Continwm Version 2.0

Continuous Monitoring & Hypothesis Testing — What's New

Code Magus Limited

www.codemagus.com

2026

Agenda

What is Continwm?

Continwm is a continuous hypothesis-testing and anomaly-detection system for time-series performance metrics.

Core idea: instead of threshold alerts, apply a **statistical test** comparing a live sample window against a historical baseline.

- **Any metric** from any broadcaster
- **Any statistical test** âĂŤ pluggable via library
- **Significance-level driven** alerts (p-value vs. α)
- Persistent **case tracking** and root-cause management

Test Decision

$p < \alpha \quad \Rightarrow$ **FAIL** (anomaly)

$p \geq \alpha \quad \Rightarrow$ **PASS** (normal)

V2 in one line

Open the test framework to **any language**, any algorithm, any alerting channel.

Pluggable Statistical Tests — The Big Leap in V2

In V1 the test algorithm was fixed inside the daemon.

V2 introduces the `cntwmifc` interface library:

- The test driver (`cntwmtd`) **fork/exec**s a separate test process for every hypothesis test.
- The test process connects via **Unix pipes** using `cntwmifc`.
- The test process can be **written in any language** that can call a C shared library or read/write a pipe.
- Multiple, **independently configured** tests can run on the same or overlapping sets of metrics simultaneously.

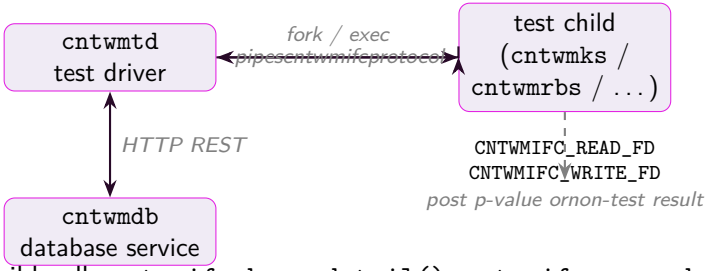
Available test modules (V2)

`cntwmtst`	Z-test (C, built-in)
`cntwmks`	KS test (C)
`cntwmrks`	KS test (R)
`cntwmrbs`	Bootstrap (R)
`*.py`	Any Python test
`*`	Any executable

Key point

Contribute your own test in R, Python, C or any other language.

How the Test Interface Works



The test child calls `cntwmifc_base_detail()`, `cntwmifc_samp_detail()`, `cntwmifc_base_summary()`, then posts its p-value via `cntwmifc_post_result()`. The driver handles all database I/O — the test child sees clean data structures.

Four Statistical Tests Ready to Use

Z-test — `cntwmtst` (C)

Two-sample Welch Z-test using *summary statistics* only.
Fast; suitable for high-frequency metrics where distributional shape is not critical.

$$z = \frac{\bar{x}_{\text{samp}} - \bar{x}_{\text{base}}}{\sqrt{s_s^2/N_s + s_b^2/N_b}}$$

KS-test — `cntwmks` (C)

Two-sample Kolmogorov-Smirnov test on *per-interval mean* values from full detail rows.

$D = \max\left|F_{\text{base}}(x) - F_{\text{samp}}(x)\right|$

Detects shifts *and* shape changes in the distribution.

R KS-test — `cntwmrks` (R)

Identical KS hypothesis using R's native `ks.test()`. Produces a **ggplot2 ECDF comparison chart** written to the test report directory for visual inspection.

Bootstrap test — `cntwmrbs` (R)

Non-parametric bootstrap test. Builds a population from baseline detail rows; the p-value is the fraction of bootstrap resample means that deviate from the baseline grand mean by at least as much as the observed sample mean. Robust to non-normal distributions.

Write Tests in R — the cntwmifc R Package

The cntwmifc R package wraps the C library. A minimal R test looks like:

```
library(cntwmifc)

token <- cntwmifc_startup(
  "my_test", CNTWMIFC_VERBOSE)

base <- cntwmifc_base_detail(token)
samp <- cntwmifc_samp_detail(token)

# ... your R statistics here ...
pvalue <- ks.test(
  samp$delta_s_value / samp$delta_count,
  base$delta_s_value / base$delta_count
)$p.value

cntwmifc_post_result(token, pvalue)
cntwmifc_closeup(token)
```

Install and go

```
R CMD INSTALL cntwmifc/R/cntwmifc
```

Data frames returned by the R package:

- epoch, sequence
- delta_count
- delta_s_value
- delta_sq_value
- delta_usec
- detail_id
- row_number

Full **ggplot2** visualisation available immediately from within the test script.

