CodeMagus

`continwm`: Continuous Monitoring System Architecture and User Guide

CML00129-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com

CodeMagus

August 28, 2026

Contents

1 Introduction **3**

2 Process Architecture **3**
2.1 Startup and Dependency Order 3
2.2 Key TCP Ports 4
2.3 Message Flow 4

3 Individual Processes **5**
3.1 `continwm` — Central Metric Daemon 5
3.1.1 Dynamic Configuration Reload 7
3.2 `cntwmdb` — Database Service 7
3.3 `cntwmtd` — Test Driver Worker 8
3.4 `cntwmtst` — Default Test Child Process 9
3.5 `cntwmks` — KS-Test Child Process (C Implementation) 9
3.6 `cntwmtc` — Test Configuration Manager 9
3.7 `cntwmbl` — Baseline Calculator 11
3.8 `cntwmht` — Legacy Hypothesis Tester 12
3.9 `cntwmsa` — Standalone Sample Analyser 12
3.10 `cntwmen` — Email Notification Dispatcher 13
3.11 `cntwmsrv` — User Management REST API Server 14

4 Web User Interface **18**
4.1 Login View 18
4.2 Application View 18

5 The cntwmifc Interface Library **19**

6 The cntwmks KS-Test Module **20**

7 cntwmrks: R Implementation of the KS Test **20**

8 The cntwmifc R Library **21**

9 cntwmtd to cntwmdb HTTP API **22**
9.1 Endpoint /continwm/server (used by `continwm`) 22
9.2 Endpoint /continwm/testdrive (used by `cntwmtd`) 22
9.3 Endpoint /continwm/testconfig (used by `cntwmtc`) 22
9.4 Endpoint /continwm/baseline (used by `cntwmbl`) 23
9.5 Endpoint /continwm/runtest (used by `cntwmht`) 23

10 The Binary Message Protocol **23**

11 The SQLite3 Database **24**
11.1 Table: sequences 24
11.2 Table: test_configs 24
11.3 Table: metric_details 24

11.4 Table: baselines . . . 25
11.5 Table: baseline_configs . . . 25
11.6 Table: baseline_detail_links . . . 26
11.7 Table: test_results . . . 26
11.8 Table: notifications . . . 27
11.9 Table: cases . . . 27
11.10 Table: users . . . 28
11.11 Table: notify_history . . . 28
11.12 Table: assignable_causes . . . 29
11.13 Table: suspect_ranges . . . 29
11.14 Table: headlines . . . 30

12 InfluxDB3 Database **31**

13 Cross-Database Relationships **31**
13.1 Data Lifecycle . . . 31
13.2 SQLite3 / InfluxDB3 Linkage . . . 32

14 apply_to_metrics Expression Evaluation **33**

15 Notification Process **33**
15.1 Notification Flow . . . 34
15.2 Email Notification . . . 34

16 Shell Scripts **35**
16.1 continwm.sh — System Startup and Control . . . 35
16.2 cntwmbl.sh — Baseline Computation Cron Wrapper . . . 36
16.3 continwm_email_notify.sh — Email Notification Handler . . . 36
16.4 cntwmtc.sh — Test Configuration Manager Wrapper . . . 36
16.5 config_test.sh — Test Configuration Integration Test . . . 36
16.6 fake_notify.sh — Test Notification Sink . . . 36
16.7 cntwmsrv_register.sh — Registration Notification Script . . . 37
16.8 cntwmrks_config.sh — R KS-Test Configuration Helper . . . 37
16.9 test_cntwmbl_config.sh — Baseline Configuration Integration Test . . . 37

17 Live Instance Directory and File Structure **37**
17.1 Log File Naming Convention . . . 38
17.2 Report Directory Naming Convention . . . 38

18 Dynamic Configuration Reload **39**
18.1 SIGUSR1 Signal Path . . . 39
18.2 Periodic Backstop . . . 39
18.3 What Gets Reloaded . . . 40
18.4 Operational Impact . . . 40

1 Introduction

`continwm` is a streaming, hypothesis-test-based continuous monitoring system developed by Code Magus Limited. It receives metric observations in real time, accumulates them into sample windows, and periodically dispatches statistical tests (such as the Kolmogorov-Smirnov or Welch's t-test) to determine whether a metric has deviated significantly from its baseline distribution. When the test indicates a statistically significant deviation the system raises a case and dispatches a notification.

The system is composed of several co-operating processes:

- `continwm` — the central metric-collection and test-dispatch daemon.
- `cntwmdb` — the database service, exclusive gatekeeper to `SQLite3` and `InfluxDB3`.
- `cntwmtd` — worker processes that receive test jobs from `continwm` and dispatch them to hypothesis-test child processes.
- `cntwmtc` — a command-line tool for managing test configurations.
- `cntwmbl` — a command-line tool for computing and managing baselines.
- `cntwmks` — a C-language two-sample KS-test child process.
- `cntwmrks` — an R-language two-sample KS-test script.
- `cntwmtst` — the default hypothesis-test child process.
- `cntwmht` — a legacy standalone hypothesis-test tool.
- `cntwmsa` — a standalone sample analyser.
- `cntwmsrv` — a user management REST API server (web UI backend).
- `cntwmen` — an email notification dispatcher.

