Environment as Code for Murex MX.3.

Every MX.3 environment — PROD, DR, UAT, SIT, DEV, on-premises or in the cloud — is described in Git instead of configured by hand. EOD, batch, health checks and housekeeping run on top of it. Same command, same result, everywhere.

WHAT YOUR MUREX TEAM GETS

- No more silent drift: One command shows every setting that differs between PROD and UAT — before it becomes an incident in the batch or a failed release weekend.
- Environment work in minutes, not days: New environment, refresh, Murex version move: change a few lines in the inventory, render, apply. The knowledge lives in the repository, not in one engineer's head.
- Evidence instead of assertions: Configuration is versioned in Git with a full history. Secrets stay in your vault (CyberArk, Ansible Vault, Infisical) and are only resolved at apply time. What is versioned in Git can be evidenced; what a script assembles at runtime cannot.
- Tooling you do not have to build first: You don't start DevOps and CI/CD for Murex with an empty playbook. MxENV3 is a base that has been running in bank operations since 1998. Your team builds its own playbooks and pipelines on top — instead of re-inventing environments, start orders and checks.

OPERATIONS

- One command, every environment: Whether PROD, DR or the fifth UAT: runprocessingscript, mxcheck, housekeeping or start read everything they need from the environment's inventory. No environment-specific scripts, no copy-paste between hosts, no surprises when the DR site is used for real.
- Thousands of EOD jobs, reliably steered: Every processing script, macro and ant task returns one consolidated result: status, timing, captured logs and the parsed Murex answer. The scheduler branches on facts instead of guessing from a shell exit code. The same job definition works on every environment, because host, user, path and parameters come from the inventory.
- Scheduler jobs generated from one table: The EOD chain is maintained once, as a table. MxENV3 generates the AutoSys job definitions (JIL) from it, with the right hosts, users, paths and date conditions per environment. The same chain in PROD and UAT, without maintaining hundreds of job definitions one by one; the way back from JIL into the table works too. Further schedulers are added on request.
- EOD alerting to whoever is on call: When an EOD job fails, MxENV3 collects the job's log tail and sends a servicedesk ticket by mail, optionally an SMS via the mail gateway. Whoever is on call registers in the on-call rota beforehand; the ticket is assigned to that person. Transport and wording come from the inventory.

IN THE TOOLKIT

- Health checks: mxcheck probes fileserver, middleware, user sessions and open files; every service must answer, compared with a known-good snapshot. Status and metrics come as JSON or plain text, with exit codes for the scheduler and thresholds from the inventory.
- Oracle SQL against the right database: runsql runs queries, DML, DDL and PL/SQL via sqlplus against financial, datamart, MLC or VaR. The connection comes from the inventory, the password from your vault. Plus dry-run, CSV/JSON output and fail-fast batches.
- Murex jobs with one consolidated result: runant, runmacro and runprocessingscript invoke Murex scripts and return command, host, timing, logs and the parsed Murex answer as one JSON — ready for the scheduler and for evidence.
- Housekeeping and archiving: Age-based archiving into verified tar.gz, then cleanup, driven by presets in the inventory: logs, GC logs, core dumps, dated exports. Dry-run first, then on one host or all.

Business Technology Consulting OÜ · Sepapaja tn 6, 15551 Tallinn, Estonia · info@mxenv3.com · https://mxenv3.com

- Diagnostics: Check business date against today, JVM heap per host with thresholds, MXML tasks and middleware ping against a git-versioned known-good snapshot. Plus purge via a node allow-list and ad-hoc commands on a whole role.
- Central inventory server: An optional per-domain server (TLS and token) serves the git-versioned inventory to thin clients. It delta-syncs version-specific templates by checksum and can run jobs on behalf of clients. Clients auto-update from it.

HOW IT WORKS

- 01 Describe your environments: Each environment is a small inventory in the standard Ansible format. Servers are grouped by Murex role (session, processing, XML server …), plus the settings that differ from the bank-wide defaults. Your team already knows the format.
- 02 Resolve values and compare: Defaults, environment, role and host settings are combined into the effective configuration. Query a value, compare two environments, render the full matrix — always with a clear answer which value applies where.
- 03 Plan, apply, operate: Configuration files are rendered for the environment's Murex version and applied to the application directory. EOD steps, checks, housekeeping and service orchestration run from the same inventory. Dry-run by default, execution on request, with exit codes your scheduler branches on.

LIMITS

MxENV3 does not patch, install or upgrade MX.3 itself, does not touch trade or market data, and replaces neither the vendor-supplied installation and administration tools nor your scheduler, your monitoring or your CI/CD. It manages what surrounds them: environment configuration, service orchestration, batch execution and the operational routine across 30 to 50 environments. On a Murex version we have not yet validated, applying the configuration with --strict stops instead of guessing.

HERITAGE

The 2026 rewrite carries the same operational knowledge, across every generation from MxGeneration 2000 to MX.3, on a new foundation: one Go binary, standard inventory, vault integration, production guard. Where a Perl tree of some 250 modules had to be installed on every Murex host and maintained for years, there is now a single file. No interpreter version, no dependencies, no compiler on the server. Copy it and run it. An invocation costs milliseconds of CPU instead of half a second and less than half the memory; rendering needs around 30× less CPU per configuration file.

Measured on identical hardware against the Perl predecessor, ten runs per case: command start-up 4 to 5 ms CPU and ~24 MB instead of ~520 ms and ~54 MB; applyconfig with 880 files ~60 ms and ~40 MB instead of ~2.2 s and ~89 MB. The figures apply to the production binary as delivered. Starting and stopping services is not part of the measurement.

NEXT STEP

Send us an anonymised overview of your Murex environments, or just the number and versions. In a 45-minute session we show you the resulting inventory, the configuration matrix and a start plan. No access to your systems needed. After that you can evaluate on a non-production environment yourself: it starts with commands that only read — configuration matrix, environment comparison, drift check against the deployed files. Nothing is written until you release it.

Licence	Annual licence per institution, scaling with the number of concurrently running environments.
Contact	info@mxenv3.com · https://mxenv3.com