Write Tests in Python

The `cntwmifc.py` pure-Python module implements the full `cntwmifc` pipe protocol.

```
from cntwmifc import (
  cntwmifc_startup, cntwmifc_closeup,
  cntwmifc_base_detail,
  cntwmifc_samp_detail,
  cntwmifc_post_result,
  CNTWMIFC_VERBOSE)
from scipy.stats import ks_2samp

token = cntwmifc_startup(
  "my_python_test", CNTWMIFC_VERBOSE)

base = cntwmifc_base_detail(token)
samp = cntwmifc_samp_detail(token)

base_means = [r['delta_s_value'] /
              r['delta_count']
              for r in base['rows']]
samp_means = [r['delta_s_value'] /
              r['delta_count']
              for r in samp['rows']]

_, pvalue = ks_2samp(base_means, samp_means)
cntwmifc_post_result(token, pvalue)
cntwmifc_closeup(token)
```

Pure Python, no compilation

`cntwmifc.py` — single file,
no build step required.

Use any Python library

- `scipy.stats` for KS, Mann-Whitney, ...
- `statsmodels` for ARIMA, regression, ...
- `sklearn` for ML-based detection
- `matplotlib` / `seaborn` for plots
- `pandas` for data manipulation

No CONTINWM needed to develop

Debug mode: run standalone
without a live test driver.

Develop and Test Without a Running System

Debug mode is activated automatically when the driver pipe environment variables are absent.

In debug mode:

- `cntwmifc_startup()` returns a valid handle immediately with no pipe I/O.
- Synthetic baseline and sample data are injected via `cntwmifc_debug_set_*` helpers.
- All retrieval functions work normally against the injected data.
- `cntwmifc_post_result()` prints to `stderr` instead of writing to the pipe.

Implication: the complete statistical logic of a new test can be developed, debugged, and validated with known datasets before the script is ever deployed as a live test process.

Run standalone (R)

```
Rscript cntwmrbs.R --sig
Rscript cntwmrbs.R --nonsig
```

Run standalone (Python)

```
python3 sample_cntwmifc.py
```

Run standalone (C)

```
cntwmks --verbose
```

Deploy = remove debug block

The same script, unchanged except for removing the `cntwmifc_debug_*` data injection, runs live in production.

Individual Test Customisation

Each **test configuration** is a named row in the `test_configs` table, specifying:

- **`apply_to_metrics`** — an expression filter selecting which metrics this test applies to (source, group, name, IP, etc.)
- **`test_process`** — the executable to run (e.g. `cntwmks`, `cntwmrbs.R`)
- **`test_process_parms`** — extra CLI arguments for the test process
- **`alpha`** — significance level per test
- **`test_size`** — sample rows before triggering
- **`tolerance`** — consecutive failures before opening a case
- **`notify_process`** + **`notify_process_parms`** — per-test alerting command

Multiple tests on one metric

A metric can simultaneously be evaluated by:
— a fast Z-test with $\alpha = 0.01$
— a KS-test with $\alpha = 0.05$
— a custom ML anomaly detector
Each with its own notification channel.

`apply_to_metrics` example

```
source like "posipmon"&&
name like ".*resp.*"
```

Live Configuration Management

Test configurations can be **added, changed, and deleted while the system is running** — no downtime required.

```
# Add a new bootstrap test
cntwmtc.sh --add-config \
  --name=bstest \
  --title="Bootstrap␣Significance␣Test" \
  --apply-to-metrics='source like "posipmon"' \
  --test-process=cntwmrbs.R \
  --notify-process=continwm_email_notify.sh \
  --notify-process-parms='ops@example.com' \
  --alpha=0.05 --test-size=50 --tolerance=3

# Change the test process for an existing config
cntwmtc.sh --edit-config \
  --test-config-id=2 \
  --test-process=cntwmks
```

A `SIGUSR1` signal is sent to `continwm` immediately after any change, reloading its in-memory configuration with no interruption to metric collection.

What can be changed live

- Test process / parameters
- Notify process / parameters
- Significance level (α)
- Sample size / tolerance
- Metric filter expression
- Title / description

The web UI (`continwm.html`) exposes the same configuration operations via a browser-based CRUD interface accessible to any authorised user.

