Executive Brief — Shunyaya Symbolic Mathematical Electrical Quantities (SSMEQ)

From Mixed Units to Zero-Centric, Portable Electrical Telemetry

Status: Public Research Release (v2.0)
Date: November 14, 2025
Caution: Research/observation only. Not for critical decision-making.

Make electrical telemetry **unitless, portable, bounded, and auditable** so voltage, current, power, frequency, power factor, THD, and unbalance can be read, pooled, alerted on, and governed the same way across vendors, voltages, and grids. SSMEQ is **representation-only**: it replaces mixed units (`V/kV`, `A/kA`, `W/kW`, `Hz`, `%`) with symbolic contrasts `e_Q` and optional bounded dials `a_Q`, without changing physics, firmware, or protection models.

At the core, SSMEQ **normalizes once, then symbolises forever**. Raw channels are first brought to a clean baseline (for example, `V_rms`, `I_rms`, `f`, signed `P`), then mapped to unitless contrasts using simple, published lenses such as:

- `e_V := ln( V_rms / V_ref )`
- `e_I := ln( I_rms / I_ref )`
- `e_f := ln( f / f_ref )`
- `e_P := asinh( P / P_scale )`

Bounded health dials turn edge-sensitive metrics into safe alignment lanes, for example:

- `a_pf := clamp( pf, -1+eps_a, +1-eps_a )`
- `a_THD := tanh( c_THD * ln( 1 + THD/THD_ref ) )`

Because the representation is **manifest-first**, every deployment freezes its lenses, anchors, clamps, and residual rules up front. AC/DC mode, nominal values (`V_ref`, `I_ref`, `f_ref`), power-factor targets `pf_ref`, and validity windows are declared once per site/profile; downstream logic consumes only `e_*`, `a_*`, and residuals like:

- `r_P := e_P_meas - ( e_V + e_I + ln( pf / pf_ref ) )`

The result is a zero-centric electrical language where **batch == stream**, multi-vendor fleets can be pooled safely, and consistency checks are built in instead of bolted on.

SSMEQ sits **between raw telemetry and everything else** — SCADA, historians, protection relays, analytics, and ML. It does **not** change Kirchhoff, Ohm, Maxwell, or your protection curves; it simply gives all of them the same clean, symbolic input layer. Human dashboards may still show volts and amps, but internal logic works on stable contrasts and bounded dials, making cross-site comparisons, fleet roll-ups, and anomaly detection far easier to trust and audit.

---

0. Why SSMEQ matters now

Today's electrical telemetry is fragmented and fragile:

- **Mixed units everywhere.** The same physical state may appear as `230 V`, `0.23 kV`, `% of nominal`, or an engineering integer. Downstream logic must constantly remember context, making **pooling, ML, and cross-vendor audits painful and error-prone**.
- **No built-in consistency check.** Real power, voltage, current, and power factor are related, but most systems treat them as separate channels. Without a simple identity residual like `r_P := e_P_meas - ( e_V + e_I + ln( pf / pf_ref ) )`, CT/PT errors, pf estimation faults, and sign flips can hide in plain sight until they become incidents.

SSMEQ addresses these structural gaps by turning **all core electrical quantities into a single, zero-centric symbolic language**, ready for policy, protection, analytics, and AI — without touching the underlying physics or your existing devices.

0.1 SSMEQ fixes the structural gaps

- **Gap 1 — Mixed units and scaling tricks.**
  The same physical state appears as `230 V`, `0.23 kV`, `% of nominal`, or an engineering integer. Every dashboard, script, and ML pipeline must remember which is which.
  Answer: SSMEQ converts each raw channel into a **single, unitless contrast** like `e_V := ln( V_rms / V_ref )`, `e_I := ln( I_rms / I_ref )`, `e_f := ln( f / f_ref )`, `e_P := asinh( P / P_scale )`. Once symbolized, everything downstream works on `e_*` (and, where needed, `a_*`) with no ambiguity.
- **Gap 2 — Hidden physics relationships.**
  Real power, voltage, current, and power factor are related, but most systems store them as independent columns with no automatic cross-check. CT/PT mistakes, wrong `pf`, and sign flips can sit undetected.
  Answer: SSMEQ exposes a simple **identity residual** such as `r_P := e_P_meas - ( e_V + e_I + ln( pf / pf_ref ) )`. When devices and sites share a manifest, `r_P` becomes a universal diagnostic lane for "is this quartet self-consistent?".

