Protocol-agnostic edge: canonical data at the source

WHITEPAPER · CONNECTIVITY & EDGE

Why normalizing industrial signals to one vocabulary — at the edge, before anything routes — is the decision that makes everything downstream simpler.

Abstract. A mixed-vendor floor speaks a dozen dialects. The common reflex is to normalize late — in the historian, the SCADA, or a cloud pipeline — after every system has already had to understand every protocol. This paper argues the opposite: normalize *at the source*, to one canonical vocabulary, before any routing decision is made. That single choice is what lets one platform serve a FANUC cell, a Siemens line, and a Modbus-fronted press without re-deriving its logic per vendor.

The problem: every vendor is its own integration project

A real industrial floor is an archaeology of controllers — FANUC over FOCAS2, open-standard machines over MTConnect, Brother over its HTTP interface, older machines fronted by Modbus PLCs, Siemens cells over S7. Each speaks a vendor-specific dialect: a spindle speed, a cycle-complete signal, and a fault code all look different on every controller.

The usual response is to handle that difference *everywhere*: a per-protocol script feeding the historian, a custom tag map in the SCADA, a transform in the cloud pipeline. Every one of those downstream systems then has to understand every protocol — and every firmware update threatens to break a script that nobody owns. The integration cost doesn't scale with the number of machines; it scales with the number of machine *types* times the number of downstream systems.

The idea: normalize once, at the edge, before routing

Canonical-at-the-source inverts that. A protocol-agnostic edge runtime polls each controller in its *native* protocol, and immediately maps every reading to a **canonical data point** — a single, shared vocabulary that does not depend on which controller produced it. A spindle reading from a 2009 FANUC and a 2024 cell become the same canonical spindle_rpm; a cycle signal becomes the same canonical cycle_time, whatever the source.

The key word is *before*. Normalization happens at the edge, **before any routing decision** — so every downstream consumer (a broker, a historian via its own subscription, an analytics layer) receives one vocabulary, not twelve. The protocol diversity is absorbed once, at the boundary, instead of leaking into every system that touches the data.

Why "at the source" beats normalizing late

Three properties fall out of doing it at the edge rather than centrally:

- **The downstream stops caring about protocols.** Your historian, your SCADA, your analytics platform read canonical signals. Add a new controller vendor and the downstream is unchanged — only the edge gains one more native collector.
- **The pipeline stays deterministic.** When the canonical shape is fixed at the source, the transform that follows is testable and replayable: the same input produces the same output, every time. That is what makes an operational data path auditable rather than a black box.
- **One semantics, many machines.** Because the vocabulary is shared, the same OEE definition, the same alarm logic, and the same reports apply across every machine and every vendor — the number a customer audit asks about traces back to a real, consistently-named signal.

How Elpis does it

EdgeConnect is the protocol-agnostic edge runtime. It collects from the controllers that ship support today — FOCAS2, MTConnect, Brother HTTP, Modbus TCP, OPC UA Client, and Siemens S7 — and normalizes every reading to the canonical vocabulary at the edge. (FANUC MT-LINKi REST integration is on the roadmap; a Linux runtime is on the roadmap.) The canonical stream is then published — to MQTT, or exposed via OPC UA Server — and EREMOS V2 turns it into OEE, alarms, and reports.

The architectural point is that the canonical layer sits *between* the floor and everything else: the protocol-agnostic core never knows about a specific downstream, and no collector formats data for a specific consumer. That separation is what the locked architecture calls the Industrial Intelligence Stack — see **/architecture** for the column model and **/capabilities/connectivity-edge** for the runtime story.

What this is *not*

- **Not a rip-and-replace.** The canonical layer sits *beside* your SCADA, historian, and MES — it feeds them canonical signals instead of vendor-specific ones; it does not take over operator HMIs, control logic, or alarm acknowledgment.
- **Not an AI claim.** Normalization is deterministic, not learned. The pipeline is testable and replayable; AI, where it appears in the platform, is decision-support outside the data path.
- **Not a universal-protocol promise.** "Protocol-agnostic" means the *architecture* doesn't bake in a vendor — not that every protocol ships today. The supported set is explicit and grows by shipping collectors, not by marketing.

Takeaways

1. Normalize at the source, to one vocabulary, **before** routing — so protocol diversity is absorbed once, not in every downstream system.
2. A fixed canonical shape at the edge is what makes the pipeline deterministic, testable, and auditable.
3. One semantics across vendors is what lets a single OEE definition, alarm model, and report set hold across a mixed floor.
4. The win is structural: adding a vendor costs one collector at the edge, not a rewrite everywhere else.
