OEE you can defend: computing segments on canonical signals

WHITEPAPER · OPERATIONAL INTELLIGENCE

Why an OEE number is only as trustworthy as the signals beneath it — and what it takes to make one that survives an audit.

Abstract. OEE is the number every plant review turns on, and it is almost always assembled the least defensible way: stitched by hand from vendor dashboards and spreadsheets, re-keyed at each level of the org, reconciled rather than derived. This paper argues that a defensible OEE is not a higher number — it is a *traceable* and *consistent* one. That means computing OEE segments from canonical signals collected and timestamped at the edge, retaining those signals, and keeping the OEE *definition* in the customer's hands. Because the canonical vocabulary is the same across every machine, the math is the same across every machine, shift, and vendor.

The problem: a number nobody can trace

Walk into a plant review and ask where the OEE figure came from. Often the honest answer is a chain: one vendor dashboard exported a runtime, another tool supplied downtime, a supervisor keyed both into a spreadsheet, a manager adjusted a category, and corporate rolled it up. Each hop is a small reconciliation. By the time the number reaches the review, it is the output of a process — not a measurement.

That number does not survive scrutiny. When someone asks *"why was this cell counted as down here but idle there?"* there is no signal to point to, only a sequence of judgement calls made days apart by people reading different screens. The classic stitched-spreadsheet OEE process — operator to supervisor to manager to corporate, each adding their own re-calculation — produces a figure that is plausible but not *defensible*. And it gets worse on a mixed-vendor floor, where each CNC, PLC, or DAQ vendor ships its own visualization layer and its own idea of what "running" means. The same machine state can be classified two different ways depending on which dashboard you happened to read.

The cost is not just effort. It is that the most important operational metric in the plant rests on data that cannot be traced back to anything real.

The idea: defensible means traceable and consistent

"Defensible" is worth defining precisely, because it is easy to hear it as "better." It is not. A defensible OEE makes no claim to be higher or lower than the stitched version. It makes two narrower, harder claims:

- **Traceable.** Every segment of OEE resolves to a signal that was actually collected from a controller, timestamped at the edge, and retained. When someone asks why a span counted as UNPLANNED_STOP rather than IDLE, the answer is a stored signal with a time on it — not a reconstruction.
- **Consistent.** The same machine state is classified the same way everywhere — across machines, across shifts, across vendors — because every machine is reasoned about in the same vocabulary.

Neither of those is a percentage. That is the point. The honest promise of a well-built OEE pipeline is not a gain; it is that the number means the same thing tomorrow as today, on machine forty as on machine one, and that you can walk it back to a real event when challenged.

Why segments on canonical signals hold up

The mechanism that delivers both properties is computing OEE through **segments** over **canonical signals**, rather than reconciling vendor outputs after the fact.

Start with the canonical foundation (the subject of the companion paper on the protocol-agnostic edge): each controller is polled in its native protocol, and every reading is normalized at the edge into one shared vocabulary before anything routes. A spindle reading from an older FANUC and a newer cell become the same canonical

signal regardless of origin. That normalization is deterministic — the same input yields the same output, every time — which is precisely what makes a downstream calculation replayable and auditable instead of a black box.

On top of that foundation, OEE is computed as **segments** — spans of machine state classified as RUNNING, PLANNED_STOP, UNPLANNED_STOP, IDLE, or SETUP. Each segment is derived from canonical signals that were collected and timestamped at the edge, then retained. Two things follow:

- **The number traces back to a real signal, not a reconciliation.** A segment is an interpretation of stored, edge-timestamped data — so any figure built from segments can be walked back to the underlying signal that produced it.
- **The math is the same across every machine and vendor.** Because the vocabulary feeding the segment logic is canonical, the segment classification does not need to be re-derived per controller type. One semantics covers the FANUC cell, the Modbus-fronted press, and the Siemens line alike. Same definitions across every plant; same math across every shift; same source-of-truth across every dashboard.

This is the difference between a number you reconcile and a number you derive. Only the second one survives the question *"show me why."*

The OEE definition stays yours

A defensible pipeline is not a defensible *opinion about OEE*. Every plant draws the lines differently: what counts as a planned stop versus an unplanned one, where setup ends and running begins, how shifts are bounded, what the targets are. Those are the shop's decisions, encoded in how it already runs.

So the segment classification, the shift schedule, and the targets are *configured to the customer's OEE definition* — not to a vendor preset. The platform supplies the discipline (canonical signals, deterministic segmentation, retention, traceability); the customer supplies the meaning (where the category boundaries sit, how shifts are cut, what "good" is). The number is computed against *your* definition, which is the only way an OEE figure can be both consistent and credible inside a specific operation. A traceable pipeline running someone else's definition would just be a defensible version of the wrong number.

How Elpis does it

EdgeConnect is the protocol-agnostic edge runtime that collects from the controllers supported today — FOCAS2, MTConnect, Brother HTTP, Modbus TCP, OPC UA Client, and Siemens S7 — and normalizes every reading to the canonical vocabulary at the edge, with each reading timestamped there. (FANUC MT-LINKi REST integration is on the roadmap; a Linux runtime is on the roadmap.) EREMOS V2, the multi-tenant analytics layer, computes **OEE via Segments** (RUNNING / PLANNED_STOP / UNPLANNED_STOP / IDLE / SETUP) on those edge-collected signals — against your OEE definition, not a vendor preset.

EREMOS V2 models the real industrial hierarchy — PLANT → AREA → LINE → EQUIPMENT → SUB_EQUIPMENT — so segments and reports resolve to actual assets, and it is multi-tenant by design: each plant's data stays isolated within the tenant boundary you define, and a multi-plant view aggregates across per-plant runtimes without losing per-site identity. The reporting outputs are the tangible side of this — shift reports, OEE summaries, downtime breakdowns — all built on the same segmented, canonical source-of-truth, so they reconcile with each other instead of disagreeing. For the column model and where this analytics layer sits, see **/architecture**; for the full pillar story, see **/capabilities/operational-intelligence**; for the outcome narrative on tight-tolerance shops, see **/solutions/precision-manufacturing**.

What this is *not*

- **Not a claim of higher OEE.** Defensible means traceable and consistent — not improved. This paper makes no quantified promise about uptime, OEE points, or savings; those wait for verified evidence in context.
- **Not Elpis's definition of OEE imposed on you.** Segment boundaries, shift schedules, and targets are configured to how your shop already defines OEE. The platform brings the discipline, not the opinion.

- **Not a rip-and-replace.** This sits *beside* your SCADA, historian, and MES — computing OEE on canonical signals rather than taking over control logic, operator HMIs, or alarm acknowledgment.
- **Not an AI claim.** Segmentation is deterministic and replayable, not learned. Where AI appears in the platform it is decision-support outside the data path — it never classifies a segment or moves a number.

Takeaways

1. A defensible OEE is **traceable and consistent**, not higher — every segment resolves to a real, edge-timestamped signal, and the same state is classified the same way everywhere.
2. Computing OEE as **segments over canonical signals** is what makes the number derivable rather than reconciled, and auditable rather than stitched.
3. Because the vocabulary is canonical, **the math is the same across every machine, shift, and vendor** — one semantics, not one per dashboard.
4. The **OEE definition stays the customer's** — classification, shifts, and targets are configured to how the shop already defines OEE, beside the existing stack, with AI kept out of the data path.