The system stores two types of persistent data: summary statistics in `SQLite3` (for fast operational queries) and raw metric detail rows in `InfluxDB3` (for efficient time-series retrieval).

All inter-process communication uses either TCP sockets (binary protocol between `continwm` and `cntwmtd`, or plain pipes between `cntwmtd` and test child processes) or an HTTP/1.1 REST API between clients and `cntwmdb`.

2 Process Architecture

2.1 Startup and Dependency Order

The `continwm.sh` startup script starts processes in the following dependency order:

1. `cntwmdb` — opens the `SQLite3` database and binds its HTTP port (default 60999).
2. `cntwmsrv` — user management REST API (depends on `cntwmdb`), binds port 7070.
3. `continwm` — connects to `cntwmdb` on startup and binds its metric (60005) and test-driver (60095) ports.
4. `cntwmtd` workers (one or more) — each connects to `continwm` on its test-driver port.

On shutdown the order is reversed so that `cntwmdb` outlives its clients.

2.2 Key TCP Ports

Port	Owner	Purpose
60005	`continwm`	Accepts metric data from broadcasters
60095	`continwm`	Accepts connections from `cntwmtd` instances
60999	`cntwmdb`	HTTP/1.1 REST API for all database operations
7070	`cntwmsrv`	HTTP/1.1 REST API for user management (web UI)

2.3 Message Flow

1. A metric probe (e.g. `cmlmdish`) sends UDP metric packets to `continwm` on port 60005.
2. `continwm` accumulates metric detail rows into per-metric sample windows. When enough detail rows have been collected (controlled by `test_size` in `test_configs`), it serialises a `hypothesis_test_t` binary structure and sends a `MESSAGE_HTEST` to a connected `cntwmtd` worker over TCP.
3. `cntwmtd` queries `cntwmdb` for baseline and sample data, then `forks`/`execs` a test child process (e.g. `cntwmks` or `cntwmrks`). The child communicates via a pair of anonymous pipes using the `cntwmifc` protocol.
4. The test child posts a p-value result back through the pipe.
5. `cntwmtd` classifies the result as PASS, FAIL, or INCONCLUSIVE and POSTs the result to `cntwmdb`.

6. If the result is FAIL and the consecutive-failure count reaches the `tolerance` threshold, `cntwmdb` marks the result `notified='Y'` and `cntwmtd` opens a case and fires the notification process.

3 Individual Processes

3.1 `continwm` — Central Metric Daemon

Source: `~/software/continwm/continwm.c`

Role: Receives UDP metric broadcasts, maintains per-metric sample windows, and dispatches hypothesis tests to `cntwmtd` workers.

Key responsibilities:

- Listens on UDP port 60005 for `metric_t` datagrams.
- Forwards each received metric to `cntwmdb` for storage in `InfluxDB3`.
- Accumulates metric samples in a `metricbatch_t` structure (start/end `metricboundary_t`) tracking count, sum, sum-of-squares, detail IDs, and wall-clock times.
- When `test_size` samples have accumulated for a metric (the minimum across all matching test configurations), serialises a `hypothesis_test_t` and sends a `MESSAGE_HTEST` to a connected `cntwmtd` over TCP port 60095.
- Listens for `MESSAGE_COMPLETE` and `MESSAGE_ERROR` responses from `cntwmtd`.
- Optionally emits self-monitoring metrics to a separate metric collector.

Command-line options:

Option	Default	Description
--metric-host	0.0.0.0	Interface to bind for metric reception
--metric-port (-p)	60005	TCP port for incoming metric broadcasts
--testdriver-host	0.0.0.0	Interface to bind for cntwmtd connections
--testdriver-port (-P)	60095	TCP port for cntwmtd connections
--source (-u)	(all)	Filter incoming metrics by source name
--group (-g)	(all)	Filter incoming metrics by group name
--metric-name (-m)	(all)	Filter incoming metrics by metric name
--max-time-out (-i)	5	pselect() maximum timeout (seconds)
--log-name-prefix (-L)	continwm	Prefix for rotating log files
--report-root-dir	(none)	Root directory for per-test report subdirectories
--notify-minimum-repeat-delay	300	Min. seconds between repeat notifications for same metric
--metric-send-host	(none)	Self-monitoring metric destination host (blank = disable)
--metric-send-port	60015	Self-monitoring metric destination port
--metric-send-group	continwm	Metric group for self-monitoring
--metric-send-interval	5	Self-monitoring interval (seconds)
--dbcommit-count (-C)	1000	Detail rows per commit to cntwmdb
--cntwmdb-host	127.0.0.1	Host of cntwmdb service
--cntwmdb-port	60999	Port of cntwmdb service
-v, --verbose		Enable verbose diagnostic output (repeatable)

The per-test parameters test_size (minimum detail rows before a test fires), tolerance (consecutive failures before notification), alpha (significance level), span (baseline window width), and lag are all stored per test configuration in test_configs and are **not** global command-line options.

