the wild

A Practitioner's Field Guide

DATADOG

Zach Groves

SWE II at Datadog

DATADOG

Quick Poll

1. Using/working on/deploying LLMs?
2. Observing those LLMs?
3. Using OpenTelemetry?

What is Opentelemetry GenAI?

A vendor-neutral standard built on top of OpenTelemetry for capturing LLM observability data e.g. spans, token usage, model metadata, and agent behavior.

Attributes:

Key	Stability	Requirement Level	Value Type	Description	Example Values
gen_ai.operation.name	development	Required	string	The name of the operation being performed. [1]	chat ; generate_content ; text_completion
gen_ai.provider.name	development	Required	string	The Generative AI provider as identified by the client or server instrumentation. [2]	openai ; gcp.gen_ai ; gcp.vertex_ai
error.type	stable	Conditionally Required if the operation ended in an error	string	Describes a class of error the operation ended with. [3]	timeout ; java.net.UnknownHostException ; server_certificate_invalid ; 500
gen_ai.agent.description	development	Conditionally Required If provided by the application.	string	Free-form description of the GenAI agent provided by the application.	Helps with math problems ; Generates fiction stories
gen_ai.agent.id	development	Conditionally Required if applicable.	string	The unique identifier of the GenAI agent.	asst_5j66UpCpwteGg4YSxUnt7lPY
gen_ai.agent.name	development	Conditionally Required If provided by the application.	string	Human-readable name of the GenAI agent provided by the application.	Math Tutor ; Fiction Writer
gen_ai.input.messages	development	Opt-In	any	The chat history provided to the model as an input. [4]	[{ "role": "user", "parts": [{ "type": "text", "content": "Weather in Paris?" }] }, { "role": "assistant", "parts": [{ "type": "tool_call", "id": "call_VSPygqKTWdrhaFErNvMV18Yl", "name": "get_weather", "arguments": { "location": "Paris" } }] }, { "role": "tool", "parts": [{ "type": "tool_call_response", "id": "call_VSPygqKTWdrhaFErNvMV18Yl", "result": "rainy, 57°F" }] }]

Session ...f3a3caba / Trace ...8e67e8f9

Flag for Annotation

May 19, 2:05 pm simple-llm-example

1.15s 0.076¢ 161 2 gpt-4o-2024-08-06 gpt-4o

Trace Span Search in currently selected span Span Details Flame Graph

chat gpt-4o 1.15s

get_weather 87.0µs

chat gpt-4o final response 53.0µs

Span Chat Evaluations 0 Logs 0 Annotations 0

LLM **chat gpt-4o final response**

Add to Dataset Test in Playground

Estimated Cost 0.053¢ gpt-4o simple-llm-example View in APM

Input Messages Input Tokens: **87** Input Cost: **0.022¢**

USER

What is the weather in Minneapolis?

TOOL

```
{
  "temperature": 58,
  "unit": "fahrenheit",
  "condition": "Partly Cloudy",
  "humidity": "62%"
}
```

Output Message Output Tokens: **31** Output Cost: **0.031¢**

The current weather in Minneapolis is 58°F and partly cloudy with 62% humidity.

Metadata

```
{
  "finish_reasons": "stop"
}
```

Tags

env:none source:undefined service:simple-llm-example cost_estimate_status:success

DATADOG

Why Care?

- **Vendor lock-in** — proprietary attributes couple you to one backend; switching means re-instrumenting
- **Reuse existing OTel APM pipelines** — spec-compliant spans route through your current Collector, sampling, and redaction rules automatically
- **Free backend integrations** — Datadog, Grafana, and others build native GenAI dashboards against the spec
- **Silent failures** — attribute name mismatches produce zeros in dashboards with no errors
- **End-to-end correlation** — one non-compliant layer in an agent trace breaks the whole chain

Third-parties such as (OpenLLMetry, OpenInference, Langtrace) invent their own namespaces

A brief history of the OTel GenAI SemConv

Late 2022
ChatGPT-3.5 release

2023/Early 2024
Third-parties such as (OpenLLMetry, OpenInference, Langtrace) invent their own namespaces

Early 2024
GenAI Special Interest group forms under OTel

Mid 2024
SemConv versions v1.27-v1.28 land but are sparse, no story for tools, agent, retrieval, multimodal, evals

Early 2025
No large “OTel” instrumentation libs, including Arize, Traceloop, and Langtrace have fully adopted

Mid 2025
SemConv 1.37 provides structured I/O messages, Followed shortly by 1.38-1.41 to fill in gaps, adding many new attributes. A real spec to believe in!

