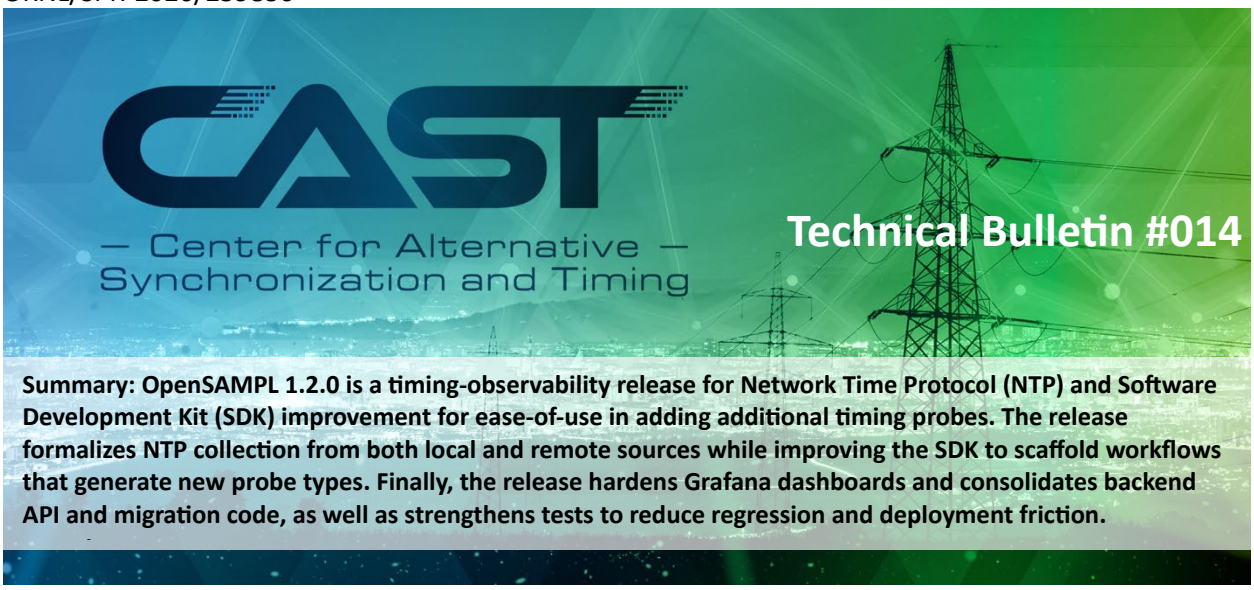
ORNL/SPR-2026/259830



Summary: OpenSAMPL 1.2.0 is a timing-observability release for Network Time Protocol (NTP) and Software Development Kit (SDK) improvement for ease-of-use in adding additional timing probes. The release formalizes NTP collection from both local and remote sources while improving the SDK to scaffold workflows that generate new probe types. Finally, the release hardens Grafana dashboards and consolidates backend API and migration code, as well as strengthens tests to reduce regression and deployment friction.

OpenSAMPL 1.2.0 Launch

Introduction

OpenSAMPL's 1.2.0 timing-observability release makes Network Time Protocol (NTP) a first-class citizen in the platform and improves the SDK to make new probe families faster to build and safer to integrate. The release formalizes NTP collection (local and remote), expands timing metrics and metadata needed to correctly interpret reference relationships, and hardens dashboards to avoid common query/type failures. In parallel, the Software Development Kit (SDK) and Command Line Interface (CLI) now support a scaffolding workflow (`opensampl create`) that can generate a new vendor/probe "starter kit," including optional timing collection mixins and a path to create the necessary database tables via `alembic` migrations.

Executive Summary

1. What Shipped in 1.2.0

OpenSAMPL 1.2.0 delivers three connected improvements that reinforce each other. First, it improves the SDK and CLI with an experimental probe scaffolding workflow that generates the baseline parser and metadata/table structures needed to stand up a new probe family inside the repository, with an option to prefill timing-collection hooks for teams building additional timing integrations. Second, it adds first-class NTP vendor/probe support through this improved extension model, including clear collection semantics, explicit loading behavior, and dashboards that treat NTP-backed "references" in a safe and consistent way. Third, it strengthens platform reliability by hardening dashboard queries, consolidating backend and migration code into the OpenSAMPL project with aligned Docker guidance, and improving tests and CI so database- and geolocation-adjacent behaviors are less likely to regress.

U.S. DEPARTMENT OF ENERGY | OFFICE OF ELECTRICITY

2. *Why this Matters*

Timing health underpins trustworthy observability: if NTP degrades, timestamps can drift and quietly undermine incident timelines, cross-system correlation, and root-cause analysis. By making NTP measurable in the same operational model as other OpenSAMPL probe families, the platform turns a hidden dependency into something teams can trend, alert on, and diagnose. At the same time, the new scaffolding workflow reduces the cost and risk of adding probe families by standardizing the generated structure for parsers, metadata models, and tables, while dashboard hardening and CI/test improvements reduce operational friction and increase confidence that new extensions can be shipped without breaking existing dashboards or environments.

