# PEM · PERSONAL ENVIRONMENTAL MONITOR

Technical Specification and Engineering Record

| Field | Value |
|---|---|
| Document reference | PEM-PUB-001 |
| Version | 1.3 —PUBLICATION APPROVED |
| Date | 20th July 2026 |
| Author | Martin Paul |
| System | PEM |
| System version described | Application service v8.8.1, bridge v1.1 (as at 7th August 2026) |
| Classification | Public |
| Consolidated from | PEM-SPEC-001 v1.0 (system specification); PEM-TD-001 v1.2 (database service read optimisation and incident record) |

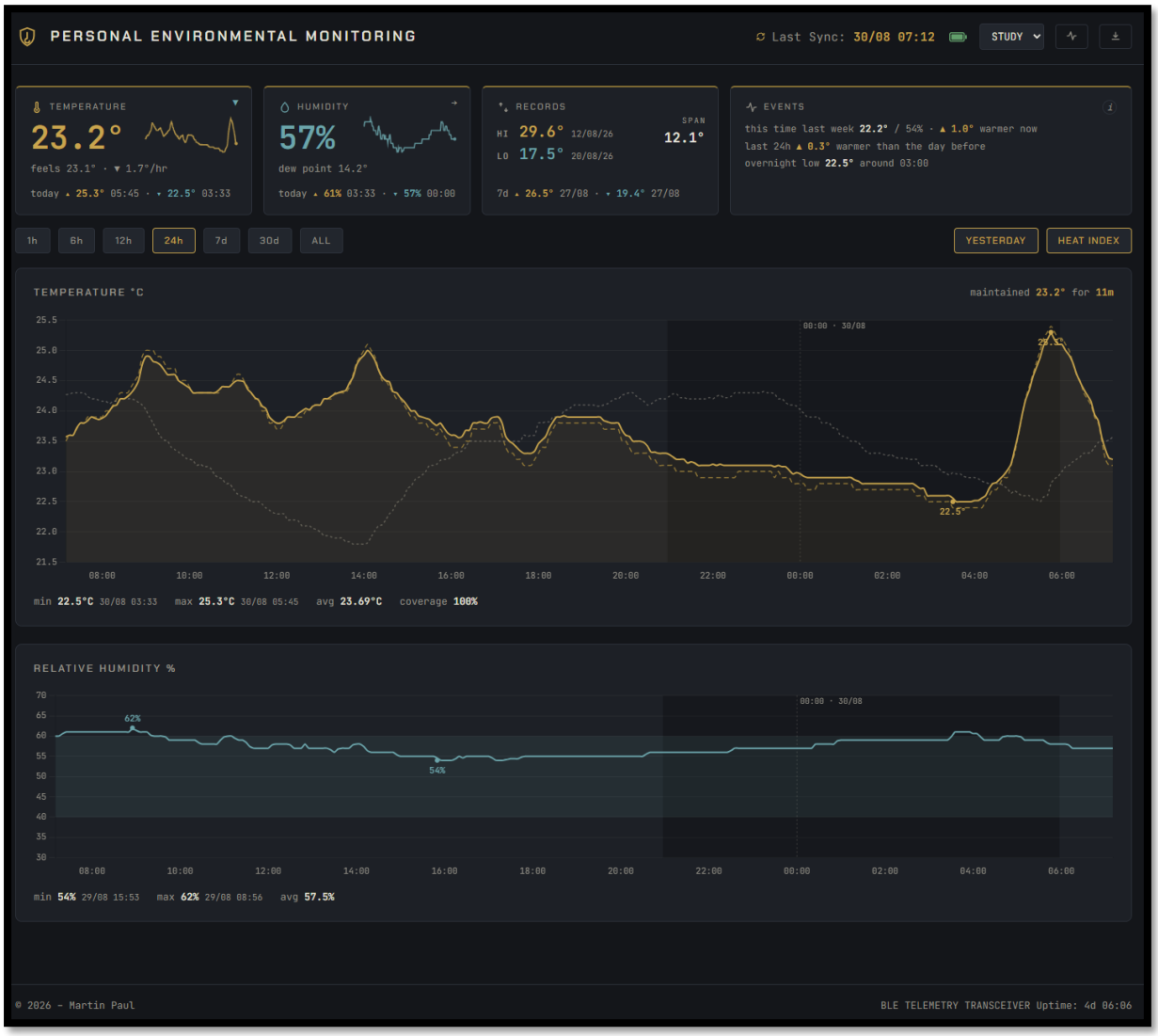
*This document describes a personal engineering project: a private environmental monitoring system built from a consumer Bluetooth sensor, a dedicated bridge host and end-point infrastructure, rather than an off-the-shelf product. It is published in the spirit of showing the working: the architecture, the data model, the measured cost engineering and one genuinely interesting production incident.*

