

OpenTelemetry Semantic Conventions and Avoiding Broken Observability

Dinesh Gurumurthy*

Staff engineer, New York City, USA

Email: Dinesh.gurumurthy@datadoghq.com

Abstract

This paper discusses the OpenTelemetry semantic conventions and how they enable a steady and accurate read on distributed systems. It reviews stability and predictability in telemetry as attribute schemas and metric schemas evolve. It describes how OTEL Weaver can be used in conjunction with its Schema Processor tool to demonstrate that observability remains stable. That need arises from the growth of microservices and cloud-native architectures: a seemingly trivial change in field names or formats related to telemetry data can break dashboards, alerts, and any other analytical queries, creating an operational blind spot. This would combine a failure scenario from real-life cases through a comparison analysis of changes, such as renaming the method attribute to `HTTP.request.method`, as well as deployment. The environment attribute is mapped to `deployment.environment.name`, among others, in the OpenTelemetry schema evolution history. Before suggesting any full solution, this method makes this work different. The OTEL Weaver will join to check schemas, create SDKs, and enforce rules with the Schema Handler, which will transform any incoming data into meaningful version goals on the fly. It is a contract between the service instruments and the watch apparatuses. Main deliverables: conversion of the pathway from reactive-issue-based to proactive-schema first journey with better dependability, as well as upkeep of the observability configs, achieved insertion of Weaver via CI/CD channels, enabled super-fast identification of major issues, and at the same time maintaining compatibility both prior and current using Schema Handler with continuous monitoring tasks. Such practices would help organizations preserve their data semantics as OpenTelemetry evolves together with its standards and ancillary services. Software engineers, site reliability engineers, and observability architects in enterprises that employ distributed systems will find this paper particularly useful.

Received: 6/11/2025

Accepted: 7/22/2025

Published: 8/3/2025

** Corresponding author.*

Keywords: OpenTelemetry; semantic conventions; observability; OTEL Weaver; Schema Processor; schema evolution; microservices.

1. Introduction

When companies deploy microservices systems and adopt cloud-native methodologies, the management of observability naturally becomes more complex. OpenTelemetry has evolved into a leading standard for collecting distributed traces, metrics, and logs. The OpenTelemetry Collector acts as a vendor-neutral agent, which particularizes it as an essential component in most modern observability architectures. Running on customer infrastructure, this collector enables the ingestion of telemetry data—traces, metrics, and logs—by plug-ins available from various sources, the transformation or enrichment of this data, and the exporting of it to one or multiple backends supporting applications used for observing systems. It can be extended using a pipeline paradigm with receivers, processors, exporters, and connectors, which compose these components as needed to address real-world requirements. YAML configuration files define the collector and all its elements. Semantic consistency is crucial for the formation of collector pipelines: attribute names and metric definitions must be stable and predictable across all services. The Semantic Conventions define a standard set of (semantic) attributes that provide meaning to data when collecting, producing, and consuming it. The Semantic Conventions specify, among other things, span names and kind, metric instruments and units, attribute names, types, meaning, and valid values.

However, evolving telemetry schemas can disrupt dashboards, alerts, and queries, leading to significant production issues. This paper will discuss how semantic conventions are maintained, the related problems of drift in those conventions, and new emerging tooling that addresses them.

1.1 Semantic Conventions

Semantic conventions are naming and structural guidelines within OpenTelemetry that help keep the collected data in a unified shape. In other words, semantic conventions define best practices for ensuring that your information maintains a consistent structure, regardless of its origin, allowing for correlation, comparison, or analysis across different tools. They evolve from earlier attempts; for instance, Elastic Common Schema (ECS) was merged into the semantic conventions of OpenTelemetry in 2023 as one canonical way to collate telemetry attributes for easy correlation, comparison, and analysis across tools.

Semantic conventions serve as the basic common language for telemetry data, much like typical road signs in any city. With such conventions in place, the organization ensures that its data is labeled with consistency of meaning attached to it, well labeled, regardless of the instrumentation being used. This standardization enables the tools of observability to work their magic on autopilot, analyzing your data, finding your problems, and surfacing insights, without you having to lift a finger.