- **Gap 3 — AC vs DC fragmentation.**
  AC feeders, DC buses, and inverter stages often use different conventions, so fleets cannot be pooled cleanly across AC/DC boundaries.
  Answer: SSMEQ **declares normalization once in the manifest** (AC RMS anchors like `V_ref`, `I_ref`, `f_ref`; DC rating anchors for bus voltage/current), then emits a common symbolic layer `e_*`, `a_*`, and `r_P` so that AC and DC assets can be compared and governed under the same rules.
- **Gap 4 — No portable "health dial".**
  Limits and ratings are expressed in prose, nameplate PDFs, or device-specific flags. There is no universal, bounded "how close to the edge are we?" signal.
  Answer: SSMEQ introduces **bounded dials** `a_Q` in `(-1,+1)` (for example, `a_pf`, `a_THD`, `a_upper`, `a_lower`) with declared anchors and slopes in the manifest. These become portable proximity and quality signals that can be fused, averaged, or alarmed across fleets.
- **Gap 5 — Vendor-locked semantics.**
  Many "smart" meters embed internal logic that cannot be replayed. Different vendors mean different meanings, even for the same label.
  Answer: In SSMEQ, **the manifest is the rulebook**: lenses, anchors, clamps, and residual formulas are published as plain-text rules. Anyone can recompute `e_*`, `a_*`, and `r_P` from raw telemetry and verify that a device's outputs match the manifest it claims to follow.

0.2 What this changes

- **For operators and engineers:**
  Keep your existing meters, relays, and SCADA. SSMEQ simply adds a **clean symbolic layer** on top: you can see at a glance whether feeders, transformers, and sites are comparable, whether power data is self-consistent, and how close each asset is to its declared limits.
- **For vendors and integrators:**
  Instead of inventing a new proprietary index for every product, you can emit `e_*`, `a_*`, and `r_P` under a published manifest. This makes your devices easier to adopt across heterogeneous fleets and reduces integration friction.
- **For analytics, forecasting, and AI:**
  Models no longer need to learn around unit quirks and naming inconsistencies. A single, zero-centric representation works across time, vendors, voltages, and mixes of AC/DC — freeing ML and analytics to focus on **patterns in behaviour**, not on cleaning up the input space.

---

1. The SSMEQ Representation (Core Idea)

SSMEQ does **not** replace your meters or physics. It simply wraps each electrical quantity `Q` (voltage, current, power, frequency, power factor, THD, unbalance, etc.) in a **small, symbolic record** that can travel across devices, vendors, and analytics stacks without losing meaning.

For each channel `Q`, SSMEQ encourages a common pattern:

- **`m_Q` — value lane (truth lane)**
  The raw magnitude you already trust (for example `V_rms`, `I_rms`, `P`, `f`, `pf`, `THD`).
- **`e_Q` — zero-centric contrast (unitless)**
  A simple, monotone transform that makes "nominal" map to `0`, higher to `+`, lower to `-`, independent of units. Example lenses:
  - `e_V := ln( V_rms / V_ref )`
  - `e_I := ln( I_rms / I_ref )`
  - `e_f := ln( f / f_ref )`
  - `e_P := asinh( P / P_scale )`
- **`a_Q` — bounded health dial (optional)**
  A compact dial in `(-1,+1)` for "how close to the edge?" or "how good is the quality?". Examples:
  - `a_pf := clamp( pf, -1+eps_a, +1-eps_a )`
  - `a_THD := tanh( c_THD * ln( 1 + THD/THD_ref ) )`
- **`band_Q` — label derived from the manifest (optional)**
  A simple label like `GREEN`, `AMBER`, `RED` derived from `a_Q` or `e_Q` using thresholds declared in the manifest (for example, `|a_pf| < 0.2 → GREEN`).
- **`manifest_id` — which rulebook was active**
  A plain-text identifier that points to the **profile** defining lenses, anchors, clamps, and band cutpoints (for example, site, feeder, or device profile).