Flexible, Per-Test Alerting Integration

Every test config has its own notification command.

When a test fails and the tolerance count is exceeded, `cntwmtd` invokes:

`notify_process` *notify_process_parms*

with the notification message on **stdin**.

Because the notification handler is a **shell command or script**, any alerting channel can be integrated without changing the Continwm codebase:

- **Email** (provided: `cntwmen` + `continwm_email_notify.sh`)
- **ServiceNow** — REST call from a shell script
- **Slack / Teams** — webhook POST
- **WhatsApp / SMS** — Twilio or similar API call
- **PagerDuty** — events API
- **Any internal ticketing system**

Notification message (stdin)

```
CONTINWM: test=kstest
CONTINWM: source=posipmon
CONTINWM: group=ecosys_...
CONTINWM: name=sessdur
CONTINWM: label=WEEKDAY.14-15
CONTINWM: result=FAIL
CONTINWM: pvalue=0.0000
CONTINWM: alpha=0.05
CONTINWM: case=15854
DIAGNOSTICS: ...
```

Alert fatigue prevention

`-notify-minimum-repeat-delay` (default 300 s) suppresses repeat alerts for the same metric.

Email Notification Out of the Box

V2 ships **cntwmen** — a purpose-built SMTP email sender — and a companion wrapper script `continwm_email_notify.sh`.

- Reads `CONTINWM:` / `DIAGNOSTICS:` lines from stdin and formats a complete MIME email with subject derived from the metric and result.
- Attaches report files from the test report directory (charts, logs, diagnostic output).
- Sends via SMTP (TLS/STARTTLS, port 465 or 587).
- Credentials loaded from `~/.continwm_email.conf` or environment variables — never stored in the test config.
- Multiple recipients supported.

Configure once, attach to any test:

```
notify-process-parms='ops@example.com sec@example.com'
```

~/.continwm_email.conf

```
CNTWMEN_FROM=continwm@example.com
CNTWMEN_SMTP_HOST=smtp.gmail.com
CNTWMEN_SMTP_PORT=587
CNTWMEN_SMTP_USER=user@gmail.com
CNTWMEN_SMTP_PASSWORD=app-pass
```

ServiceNow / Teams / Slack

Replace `continwm_email_notify.sh` with any script that reads stdin and posts to your chosen channel.

Named Baseline Policies

V2 supports **multiple named baseline configurations**, each defining a different historical reference window.

Each **baseline config** specifies:

- `name` / `title` — descriptive label
- `apply_to_metrics` — metric filter expression
- `span` — width of the historical window (days)
- `lag` — days before now to end the window
- `ttl_baseline` — days between automatic baseline refreshes

Baselines are computed by `cntwmbl` (baseline calculator) and stored in the `baselines` SQLite3 table via queries against **InfluxDB3** raw detail data.

`cntwmbl` runs automatically via cron (`cntwmbl.sh`) refreshing any baseline whose TTL has expired.

Example baseline policies

Name	Window
last-week	7d, lag 1d
last-month	30d, lag 1d
last-quarter	90d, lag 7d
weekday-am	14d, lag 0

Seasonal baselines

Different baselines for weekday vs. weekend, business hours vs. overnight — configured by label matching in `apply_to_metrics`.

Cases, Root Causes, and Suspect Ranges

When a test fails repeatedly (consecutive failures $\geq$ tolerance), a **case** is automatically opened and tracked in the `cases` table.

Case lifecycle:

- Opened on repeated failure; linked to the triggering test result and metric.
- Remains **open** (`O`) until explicitly closed by an operator.
- Operators **assign a root cause** from the `assignable_causes` catalogue.
- Case is **closed** (`C`) after remediation; can be reopened if the problem recurs.
- **Suspect ranges** record time windows where known external events (maintenance, deployments) may explain anomalies.

Assignable causes catalogue

A managed table of pre-defined root cause categories:
Unknown / No cause determined (built-in)
Planned maintenance window
Application deployment
Infrastructure change
External dependency failure
Configuration change
. . . (customer-defined)

Fully managed via the **web UI** (Configs $\rightarrow$ Causes tab) or command-line tools.

The Continwm Web UI

`continwm.html` — a self-contained single-page application (SPA) with no build step or framework dependency.