For example, an HTTP request trace will carry with it attributes such as *http.request.method*, *user_agent.original*, and *http.request.status_code*, so that key details are consistent across all spans based on HTTP. Similarly, for database queries, attributes such as *db.system*, *db.query.text*, and *db.user* standardize the

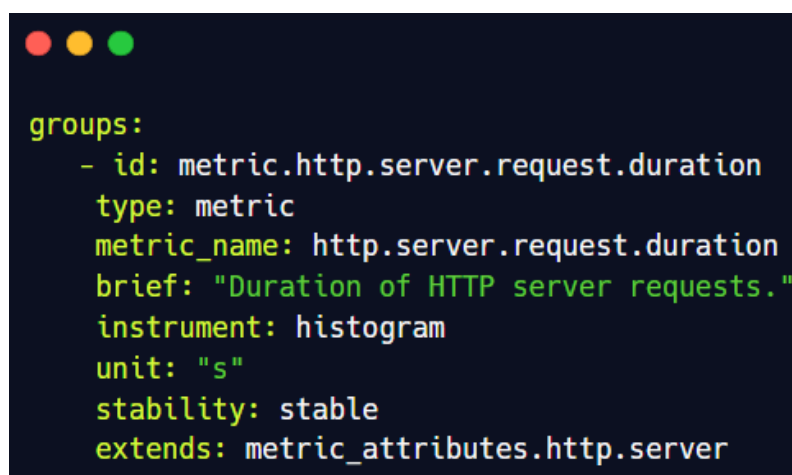
emitted telemetry regardless of the database driver.

Semantic conventions in OpenTelemetry constitute a formally defined vocabulary that prescribes how telemetry attributes must be named and organized. Their purpose is to impose a single logical shape on data emitted by heterogeneous runtimes and libraries, so that every span, metric, or log can be understood in the same way regardless of its origin. Shkuro et al. emphasise that, by turning attribute names and value types into an explicit contract, semantic conventions “provide producers with well-defined expectations for the shape of telemetry and allow consumers to rely on a common set of dimensions when querying the data” [2].

When this uniform contract is respected end-to-end, analytical back-ends can correlate, compare, and aggregate signals without ad-hoc field mapping. The practical use is shown in trace compression by Chen and his colleagues. Since all assisting applications write net and RPC metadata with repeating OpenTelemetry keys, extra attributes can be securely packed and then unpacked later on. This results in significant bandwidth conservation while maintaining 100% analytical acuity [1].

These conventions are also based on earlier domain-specific schemas, such as the Elastic Common Schema. Since 2023, the active process of converging ECS into OpenTelemetry’s specification has made that combined set yet more cemented as a canonical reference to structure observability attributes across the industry.

The OpenTelemetry Semantic Conventions Registry is an open catalog of telemetry definitions, developed by various Special Interest Groups. This registry comprises more than 900 attributes and signals, neatly arranged into 74 domains, providing a solid framework for steady instrumentation [3]. An example of a Semantic convention YAML definition is below. Attribute *stability* defines the stage of evolution of a specific attribute. Stability can be categorized into three values: experimental, developmental, or stable, as illustrated in Figure 1. Once a convention is marked stable, it is ready for production use. Once the conventions are marked stable, breaking changes are no longer allowed.



```
groups:
  - id: metric.http.server.request.duration
    type: metric
    metric_name: http.server.request.duration
    brief: "Duration of HTTP server requests."
    instrument: histogram
    unit: "s"
    stability: stable
    extends: metric_attributes.http.server
```

Figure 1: Semantic Convention YAML for HTTP Server Request Duration Metric

2. Materials and Methodology

The study draws on an analysis of seven key sources: the Tracezip work on efficient distributed-trace compression [1], the positional paper on a schema-first approach by Shkuro and his colleagues [2], and SRECon Day 2 – What a Day! report by Marques [3], the paper on semantic versioning by Lam and his colleagues [4], the practical implementation description in the Kieker Observability Framework by Yang and his colleagues [5], the Mint study on tracing optimization by Huang and his colleagues [6], and the comparative overview of observability tooling and fragmentation challenges by Janes and his colleagues [7].

The theoretical foundation is based on research demonstrating the need for unified and predictable telemetry schemas. In particular, Chen and his colleagues demonstrated that trace compression is both safe and effective when identical attribute keys are strictly adhered to [1], and Shkuro and his colleagues argued for the benefits of a schema-first design, enabling compile-time validation and preventing unforeseen changes to data formats [2].