3.1.1 Dynamic Configuration Reload

When a test_configs row is changed via cntwmtc, continwm must pick up the change without restarting. This is implemented via SIGUSR1:

1. On startup continwm POSTs a register_continwm_pid action to cntwmdb, registering its PID.
2. When cntwmdb processes any test_configs mutation (insert, delete, or update), it calls notify_continwm() which sends SIGUSR1 to the registered PID.
3. The SIGUSR1 handler sets a volatile sig_atomic_t flag g_reload_testconfigs.
4. At the top of every main-loop iteration, continwm checks the flag and, if set, calls reload_test_configs() which clears and reloads both the testconfigs and metrics_to_test hash tables from cntwmdb.

A periodic backstop (every 300 seconds by default) triggers an additional reload in case any signal is missed. In-progress metric accumulation is not disturbed: only the compiled test-configuration and metric-to-test-set caches are rebuilt.

3.2 cntwmdb — Database Service

Source: ~/software/continwm/cntwmdb.c

Role: Single-process HTTP/1.1 REST server — the exclusive gatekeeper to both SQLite3 and InfluxDB3.

Key responsibilities:

- Holds the only open SQLite3 connection and InfluxDB3 write/query connection. Because it is single-threaded, all database writes are naturally serialised without locking.
- On startup, calls ensure_tables() which creates all required SQLite3 tables and applies any pending schema migrations (e.g. adding new columns) automatically.
- Accepts JSON-body POST requests from all other processes.
- Manages all sequences, test configurations, baselines, test results, notifications, and cases.

Key command-line options:

Option	Default	Description
--host (-H)	0.0.0.0	Interface to bind for HTTP connections
--port (-p)	60999	HTTP listen port
--database (-D)	(required)	Path to SQLite3 database file
--dbcommit-count (-C)	1000	Rows accumulated before a SQLite3 COMMIT
--log-name-prefix (-L)	cntwmdb	Prefix for rotating log files
--influxdb3-host	localhost	InfluxDB3 server hostname
--influxdb3-port	8086	InfluxDB3 server port
--influxdb3-database	continwm	InfluxDB3 database/bucket name
--influxdb3-token	(required)	InfluxDB3 API bearer token
--influxdb3-tls	off	Enable TLS for InfluxDB3 connections
--influxdb3-query-time-range-limit	86400	Max InfluxDB3 query window (seconds)
-v, --verbose		Verbose diagnostic output (repeatable)

3.3 cntwmtd — Test Driver Worker

Source: ~/software/continwm/cntwmtd.c

Role: Worker process that accepts hypothesis-test jobs from continwm, retrieves data from cntwmdb, and dispatches test child processes.

Processing loop:

1. Connects to continwm on the test-driver TCP port.
2. Receives a MESSAGE_HTEST binary message.
3. Queries cntwmdb for the relevant test configurations.
4. For each matching configuration, queries cntwmdb for baseline data and sample data (from InfluxDB3).
5. fork/execs the test_process binary (e.g. cntwmks or Rscript cntwmrks.R).
6. Communicates with the child via the cntwmifc pipe protocol.
7. Receives the p-value, classifies P/F/I, and POSTs to cntwmdb.
8. If the result is notified='Y': opens a case and fires notify_process via popen().

9. Sends MESSAGE_COMPLETE back to continwm.

Key command-line options:

Option	Default	Description
--continwm-host (-h)	127.0.0.1	Host address of continwm daemon to connect to
--continwm-port (-p)	60095	Test-driver port of continwm daemon
--log-name-prefix (-L)	cntwmtd	Prefix for rotating log files
--test-log-dir	(none)	Directory for child hypothesis-test log files
--cntwmdb-host	127.0.0.1	cntwmdb host
--cntwmdb-port	60999	cntwmdb port
-v, --verbose		Verbose diagnostic output (repeatable)

3.4 cntwmtst — Default Test Child Process

Source: ~/software/continwm/cntwmtst.c

Role: The default statistical test child. Implements Welch's two-sample t-test comparing interval-mean values from the sample window against those from the baseline.

Protocol: Uses the cntwmifc C library (see Section 5) to communicate with cntwmtd over a pair of anonymous pipes.

3.5 cntwmks — KS-Test Child Process (C Implementation)

Source: ~/software/cntwmks/cntwmks.c

Role: A two-sample Kolmogorov-Smirnov test child process written in C. It receives baseline and sample data via the cntwmifc pipe protocol, computes per-interval mean values (delta_s_value / delta_count), applies the KS test, and posts the resulting p-value.

This is the production KS-test child for maximum performance. For an R-language equivalent see cntwmrks (Section 7).