DATADOG

Where we are now in the post-1.37 era

SemConv Status: still in “Development”

Finally mature enough to express modern LLM and agent observability

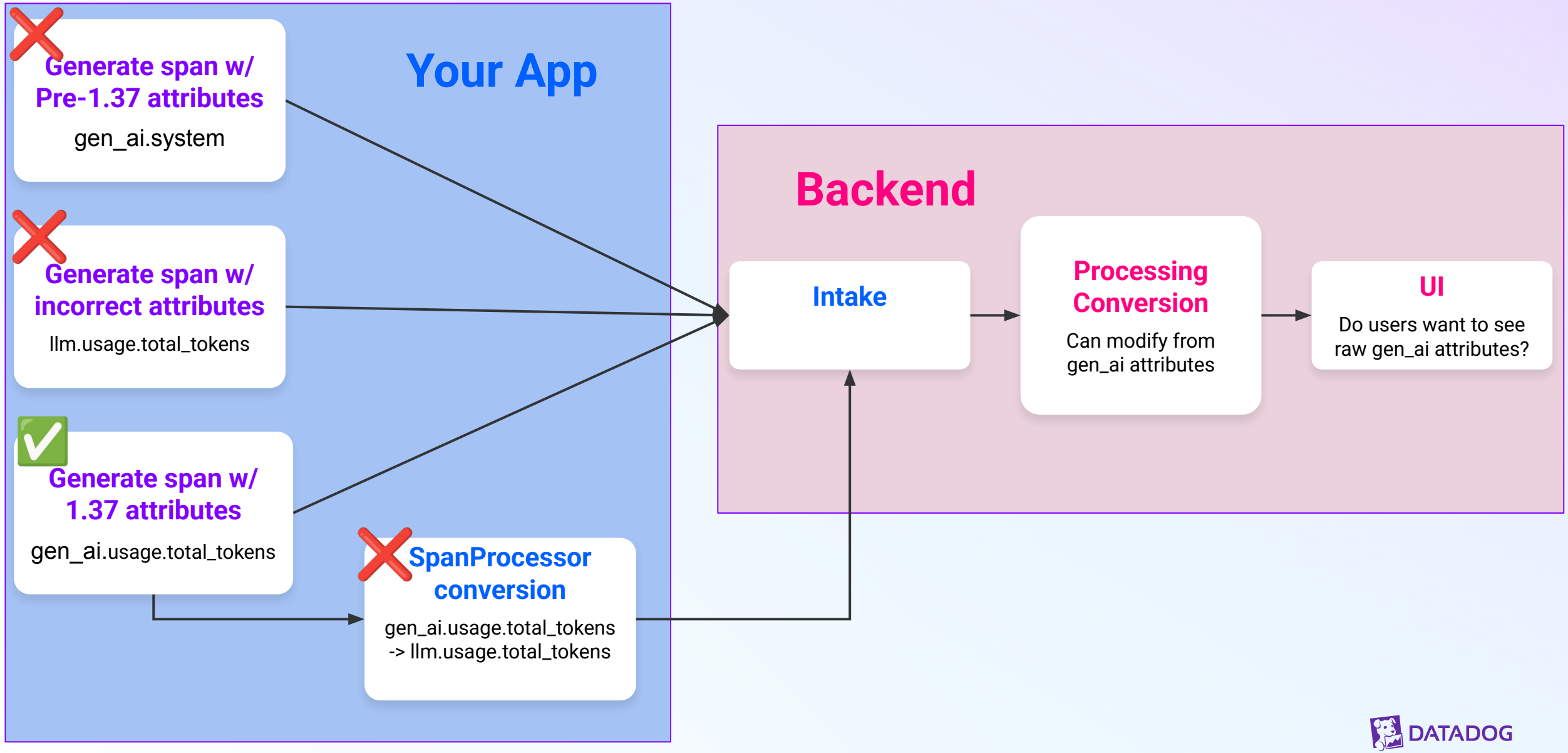
- Migration: The current recommendation (if a package already supported pre-1.37 SemConv) is to put post-1.37 SemConv implementation behind a feature flag OTEL_SEMCONV_STABILITY_OPT_IN=gen_ai_latest_experimental

- This works… but now it’s almost as if OTel GenAI is two different SemConvs for now…. if only it were that simple… remember all of those third party namespaces?

The implementation landscape

In what way is OTel GenAI actually being put into practice?

Where can GenAI go wrong?