Methodologically, the study combines several complementary approaches. First, it conducts a comparative analysis of schema evolution—such as the renaming of HTTP attributes during convention stabilization from *http.method* to *http.request.method*—and its impact on the operability of an observability pipeline [4]. Second, it conducts a systematic review of the Semantic Conventions Registry and policy implementations in the Weaver CLI, examining compliance with naming conventions, attribute stability, and the automated generation of schemas and SDKs [3, 5]. Third, it analyzes real-world failure cases caused by the change from *deployment.environment* to *deployment.environment.name*, which resulted in “empty” dashboards and nonfunctional alerts [6].

3. Results and Discussion

3.1 OTEL Weaver: Ensuring Registry Compliance

The Semantic Conventions Registry is an open catalog of nine Special Interest Groups (SIGs). These conventions ensure interoperability across observability tools, reduce ambiguity, and support consistent dashboards and alerts. It falls to the OpenTelemetry community to maintain this registry and also introduce the Weaver CLI, which enables schema-first telemetry enforcement by generating all the OTel SDKs. Through Weaver, developers have an easy way to define and validate a schema registry, as well as manage it, ensuring that signals always take on a consistent structure and naming convention. Standardization raises the level of seamlessness between different services and tool interactions. It enables the generation of type-safe client SDKs within the same CI/CD pipeline, thereby enforcing data governance and privacy compliance. Use Weaver in your development flow to establish more repeatable and maintainable observability practices that will prevent broken dashboards and alerts caused by semantic mismatches. The latest survey clearly illustrates how the fragmentation of tools will hinder adoption without shared semantics [7].

OTEL Weaver is a CLI tool developed to:

- Parsing and validating semantic conventions

- Enforce policies (e.g., naming, stability)
- Generate SDK clients, documentation, and schemas
- Generate Diff registry versions to detect breaking changes

Weaver takes the registry and policies as input for processing the commands on the registry. Weaver uses policies per registry to ensure the integration of the semantic registry. It supports extensible policy definitions using Rego, allowing teams to enforce custom constraints. The policies drive most checks that Weaver can run on the registry, as shown in Figure 2.

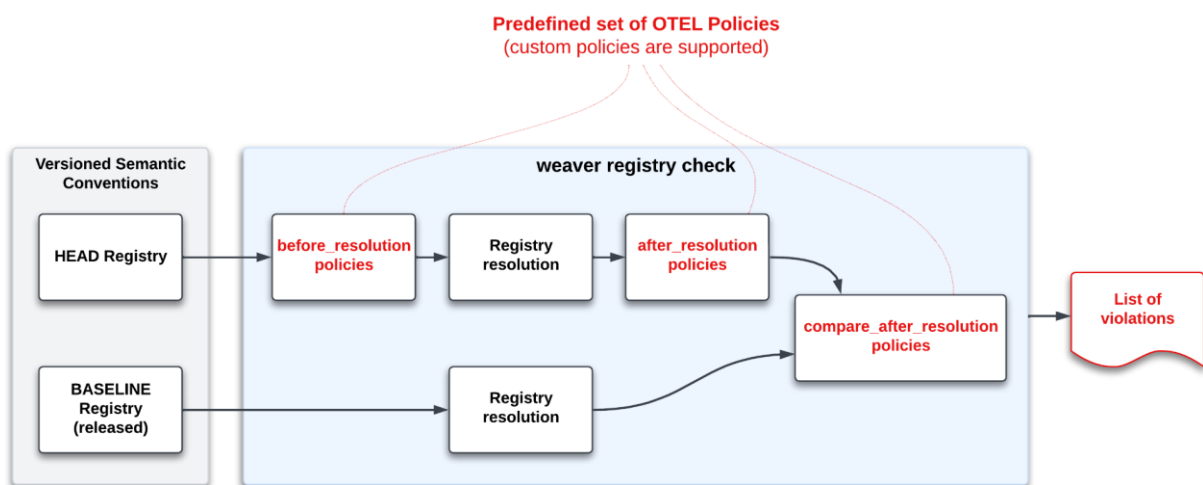


Figure 2: Weaver registry check

In the Otel ecosystem, the Weaver registry *generate* command places a critical role. The generate command generates the conventions for each OTel SDK, along with documentation. The documentation is a Markdown catalog referred to on the public OpenTelemetry website, as shown in Figure 3.