## Contents

# 1. Introduction

PEM records the temperature and relative humidity of one room at one-minute resolution and presents the history as a private instrument-panel dashboard. Commercial products exist that do this. *PEM exists because building the instrument was the point*: every layer, from decoding the sensor's Bluetooth advertisements to the economics of the database engine underneath the charts, was a problem worth solving deliberately.

The system has three defining commitments. First, full-resolution data is never thrown away: one reading per minute, kept forever, with all long-range views served from derived aggregates rather than by degrading the source. Second, no silent failure anywhere: every rejected line, skipped minute and stale sensor is surfaced, logged or rendered. Third, least privilege by construction: the public face of the system discloses nothing, and every access tier gets exactly the capability it needs and no more.





*Screenshot of PEM instrument panel and System Status*

## 2. System architecture

A battery-powered commercial Bluetooth Low Energy temperature and humidity sensor broadcasts its current reading every few seconds. A dedicated network-segregated host (the bridge) listens passively, with no pairing involved, and posts one reading per minute over HTTPS to the end-point infrastructure. The application service stores raw readings and hourly rollups in a managed SQL database and serves the dashboard. The sensor's local history, synchronised by the vendor application, provides a backfill path that makes a bridge outage recoverable.

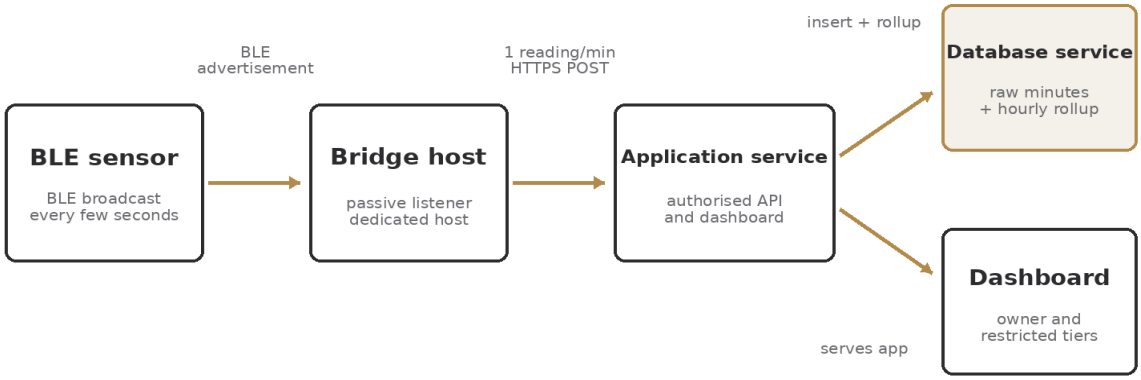



*Figure 1 — PEM data path from sensor to dashboard*

| Component | Detail |
|---|---|
| Sensor | Commercial BLE temperature/humidity sensor, battery powered, with local history buffer |
| Bridge | Dedicated network-segregated host with independent DNS filtering |
| Bridge software | Passive listener with automatic startup and recovery, plus a local retry buffer |
| Application service | Hosted application service, deployed through the deployment interface |
| Database | Managed SQL database: immutable minute data, derived hourly rollups and status metadata |
| Dashboard | Browser application served from the application service, using client-side charting and web fonts |
| Backfill | Vendor application CSV export, imported through the dashboard; overlaps are duplicate-safe by design |

## 3. The sensor and the BLE decode

The sensor broadcasts temperature, relative humidity, battery state and framing information in manufacturer-specific BLE advertisements. No pairing is required; the bridge listens passively, so the sensor remains independent and the vendor application continues to work in parallel. The decoder was validated against the physical unit and an independent reference implementation.

| Field | Representation | Handling |
|---|---|---|
| Temperature | Signed numeric value | Decoded to degrees Celsius |
| Relative humidity | Unsigned numeric value | Decoded to %RH |
| Battery state | Compact status value | Normalised to the sensor's reported levels |

| Field | Representation | Handling |
|---|---|---|
| Framing information | Device / frame marker | Used for validation only |

Known invalid advertisement frames are rejected before ingestion, matching the behaviour of the reference implementation.