3.6 cntwmtc — Test Configuration Manager

Source: ~/software/continwm/cntwmtc.c

Role: Command-line tool for CRUD operations on the test_configs table via the cntwmdb HTTP API.

Command-line options:

Option	Description
--list-configs	Display all test_configs rows
--add-config	Insert a new test_config row
--delete-config	Delete a row (requires --test-config-id)
--edit-config	Update one field (requires --test-config-id)
--test-config-id=<id>	Row selector for delete/edit
--name=<string>	Unique test name
--title=<string>	Human-readable title
--description=<string>	Free-text description
--apply-to-metrics=<expr>	Expeval expression for metric selection (see Section 14)
--test-process=<path>	Test child executable (e.g. cntwmks)
--test-process-parms=<parms>	Additional CLI arguments for test child
--notify-process=<path>	Notification command
--notify-process-parms=<parms>	Additional arguments for notification command
--alpha=<float>	Significance level (e.g. 0.05)
--test-size=<integer>	Min detail rows before triggering a test (> 0)
--tolerance=<integer>	Consecutive failures before a case is opened (> 0)
--cntwmdb-host=<host>	cntwmdb host (default 127.0.0.1)
--cntwmdb-port=<port>	cntwmdb port (default 60999)

Example usage:

```
# List all test configurations
cntwmtc --list-configs

# Add a KS-test configuration
cntwmtc --add-config \
   --name=kstest \
   --title='KS Test' \
   --apply-to-metrics='source like ".*"' \
   --test-process=cntwmks \
   --test-process-parms='' \
   --alpha=0.05 \
   --test-size=100 \
   --tolerance=3

# Edit the test_size for an existing configuration
cntwmtc --edit-config --test-config-id=1 --test-size=200
```

```
# Delete a test configuration
cntwmtc --delete-config --test-config-id=5
```

The `--list-configs` output shows one labeled block per row:

```
--- test_config_id: 1 ---
  name                  : default
  title                 : Default Hypothesis Test
  apply_to_metrics      : source like ".*"
  test_process          : cntwmht
  alpha                 : 0.05
  test_size             : 100
  tolerance             : 3
  last_changed_by       : config_test.sh
  last_changed_date     : 2026-01-15 08:30:00

1 row(s).
```

3.7 cntwmbl — Baseline Calculator

Source: `~/software/continwm/cntwmbl.c`

Role: Computes and stores baseline summary statistics from `InfluxDB3` historical data into the `SQLite3` `baselines` table. Also manages the `baseline_configs` table which defines named policies (span, lag, TTL) controlling baseline computation.

Action flags (at most one):

Option	Description
`--list-configs`	Print all baseline_configs rows
`--add-config`	Insert a new baseline_config
`--delete-config`	Delete a baseline_config and its baselines
`--edit-config`	Update one field of a baseline_config
`--baseline-config-id=<id>`	Row selector for delete/edit
`--name=<string>`	Unique config name
`--title=<string>`	Human-readable title
`--description=<string>`	Free-text description
`--apply-to-metrics=<expr>`	Expeval expression for metric selection
`--span=<days>`	Baseline window width in days
`--lag=<days>`	Days of lag before the window
`--ttl-baseline=<days>`	Days until a baseline must be recomputed

When invoked with *no* action flag, `cntwmbl` iterates all `baseline_configs` rows, checks whether each baseline's TTL has expired, and recomputes expired baselines.

This is the normal cron usage via cntwmbl.sh.

3.8 cntwmht — Legacy Hypothesis Tester

Source: ~/software/continwm/cntwmht.c

Role: A legacy standalone hypothesis-test utility, spawned directly by the older continwm test path via POST /continwm/runtest. Performs a two-sample Z-test using pre-aggregated baseline statistics from SQLite3.

Command-line options:

Option	Default	Description
--readfd=<fd> (-r)	(required)	Pipe FD to read metric batch from
--writefd=<fd> (-w)	(required)	Pipe FD to write result to
--instance-number=<n> (-i)	1	continwm instance number
--log-file-spec=<spec> (-l)	(none)	Log file path specification
--cntwmdb-host	127.0.0.1	cntwmdb service host
--cntwmdb-port	60999	cntwmdb service port
-v, --verbose		Verbose diagnostic output

3.9 cntwmsa — Standalone Sample Analyser

Source: ~/software/continwm/cntwmsa.c

Role: Re-runs a hypothesis test from a stored test_results row without waiting for new metric data. Useful for re-analysis and debugging.

Key responsibilities:

- Reads a test_results row by --results-id to obtain baseline_id and the detail-ID range.
- Fetches baseline detail from SQLite3 baseline_detail_links and InfluxDB3.
- Fetches sample detail from InfluxDB3 using the stored detail-ID range.
- Forks the configured test child and drives the full cntwmifc pipe protocol with it.
- Writes the new result back to cntwmdb.