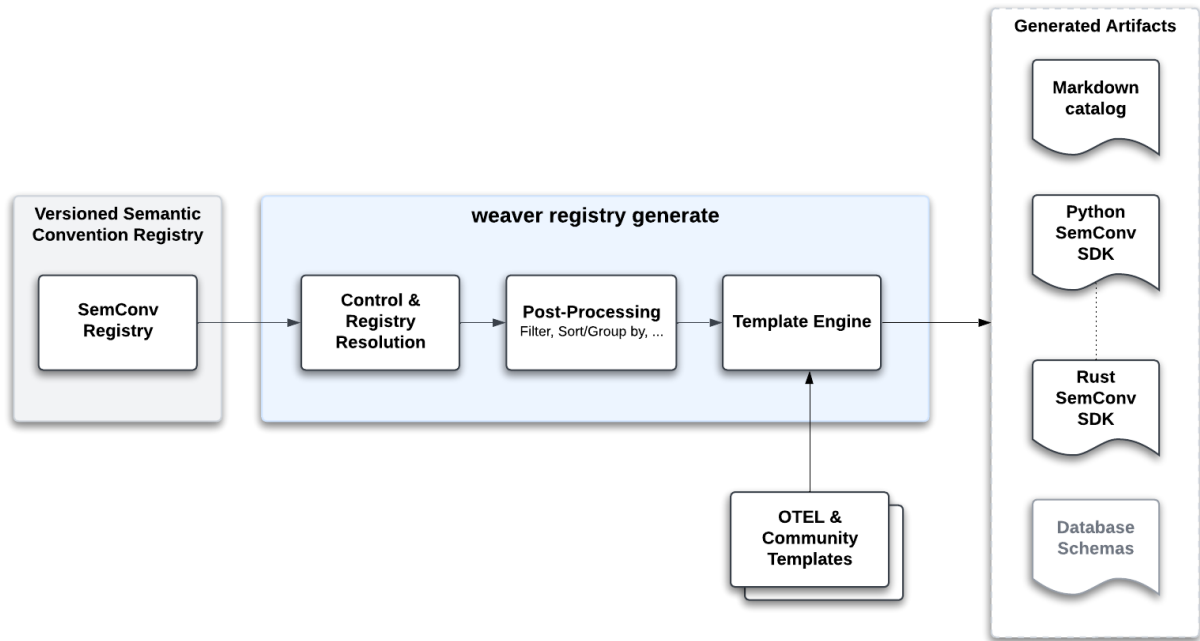


Figure 3: Weaver Registry Generate Workflow for Semantic Convention Artifact Production

The next step is to use the Weaver diff command to generate the schema transformation and migration guides. Schema transformation files play a crucial role in the automated translation of telemetry within the collector, as illustrated in Figure 4.