First-class NTP observability

1. *Why NTP as a first-class vendor family?*

NTP is now integrated as a supported vendor/probe family rather than being handled as an external preprocessing step. The practical motivation is that time synchronization should be observable and debuggable using the same OpenSAMPL concepts used elsewhere, instead of relying on implicit assumptions that system time is "good enough." With this change, timing quality becomes part of the platform's measurable surface area, which makes it easier to detect degradation early and to explain timing-related anomalies when other signals appear inconsistent.

2. *Where does the NTP extension live?*

NTP support is implemented through OpenSAMPL's extension model (for example under `opensampl.vendors.ntp`), keeping the integration consistent with existing vendor/probe patterns. This avoids introducing special-case architecture and helps ensure that the NTP work remains maintainable over time, while also setting a clear precedent for future timing-related probe families. The intent is that NTP behaves like a native part of OpenSAMPL rather than a bolt-on data source.

3. *Collection modes: local and remote*

The NTP probe supports two distinct collection modes to match how NTP is used in real environments. In `local` mode, OpenSAMPL uses host-available tools such as `chronyc`, `ntpq`, `timedatectl`, and `systemctl` to infer synchronization state and collect whatever metrics the system can expose. In `remote` mode, OpenSAMPL performs a direct NTP query using `ntplib` and extracts timing-related values such as offset, delay, stratum, and root dispersion from the response. These two modes enable both "is this host correctly synchronized?" and "from this collector, what does upstream time quality look like?" monitoring patterns, which is especially important across segmented networks and multi-site deployments.

4. *Metrics: what you can now trend and alert on*

OpenSAMPL 1.2.0 adds a set of NTP-focused metrics designed to reflect both acute failures and gradual degradation. Jitter, delay, stratum, reachability, root delay, root dispersion, poll interval, and sync health can now be captured and stored as time-series data. The intent is to make the

warning signs visible before time drift becomes operationally obvious, and to provide enough protocol-relevant context that operators can distinguish between network transport problems, upstream instability, and local clock-discipline issues.

5. *Probe identity model: target vs collection probe*

Because NTP measurements have an inherent "observer versus observed" relationship, the NTP model uses two probe identities: a target probe representing the NTP server or local host being measured, and a collection probe representing the host performing the observation. OpenSAMPL writes time-series rows for the target probe while using the collection probe as the reference dimension, which keeps the data aligned with existing reference semantics used for joins and filtering. Importantly, this preserves the meaning of "reference" as an OpenSAMPL modeling concept without implying that GNSS is the underlining timing reference.

6. *Metadata and loading semantics*

Each NTP artifact carries header metadata that describes the relationship between the target and collection probes, along with the time-series rows for the metrics collected. During load, OpenSAMPL ensures there are entries for both the collection host and target probe in `probe_metadata`, creating new ones or updating existing ones as needed. For both, it will also add or update the NTP specific context to `ntp_metadata`, including a `reference: true` flag for the collection host. After the metadata is validated, the mechanism will next enter each collected metric into `probe_data` using the collection probe as the reference dimension. The impact is that NTP data remains filterable and comparable through the same reference and metadata tables used across vendors, while still retaining the crucial "who observed whom" context needed for correct interpretation.

7. *Jitter semantics: measured vs estimated*

Due to the mechanism employed to collect Remote NTP samples, the responses themselves do not contain RFC5905 peer jitter. OpenSAMPL fills the `jitter_s` metric for remote probes with a documented surrogate derived from round-trip delay and root dispersion so that they retain an `NTP Jitter` value. Local `chronyc` and `ntpq` collection paths continue to populate `jitter_s` from the jitter/RMS fields exposed by those tools, which are based on the local daemon's sample history. This keeps dashboards populated and supports consistent trending, but it also means operators should interpret jitter differently depending on whether the collection path is local (measured) or remote (estimated from limited information).

8. *Geolocation behavior at ingest time*

Geolocation is intentionally handled at metadata ingest time rather than in Grafana to keep dashboards reproducible from database state alone. The loader uses a `ENABLE_GEOLOCATE` environment variable and the geolocation helper to decide whether and how to create `locations` rows: explicit overrides take precedence; public IP resolution can trigger a lookup via `ip-api.com`; private or loopback addresses can map to default lab coordinates; and if the system cannot confidently name and place a location, it skips location creation. This design isolates external lookups to ingestion and avoids dashboards that depend on live enrichment behavior.

9. *Dashboard semantics and reference-safe terminology*

Dashboards were updated to be more robust and more semantically explicit for NTP-backed timing paths. “Reference” is used intentionally to mean the OpenSAMPL reference dimension that powers joins and variable resolution, not a claim about the physical provenance of time (for example, GNSS truth). Compact source/reference metadata views were also added so operators can quickly understand what a timing series is anchored to, which reduces ambiguity during triage and makes the NTP data easier to reason about in mixed-vendor dashboards.