Command-line options:

Option	Default	Description
--results-id=<id> (-r)	(required)	test_result_id of the row to re-run
--test-program=<path> (-t)	(required)	Test child executable path
--test-parameters=<p> (-T)	(none)	Extra CLI arguments for test child
--log-file-spec=<spec> (-l)	(none)	Log file path specification
--cntwmdb-host	127.0.0.1	cntwmdb host
--cntwmdb-port	60999	cntwmdb port
-v, --verbose		Verbose diagnostic output

3.10 cntwmen — Email Notification Dispatcher

Source: ~/software/continwm/cntwmen.c

Role: Reads notification content from stdin, constructs a MIME email with report-directory attachments, and delivers it via SMTP using libcurl and GMime-3.0.

Usage:

```
cntwmen [options] recipient@addr ...
```

Command-line options and environment variable equivalents:

Option	Environment variable	Description
--from=ADDR	CNTWMEN_FROM	Sender address
--smtp-host=HOST	CNTWMEN_SMTP_HOST	SMTP server
--smtp-port=PORT	CNTWMEN_SMTP_PORT	SMTP port (default 587)
--smtp-user=USER	CNTWMEN_SMTP_USER	SMTP username
--smtp-password=P	CNTWMEN_SMTP_PASSWORD	SMTP password
--smtp-ssl	CNTWMEN_SMTP_SSL=1	Use implicit TLS (SMTPS, port 465). Without this flag, plain cleartext SMTP is used with no SSL/TLS or STARTTLS negotiation.
--subject=SUBJ		Override subject line
-v, --verbose		Verbose output

All options may also be supplied via the listed environment variables; credentials stored in $HOME/.continwm_email.conf and sourced by continwm_email_notify.sh will not appear on the process command line.

SSL/TLS modes: When --smtp-ssl (or CNTWMEN_SMTP_SSL=1) is specified,

cntwmen connects using implicit TLS (the smtps:// scheme, typically port 465). When --smtp-ssl is *not* specified, the connection uses plain cleartext SMTP (the smtp:// scheme) with no SSL/TLS or STARTTLS negotiation. This legacy mode is appropriate for internal SMTP relays, local MTA delivery, or environments where transport encryption is handled at another layer.

Credentials: SMTP credentials (--smtp-user and --smtp-password) are *optional* when using plain (legacy) SMTP. The plain SMTP protocol does not mandate authentication; local MTAs and internal relays typically accept unauthenticated connections. When no credentials are supplied in plain SMTP mode, cntwmen proceeds without issuing an AUTH command. Credentials *are* expected when using SMTPS (--smtp-ssl); the server will reject the connection if it requires authentication and none is provided.

3.11 cntwmsrv — User Management REST API Server

Source: ~/software/continwm/cntwmsrv.c

Role: An HTTP/1.1 REST API server providing user authentication, registration, and session management. Acts as the backend for the web UI (continwm.html) and any other HTTP client requiring authenticated access to continwm data.

Key responsibilities:

- Accepts HTTP/1.1 requests on port 7070 (default).
- Manages user accounts in the users SQLite3 table via cntwmdb.
- Authenticates requests using a SessionId UUID header; sessions expire after 3600 seconds of inactivity.
- Enforces a four-level permission model: 0 = login denied, 1 = read-only, 2 = read/write, 9 = admin.
- Invokes a configurable --register-script for new-user registration and password-reset notifications. **This option is mandatory**; cntwmsrv will not start without it.

Command-line options:

Option	Default	Description
`--host=<addr>`	0.0.0.0	Bind address
`--port=<port>`	7070	HTTP port
`--cntwmdb-host=<host>`	127.0.0.1	`cntwmdb` host
`--cntwmdb-port=<port>`	60999	`cntwmdb` port
`--register-script=<path>`	(required)	Register script
`--default-user-permissions=<int>`	1	Default permissions
`--log-name-prefix=<prefix>`	(none)	Log file prefix
`-v, --verbose`		Verbose output

Permission model:

Level	Access
0	Login rejected with "Insufficient permissions to use this system"
1	Read-only — all `list` endpoints
2	Read/write — includes `add`, `edit`, `delete`
9	Admin — includes `/continwm/who`

Any authenticated user may logoff regardless of permission level.

Authentication: All authenticated endpoints require the `SessionId` HTTP header set to the UUID string returned by `/continwm/login`.

Password scheme:

```
secret_value = base64(sha1(lower(email) + password))
```

The secret is computed on the client; the plaintext password is never transmitted.

Headlines feature: `cntwmsrv` maintains a per-session cursor (`last_headline_id`) tracking the last headline seen by each client.