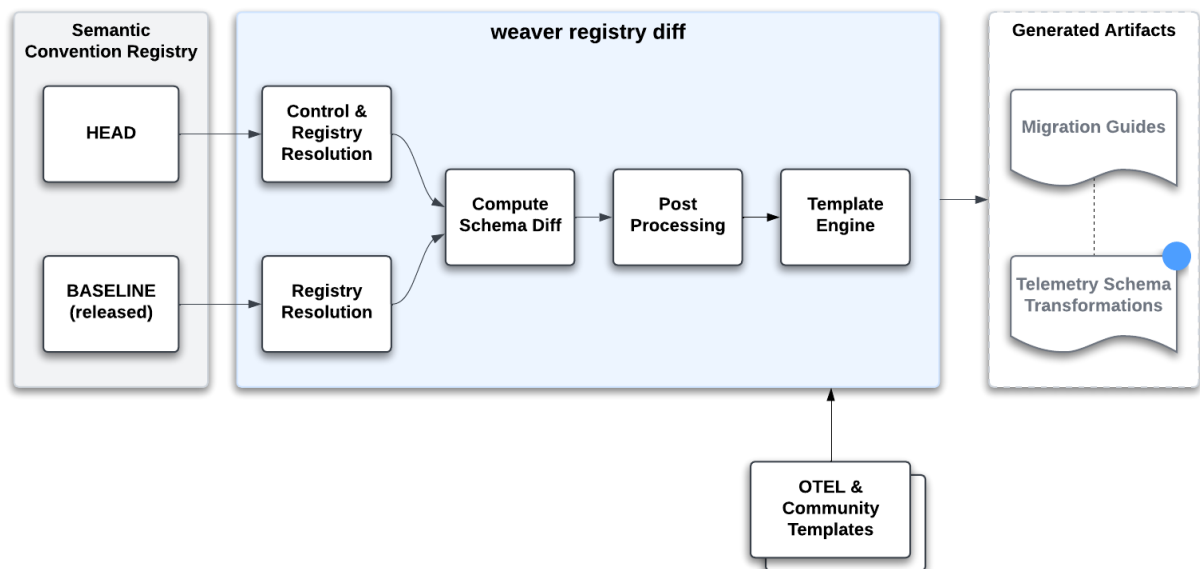


Figure 4: Weaver Registry Diff Workflow for Comparing Telemetry Schema and Creating Artifacts

Weaver can be integrated into CI/CD in a way that checks telemetry produced by the app at build time, allowing developers to catch semantic issues as early as possible. This is still in the process of piloting telemetry testing

during development, but it should allow us to identify the problems of semantic drift during development. Weaver can help enforce an application that produces telemetry to match the registry. It generates a report, whether any semantics are present or not, that the developer can further analyze.

3.2 Schema Evolution Problems

Although semantic conventions offer significant advantages, they also present challenges as they evolve. Schema evolution is the process of modifying the structure of data schemas while ensuring their compatibility with the existing data. In the case of OpenTelemetry, it becomes challenging when the semantic conventions change.

For example, by the time HTTP semantic conventions became stable, several attributes, such as HTTP method, became *http.request.method* and *http.client_id* became *client.address*. Such a change, although it improves the quality and consistency of the conventions, can disrupt existing observability pipelines if not managed correctly [4].

A service team upgraded their OpenTelemetry SDK version and, unknowingly, started producing telemetry with a newer semantic convention, version 1.27. In this updated version, the attribute name '*deployment.environment*' has been changed to '*deployment.environment.name*'.

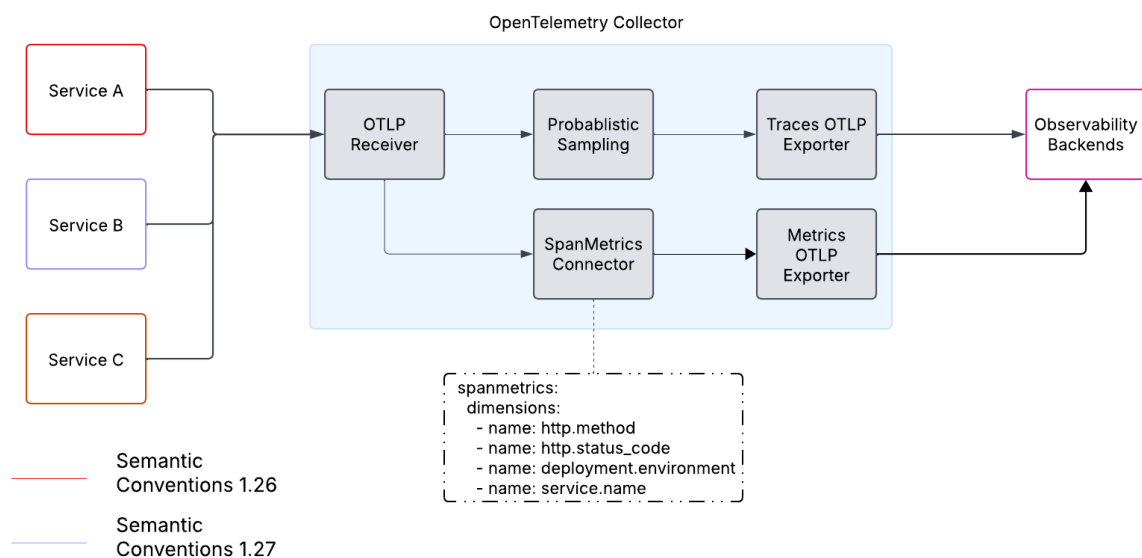


Figure 5: OpenTelemetry Collector Pipeline with Probabilistic Sampling and SpanMetrics Connector

As a result, metrics were no longer computed over the environment dimension, leading to empty dashboards and non-functional alerts. This left the team "flying blind" without alerts, creating a significant operational risk. These issues happened when HTTP semantics were locked, causing client setups to break simply because the HTTP method was changed to an HTTP request method.