SDK Improvements: scaffolding new probe types

1. *What is the SDK feature?*

OpenSAMPL 1.2.0 adds an SDK/CLI workflow to scaffold a new vendor/probe family inside a local repository clone using `opensampl create`. The goal is to reduce the upfront work and inconsistency that often come with new integrations by generating a standard starting point: parser structure, metadata model/table definitions, and the connective tissue needed to fit cleanly into the OpenSAMPL ecosystem.

2. *Recommended workflow*

The intended workflow begins by cloning the repository and setting up a development environment, then running `opensampl create` with a config file that describes the new probe type. If the new probe will also collect timing signals, the `--collect-mixin` flag can be used to prefill the collection-related functions so implementers start from a consistent template. After completing the generated parser and metadata model (and any required collector mixins), developers can initialize or update the database using `opensampl init` or `opensampl create --update-db ...` to create the new tables needed for the probe family.

3. *Configuration drives the generated scaffold*

The scaffolding process is configuration-driven so probe families are defined explicitly rather than implicitly. The config specifies the probe type name and can optionally override class, module, ORM, and table naming, while `metadata_fields` defines the metadata columns that will exist for the new probe type (with optional SQLAlchemy type hints and a default of `Text`). This approach makes schema and metadata decisions visible and repeatable, reduces the likelihood of naming/type drift across vendors, and speeds up initial implementation because the structure is generated rather than hand-written. The generated structure includes implementation instructions directly in the functions, making the experience much easier for developers to proceed with confidence.

4. *Database implications and contribution expectations*

Because probe scaffolding can create or update database structures, the workflow intentionally ties code generation to a supported database update path. When a new probe type is intended for upstream contribution, the expectation is that alembic revisions will accompany the generated code to apply any required database changes. This ensures the repository’s migration history stays coherent and other users can adopt the probe type without manual database surgery.

Reliability, packaging, and test/CI hardening

1. *Dashboard query hardening*

Dashboard queries and variables were hardened to handle empty or unset filters more gracefully and to avoid failures caused by varchar-versus-UUID mismatches. The operational effect is fewer brittle dashboards, especially during exploratory usage, partial deployments, or upgrade transitions where templating variables may temporarily resolve to empty values or inconsistent types.

2. *Consolidation of backend API and database migrations into OpenSAMPL*

OpenSAMPL now includes the backend API and alembic migration code within the main project, along with Docker image information. This consolidation is primarily about release coherence: schema migrations, API behavior, and probe/collector functionality evolve together under a single version, reducing the risk of drift and making deployments and upgrades simpler to plan and execute.

3. *Developer experience improvement via Docker Compose*

A developer-oriented docker-compose flow installs OpenSAMPL as editable on the backend image. This is designed to shorten iteration cycles and reduce environment friction, especially for changes that span API code, migrations, and collection/loading semantics, where developers benefit from quickly testing real end-to-end behavior in a containerized environment.

4. *Tests and Continuous Integration (CI) improvements*

This release expands unit and integration-style coverage for the NTP collection and loading paths, geolocation helpers, and seeded database defaults to reduce the risk of subtle correctness regressions in timing workflows. Integration-style tests were also reworked to use the project MockDB harness instead of requiring a locally spawned PostgreSQL instance, improving portability and reliability for developers and CI. At the same time, CI now installs PostgreSQL/PostGIS tooling so environments that do rely on `pytest-postgresql`-style provisioning can still be supported when needed.

5. *Bug fixes with practical impact*

Several fixes improve correctness and reduce noise in day-to-day development and CI. MockDB seeded default metric UUID handling now correctly points to the `UNKNOWN` metric as intended, preventing mis-seeded identifiers from distorting tests. Random data duration generation was corrected so it no longer always produces a one-hour duration, improving the realism of demo/test behavior. CI ordering was fixed to prevent lint failures caused by workflow sequencing, and the `mkdocs-click` variable assignment issue was resolved to improve documentation build reliability.

Conclusion

OpenSAMPL 1.2.0 makes timing health measurable and interpretable by integrating NTP as a first-class probe family with clear collection/loading semantics, richer NTP metrics, and

dashboards that are explicit about reference meaning. It also reduces the cost of extending the platform by adding an experimental scaffolding workflow that standardizes how new probe families are generated and brought into the database. Combined with dashboard hardening, repository consolidation, and stronger tests and CI, the release improves both operational trust in time-based data and the maintainability of future extensions.

The Center for Alternative Synchronization and Timing (CAST) at Oak Ridge National Laboratory (ORNL) performs research, development, testing, evaluation, and technical assistance to enabled resilient timing and synchronization for the power grid. Working closely with power utilities, timing hardware and software vendors, network operators, and federal stakeholders, CAST helps develop and validate alternative timing architectures to augment GPS time. CAST also translates and transfers ORNL's research and development (R&D) advances in secure timing and grid communications to power sector applications, and engages across the broader timing community to develop best practices to ensure the resilience of US critical infrastructure. CAST is sponsored by DOE's Office of Electricity. Visit https://cast.ornl.gov for more information.