# 4. The bridge

- The bridge is a dedicated network-segregated host running a passive BLE listener. It starts automatically and is designed to recover after host or power interruptions.
- Credentials are held in operating-system protected storage and are not embedded in the bridge code.
- Undeliverable readings are buffered locally and retried in batches after connectivity returns. Duplicate-safe ingestion means retries require no coordination.

# 5. Data model and conventions

## 5.1 Storage model

```
The storage model has two principal layers: an immutable minute-resolution readings store
and a derived hourly aggregate store. Supporting status and summary metadata are held
separately. Data is indexed by sensor location and time so common range queries avoid
unnecessary scans.
```

Indexes are retained only where they support common query paths. A humidity-specific index was deliberately dropped because the database service meters index maintenance while long-range humidity extremes are served from the rollup. The design favours efficient per-location time scans without taxing every write for a rare fallback path.

## 5.2 Conventions

- Timestamps are naive wall-clock epochs. The sensor's local calendar time is stored and rendered without any timezone or DST arithmetic anywhere in the system, deliberately: the question PEM answers is what the room was like at ten past nine, not what UTC instant that was.
- Readings are immutable. Raw ingestion is conflict-safe and ignores duplicate records, so overlapping imports cost no data writes and any re-import is safe.
- The hourly rollup is derived data, recomputed for touched hours on every ingest, and can always be rebuilt from raw. The raw table remains the single source of truth.
- Chart bucket sizes are chosen server-side from a fixed whitelist and inserted as validated integer constants rather than bound as generic numeric parameters. The reason is Incident INC-01 (Section 11).

# 6. Ingest paths

## 6.1 Live: the bridge (primary)

The bridge decodes advertisements continuously and, on each minute boundary, posts the freshest reading with its location. If the freshest reading is stale, indicating the sensor is out of range or off, the minute is skipped with a logged warning rather than posted. Battery level is included only when it changes, plus once per session start.

### 6.2 Backfill: vendor app CSV (secondary)

The vendor app's emailed export can be dropped anywhere on the dashboard, which is a whole-page drop target. The parser handles the export's genuine quirks: a UTF-8 byte-order mark stamped on every line, a preamble row, DD/MM/YYYY dates and 12-hour AM/PM times. Rejected lines are reported with line numbers and reasons; nothing is skipped silently. Because duplicates cost nothing, generous date overlaps are encouraged. The sensor's ~20-day internal buffer is the only hard deadline in the system.

## 7. API surface

The application exposes a small authorised interface divided into read, ingest, import and export functions. Access decisions are enforced server-side; restricted roles are read-only and authenticated activity is logged.

| Interface | Access | Purpose |
|---|---|---|
| Read interface | Owner and restricted read-only roles | Supplies authorised dashboard data and summaries |
| Ingest interface | Owner and bridge identity | Receives validated live sensor readings |
| Import interface | Owner only | Supports historical backfill with duplicate-safe handling |
| Export interface | Owner only | Provides an owner-controlled full-history backup |

Read-only restrictions and denial are enforced server-side. Unknown requests and application failures return generic responses without exposing internal implementation details.

## 8. Access control

Unauthenticated visitors receive a neutral access gate that discloses no application content. Dashboard markup is returned only after successful authorisation.

### 8.1 Tiers

| Role | Capability | Provisioning |
|---|---|---|
| Owner | Full application access | Individually provisioned credential |
| Restricted | Read-only dashboard access | Individually issued and revocable credential |

Credentials are individually issued, revocable and constrained to the minimum capability required. Formatting and validation rules are enforced by the application rather than relying on client behaviour.

Restricted credentials can also be used on lower-privilege portable devices, keeping higher-privilege credentials off them. Least privilege is applied by construction.

### 8.2 Issue, log, revoke

- Authorised enrolment establishes access without exposing internal route behaviour. Invalid or malformed requests receive the same neutral public response.
- Authenticated activity is logged by credential identity, providing per-user accountability and a durable audit trail.
- Credentials can be revoked independently without affecting other users or devices.

