

Question

How application can influence span status set by instrumentation library

<https://github.com/open-telemetry/opentelemetry-specification/issues/4131>

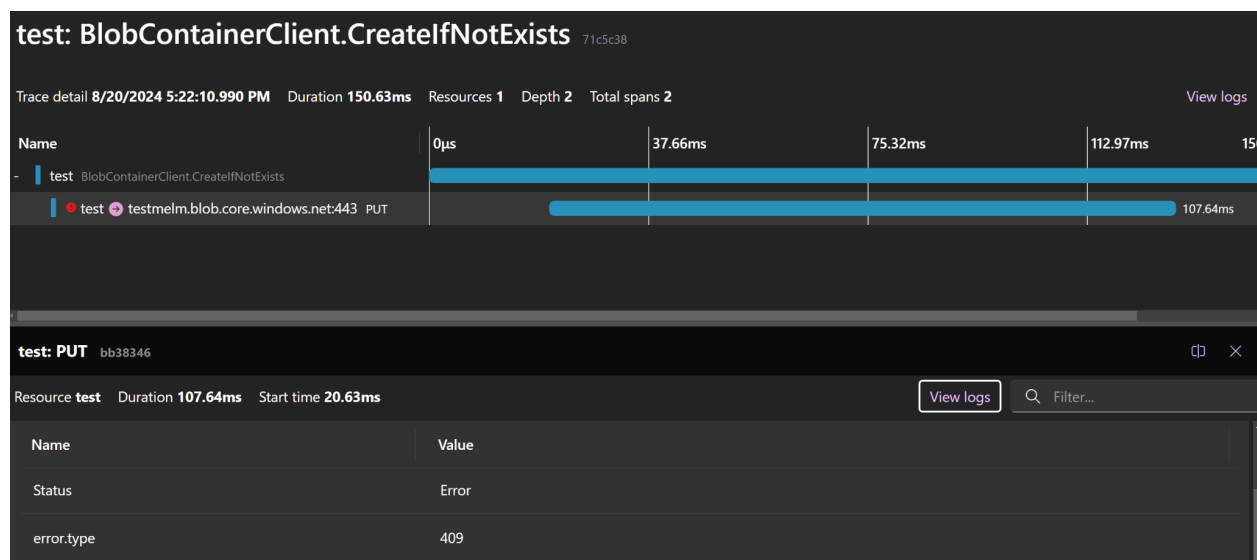
Related

[Convention for cancelled spans](#)

[HTTP span status: use SHOULD instead of MUST for errors](#)

Problems

- 404 (and similar) is error or not-an-error depending on the context
- Tail-based sampling relies on errors
- Alerts/monitoring based on error status is unreliable



Solutions

1. Do nothing

- Backends should detect when outer span status is OK/UNSET and nested is ERROR. They should visualize/explain that inner error is not that important (was handled or not an error)
- Users should change their habits and stop considering inner span status to be indicative of an error:

- Tail-based sampling should use more sophisticated policies
- Alerts should be more complicated - e.g. focus on server-side rather than client-side telemetry

Cons:

- Does not work well for metrics: there is no relationship between outer/inner metrics. Outer metric may not exist
- HTTP/gRPC span status set to ERROR does not mean anything on its own

2. Change span status in SpanProcessor.OnEnding

- Users can leverage `SpanProcessor.OnEnding` callback to override status set by instrumentation library
- They will need to filter out specific spans whose status should be changed to UNSET.
 - E.g. all `client` spans sent to specific `server.address` with specific `http.request.method` that have specific `http.response.status_code`.
 - This can be done using context/thread-local/etc

Cons:

- Does not work for metrics
- Hard, error-prone

3. Scoped hooks into instrumentations

```
try (var scope = Instrumentation.customize(
    span -> span
        .setAttribute("error.type", null)
        .setStatus(UNSET),

    measurement -> measurement
        .setAttribute("error.type", null))) {

    client.put("https://storage/create-if-not-exists");
}
```

- New API that would allow app/another instrumentation to register a scoped callback executed at the end of each span/measurement. Callback can read and modify spans and/or measurements.
- Useful for enrichment - <https://github.com/open-telemetry/oteps/pull/207>
- Maybe useful for redaction/sanitization

Cons:

- Needs cross-signal instrumentation API <https://github.com/open-telemetry/oteps/pull/165>
- Tricky API design

Notes:

- Maybe Developer Experience WG could take it over?

4. Native instrumentations provide API

```
var req = new HttpRequestMessage(HttpMethod.Put,  
    "https://storage/create-if-not-exists");  
  
HttpMetricsEnrichmentContext.AddCallback(req,  
    ctx => ctx.SetAttribute("error.type", null));  
  
await httpClient.SendAsync(req);
```

That's similar to what [native .NET HTTP metrics](#) provide now.

Cons:

- Don't have native instrumentations for everything
- API is instrumentation-specific

Proposal

Short term: Recommend changing span status in `SpanProcessor.OnEnding` (Opt2)

Long term: We need to look into customization and enrichment - this comes up a lot (Opt3)