This gives every consumer — protection logic, SCADA, dashboards, and ML — a **standard shape** for all channels, across vendors and across sites.

1.1 e_Q — zero-centric contrast (unitless lane)

- **Definition:**
  For each quantity `Q`, choose a nominal anchor `Q_ref` (for example, rated voltage, rated current, nominal frequency, target `pf`) and map:
  - `e_V := ln( V_rms / V_ref )`
  - `e_I := ln( I_rms / I_ref )`
  - `e_f := ln( f / f_ref )`
  - `e_P := asinh( P / P_scale )`
- **Properties:**
  - `e_Q = 0` at nominal, `e_Q > 0` when above, `e_Q < 0` when below.
  - Unit-invariant: whether `V` arrives as volts, kilovolts, or engineering integers, the same `Q_ref` and lens yield the same `e_V`.
  - Vendor-portable: any device that agrees on `Q_ref` and the lens will produce the same `e_Q`.

1.2 a_Q — bounded health dial in (-1,+1)

Where needed, SSMEQ adds an optional **health dial** `a_Q` to express proximity to limits, quality, or risk:

- **Examples:**
    - Power factor: `a_pf := clamp( pf, -1+eps_a, +1-eps_a )`
    - Harmonics: `a_THD := tanh( c_THD * ln( 1 + THD/THD_ref ) )`
    - Headroom to upper limit: `a_upper := tanh( k_upper * ( e_Q - e_Q_upper ) )`
    - Margin to lower limit: `a_lower := tanh( k_lower * ( e_Q_lower - e_Q ) )`
- **Why bounded:**
  `a_Q` in `(-1,+1)` is easy to combine, average, and alarm across fleets without any single source dominating due to scale alone.

1.3 Residuals and identities (r_P and friends)

Electrical quantities obey known relationships. SSMEQ **makes the residual explicit** so it can be monitored:

- **Power consistency residual:**
    - `r_P := e_P_meas - ( e_V + e_I + ln( pf / pf_ref ) )`

When `r_P` is close to `0`, your quartet `(V,I,P,pf)` is self-consistent under the declared manifest. Persistent bias or jumps in `r_P` point to CT/PT errors, bad pf estimation, mis-wiring, or scaling mistakes.

Other identities can be added similarly (for example, `P^2 + Q^2 ≈ S^2`, unbalance metrics, or neutral current relationships), each with an explicit residual lane.

1.4 manifest_id — profiles for sites, feeders, and devices

All lenses, anchors, clamps, and residual recipes live in a **profile manifest**:

- Nominal anchors such as `V_ref`, `I_ref`, `f_ref`, `pf_ref`.
- Scaling constants like `P_scale`, `THD_ref`, `c_THD`.
- Clamp parameters `eps_a`, `eps_pf`, slopes `k_upper`, `k_lower`.
- Band cutpoints for `GREEN/AMBER/RED` and similar labels.

The string `manifest_id` is carried with each SSMEQ record so anyone can replay the mapping from `m_Q` to `e_Q`, `a_Q`, and `r_P` using the same rulebook.

One-line takeaway for Section 1

SSMEQ turns every electrical channel into a **small, replayable record** — `m_Q` (value), `e_Q` (zero-centric contrast), `a_Q` (bounded health), `band_Q` (label), and `manifest_id` (rulebook) — so different devices and sites can finally speak the same electrical language.