### 8.3 A considered risk

High-resolution environmental history can reveal patterns of occupancy and building use and is therefore treated as private data. Restricted access is read-only, individually revocable and rotated when a sharing occasion ends. This trade-off was made consciously rather than discovered later, which is the whole argument for writing systems like this down.

## 9. The dashboard

| Element | Behaviour |
|---|---|
| Header | Newest-reading age, self-updating and turning amber at five minutes' staleness; battery glyph (green/amber/red, hidden when unknown); location selector; backup button with last-backup tooltip (owner only); LIVE indicator with per-minute countdown on the live view |
| Temperature tile | Current value; feels-like (Rothfusz heat index); 30-minute windowed-mean trend arrow with °/hr; 24h sparkline in which gaps break the line |
| Humidity tile | Current %RH; dew point (Magnus formula); 24h sparkline |
| Records tile | All-time high and low with dates; an envelope gauge tracking the current temperature between them |
| Events tile | Up to three lines by priority: records set in the last 48h, coverage problems this week, day-on-day comparison, overnight low, week-ago comparison; quiet when nothing to report |
| Charts | Ranges 1h (live) to ALL; yesterday ghost, heat-index overlay and comfort bands as toggles; night shading; midnight markers; min/max markers anchored to true raw extremes; coverage percentage in every statline; missing data breaks the line rather than interpolating |

Range and toggle preferences persist locally. Long-range views are served from the rollup with averages weighted by row count, so they match the raw data exactly.

## 10. The economics: optimising database service reads

This section is the engineering heart of the document. It records how a dashboard that innocently charted some temperatures was measured consuming its database allowance at an unsustainable rate, how the cost model was derived, and how one schema decision cancelled the problem permanently. All figures are real measurements from July 2026.

### 10.1 The billing model

The database service meters rows read, defined as every row the SQL engine touches while answering a query, not the rows returned to the caller. Two consequences drive everything that follows. First, aggregation is metered at input size: a chart point averaged from 1,440 raw readings costs 1,440 reads regardless of the single value returned. Second, indexes eliminate waste, not work: they stop whole-table scans, but they cannot make a legitimate aggregation over N rows cost less than N. Reducing that cost requires reading pre-aggregated data instead.

## 10.2 The measured baseline (22 July 2026)

With the daily counter starting at zero and a table of roughly 4,600 rows:

| Action | Cost in rows read |
|---|---|
| One dashboard visit (24h view) | 19,350 |
| One CSV import (~1,600 rows) including auto-refresh | 12,190 |
| One click of the ALL range | 37,450 |

Decomposing the page load found two freeloaders: a locations dropdown that counted every row in the table on every visit, and four independent extremes queries that each re-swept a range the main aggregate had already covered.

## 10.3 The ALL multiplier and the projection

On the ALL range, every range-scoped query sweeps the entire table. With six to seven such sweeps per load, the model is simply: one ALL click ≈ 6.5 × the table row count. Measured 37,450 against a predicted ~30,000 plus page furniture, the model holds. At one-minute logging the table grows by 525,600 rows per sensor per year, which turns the model into a countdown:

| Point in time | Table rows (1 sensor) | Cost of one ALL click | ALL clicks to exhaust the service allowance |
|---|---|---|---|
| 22 July 2026 | ~4,600 | ~37k | ~130 |
| +1 month | ~48,000 | ~312k | 16 |
| +3 months | ~134,000 | ~871k | 5 |
| +12 months | ~530,000 | ~3.45M | 1 |
| +12 months, 2 sensors | ~1,060,000 | ~6.9M | 0 |

At one year of single-sensor data, a single click of ALL would consume a full day's service allowance. Storage, by contrast, was never the issue: roughly 26 MB per sensor-year.