DATADOG

My Personal Totally Objective Tier List

Of various SDKs that emit “OTel”

DATADOG

D Tier — Doesn't preserve gen_ai.*; converts SemConv data away

OpenInference

Emits incorrect llm.* attributes by default

- Adapter for upstream 1.37 frameworks strip gen_ai.* and rewrite to llm.*
  - Only usable with ingestion that SemConvs to OpenInference

Langfuse

Emits incorrect langfuse.* attributes by default

Backend that consumes OTel via /api/public/OTel

- Adapter for upstream pre-1.37 attribute strip gen_ai.* and rewrite to langfuse.*
  - No public v1.37 mapping
  - Only usable with ingestion that SemConvs to Langfuse

C Tier — Uses gen_ai.* to some extent but needs improvement

Langtrace

- Uses old message structures
- Partial 1.37 alignment — Correctly emits stable `gen_ai.request.*` but is missing many other attributes
- Committed to migrating
- Parallel `langtrace.*` namespace

AgentOps

Flat-indexed `gen_ai.prompt.{i}.role/.content` style

- No structured 1.37 messages
- Agent-first data model with `agentops.*` as the primary namespace `gen_ai.*` second
- Custom `gen_ai.openai.*` extensions outside the SemConv

Vercel AI SDK

Native 1.37 emission

- Proprietary `ai.*` namespace is primary
- Partial `gen_ai.*` coverage exists but is limited
- Third-party ai-sdk-otel-adapter SpanProcessor remaps `ai.*` attributes to `gen_ai.*` for standard backends

B Tier — Right names, wrong types

Spring AI / Micrometer

- Submits everything as strings
- Distinct failure mode from SemConv drift — names match, type system doesn't
- Backend has to coerce "42" to int for token counts on every span if you want useful metrics capabilities etc.

A Tier — Moving in the right direction and very close

OpenLLMetry

Renamed their `llm.*` namespace to `gen_ai.*` in early 2026

- Per-instrumentor PRs landing — OpenAI / Anthropic / Bedrock / Gemini done, others not
- Messages PR adds attribute names without the structured-message implementation
- Dual-emits legacy attrs for back-compat

OpenLIT

SemConv-aligned third-party

- Actively tracks the spec, correct span naming pattern and metric names
- Some incorrect provider name values (gemini not gcp.gemini, xai not x_ai)
- Incorrect reasoning tokens attribute
- Tool calls flattened onto the parent span as attributes instead of child execute_tool spans

S Tier — Reference implementation & verified third-party

OTel python-contrib

Cleanest 1.37 emit available

- opt-in via `OTEL_SEMCONV_STABILITY_OPT_IN` flag
- Default still emits v1.28-derived schema

LiteLLM

SemConv-aligned

- opt-in via `OTEL_SEMCONV_STABILITY_OPT_IN` flag
- Emits deprecated gen_ai.system along gen_ai.provider.name

Code comments admit they know

S Tier Cont. — 1.37+ emission

Microsoft Agent Framework

Both Python and .NET

- Possibly the cleanest first-party framework today
- .NET source pins the SemConv version directly in code

Pydantic AI

Defines the full 1.37+ message-part schema

- Their evaluation extensions got adopted into v1.38

AWS Strands

Native 1.37 emission

- Pretty close to perfect
- Cross-validated with vendor ingesting OTel

Takeaways

We’re headed in the right direction

MASSIVE IMPROVEMENTS

- We’ve come so far!
- Rapid adoption and movement towards aligning on the 1.37+ SemConv
- Demonstrates that the SemConv fits use cases for GenAI monitoring far better than its predecessor

The Shift

Frameworks emitting OTel natively

- Pydantic AI, MS Agent Framework, AWS Strands, Spring AI now emit OTel directly — no third-party wrapper required.
- The monkey-patching ecosystem (OpenLLMetry, OpenInference's per-vendor packages) is partly a workaround for frameworks that didn't ship OTel themselves

Potential side effects of using wrappers

- If you’re seeing duplicates for spans, validate that the wrapper isn't doubling up on what the framework already does, choose which you want.
- Double check if a wrapper adds span processors.

