Discussion has moved to this OpenTelemetry RFC.

Zero-Source-Code-Modification OpenTel Instrumentation (i.e., “OpenTel Agents”)

A proposal for a cross-language specification

Status: RFC / draft
Feedback requested: any and all

Background

The purpose of OpenTelemetry is to make robust, portable telemetry a built-in feature of cloud-native software. For some software and some situations, that instrumentation can literally be part of the source code. In other situations, it’s not so simple: for example, we can’t necessarily edit or even recompile some of our software, the OpenTelemetry instrumentation only exists as a plugin, or instrumentation just never rises to the top of the priority list for a service-owner.

One way to navigate situations like these is with a software layer that adds OpenTelemetry instrumentation to a service without modifying the source code for that service. In the APM world, these software layers are often called “agents”, though that term is overloaded and ambiguous so we will try to use it sparingly in this document.

Suggested Reading

- This GitHub issue: Propose an "Auto-Instrumentation SIG"
- Rough notes from the June 11, 2019 meeting following this ^^ issue

Why “cross-language”?

Many people have correctly observed that “agent” design is highly language-dependent. This is certainly true, but there are still higher-level “product” objectives for OpenTelemetry that can guide the design choices we make across languages and help users form a consistent impression of what OpenTelemetry provides (and what it does not).

Requirements

Without further ado, here are a set of requirements for “official” OpenTelemetry efforts to accomplish zero-source-code-modification instrumentation (i.e., “OpenTelemetry agents”) in any given language:

- No *manual* source code modifications allowed
- Licensing must be permissive (e.g., ASL / BSD)

- Packaging must allow vendors to “wrap” the OpenTelemetry piece into a single asset that’s delivered to customers
    - That is, vendors do not want to require users to comprehend both an OpenTel package *and* a vendor-specific package
- Explicit, whitebox OpenTelemetry instrumentation must interoperate with the “automatic” / zero-source-code-modification / blackbox instrumentation.
    - If the blackbox instrumentation starts a Span, whitebox instrumentation must be able to discover it as the active Span (and vice versa)
    - Relatedly, there also must be a way to discover and avoid potential conflicts/overlap/redundancy between explicit whitebox instrumentation and blackbox instrumentation of the same libraries/packages
        - That is, if a developer has already added the “official” OpenTelemetry plugin for, say, gRPC, then when the blackbox instrumentation effort adds gRPC support, it should *not* “double-instrument” it and create a mess of extra spans/etc
- The code in the OpenTelemetry package must not take a hard dependency on any particular vendor/vendors (that sort of functionality should work via a plugin or registry mechanism)
    - Further, the code in the OpenTelemetry package must be isolated to avoid possible conflicts with the host application (e.g., shading in Java, etc)

Nice-to-have properties

- A way to discover and avoid potential conflicts/overlap/redundancy between explicit whitebox instrumentation and blackbox instrumentation of the same libraries/packages
    - That is, if a developer has already added the “official” OpenTelemetry plugin for, say, gRPC, then when the blackbox instrumentation effort adds gRPC support, it should *not* “double-instrument” it and create a mess of extra spans/etc
- Run-time integration (vs compile-time integration)
- Automated and *modular* testing of individual library/package plugins
    - Note that this also makes it easy to test against multiple different versions of any given library
- A fully pluggable architecture, where plugins can be registered at runtime without requiring changes to the central repo at github.com/open-telemetry
    - E.g., for ops teams that want to write a plugin for a proprietary piece of legacy software they are unable to recompile