- `GET /continwm/headlines/count/<n>` — returns the n most recent headlines ordered by `headline_id` DESC and advances the session cursor so that `/headline/next` polls for rows strictly after these.
- `GET /continwm/headlines/age/<secs>` — returns headlines whose `timestamp` is within the last `secs` seconds (does not update the session cursor).
- `GET /continwm/headline/next` — long-poll endpoint. If a headline with `headline_id > last_headline_id` already exists it is returned immediately; otherwise the request blocks for up to 5 seconds watching a `g_headline_epoch` counter incremented by the `SIGUSR2` handler. `cntwmdb` sends `SIGUSR2` to `cntwmsrv` (via `notify_cntwmsrv()`) whenever it commits a new headline row. If no headline arrives within 5 seconds the endpoint returns {`"Result":"Success","head`

Headlines are inserted by `cntwmtd` (action `insert_headline` on `POST /continwm/headlines`) whenever a notified (`notified='Y'`) test result is written. Test child processes may

also push custom headlines via a `headline` message in the `cntwmifc` pipe protocol. The `headlines` `SQLite3` table is described in Section 11.14.

REST endpoints:

Perm	Method	Path	Description
None	POST	`/continwm/register/newuser`	Register a new user
None	POST	`/continwm/register/newpassword`	Request password reset
None	POST	`/continwm/login`	Authenticate; returns `SessionId`
None	GET	`/continwm/version`	Return version string
≥ 1	GET	`/continwm/logoff`	Invalidate session
≥ 1	GET	`/continwm/baseline_configs/list`	List baseline configs
≥ 1	GET	`/continwm/baseline_configs/fetch/<id>`	Fetch one baseline config
≥ 1	GET	`/continwm/test_configs/list`	List test configs
≥ 1	GET	`/continwm/test_results/list`	List test results
≥ 1	GET	`/continwm/test_results/fetch/<id>`	Fetch one test result
≥ 1	POST	`/continwm/test_results/files/list`	List report files
≥ 1	POST	`/continwm/test_results/file/download`	Download report file
≥ 1	GET	`/continwm/notifications/list`	List notifications
≥ 1	GET	`/continwm/cases/list`	List all cases
≥ 1	GET	`/continwm/cases/list/open`	List open cases
≥ 1	GET	`/continwm/cases/list/closed`	List closed cases
≥ 1	GET	`/continwm/assignable_causes/list`	List assignable causes
≥ 1	GET	`/continwm/suspect_ranges/list`	List suspect ranges
≥ 1	GET	`/continwm/headlines/count/<n>`	Last n headlines (advances cursor)
≥ 1	GET	`/continwm/headlines/age/<secs>`	Headlines within last n seconds
≥ 1	GET	`/continwm/headline/next`	Long-poll for next headline
≥ 2	POST	`/continwm/baseline_configs/add`	Add baseline config
≥ 2	POST	`/continwm/baseline_configs/edit`	Edit config field
≥ 2	POST	`/continwm/baseline_configs/delete`	Delete baseline config
≥ 2	POST	`/continwm/test_configs/add`	Add test config

4 Web User Interface

File: `~/software/continwm/continwm.html`

The `continwm` web UI is a self-contained single-page application (SPA) that communicates with the `cntwmsrv` REST API. Styled in the Continwm magenta brand theme (`#E323E3`). No build step is required; the file is served as a static HTML page.

4.1 Login View

The login view presents three card tabs:

Tab	Function
Sign In	Authenticate with email and password
Register	Create a new user account
Forgot Password	Request a password-reset notification

The target API server URL is configurable from the login page and is persisted in `localStorage` under the key `continwm_server`.

4.2 Application View

After successful login, the application view is shown. The default landing tab is **Headlines**. The top-level tabs are:

Tab	Sub-tab	Content
Headlines	—	Live feed of test-result headlines
Configs	Tests	Full CRUD for `test_configs` rows
Configs	Baselines	Full CRUD for `baseline_configs` rows
Configs	Causes	Full CRUD for `assignable_causes` rows
Results	—	Read-only table of `test_results`
Cases	All / Open / Closed	Expandable cases list; cause assignment; close/reopen
Notifications	—	Read-only notifications list

Add/Edit modals support `test_configs`, `baseline_configs`, and `assignable_causes` rows. The Edit modal issues one API call per changed field, matching the server's per-column update endpoint contract.

The **Cases** panel groups rows by `case_number` and expands on demand to show the associated test results, attached report files, baseline configs used, and metric identity. A context menu (permission ≥ 2) provides:

Action	Description
Edit	Update individual case fields one at a time
Cause	Assign an `assignable_cause` to the case
Close/Reopen	Toggle `status` between `O` (open) and `C` (closed)

The **Causes** sub-tab manages the `assignable_causes` table — the catalogue of pre-defined root cause categories available for assignment to open cases. The built-in cause `assignable_cause_id=0` ("Unknown / No cause determined") is always present and cannot be edited or deleted.

5 The cntwmifc Interface Library

Source: `~/software/cntwmifc/`

The `cntwmifc` library provides the communication protocol between a `cntwmtd` worker and a hypothesis-test child process. Communication uses a pair of anonymous pipes (inherited from `fork`/`exec`) identified by the environment variables `CNTWMIFC_READ_FD` and `CNTWMIFC_WRITE_FD`.

