x_make_telemetry_vector_x â€” Bonsai Discipline For Event Streams

The visitor constellation spews telemetry with the enthusiasm of a ruptured mainline. I built `x_make_telemetry_vector_x` so every service stops improvising envelopes and starts emitting disciplined payloads the command center can trust. Feed it a raw dict, get back a UTC-normalised shell with version tags, metadata, and the scars baked in.

Mission Log

- Canonicalise event identifiers, sources, actions, and timestamps without hand-written shims.
- Stamp every record with a telemetry version and ingestion time so audits can reconstruct the blast radius of a bug.
- Ship type hints and a Pydantic model so static analysis whines before production does.
- Replace the copy-pasted normalisers littering orchestrators and visitors, one repo at a time.

Instrumentation

- Python 3.10 or newer.
- `pydantic>=2.6` and `typing-extensions>=4.9` â€” already declared in `pyproject.toml`.
- Ruff, Black, MyPy, and Pyright for the QA gauntlet; pytest for behavioural assurance.
- Optional: cProfile via `scripts/profile_cpu.py` when you want proof that normalisation overhead stays negligible.

Operating Procedure

1. Create your environment: `python -m venv .venv` then `\.venv\Scripts\Activate.ps1`.
2. Install the tooling: `python -m pip install --upgrade pip build` followed by `pip install -e .[dev]` once we finish wiring extras.
3. Normalise a payload: ```python from x_make_telemetry_vector_x import normalize_payload

envelope = normalize_payload({ "id": "abc-123", "source": "visitor", "action": "ingest", "timestamp": "2025-11-12T10:30:00Z", "payload": {"repo": "x_make_common_x"}, }) `` ` 4. Archive profiling and benchmark evidence under`Change Control/0.20.13/evidence/` before merging replacements.

Evidence Checks

Check	Command
Formatting sweep	`python -m black .`
Lint interrogation	`python -m ruff check .`
Type audit	`python -m mypy .`
Static contract scan	`python -m pyright`
Unit tests	`pytest tests/test_telemetry_vector.py`
Package build	`python scripts/package_telemetry_vector.py`

Integration Doctrine

- Replace any service-level normaliser with `normalize_payload` and log the before/after benchmarks in Change Control.
- Keep `ingested_at` timestamps tied to UTC and document any downstream truncation before shipping.
- Record adoption notes in `Change Control/0.20.13/evidence/x_make_telemetry_vector_x-adoption.md` for every repo you touch.
- Legacy clone automation remains archived in-tree until the next freezeâ€"do not revive it without my signature.

Sole Architect's Note

I architected this package as part of the 0.20.13 cleanup offensive. No third-party improvisation, no half-measures. If a consumer leaks divergent telemetry again, the fault lies with whoever ignored this kit.

Staffing Reality Check

- Reproducing this without the lab's LLM assistance demands a senior Python engineer fluent in Pydantic, telemetry pipelines, and QA automation.
- Timeline: 4â€"5 engineer-weeks to reconstruct the schema, benchmarks, and evidence bundle to my standard.
- Cost band: USD 35kâ€"45k, excluding the operational burn.

Technical Footprint

- Core Language: Python 3.10+ with `datetime`, `typing`, and Pydantic v2.

- External Hooks: None beyond standard library and declared dependencies.
- Quality Net: Ruff, Black, MyPy, Pyright, pytest.
- Integration Points: `x_make_common_x`, the visitor orchestrators, and any service that wants telemetry consistency without growing yet another bonsai branch.