---

2. How SSMEQ Rides Beside Existing Telemetry

SSMEQ does **not** replace your meters, relays, or SCADA tags. It adds a **thin symbolic layer** that can be produced at any of three points:

- Inside the device (firmware or embedded script).
- At the gateway or substation server (edge script).
- In the data pipeline / historian (batch or stream job).

In all cases, the **raw channels stay exactly as they are**; SSMEQ just emits a small, replayable record for each electrical quantity `Q`.

2.1 Minimal SSMEQ record (illustrative)

For a single channel `Q`, a minimal SSMEQ record looks like:

- `timestamp` — when the sample was taken.
- `channel_id` — which feeder / phase / device / measurement this is.
- `m_Q` — raw magnitude (for example, `V_rms`, `I_rms`, `P`, `f`, `pf`, `THD`).
- `e_Q` — zero-centric contrast (for example, `e_V`, `e_I`, `e_f`, `e_P`).
- `a_Q` — optional bounded health dial (for example, `a_pf`, `a_THD`, `a_upper`, `a_lower`).
- `band_Q` — optional label such as `GREEN`, `AMBER`, `RED`.
- `manifest_id` — which profile defined the lenses, anchors, clamps, and bands.

A row in CSV or JSON might therefore carry both the **legacy tags** and the **symbolic tags** side by side, with no changes to existing consumers that do not yet understand SSMEQ.

2.2 Where SSMEQ sits in your architecture

- **Inside devices:**
  Meters and relays that already compute `V`, `I`, `P`, `pf`, `THD` can add a tiny script to emit `e_Q`, `a_Q`, and `band_Q` under a declared `manifest_id`.
- **At edge gateways:**
  If devices are “dumb” or proprietary, an edge script can read their raw outputs, apply the manifest, and write SSMEQ fields into an **adjacent stream or topic** without touching the original SCADA tags.
- **In the data pipeline:**
  Analytics teams can backfill SSMEQ on historical data by applying the same manifest logic in batch, so models can train on `e_*`, `a_*`, and residuals like `r_P` without modifying any upstream infrastructure.

2.3 Minimal compliance (Day-1)

On Day-1, an SSMEQ adoption can be as simple as:

- Declare a **manifest** with `V_ref`, `I_ref`, `f_ref`, `pf_ref`, `P_scale`, and any `THD_ref` or slopes.
- For each sample, compute:
  - `e_V := ln( V_rms / V_ref )` (and similar for `e_I`, `e_f`, `e_P`).
  - Optional `a_pf`, `a_THD`, or limit dials such as `a_upper`, `a_lower`.
- Store or transmit these alongside the original channels with the chosen `manifest_id`.

No protection setting, wiring, or SCADA register needs to change. Existing tools keep working; new tools gain a **clean, zero-centric electrical language** from the same data.

One-line takeaway for Section 2

SSMEQ is a **sidecar representation**: keep all your existing electrical telemetry, and add a small, manifest-driven symbolic record beside it so every system sees the same, replayable view of what “normal”, “margin”, and “drift” mean.

---

3. Core Invariants (Copy-Ready)

SSMEQ is designed so that **anyone, anywhere** can replay the same mapping from raw electrical quantities to symbolic lanes and get the **same answer**, provided they share the manifest.

3.1 Value lane invariance (m_Q stays as-is)

- SSMEQ never changes the underlying physics or raw channels.
- `m_Q` remains the truth lane — for example `V_rms`, `I_rms`, `P`, `f`, `pf`, `THD`.
- Existing protection, billing, and control logic may continue to use `m_Q` exactly as before; SSMEQ only **adds** symbolic lanes, it does not replace or reinterpret `m_Q`.

3.2 Zero-centric encoding (e_Q properties)

For each quantity `Q`, `e_Q` is constructed so that:

- `e_Q = 0` at the declared nominal (for example `V_rms = V_ref`).
- `e_Q > 0` when the magnitude is above nominal; `e_Q < 0` when below.
- The mapping is **monotone** in the physical magnitude (no reversals).
- The mapping is **unit-invariant**: whether raw input arrives as volts, kilovolts, or scaled integers, a shared `Q_ref` and lens yield the same `e_Q`.