## 10.4 Options considered

| Option | Description | Verdict |
|---|---|---|
| A: coarser exports | Log at 15-minute or hourly intervals, shrinking the table | Rejected: destroys the forensic value of the dataset. Solves the problem by deleting the product. |
| B: hourly rollup table | Keep 1-minute raw; serve long views from pre-aggregated hourly rows | Accepted. Retains full detail, caps long-view cost permanently. |
| C: do nothing | Acceptable at then-current volume | Viable for one to two months only, per the projection. |

## 10.5 The v6 rollup design

```
The hourly rollup stores one aggregate row per location and hour, retaining average,
minimum, maximum and sample count so long-range views preserve extremes and coverage
without scanning minute-resolution history.
```

Storing minimum and maximum alongside the average keeps long-range extremes honest, while a sample count carries coverage information forward. At ingest time the application service recomputes only the hourly aggregates touched by incoming data; a one-time migration backfilled history in 4.62 milliseconds. Query routing after v6 is unchanged in principle: short ranges read raw minutes and long ranges read the rollup, which accumulates only 8,760 rows per year.

## 10.6 Results

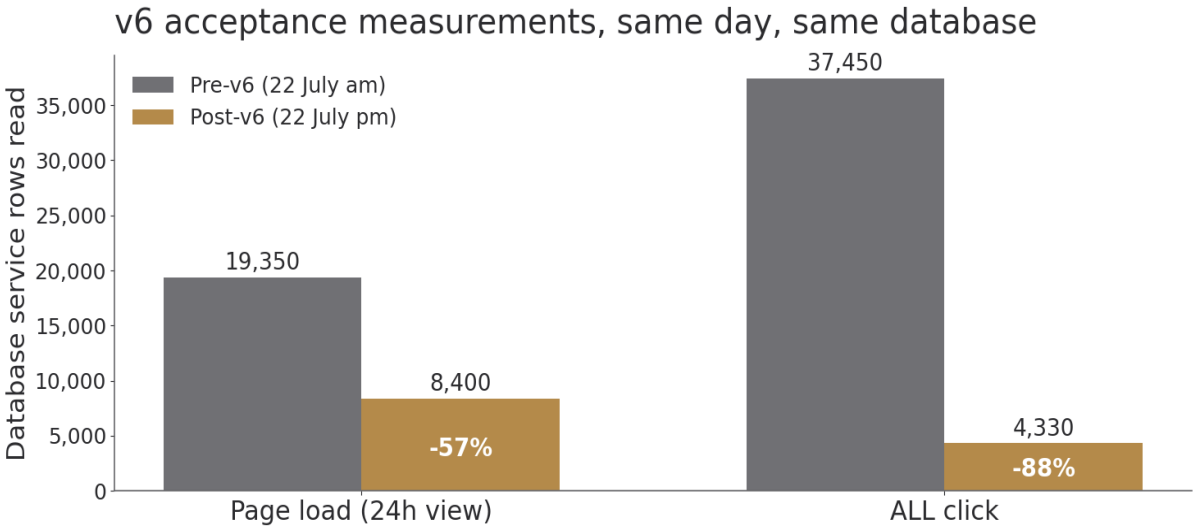



*Figure 2 — v6 acceptance measurements, same day, same database*

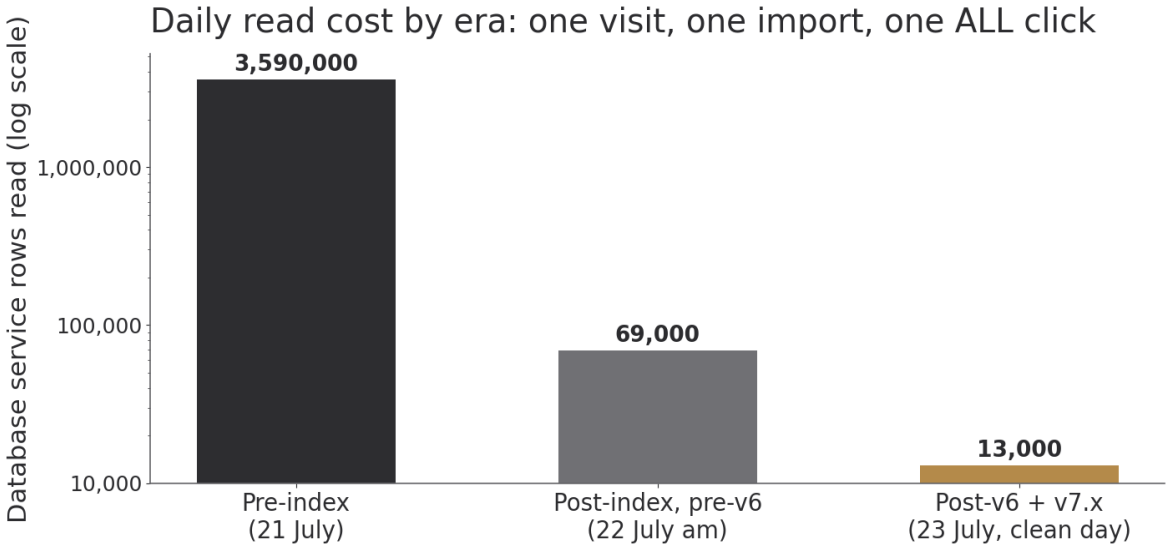



*Figure 3 — daily read cost by era; note the logarithmic scale*

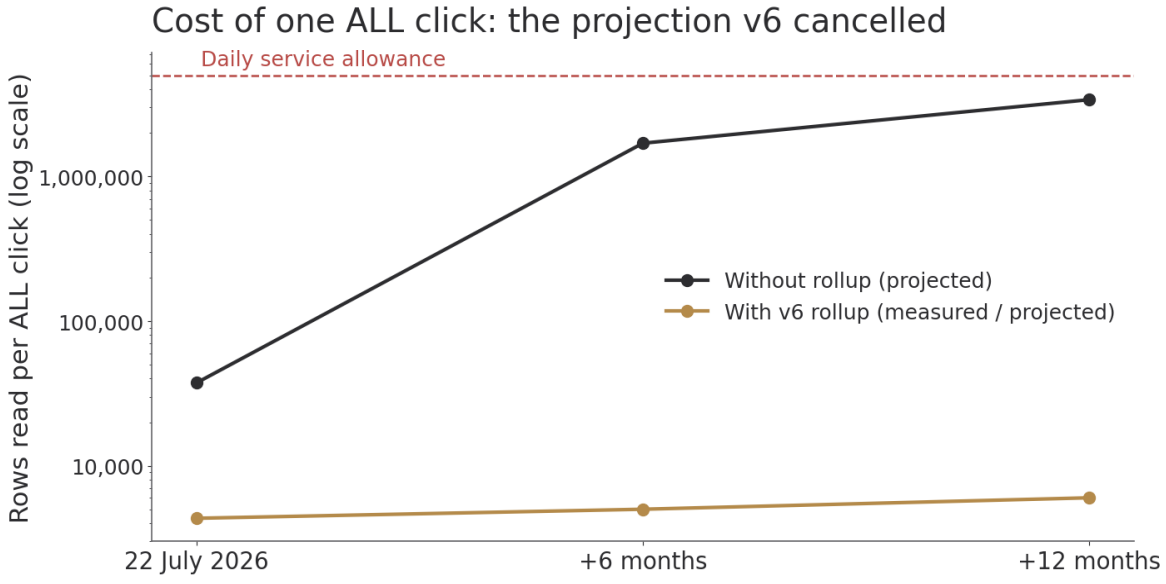



*Figure 4 — the projection v6 cancelled*