C API: The header `cntwmifc.h` and library `libcntwmifc.a` provide:

- `cntwmifc_startup()` — initialise the protocol
- `cntwmifc_base_detail()` — retrieve baseline detail rows
- `cntwmifc_samp_detail()` — retrieve sample detail rows
- `cntwmifc_post_result(p_value)` — post p-value result
- `cntwmifc_post_nontest(reason)` — report no test possible
- `cntwmifc_closeup()` — graceful shutdown

Each call reads/writes a typed binary message over the pipe pair. The `cntwmifc_detail_row_t` structure carries:

Field	Type	Description
epoch	uint32_t	Unix epoch seconds
metric_type	uint32_t	Metric type code
sequence	uint32_t	Monotonic sequence number
detail_id	uint32_t	InfluxDB3 detail row identifier
delta_usec	uint64_t	Interval duration (microseconds)
delta_count	double	Events in the interval
delta_s_value	double	Sum of metric values
delta_sq_value	double	Sum of squared metric values

Debug mode: When CNTWMIFC_READ_FD/CNTWMIFC_WRITE_FD are not set, the library enters debug mode, enabling stand-alone testing of test child scripts without a running continwm system.

For the R-language wrapper see Section 8.

6 The cntwmks KS-Test Module

Source: ~/software/cntwmks/cntwmks.c

cntwmks is a C implementation of the two-sample Kolmogorov-Smirnov test for use as a cntwmtd test child process. It is configured in test_configs with test_process=cntwmks.

Algorithm:

1. Retrieves baseline and sample detail rows via cntwmifc.
2. Computes per-interval mean values: $\bar{x}_i = \texttt{delta\_s\_value}_i / \texttt{delta\_count}_i$.
3. Applies the two-sample KS test to the vectors of baseline and sample means.
4. Posts the resulting p-value.

7 cntwmrks: R Implementation of the KS Test

Source: ~/software/cntwmrks/cntwmrks.R

cntwmrks is an R-language implementation of the two-sample Kolmogorov-Smirnov test, equivalent in function to cntwmks but written as an R script. It is configured in test_configs as:

```
test_process       = /usr/bin/Rscript
test_process_parms = /path/to/cntwmrks.R
```

Algorithm:

1. Calls `cntwmifc_samp_detail()` and `cntwmifc_base_detail()` to retrieve detail rows.
2. Extracts `delta_s_value / delta_count` for each row.
3. Runs R's built-in `ks.test()` on the two mean vectors.
4. Posts the p-value via `cntwmifc_post_result()`.

Debug mode: When `CNTWMIFC_READ_FD` and `CNTWMIFC_WRITE_FD` are not set, the script uses `cntwmifc_debug_set_test_config()` and `cntwmifc_debug_make_base/samp` to inject synthetic data, enabling stand-alone interactive testing.

The companion script `cntwmrks_config.sh` registers the `cntwmrks` test configuration in the `test_configs` table.

8 The cntwmifc R Library

Source: `~/software/cntwmifc/R/cntwmifc`

The `cntwmifc` R package wraps `libcntwmifc.a` so that R test scripts can participate in the continwm pipe protocol without writing any C code. The package is documented separately in [**?**].

Installation:

```
R CMD INSTALL ~/software/cntwmifc/R/cntwmifc
```

Exported functions:

Function	Description
`cntwmifc_startup(name, flags)`	Connect to driver; returns opaque token
`cntwmifc_closeup(token)`	Send close message, free resources
`cntwmifc_base_detail(token)`	Data frame of baseline detail rows
`cntwmifc_samp_detail(token)`	Data frame of sample detail rows
`cntwmifc_post_result(token, pvalue)`	Send p-value to driver
`cntwmifc_post_nontest(token, reason)`	Report no test possible
`cntwmifc_testnum(token)`	Current test sequence number
`cntwmifc_instance(token)`	`continwm` instance number
`cntwmifc_is_debug_mode(token)`	Returns 1 in debug mode

Detail data frame columns:

Column	Type	Description
epoch	integer	Unix epoch seconds
metric_type	integer	Metric type code
sequence	integer	Monotonic sequence number
delta_usec	integer	Interval duration (μs)
delta_count	double	Events per interval
delta_s_value	double	Sum of metric values
delta_sq_value	double	Sum of squared metric values
detail_id	integer	InfluxDB3 row identifier
row_number	integer	1-based position in array

Startup flags:

Flag	Value	Effect
CNTWMIFC_VERBOSE	1	Verbose logging to stderr
CNTWMIFC_TRACE	2	Trace logging and hex dumps

9 cntwmtd to cntwmdb HTTP API

All processes communicate with cntwmdb via HTTP/1.1 POST requests to http://host:60999/cor with a JSON body. The response is always a JSON object containing a result field.

9.1 Endpoint /continwm/server (used by continwm)

Handles metric detail storage, test-config loading, and metric-detail lookups.

