

Restore to the Second Before the Mistake

Point-in-time recovery, monitoring that survives an outage, and repeatable releases for utility software that cannot afford a bad day.

PRODUCT	ControlPlane
AUDIENCE	IT operations and SRE teams, infrastructure and platform leads, IT and OT security, CIOs

Every operations team has a bad day

Sooner or later, it happens. A data correction script meant for the test environment runs against production. A release includes a change that damages a table nobody thought it touched. An integration writes bad values for an hour before anyone notices.

In most industries, that is an expensive afternoon. In a utility, the data behind it feeds customer bills, wholesale settlement, outage reporting, and regulatory filings. A bad day in the meter data system can turn into weeks of rebills and explanations.

The usual recovery options are not good. Restoring last night's backup throws away every read that arrived since, and all of them must then be recovered and reprocessed. Restoring over the top of production takes the system offline, destroys the evidence of what went wrong, and leaves no way to check the result before committing to it. And the whole procedure often lives in the heads of a few people who know the right commands.

Why recovery is harder than it looks

The mistake is found late. Bad data is usually discovered hours after it was written, with good data continuing to arrive on top of it. The right recovery point is almost never "last night."

Restoring in place is all or nothing. Once you overwrite production, you cannot compare before and after. If the restore point was wrong, you have made the problem worse.

Monitoring goes dark with the system. When the monitoring runs inside the same environment it watches, the dashboards and alerts fail at the moment they are most needed.

Environments drift. Hand-edited configuration, one-off fixes, and manual upgrades mean no two environments are quite the same. Rolling back a bad release becomes guesswork.

Utilities run in restricted places. Many utility systems are on-premises, and some are air-gapped, as critical infrastructure often must be. Tools that assume a connection to the outside world do not work there.

Recovery depends on a few people. If restoring a database requires command-line access and deep platform knowledge, recovery waits for whoever has both.

Principles of a good approach

1. **Restore to a precise moment**, down to the second before the mistake, not to the last scheduled backup.
2. **Verify before you cut over**. Production should keep serving while the restore runs, and the team should verify the result before switching to it.
3. **Keep monitoring available during an outage**, when it matters most.
4. **Make every release versioned and repeatable**, with upgrade and rollback built in and a full release history.
5. **Run where the utility runs**, including on-premises and air-gapped sites, without external dependencies.
6. **Make recovery operable by the team on shift**, through a guided, auditable console rather than shell access and memory.

What it looks like in practice

ControlPlane is the deployment and operations console for the Grid Data family. It installs every application as a versioned release, so each environment runs a known, reproducible configuration, and then stays with those applications in production.

Backup and point-in-time restore. ControlPlane runs scheduled and on-demand database backups. When something goes wrong, an operator can restore any environment to any second within the retention window while production keeps serving. The team checks the restored data before anything changes. Only when they are satisfied do they cut over. This is available for the Grid Data Management Platform and other family applications.

Monitoring that survives the outage. Monitoring and alerting keep reporting through an outage of the application environment. Dashboards are embedded in the console under single sign-on.

Versioned, repeatable deployments. Operators install, upgrade, resize, and roll back complete releases with a full release history. Automated releases follow the same controlled process as releases from the console, so what was tested is what gets deployed.

On-premises and air-gapped sites. ControlPlane can deploy entirely inside the utility's network with no external dependencies, a common requirement for critical infrastructure. The same releases run in large production environments and in small demonstration setups.

No command line required. Operators view live logs and system events across components from the browser. Credentials are encrypted at rest. Recovery becomes a guided procedure the team on shift can follow.

ILLUSTRATIVE EXAMPLE (HYPOTHETICAL)

On a Tuesday afternoon, a script intended for the test environment runs against production and overwrites a set of meter channel configurations. The next morning, the VEE team sees an unusual rise in exceptions and traces it to the change. Using the audit trail, the operations team identifies the second before the script ran. They restore to that moment while production keeps receiving and validating reads. They review the restored channel configurations, confirm the values are correct, and plan how to bring in the reads that arrived after that moment. Then they cut over. Throughout, the monitoring stays up, and nobody needs shell access to the database servers. A month later, a release introduces a regression; the team rolls back to the previous version from the release history and investigates without pressure.

Questions to ask any vendor

When evaluating operations and recovery tooling from any provider, ask:

- Can we restore to a precise second, or only to a scheduled backup?
- Can we verify a restore before cutting over?
- Does production keep serving while the restore runs?
- Does monitoring keep working when the application environment is down?
- Is every release versioned, with upgrade, rollback, and history?
- Does it work on-premises and air-gapped, with no external dependencies?
- Can the operations team run a recovery from a console, without command-line access?

Where to go from here

The bad day will come. What matters is whether recovery is a precise, verified, routine procedure or a late-night improvisation. Utilities should expect to restore to the second before the mistake, check the result before cutting over, and keep watching the whole time.

Learn more about [ControlPlane](#), or [talk to our team](#) about how your Grid Data platforms are deployed and protected.

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