PERINIMBLE · WHITE PAPER · GRID DATA FAMILY

Every Read Kept, Every Decision on the Ledger

Why a meter data system should never change a read to correct it, and how a decision ledger lets a utility prove where every billed value came from

PRODUCT	Grid Data Management Platform
AUDIENCE	Billing and meter data operations, VEE analysts, IT and integration leads, audit and regulatory teams

Where meter data quality really gets decided

Most talk about meter data quality is about VEE: the rules that validate reads, the methods that estimate missing ones, and the exceptions analysts work. Under all of that sits a design choice that gets much less attention. When VEE decides a value is wrong or missing, what does it do to the data?

In many meter data systems the answer is that it changes it. The estimate goes into the slot where the read was. The corrected value replaces the bad one. A flag or a version number records that something happened. Later, a late actual arrives, an analyst overrides the estimate, and a bill has already gone out. Then "what did we bill from, and why?" becomes an investigation.

The Grid Data Management Platform takes a different approach. Reads are never changed to correct them. Every correction is a decision, and every decision goes on a ledger.

Why correcting in place breaks down

The read and the judgment get mixed up. Once an estimate sits where the read was, the system can no longer separate what the meter reported from what VEE concluded. Each later step has to guess.

Late data has to fight the earlier edits. A read for last Tuesday can arrive on Friday, after Tuesday was validated, estimated and billed. If the estimate was written into the data, the system needs special rules to work out whether the new read should replace it, and for which intervals.

Manual work gets lost. An analyst's override can be overwritten by the next automatic estimate, or block a better actual that arrives later. Both are common sources of billing disputes.

History becomes a rebuild. Auditors and regulators ask what you knew, when you knew it, and what you billed from. If values were changed in place, the answer has to be pieced together from logs, if the logs exist.

Replays become risky. If reprocessing can rewrite values, every recovery puts the data you already billed at risk, and teams hesitate to replay.

Three layers, kept apart

The platform keeps three things separate that many systems combine.

1. **What was received.** Interval and register reads are accepted once. A resend, retry or replay is recognized and not stored again. Reads are organized into a day for each meter and channel. When late or corrected data arrives, the platform creates a new version of the day, and the earlier version stays on record as superseded.
2. **What was decided.** Every judgment about a day, from accepting the received values to estimating, overriding or finalizing it, is a separate entry on a decision ledger. Each entry records who or what made it, when, and why. An estimate or override covers only the intervals it changes. A decision is never rewritten. A correction is a new entry that points back to the one it replaces.
3. **What is served.** The value that billing, analytics and other systems use is worked out from the first two layers. A configurable precedence decides which decision wins, so finalized days and analyst overrides always beat automatic estimates. The served answer can always be rebuilt from the ledger, and it can always be traced back to it.

Findings are signals; configuration decides the action

Validation in the platform doesn't change values. It raises signals. A gap, a negative delta or a register that disagrees with its intervals becomes a signal. Configuration decides what each signal leads to, for each group of meters: estimate, open an exception for review, flag only, notify someone, or open a service order. Several actions can follow from one signal.

Only the actions that change a value write a decision. So the ledger holds the corrections the utility actually chose, not everything a rule happened to notice. The same finding can lead to an estimate for residential meters and to an analyst review for large commercial accounts, and the ledger shows which path each meter took.

What the ledger makes possible

- **A late actual replaces an estimate without overwriting it.** When stored intervals arrive after an outage, they replace the estimate only for the intervals they cover. The estimate stays on the ledger, marked as replaced.
- **Analyst work is protected.** An override ranks above automatic estimates, so the next VEE run can't silently undo it.
- **Finalized days stay final.** Once a day is finalized for billing, no further decisions can be added to it. A change after that goes through the governed rebill path.
- **Any past answer can be rebuilt.** The platform can reconstruct the value it was serving for any meter, channel and day at any earlier moment.
- **Bills point to decisions.** Billing determinants keep links to the ledger entries they were built from, so "which data was this bill built on?" has an exact answer.
- **Decisions keep their context.** Every decision can be traced to the VEE run and rules that produced it, and a VEE result based on out-of-date data is never applied.

- **Replays are safe.** Duplicate reads are recognized on arrival, so a replay cannot count usage twice, and it cannot stack duplicate corrections.

> **A WORKED EXAMPLE (HYPOTHETICAL)**
>
> A storm cuts communications to a group of meters for three days. VEE raises missing-data signals. For these residential meters, configuration says estimate, so an estimate decision is added to the ledger for each affected day. The billing cycle closes on time, using the estimates.
>
> When communications return, the meters upload their stored intervals. That same morning, the head-end recovers from its own outage and resends a batch it had already sent before the storm.
>
> - The resent batch is recognized and set aside, so nothing is counted twice.
> - The uploaded intervals become new versions of the three days. For every interval they cover, the actual now outranks the estimate. Where a meter is still missing two hours, the estimate stays in place for just those two hours.
> - The estimate decisions stay on the ledger, now replaced by the actuals.
> - Billing sees that a closed cycle's data has changed, and the account goes through the normal rebill path.
>
> Two weeks later the customer calls. The analyst can show the original estimate and the rule that produced it, when the actuals arrived, which intervals they replaced, which ledger entries the first bill and the rebill were each built from, and what the platform was serving on the day the first bill went out. If the customer escalates to the regulator, the utility answers with records, not a reconstruction.

Questions to ask any MDMS vendor

- When VEE estimates a missing interval, is the estimate written into the read data, or recorded separately so the received read stays intact?
- When a late actual arrives, does it replace the estimate only for the intervals it covers? Show us.
- Can an automatic VEE rerun overwrite an analyst's manual override?
- For one meter and day, can you list every decision: who or what made it, when, why, and what it replaced?
- Can you show exactly what value you were serving for that meter and day on a past date?
- Can you show which decisions a specific bill was built from?
- Can the same validation finding lead to different actions for different customer classes, without code changes?
- If a head-end resends a batch, or we replay a week of traffic, what happens?

The foundation under every bill

Validation, estimation, billing and settlement all depend on knowing the difference between what a meter reported and what the utility decided about it. Keep the two apart, record each decision on a ledger that is never rewritten, and serve values that can always be traced to that ledger. The utility can then answer, with evidence, what it knew, when it knew it, and why it billed what it billed.

Learn more about the Grid Data Management Platform, or talk with our team.

ABOUT PERINIMBLE

PeriNimble builds mission-critical grid data software for energy and utility companies. Founded in 2022 and headquartered in Bloomington, Minnesota, we support more than 15 million meters and over 1 billion interval reads per day in production.

Built for systems that can't afford failure. · perinimble.com · perinimble.com/contact