Login: sign in, register, or request password reset. Credentials are hashed client-side (SHA1+base64); the plaintext password never leaves the browser.

Four-level permission model:

- **0** — login denied
- **1** — read-only
- **2** — read/write (add/edit/delete)
- **9** — admin (session management)

Session via `SessionId` UUID header; 1-hour idle timeout.

Application tabs

Tab	Purpose
Headlines	Live event feed (default)
Configs / Tests	CRUD for test configs
Configs / Baselines	CRUD for baseline configs
Configs / Causes	CRUD for root causes
Results	Test results history
Cases	Case management
Notifications	Notification log

Headlines — Live Anomaly Feed

The **Headlines tab** is the default landing page after login. It provides a **live, push-updated feed** of test-result events.

How it works:

- When a notified (failing) test result is written, `cntwmtd` inserts a human-readable headline row.
- The browser loads the last 100 headlines on login, then starts a **long-poll loop** for new events.
- Each browser session tracks its own cursor (`last_headline_id`); no headline is shown twice.
- New headlines appear **at the top of the list** in real-time without any manual refresh.
- Test child processes can also **push custom headlines** via the `cntwmifc` pipe protocol.

Headline format

Metric title: *metric name*:
source=*src*, group=*grp*, IP=*ip*.
Baseline: *baseline title*.
Test: *test name*: *test title*,
label=*label*.
Result: p-value = p alpha = α.

Instant visibility

No polling interval delay.
Server-sent wake-up via
`SIGUSR2` from `cntwmdb`
on every new headline insert.

InfluxDB3 for Time-Series Depth

All raw metric detail rows are stored in **InfluxDB3** — a modern columnar time-series store with Apache Arrow / DataFusion SQL query support.

- Every metric interval is stored as a `detail` measurement row with nanosecond timestamps.
- Both **sample windows** and **baseline windows** are retrieved by time-range + detail-ID queries against InfluxDB3.
- Baseline computation spans days or weeks of history without loading data into SQLite3.
- InfluxDB3 Core's Parquet file scan limit is handled transparently by chunked day-by-day queries inside `cntwmdb`.

`detail` measurement fields

`detail_id`	tag
`source`	tag
`mgroup`	tag
`name`	tag
`metric_type`	field
`sequence`	field
`delta_usec`	field
`delta_count`	field
`delta_s_value`	field
`delta_sq_value`	field
`epoch`	field

SQLite3 for metadata

Configs, results, cases, users:
fast relational access without time-series overhead.

Monitor the Monitor

Continwm monitors itself.

The `continwm` daemon can be configured to broadcast its own operational metrics to a second monitoring instance (or back to itself):

- Tests executed per interval
- Tests passed / failed
- Message queue depth
- Metric ingestion rate

Configured via `-metric-send-host`, `-metric-send-port`, `-metric-send-group`, and `-metric-send-interval`.

The same hypothesis-testing machinery that monitors your application **alerts you when monitoring health degrades**.

Standalone Re-Analysis — `cntwmsa`

Re-run any stored test result against a different test algorithm without waiting for new data:

```
cntwmsa --results-id=1642982
   --test-program=cntwmrbs.R
```

Ideal for **post-incident investigation** and algorithm comparison.

Report files

Each test run can write charts, logs, and diagnostic files to a **per-test report directory**, viewable directly in the web UI.

Continwm V2 — What's New at a Glance

Test framework

- Pluggable test processes via `cntwmifc`
- C, R, and Python interfaces provided
- 4 statistical tests ready to use
- Debug mode for offline development

Configuration

- Per-test metric filter expressions
- Live config reload — zero downtime
- Multiple baseline policies
- Per-test α, sample size, tolerance

Alerting

- Per-test notification command
- Email SMTP built-in (`cntwmen`)
- Any channel via script hook
- Alert fatigue suppression

Visibility

- Headlines live event feed
- Long-poll, server-push updates
- Custom headlines from test scripts
- Test report file viewer in web UI

Case management

- Automatic case opening on repeated failure
- Root cause assignment catalogue
- Suspect ranges for known events
- Open / close / reopen workflow

Infrastructure

- InfluxDB3 deep history for baselines
- Standalone re-analysis tool
- Self-monitoring metric broadcast
- Full REST API for all operations

Thank You

Continwm Version 2.0

Code Magus Limited

www.codemagus.com

Questions?