Examples (repeated for clarity):

- `e_V := ln( V_rms / V_ref )`
- `e_I := ln( I_rms / I_ref )`
- `e_f := ln( f / f_ref )`
- `e_P := asinh( P / P_scale )`

3.3 Manifest-governed lenses and anchors

- All choices of anchors and lenses are **declared, not guessed**.
- A profile manifest fixes, for example:
  - `V_ref`, `I_ref`, `f_ref`, `pf_ref`
  - `P_scale`, `THD_ref`, `c_THD`
  - Any special handling for AC vs DC channels
- Changing anchors or lenses means **issuing a new profile** with a different `manifest_id`; historical data is never silently reinterpreted.

3.4 Residual consistency (r_P and others)

- Known relationships (like the link between `V`, `I`, `P`, and `pf`) are turned into **explicit residual lanes**.
- A canonical power residual is:
  - `r_P := e_P_meas - ( e_V + e_I + ln( pf / pf_ref ) )`

- When sensors and scaling are correct under the manifest, `r_P` stays near `0`.
- Persistent bias, step changes, or phase-dependent drift in `r_P` highlight CT/PT errors, pf estimation bugs, sign mistakes, or wiring issues without needing bespoke logic per site.

3.5 Bounded dials (a_Q in (-1,+1))

- Health dials `a_Q` (for example `a_pf`, `a_THD`, `a_upper`, `a_lower`) always live in `(-1,+1)`.
- This makes them safe to:
  - Combine across sensors and sites.
  - Average over time.
  - Feed directly into thresholds and bands.
- A typical pattern is:
  - `a_THD := tanh( c_THD * ln( 1 + THD/THD_ref ) )`
  - `a_pf := clamp( pf, -1+eps_a, +1-eps_a )`

When multiple dials are fused (for example, “overall feeder health”), they can use the same alignment pipeline as core SSM:

- `a_c := clamp( a_raw, -1+eps_a, +1-eps_a )`
- `u := atanh( a_c )`
- `U += w*u ; W += w`
- `a_fused := tanh( U / max( W, eps_w ) )`

This ensures **batch == stream** for any fused dial, given the same weights `w`.

3.6 Bands and labels (band_Q)

- Labels like `GREEN`, `AMBER`, `RED` are **derived, not hard-coded**.
- Each band is defined by ranges over `e_Q` or `a_Q`, for example:
  - `GREEN` if `|a_Q| <= 0.3`
  - `AMBER` if `0.3 < |a_Q| <= 0.7`
  - `RED` if `|a_Q| > 0.7`
- Band rules are written in the manifest; changing a band definition creates a new profile, not a silent shift in meaning.

One-line takeaway for Section 3

SSMEQ is governed by **simple, replayable invariants** — raw values preserved, `e_Q` zero-centric, `a_Q` bounded, residuals explicit, and all lenses and bands frozen in a manifest — so any party can recompute and trust the same symbolic view of their electrical world.

4. Minimal Adoption (Paste-Ready)

SSMEQ is meant to be **easy to adopt**: declare one profile manifest, add a small script, and start emitting symbolic lanes beside your existing telemetry.

4.1 Profile manifest — MUST

Before emitting any SSMEQ fields, define a **profile manifest** that fixes:

- **Anchors and scales**
  - `V_ref`, `I_ref`, `f_ref`, `pf_ref`
  - `P_scale` (for example, rated kW in watts or a chosen engineering scale)
  - Any `THD_ref`, unbalance references, or other quality baselines
- **Lenses**
  - `e_V := ln( V_rms / V_ref )`
  - `e_I := ln( I_rms / I_ref )`
  - `e_f := ln( f / f_ref )`
  - `e_P := asinh( P / P_scale )`
- **Health dials and residuals**
  - Formulas for `a_pf`, `a_THD`, limit dials such as `a_upper`, `a_lower`
  - Residuals such as `r_P := e_P_meas - ( e_V + e_I + ln( pf / pf_ref ) )`
- **Bands and labels (optional)**
  - Thresholds for `GREEN/AMBER/RED` or other labels over `e_Q` or `a_Q`

Assign this profile a clear **`manifest_id`** (for example, `PLANT_A.LV_FEEDERS.v1`).