The 21 July pre-index era peaked at 3.59 million rows read in a single day, the day that prompted the investigation. After v6, a full page load on the most expensive view costs roughly 5,000 reads and an ALL click 4,330, dominated by fixed page furniture rather than table size. Roughly 600 ALL clicks per day would now be required to exhaust the service allowance, and the curve stays effectively flat for years, absorbing a second sensor without redesign.

### 10.7 Write amplification, the sequel

With reads solved, the write meter told its own story: 56,000 rows written in a day, driven by overlapping re-imports. Two mechanisms were identified. The original upsert rewrote identical rows, paying full write cost to change nothing, and the database service meters index maintenance, so each raw row incurred several metered writes across the primary and secondary indexes. The fix was duplicate-safe raw insertion plus removal of the unnecessary humidity index. Verification was satisfying: an immediate full-database re-import of 3,271 rows moved the write counter by roughly 60, all of it the rollup refresh, and nothing at all for duplicate rows.

## 11. Incident INC-01: the chart that lost its labels

One production incident is worth telling in full, because the root cause had been latent since day one and the reason it stayed hidden is a genuinely good story about luck.

### 11.1 Symptom

Following a feature deployment, the 24-hour and 7-day views rendered with no x-axis labels, no midnight markers and no min/max markers, while the 6h, 12h, 30d and ALL views rendered perfectly. Chart lines and statistics were unaffected everywhere.