3.3 The Schema Processor: A Decoupling Layer

The Schema Processor fixes this issue by mapping the incoming telemetry data to a target semantic version. This mapping lets organizations set up their OpenTelemetry collector pipeline configuration with a stable semantic version that aligns with the processor's target version.

Place the Schema Processor as the first processor in the collector pipeline to modify or upgrade incoming telemetry signals, ensuring they conform to the desired version. This way, all later processing and visualization tools work correctly, regardless of the version of semantic rules used by the data source.

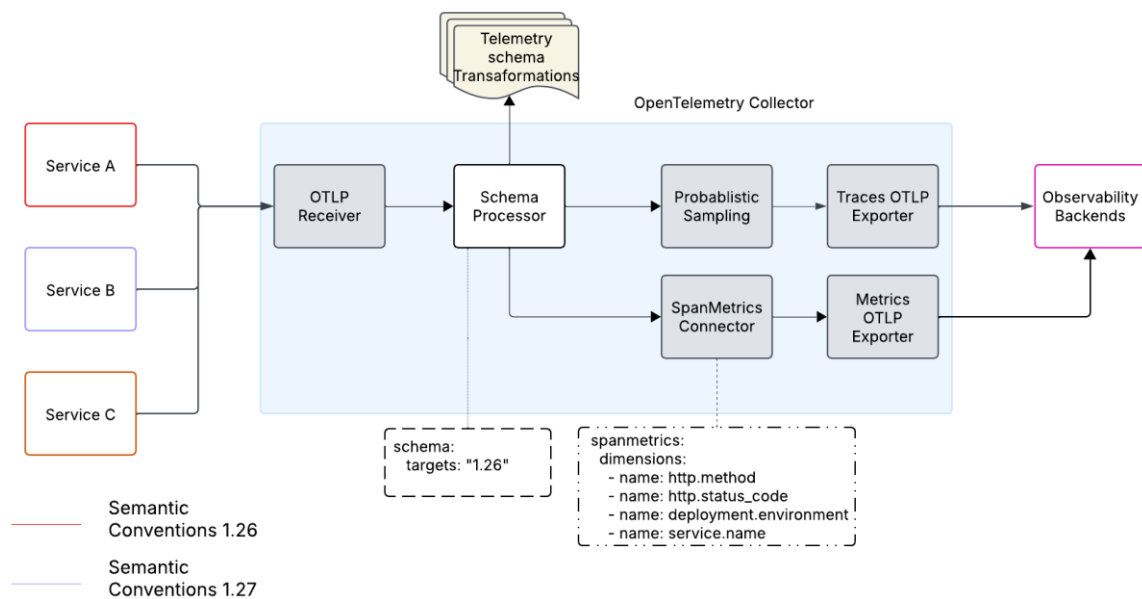


Figure 6: OpenTelemetry Collector Pipeline with Schema Processor and Telemetry Schema Transformations

The Schema Processor performs these transformations using:

1. OTLP data schema URL - The OTLP data format includes an optional field *schema_url* at resource levels and signal levels that specifies the semantic version of the payload format¹
2. Telemetry schema transformations—Weaver produces these when running the *registry diff*. These transformations specify how to convert between different schema versions.

Using a schema processor, service owners have been effectively decoupled from collector configuration and downstream dependencies, allowing individual services to upgrade or adjust their SDKs without forcing simultaneous changes to the collector setup [5].

The current schema file format exhibits several practical limitations that have become apparent through its use in real-world scenarios and manifest as constraints within the Schema Processor. First, because the *schema_url* field in OTLP payloads is optional and is not set by all SDKs, the processor simply ignores any

payloads lacking this field. Second, non-bijective or breaking transformations cannot be handled automatically; any change that is not strictly reversible falls outside the processor's capabilities. Third, complex transformations are not supported by the version 1.0 format, hence expressiveness. Lastly, manual schema updates are performed without validation or consistency checking. As a result of such deficiencies, OpenTelemetry has directed efforts towards enhancements. Weaver is expanding to support multiple registries, reflecting the common need for vendors to define their layers on top of the core registry. A new, self-contained schema file format is under development to enable both upward and downward transformations. Efforts are also underway to harden the Schema Processor for production-level reliability and to integrate an OTEL Weaver live-check feature for compliance and coverage testing during development. In parallel, type-safe client SDK generation is being pursued to provide stronger guarantees for SDK users, and a broader schema-first approach to telemetry—where schemas are defined before implementation—is being promoted. These initiatives will be codified in a forthcoming OTep that overhauls the existing schema file format [6].