4.2 Device / gateway script — MUST

At the meter, relay, or gateway level, implement a **small SSMEQ script**:

- Read existing raw channels `V_rms`, `I_rms`, `P`, `f`, `pf`, `THD` (and others as available).
- Look up the active `manifest_id`.
- Compute:
  - `e_V`, `e_I`, `e_f`, `e_P` from the manifest lenses.
  - Any required dials `a_pf`, `a_THD`, `a_upper`, `a_lower`.
  - Residuals such as `r_P`.
- Emit or store these as **extra fields** beside the original channels, always tagged with `manifest_id`.

No change is made to the legacy registers or payload — SSMEQ is a **sidecar**, not a replacement.

4.3 Pipeline / historian — SHOULD

If you cannot modify devices immediately, you can still adopt SSMEQ via your **data pipeline**:

- Ingest raw telemetry into your historian as usual.
- Run a batch or streaming job that:
  - Applies the manifest lenses to compute `e_*`, `a_*`, and `r_P`.
  - Writes these symbolic fields into adjacent columns or topics, stamped with `manifest_id`.
- Keep the mapping code under version control so it matches the published manifest.

This allows you to:

- Backfill historical data with SSMEQ fields.
- Train analytics and ML models on a **clean, zero-centric representation** without touching upstream systems.

4.4 Analytics, dashboards, and AI — CAN

Once SSMEQ fields are available, downstream consumers can:

- Plot `e_V`, `e_I`, `e_f`, `e_P` and `r_P` to see **drift and inconsistencies** directly.
- Use `a_pf`, `a_THD`, `a_upper`, `a_lower` to drive alarms and visual badges (for example, colour-coded panels).
- Feed `e_*`, `a_*`, and `r_P` into ML models as **stable features** that mean the same thing across devices, sites, and vendors.
- Build portfolio-level views where hundreds of feeders and plants share the same symbolic axes.

No special treatment is required: SSMEQ fields are just extra columns with **explicit, documented meaning**.

4.5 One-minute acceptance checklist

A deployment can call itself “SSMEQ-aligned” when all of the following are true:

1. A **profile manifest** exists and is published, naming `V_ref`, `I_ref`, `f_ref`, `pf_ref`, `P_scale`, lenses, dials, and residuals.
2. Every symbolic field (`e_*`, `a_*`, `r_P`, `band_Q`) is tagged with a `manifest_id`.
3. Any party with access to the raw channels and the manifest can recompute `e_*`, `a_*`, and `r_P` and obtain the **same values** within numeric tolerance.

4. Changes to anchors, lenses, or band rules are made by **minting a new manifest_id**, not by silently editing the old one.

One-line takeaway for Section 4

Adopting SSMEQ means **publishing one manifest and adding one script** — after that, every device, site, and model can share the same, replayable symbolic view of electrical reality.

---

5. Conclusion — What Changes When SSMEQ Is On

With SSMEQ, electrical telemetry stops being a patchwork of units and vendor quirks and becomes a **single, zero-centric symbolic language** that any tool can replay.