Lesson: Don't trust compatibility claims at face value

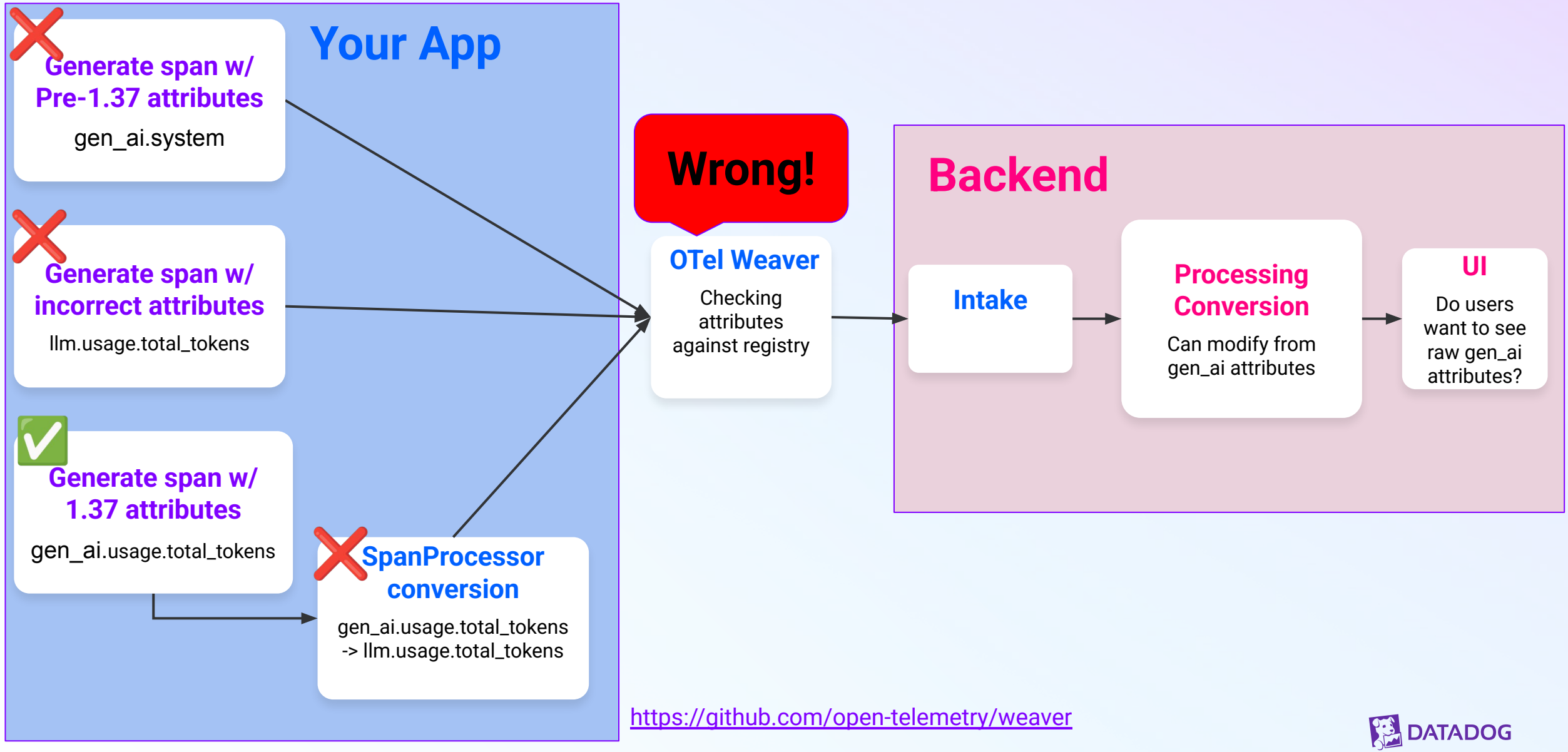
You don’t quite know what you’re going to get until you try it out.

- Adds attribute names without the structured-message implementation — AgentOps, Langtrace, Openllmetry
- Proprietary ai.* namespace as primary - Vercel
- Convert SemConv-aligned data away before the backend — OpenInference, Langfuse
- Use right names with wrong types — Spring AI / Micrometer

Validate emitted telemetry

- Look at raw span json and use LLM to verify structure against SemConv
- Try using OTel Weaver

Where can GenAI go wrong?



https://github.com/open-telemetry/weaver

DATADOG

X Tier — Custom in-house instrumentation

Telemetry for the people by the people

- Many production LLM apps emit OTel from custom in-house code, not from a library
- Quality ranges from SemConv-perfect to wildly off, token counts under made-up keys, ad-hoc message shapes, attributes named after internal jargon, messages on random non GenAI spans etc. `observe_my_ai.2.<my-company>.customer_id`
- Where "validate emitted telemetry" matters most

In-house instrumentation advice

The cause is the cure

- In-house telemetry is a lot of work… but luckily we can use AI to monitor AI!
  - Be careful LLMs use what's most common, and right now most OTel instrumentation out there is pre-1.37 SemConv. Make sure to specify and pass it the docs.

Validate emitted telemetry

- Look at raw span json and use LLM to verify structure
- OTel Weaver can be a great tool for validation
- Verify with your vendor
  - What actually matters to you is if the spans look how you want them in a UI, work with your vendor to come to the best SemConv implementation

The 1.37+ SemConv is good and ever improving!

The implementations are getting there!

In-house instrumentation is easier and better than ever before!