These improvements are part of a broader shift toward schema-first observability, aligning with API design best practices, such as RPC and GraphQL, where the schema serves as the definitive contract between services and clients. The upfront definition of telemetry schemas delivers compile-time validation, richer tooling support, more predictable instrumentation, and an “observability-by-design” mindset. It does not only propagate consistency across services—eliminating the risk of breaking changes that can ripple through whole systems—it brings development and operations teams together by making shared expectations for telemetry data explicit.

4. Conclusion

Strong, steady telemetry information is key as watchfulness lines become more complex. The meaning rules in OpenTelemetry lay a vital foundation for clear and useful telemetry information from distributed systems. Yet, changing such rules gives groups huge tasks to ensure their check lines work just right. If not dealt with properly, shifts in meaning rules can disrupt control panels, complicate searches, and make repairs a significant hassle. The problems have been mirrored for decades in the evolution of databases and APIs, where, again, versioning schemas and compatibility have been matters of enormous academic and industrial concern.

OTEL Weaver and the Schema Processor represent a cultural shift in tooling from reactive remediation to proactive enforcement of telemetry standards by adopting a schema-first design approach, enabling teams to ensure stable, predictable, and effective observability systems. OTEL Weaver enables them to validate, document, and enforce policies on semantic conventions, ensuring everything aligns with best practices. The Schema Processor enables the easy evolution of semantic conventions by transforming incoming telemetry data to match a target schema version, thereby decoupling service instrumentation from collector configuration. As the OpenTelemetry ecosystem continues to mature, multi-registry support, working in conjunction with improved schema file formats and schema-first approaches, is planned to make observability pipelines even more resilient while maintaining high flexibility. Organizations that adopt these tools and practices will be better positioned to maintain effective observability even as their systems and the OpenTelemetry standard evolve.

References

- [1] Z. Chen, J. Pu, and Z. Zheng, “Tracezip: Efficient Distributed Tracing via Trace Compression,” *Proceedings of the ACM on Software Engineering*, vol. 2, no. ISSTA, pp. 411–433, Jun. 2025, doi: <https://doi.org/10.1145/3728888>.
- [2] Y. Shkuro, B. Renard, and A. Singh, “Positional Paper: Schema-First Application Telemetry,” *Arxiv Software Engineering*, 2022, doi: <https://doi.org/10.48550/arXiv.2206.11380>.
- [3] R. Marques, “SRECon Day 2 – What a Day!” *LinkedIn*, Mar. 27, 2025. https://www.linkedin.com/posts/richard-marques-26b3a14_srecon-soubradesco-activity-7311031141212340224-4YYP (accessed Jun. 15, 2025).
- [4] P. Lam, J. Dietrich, and D. J. Pearce, “Putting the semantics into semantic versioning,” *Proceedings of the 2020 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Nov. 2020, doi: <https://doi.org/10.1145/3426428.3426922>.
- [5] S. Yang, D. G. Reichelt, R. Jung, M. Hansson, and W. Hasselbring, “The Kicker Observability Framework Version 2,” *Arxiv Software Engineering*, pp. 11–15, May 2025, doi: <https://doi.org/10.1145/3680256.3721972>.
- [6] H. Huang et al., “Mint: Cost-Efficient Tracing with All Requests Collection via Commonality and Variability Analysis,” *Arxiv Software Engineering*, pp. 683–697, Feb. 2025, doi: <https://doi.org/10.1145/3669940.3707287>.
- [7] A. Janes, X. Li, and V. Lenarduzzi, “Open tracing tools: Overview and critical comparison,” *Journal of Systems and Software*, vol. 204, p. 111793, Oct. 2023, doi: <https://doi.org/10.1016/j.jss.2023.111793>.