5.1 The shift in one view

When SSMEQ is active, each sample carries:

- **Raw truth preserved:**
  - `m_Q` stays exactly as your meters emit it (`V_rms`, `I_rms`, `P`, `f`, `pf`, `THD`, etc.).
- **Zero-centric contrasts:**
  - `e_Q` lanes such as `e_V := ln( V_rms / V_ref )`, `e_I := ln( I_rms / I_ref )`, `e_f := ln( f / f_ref )`, `e_P := asinh( P / P_scale )`.
- **Bounded health dials:**
  - Optional `a_Q` in `(-1,+1)` such as `a_pf`, `a_THD`, `a_upper`, `a_lower`, safe to combine and compare across fleets.
- **Explicit residuals:**
  - Power consistency via `r_P := e_P_meas - ( e_V + e_I + ln( pf / pf_ref ) )` as a standing check on CT/PT scaling, pf estimation, and wiring.
- **Profile awareness:**
  - A `manifest_id` that tells every consumer exactly which lenses, anchors, and band rules were used.

The result is an electrical layer where **"normal", "margin", and "drift" have the same meaning** across sites, vendors, and time.

5.2 Micro-adoption playbook

- **Day-1 — Add symbolic fields, change nothing else.**
  - Publish a minimal manifest with `V_ref`, `I_ref`, `f_ref`, `pf_ref`, `P_scale`.
  - Compute and emit `e_V`, `e_I`, `e_f`, `e_P` beside existing channels, tagged with `manifest_id`.
- **Day-7 — Turn on health and residuals.**
  - Add `a_pf`, `a_THD`, simple limit dials (`a_upper`, `a_lower`), and `r_P`.
  - Start plotting these lanes in dashboards to see where inconsistencies and edge-cases really live.
- **Day-30 — Make decisions symbolically.**
  - Let alarms, reports, and ML models consume `e_*`, `a_*`, and `r_P` as primary features.
  - Keep raw tags as a safety lane, but let the **symbolic layer** drive fleet-level comparisons, tuning, and audits.

5.3 Human protections (why this matters)

- **For operators and field engineers**
  - Easier to spot mis-scaled feeders, suspect CT/PTs, and wiring issues via `r_P` and dials like `a_upper`, `a_lower`.
  - Less guesswork when comparing assets across plants or vendors — one common symbolic scale.
- **For vendors and integrators**
  - A neutral, open representation that reduces integration friction and avoids proprietary "indices" that cannot be replayed.
- **For auditors, regulators, and analytics teams**
  - A transparent mapping from raw data to symbolic views: anyone can recompute `e_*`, `a_*`, and `r_P` from the manifest and raw channels and check that conclusions are grounded in the same rulebook.

One-line closing

Keep your volts, amps, and watts exactly as they are — **add a small, manifest-driven symbolic layer**, and let every device, site, and model finally speak the same electrical language.

Minimal Profile Manifest — Shunyaya Symbolic Mathematical Electrical Quantities (SSMEQ)

Copy-paste template for lenses, anchors, dials, and residuals

---

Below is a **minimal, paste-ready manifest** you can adapt for a plant, feeder group, or device family.
Adjust only the values and IDs; keep the structure and formulas replayable.

