OpenEMS Backend InfluxDB Data Model and Edge-ID Collision

> [!warning] Verification status
> **The storage-key collision vulnerability is confirmed in source code, and the repository contains a real configured collision.** However, the vulnerability alone does not prove that Hillary data contaminated the current Sezibwa Pi2 history. Historical causation must be established from production InfluxDB data, connection periods, and gateway logs before any records are rewritten or deleted.

Executive conclusion

- The configured backend Timedata.InfluxDB component writes edge data into a shared InfluxDB measurement, currently `data`.
- It does not use the complete edge ID as the storage discriminator. It parses the trailing number and stores that number in the OEM edge tag, normally `edge=<number>`.
- Two edge IDs with the same parsed number are indistinguishable to this timedata store and its history queries.
- The repository contains `hillary-test-pi-1` and `aaron-pi-1`; both map to `edge=1`. This is a genuine collision risk.
- The current Sezibwa Pi2 edge definition is `sezibwa-rentals-gw-pi2`, which maps to `edge=2` and does not collide with Hillary.
- `nfetestpi1` is a hostname/deployment user in this repository, not the backend edge ID, and should not be used as evidence of an edge-tag collision.

Confirmed backend data model

The relevant implementation is:

`io.openems.backend.timedata.influx/src/io/openems/backend/timedata/influx/TimedataInfluxDb.java`

For every accepted data timestamp, the backend constructs an InfluxDB point using:

```
Point.measurement(this.config.measurement())
    .addTag(this.oem.getInfluxdbTag(), String.valueOf(influxEdgeId))
    .time(timestamp, WritePrecision.MS);
```

The deployment configuration sets `measurement="data"`. “Measurement” is the correct InfluxDB term; it is analogous to, but not identical with, a relational database table. All edges handled by this configured timedata provider share that measurement unless another Timedata.InfluxDB instance or measurement is configured.

Channel addresses such as `meter0/ActivePower` and `_sum/EssSoc` are written as fields. History queries parse the requested edge ID through the same numeric function and filter using the resulting numeric tag.

Confirmed edge-ID parser behavior

`InfluxConnector.parseNumberFromName(edgeId)` uses:

```
\D++(\d++)$
```

This extracts the trailing numeric run rather than preserving the full edge ID:

Configured value	Role in this repository	Influx tag value	Finding
`hillary-test-pi-1`	Hillary test edge ID	`1`	Collides with `aaron-pi-1`
`aaron-pi-1`	Aaron edge ID	`1`	Collides with `hillary-test-pi-1`
`sezibwa-rentals-gw-pi2`	Current Sezibwa Pi2 edge definition	`2`	No suffix collision with Hillary
`nfetestpi1`	Hostname/deployment user	Not applicable as currently configured	Not the backend edge ID

If an edge ID contains no parseable trailing number, the timedata write is not stored. This is **not completely silent**: `TimedataInfluxDb.writeData()` logs a warning stating that it could not parse the numeric Influx Edge-ID, then returns without writing the notification.

What a collision does

InfluxDB identifies a point by measurement, tag set, and timestamp. Once two gateways share the same numeric edge tag:

- History queries cannot distinguish which gateway supplied a value.
- Different channel fields can appear together under the same edge series.

- If both gateways write the same field at the same timestamp, later writes can replace or combine with the existing point according to InfluxDB point-update behavior.
- Backend caches keyed by the parsed integer, such as timestamped-channel tracking, can also conflate the two edges.

The resulting history may therefore be mixed, overwritten, or both; “merge” should not be interpreted as guaranteed preservation of every value from both sources.

What is and is not proven about the incident

Proven: the implementation is collision-prone, and two repository edge configurations currently resolve to `edge=1`.

Not yet proven: that this caused the reported Hillary-to-Sezibwa contamination. If the affected Sezibwa history was requested under `aaron-pi-1`, the collision is a strong causal candidate. If it was requested under `sezibwa-rentals-gw-pi2`, the `1`-versus-`2` mapping means this specific mechanism does not explain it.

Confirm the affected edge ID and examine production `edge=1` data before finalizing the root-cause statement.

Immediate containment

1. Inventory every edge registered with or connecting to the production backend.
2. Compute the parsed numeric ID for each and identify duplicates before making changes.
3. Disconnect colliding test rigs from production or assign a globally unique temporary numeric suffix.
4. Back up InfluxDB and the backend/metadata configuration before renaming an edge.
5. Update all coupled references together: metadata, API keys, gateway configuration, monitoring, dashboards, automation, and operational documentation.
6. After each change, verify websocket connectivity, live channels, new timedata writes, history queries, and monitoring alerts.

> [!note] Unique numeric suffixes are containment, not the permanent design
> A naming convention reduces immediate risk but remains brittle and easy to violate as the fleet grows.

Permanent backend correction

1. Use the complete immutable edge ID as the InfluxDB discriminator, or resolve the edge to another stable, globally unique identifier maintained by backend metadata.
2. Validate uniqueness at edge registration and backend startup. Reject duplicate storage identifiers instead of accepting ambiguous writes.
3. Fail visibly for invalid identifiers, with actionable logs and operational alerts.
4. Add automated tests covering identical suffixes, multi-digit suffixes, IDs without numbers, reconnects, resends, and history-query isolation.
5. Review other structures keyed by the parsed integer, including timestamped-channel tracking and aggregated timedata, so the fix covers more than the point tag.

Migration requirements

Changing the tag key or value can make existing history unreachable to current queries. The implementation therefore needs an explicit migration plan, such as:

- dual-write the legacy numeric tag and the new full-ID tag for a controlled period;
- support query fallback across both schemas during migration;
- backfill only records that can be attributed confidently;
- retain an immutable backup and a tested rollback path; and
- define a cutover date after which the legacy numeric identity is no longer accepted.

Historical-data audit before cleanup

1. Identify exactly which logical Sezibwa edge displayed Hillary channels.
2. Determine when `hillary-test-pi-1` and `aaron-pi-1` were simultaneously connected to the same production backend and bucket.
3. Inspect `edge=1` by time range, unique channel inventory, device serial numbers, meter topology, and gateway connection logs.
4. Classify records as confidently Hillary, confidently Aaron/Sezibwa, or ambiguous.
5. Quarantine ambiguous records. Do not delete or rewrite them merely from channel-name assumptions.
6. Document any irrecoverable overlap where the same field and timestamp may have been overwritten.

Approval boundary

Approved direction: immediate containment, a reviewable backend fix, collision tests, and a documented migration design.

Requires separate approval: irreversible deletion, reassignment, or rewriting of production historical data. That work must follow the backup and attribution audit above.

Revision #2
Created 2026-08-20 11:23:09 UTC by hillary.arinda
Updated 2026-08-20 13:09:31 UTC by aaron.tushabe