

A Whole Utility You Can Test Against

Synthetic territories, networks, meters, and reads, delivered in your system's own data model, so testing no longer depends on customer data.

PRODUCT

Grid Data Simulator

AUDIENCE

MDMS and CIS program managers, QA and test leads, integration architects, IT leadership

The testing gap nobody budgets for

Every utility system that touches meter data needs to be tested against something that looks like a utility. A new MDMS, a CIS upgrade, a change to VEE rules, an outage integration, a head-end replacement: each one has to prove it handles real volumes, real relationships, and real mess before it goes live.

Most programs end up choosing between two poor options.

The first is a copy of production. It has the right scale and the right shape, but it is full of customer names, addresses, and interval usage that reveals when people are home. Privacy policy, regulators, and common sense all limit how widely that data can travel, and it rarely belongs in a vendor's test lab. Masking helps, but it tends to break the relationships that make the data useful, and it still carries the risk of re-identification.

The second is a hand-made dataset. It is safe, but small: a few dozen meters, clean reads, no storms, no meter exchanges in the middle of a billing cycle, no late data. It proves the happy path and very little else.

The result is familiar. Defects surface in production the first time a storm takes out a feeder, the first time a move-out and a move-in land on the same day, or the first time the system meets a full day of reads at utility scale.

Why it is hard to do well

A utility is a web of relationships. Premises, service points, meters, channels, customers, contracts, and the network they hang on all have to agree with each other, and they change over time. Test data that gets one of those relationships wrong tests the wrong thing.

Data has to arrive the way it really would. Reads come on their own cadences, some late and some out of order. A meter exchange arrives as a specific transaction from the CIS, followed by reads from the new meter. A system tested with data that arrives neatly will behave differently when it does not.

Realism has to be dialed in. Storms, DER and EV growth, broken meters, and duplicate reads are exactly the conditions a test needs to cover, and exactly the ones hand-made data leaves out.

You need to know the right answer. To test VEE, it is not enough to feed in bad reads. You need to know, in advance, which reads should be flagged on which meters, so you can tell whether each rule fired where it should.

Results must be repeatable. A failure you cannot reproduce is an anecdote. After a fix, the team needs to run exactly the same scenario again.

Principles of a good approach

1. **Synthetic, not masked.** Generate the utility from scratch, so there is no customer data to protect.
2. **Complete.** Cover the network from substation to meter, across every commodity the utility runs, with the premises, customers, contracts, meters, and reads that go with it.
3. **Grounded in real places.** Premises on real addresses and assets on real roads make the data behave like a real territory, at urban, suburban, or rural density.
4. **Delivered in the target system's own data model.** If a translation layer sits between the test data and the system under test, you end up testing the translation.
5. **Realism on demand.** Storms, adoption curves, faults, and dirty data should be options you switch on, not projects you build.
6. **An answer key.** Every injected fault should come with a record of what the system should find.
7. **Reproducible by design.** The same specification and seed should produce the same utility, every time.

What it looks like in practice

Grid Data Simulator starts from a territory. A user draws one on the map, searches for a place, or describes the scenario in plain English and reviews the drafted specification before anything is generated. From that, the simulator builds a complete synthetic utility: electric, gas, and water networks from substation to meter; premises and service points on real addresses; customers, contracts, and service agreements; meters and their channels; and interval, register, and SCADA data.

It then delivers that utility the way the real systems would. Master data arrives in the target system's own data model. Reads arrive on realistic cadences. Different systems receive what they would receive in production, such as meter data to the MDMS and SCADA to network analytics.

Realism is available on demand:

- **Storms**, restored by a finite number of crews, with estimated restoration times and nested outages, and meters that fall silent while they are out.
- **DER and EV adoption** that grows over time.
- **VEE fault patterns** delivered with an expected-findings manifest, so every rule can be checked against exactly the meters it should fire on.
- **Dirty data**, including duplicates, drift, and late or out-of-order reads.

The simulator also plays the other systems in the conversation. Change a record, and it publishes exactly the transaction a CIS would send, whether that is a meter exchange, a reversal, a removal and install, a move-out and move-in, or an attribute change, followed by the reads that come after. For command and control, it runs

the full round trip from CIS to MDMS to head-end system, covering disconnect, reconnect, on-demand read, power status, and ping, with the simulator on both ends and the MDMS under test in the middle.

Every scenario is reproducible from its specification and seed, and the simulator is built for millions of meters, so load and scale testing can happen before go-live instead of during it.

ILLUSTRATIVE EXAMPLE (HYPOTHETICAL)

A utility preparing to replace its MDMS draws a territory that mixes a dense city center with rural feeders, and generates a synthetic utility across electric and gas. The test team switches on VEE fault patterns and compares the new system's exceptions against the expected-findings manifest. One rule flags fewer meters than the manifest says it should, and the defect is fixed weeks before cutover. Next, they run a storm scenario and confirm that estimated reads for silent meters are handled correctly once power is restored. Finally, they trigger a meter exchange in the middle of a billing cycle. After each fix, they regenerate the same scenario from the same seed and confirm the result.

Questions to ask any vendor

When evaluating a test data approach from any provider, ask:

- Is the data fully synthetic, or derived from customer data?
- Does it cover the whole network and every commodity we run?
- Is it delivered in our target system's own data model, or does it need translation?
- Can it simulate storms, DER growth, VEE faults, and dirty data on demand?
- Does it come with an answer key for injected faults?
- Can it send CIS lifecycle transactions and run a command-and-control round trip?
- Can we reproduce any scenario exactly, from a specification and seed?
- Has it been built for millions of meters?

Where to go from here

Good testing needs a utility to test against. Building one synthetically removes the privacy risk of production copies and the blind spots of hand-made data, and it gives every test a known right answer.

Learn more about [Grid Data Simulator](#), or [talk to our team](#) about your next MDMS or integration program.

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