SSMEQ_PROFILE_MANIFEST (illustrative template)

```
manifest_id      = PLANT_A.LV_FEEDERS.v1
description      = "LV feeders — Plant A — SSMEQ profile"
scope            = "LV switchboard, outgoing feeders, 400 V class"

# Anchors and scales (per-phase RMS unless stated)
V_ref            = 400.0        # volts (line-to-line nominal)
I_ref            = 100.0        # amps  (typical rated feeder current)
f_ref            = 50.0         # Hz    (nominal grid frequency)
pf_ref           = 0.98         # target power factor (lagging)
P_scale          = 40000.0      # watts (e.g., 40 kW rating for e_P =
asinh(P/P_scale))

# Quality references
THD_ref          = 5.0          # percent THD at acceptable quality
baseline
c_THD            = 0.25         # slope factor for THD dial

# Small epsilons and clamps
eps_a            = 1e-6         # for dials (avoid hitting exactly -1 or
+1)
eps_w            = 1e-9         # for fused dials (denominator guard in
tanh(U / max(W, eps_w)))

# Lenses (zero-centric contrasts)
# e_V := ln( V_rms / V_ref )
# e_I := ln( I_rms / I_ref )
# e_f := ln( f / f_ref )
# e_P := asinh( P / P_scale )

lens_e_V         = "e_V := ln( V_rms / V_ref )"
lens_e_I         = "e_I := ln( I_rms / I_ref )"
lens_e_f         = "e_f := ln( f / f_ref )"
lens_e_P         = "e_P := asinh( P / P_scale )"

# Health dials (bounded in (-1,+1))
# a_pf := clamp( pf, -1+eps_a, +1-eps_a )
# a_THD := tanh( c_THD * ln( 1 + THD/THD_ref ) )
```

```
dial_a_pf         = "a_pf := clamp( pf, -1+eps_a, +1-eps_a )"
dial_a_THD        = "a_THD := tanh( c_THD * ln( 1 + THD/THD_ref ) )"

# Optional limit dials (example for upper voltage margin)
# e_V_upper is e_V at upper allowable voltage (e.g., +10% of nominal)
V_upper_factor    = 1.10
e_V_upper         = "ln( V_upper_factor )"
k_upper           = 1.0          # slope for upper margin dial
dial_a_upper      = "a_upper := tanh( k_upper * ( e_V - e_V_upper ) )"

# Power identity residual
# r_P := e_P_meas - ( e_V + e_I + ln( pf / pf_ref ) )

residual_r_P      = "r_P := e_P_meas - ( e_V + e_I + ln( pf / pf_ref ) )"

# Bands over a generic dial a_Q (e.g., a_pf or a_THD)
# These are illustrative GREEN / AMBER / RED cutpoints

band_GREEN        = "abs(a_Q) <= 0.3"
band_AMBER        = "0.3 < abs(a_Q) <= 0.7"
band_RED          = "abs(a_Q) > 0.7"

# Notes:
# 1) Any change to V_ref, I_ref, f_ref, pf_ref, P_scale, lenses, or bands
#    MUST mint a new manifest_id (e.g., PLANT_A.LV_FEEDERS.v2).
# 2) All SSMEQ records emitted under this profile MUST carry manifest_id
unchanged.
```

How to use this manifest template

- **Step 1 — Copy and rename.**
  Copy the template into a text file (for example, `SSMEQ_PROFILE_PLANT_A_LV.txt`) and change:
  - `manifest_id`
  - `description`, `scope`
  - Numeric values for `V_ref`, `I_ref`, `f_ref`, `pf_ref`, `P_scale`, `THD_ref`, and limits.
- **Step 2 — Freeze and publish.**
  Store the manifest under version control and make it visible to anyone who will:
  - Configure devices or gateways.
  - Compute `e_*`, `a_*`, and `r_P` in pipelines.
  - Build analytics or ML models using SSMEQ fields.
- **Step 3 — Stamp every record.**
  Ensure every SSMEQ record emitted by devices, gateways, or jobs **carries `manifest_id` exactly as written** so that any party can:
  - Take raw channels `V_rms`, `I_rms`, `P`, `f`, `pf`, `THD`.
  - Apply the formulas in the manifest.
  - Reproduce `e_V`, `e_I`, `e_f`, `e_P`, `a_pf`, `a_THD`, and `r_P` within numeric tolerance.
- **Step 4 — Version by addition, never by edit.**
  If you later change anchors, scales, or band cutpoints (for example, upgrade feeders or adjust limits), do **not** edit the old manifest in place.

  - Create a new manifest with a new `manifest_id` (for example, `PLANT_A.LV_FEEDERS.v2`).
  - From that point onward, emit SSMEQ records using the new `manifest_id`.
  - Historical data remains tied to the original rulebook.

One-line takeaway for the manifest template

A single, published profile manifest — with anchors, lenses, dials, residuals, and bands spelled out in plain text — is all you need for any device, script, or model to speak **the same SSMEQ language** for electrical quantities.

OMP