Eiffel and OpenTelemetry traces

2024-02-15

What is OpenTelemetry?

Emerging industry standard for exporting metrics, logs, and distributed traces from applications.

Traces have the most mature spec and implementation, and it’s what we’re planning to talk about today.

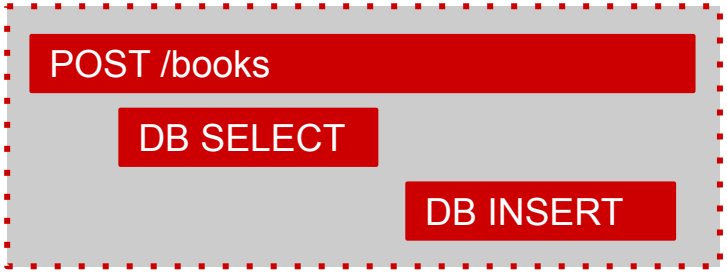
A primer on traces and spans

A trace is a collection of spans.

A span describes what happens, when, and for how long.

A span can declare relationships to other spans, most notably its parent.

A span has attributes that provide additional context and enables searching and aggregating spans.



How do traces and Eiffel events relate?

They both describe something that happens, when it happens, and how it relates to other things that happen.

Spans are usually rather close to the code (potentially mapping 1:1 with functions/methods) but could describe high level, abstract matters.

Eiffel events are almost mostly high level and focus more on relationships to other Eiffel events rather than being vessels of information by themselves.

Do you really need both traces and Eiffel events?

Spans are often sampled and/or not stored for a very long time. Eiffel events are never sampled and usually stored for a long time.

Spans only announce what has happened. They aren't meant to be used to trigger anything else.

Spans are sent when they've ended, so they can't be used to monitor ongoing activities.

Conclusion: Traces and Eiffel events serve different purposes and only overlap to some extent.

Some reasons for connecting spans and Eiffel events

Your build emits traces but your CI pipeline doesn’t.

Your CI pipeline emits traces and you want to send Eiffel events for e.g. artifacts.

Connecting spans with each other

A span is connected to its parent via the parent id attribute.

Context propagation transfers the current trace context across a boundary so that new, connected, spans can be created.

The W3C Trace Context standard is commonly used for this, e.g. in HTTP:

```
GET /books/1 HTTP/1.1
traceparent: 00-0af7651916cd43dd8448eb211c80319c-b7ad6b7169203331-01
tracestate: congo=t61rcWkgMzE
```

Proposal: Add optional trace context to Eiffel events

```
{
  "meta": {
    "id": "3de1c273-b961-4ecc-9f78-d000bcc93edd",
    "type": "EiffelArtifactCreatedEvent",
    "version": "3.3.0",
    "tracecontext": {
      "traceparent": "00-0af7651916cd43dd8448eb211c80319c-b7ad6b7169203331-01",
      "tracestate": "congo=t61rcWkgMzE"
    },
    ...
  },
  "data": {...}
}
```

Eiffel

Proposal: Use span attributes to connect spans to Eiffel activities

A span indicating that it's processing a particular Eiffel event:

```
com.github.eiffel-community.meta.id: 3de1c273-b961-4ecc-9f78-d000bcc93edd
com.github.eiffel-community.meta.type: EiffelArtifactCreatedEvent
```