### 11.2 Diagnosis

The served client code was exercised verbatim in a real client-side charting instance and rendered correctly, eliminating the client. Live interrogation of the deployed chart object showed zero axis ticks generated. Fetching the live 24h payload produced the decisive evidence: the response declared a 300-second bucket while returning 1,441 points spaced 60 seconds apart, with the first timestamp not aligned to a 300-second boundary. The GROUP BY was not grouping.

### 11.3 Root cause

The production database binding layer represented JavaScript numeric parameters as floating-point values. The bucketing expression therefore executed as floating-point division rather than integer bucketing, so every raw row formed its own group. The defect had existed since version one and stayed invisible for three distinct reasons: a 60-second bucket over 1-minute data is the identity anyway; long-range views read the hourly rollup; and the earliest test windows happened to align with bucket boundaries. When a later 24-hour window began at a non-aligned time, that luck expired and every feature keyed on aligned timestamps failed together.

### 11.4 Fix and lesson

The bucket value, chosen server-side from a hard-coded whitelist, is now inserted as a validated integer constant rather than bound as a generic numeric parameter. The failure was reproduced in a local SQL harness by deliberately using floating-point binding and the fix verified with correctly aligned groups. A side effect of the repair was that the 24h and 7d views became genuinely bucketed for the first time, cutting their payloads from about 1,440 points to about 288.

The standing lesson: never assume production parameter types match those used by a local SQL test harness when arithmetic depends on integer semantics. Type-sensitive calculations must be verified

against the deployed database service at the payload level: point spacing and alignment, not just query results.

## 12. Security and privacy posture

Transport is HTTPS end to end. Authorisation credentials are held in protected platform and operating-system stores and compared using timing-safe checks. Public requests receive no application markup until authorised. Restricted access is least-privilege, individually accountable and revocable. The ingest identity is deliberately constrained and cannot perform administrative database operations. The bridge host is contained by network segregation, filtered DNS and restrictive firewall policy.

## 13. Standing engineering rules

- Never assume a bound numeric parameter has local-test integer semantics in production; use validated integer constants for type-sensitive arithmetic (INC-01).
- Client script lives inside a server-side template: always verify a build by extracting and parsing the served scripts, not just the source file.
- Database console SQL is written comment-free and single-line: the console collapses line breaks, after which a double-hyphen comment consumes the rest of the statement.
- No silent error suppression anywhere: failures are surfaced, logged or rendered.
- Credential values are validated against deployment-safe formatting rules.
- Ownership identifiers appear on authenticated pages only; public-facing gates disclose nothing.
- Files are delivered complete, with consistent line endings, and every previous version is archived.
- Every change ships tested: local SQL exercises that simulate the production database service's binding behaviour where relevant, endpoint tests through the actual request handler, and served-script parsing.

## 14. Version history (condensed)

| Version | Date (2026) | Summary |
|---|---|---|
| v1 | 20 Jul | Launch: wall-clock schema, CSV upload with defensive parser, gated dashboard, charts |
| v2–v5 | 21 Jul | Identity, records tile, night bands, yesterday ghost, trend, coverage honesty, extra ranges, preferences |
| v6 | 22 Jul | Hourly rollup; 30d/ALL and stats served from it; consolidated extremes; indexed locations (88% ALL-click cut) |
| v7–v7.2 | 22–23 Jul | Events tile, whole-page upload, write-amplification fixes, INC-01 root cause fixed |
| v7.3–v7.7 | 24 Jul | Layout maturity: tile sparklines, records envelope gauge, aligned grid |
| v7.8 | 29 Jul | Public-facing access-control refinement |
| v8.0–v8.7 | 2–5 Aug | Live ingest commissioned: bridge online, freshness age, battery glyph, live view, export/backup, optional AI summary; 20,979-row backfill to 100% coverage |
| v8.8–v8.8.1 | 7 Aug | Restricted guest access, independent revocation and audit logging |

## 15. Cost postscript

The hosted platform is provisioned with substantial headroom: after the July optimisation work, an ordinary day of normal use lands at roughly 30,000 to 50,000 rows read and 2,000 to 3,000 written, a small fraction of the service allowance, on a trajectory that stays flat regardless of table growth. A year of one-minute readings occupies roughly 26 MB. The most expensive thing in the whole system remains, by a comfortable margin, the time spent building it, which was of course the point.

<EOF>