9.2 Endpoint /continwm/testdrive (used by cntwmtd)

Handles baseline and sample detail retrieval, result writing, case management, and notification history.

9.3 Endpoint /continwm/testconfig (used by cntwmtc)

Live management of the test_configs table. All writes are committed before the response is returned.

Action	Key Request Fields	Response
`get_test_configs`	(none)	test_configs array
`insert_test_config`	name, title, description, apply_to_metrics, test_process, test_process_parms, notify_process, notify_process_parms, alpha, test_size, tolerance	test_config_id
`delete_test_config`	test_config_id	success/error
`update_test_config_column`	test_config_id, column, value_str or value_int or value_dbl	success/error

Updateable columns for `update_test_config_column`: `name`, `title`, `description`, `apply_to_metrics`, `test_process`, `test_process_parms`, `notify_process`, `notify_process_parms` (via `value_str`); `test_size`, `tolerance` (via `value_int`); `alpha` (via `value_dbl`).

9.4 Endpoint /continwm/baseline (used by `cntwmbl`)

Handles baseline computation and baseline-config CRUD.

9.5 Endpoint /continwm/runtest (used by `cntwmht`)

Handles direct hypothesis test invocation from the legacy tool.

10 The Binary Message Protocol

`continwm` and `cntwmtd` communicate over TCP using a lightweight binary framing protocol. Each message starts with a fixed-size header:

Field	Type	Description
`magic`	uint32_t	Magic number for sanity check
`type`	uint32_t	Message type (`MESSAGE_HTEST`, `MESSAGE_COMPLETE`, `MESSAGE_ERROR`)
`length`	uint32_t	Payload length in bytes

The `MESSAGE_HTEST` payload is a serialised `hypothesis_test_t` structure containing the metric identification, baseline config ID, sample boundary detail IDs, counts, sums, wall-clock timestamps, and the significance level.

11 The SQLite3 Database

The SQLite3 database at $CONTINWM_HOME/database/continwm.db holds all operational state except raw metric detail rows (which live in InfluxDB3). The schema is maintained by cntwmdb's ensure_tables() function, which creates tables on first run and applies migrations automatically on startup.

11.1 Table: sequences

Global sequence counter table.

Column	Type	Description
seq_name	TEXT (PK)	Sequence name (e.g. cntwmseq)
seq_value	INTEGER	Current sequence value

11.2 Table: test_configs

Test configuration: which test process runs for which metrics and what criteria apply.

Column	Type	Constraints	Description
test_config_id	INTEGER	PK	Surrogate primary key
name	TEXT	UNIQUE	Config name
title	TEXT		Human-readable title
description	TEXT		Full description
apply_to_metrics	TEXT		Expeval expression
test_process	TEXT		Executable (e.g. cntwmks)
test_process_parms	TEXT		CLI arguments
notify_process	TEXT		Notification command
notify_process_parms	TEXT		Notification arguments
alpha	REAL		Significance level
test_size	INTEGER	DEFAULT 100	Min detail rows
tolerance	INTEGER		Consecutive failures
last_changed_by	TEXT		Audit: changed by
last_changed_date	TEXT		Audit: changed date

11.3 Table: metric_details

Display metadata for each known metric.

Column	Type	Constraints	Description
metric_detail_id	INTEGER	PK	Surrogate key
source	TEXT	UQ(source,mgroup,name)	Metric source
mgroup	TEXT		Metric group
name	TEXT		Metric name
source_ip	TEXT		Source IP address
title	TEXT		Display title

11.4 Table: baselines

One row per baseline computation run for a given metric and baseline configuration. Metric identity is stored as a foreign-key reference to metric_details rather than repeating the source/mgroup/name columns.

Column	Type	Constraints	Description
baseline_id	INTEGER	PK	Surrogate primary key
baseline_config_id	INTEGER	DEFAULT 0	FK → baseline_configs
metric_detail_id	INTEGER		FK → metric_details
label	TEXT		Time-window label
timestamp	TEXT		When baseline was computed
sum_count	REAL		Event count
sum_values	REAL		Sum of delta_s_value
sum_sq_values	REAL		Sum of $\texttt{delta\_s\_value}^2$
sum_usec	INTEGER		Sum of delta_usec
last_changed_by	TEXT		Audit column
last_changed_date	TEXT		Audit column

Indexes: baselines_ix1 on (metric_detail_id, label);baselines_ix2 on (baseline_config_id). Source, group and name are recovered by joining metric_details on metric_detail_id.

11.5 Table: baseline_configs

Named policies that control baseline computation. Each row defines the time window (span, lag), the minimum recomputation interval (ttl_baseline), and an apply_to_metrics expression that selects which metrics the policy governs.

Column	Type	Constraints	Description
baseline_config_id	INTEGER	PK	Surrogate primary key
name	TEXT	UNIQUE(1)	Short identifier
title	TEXT	UNIQUE(1)	Human-readable title
description	TEXT		Free-text description
apply_to_metrics	TEXT		Expeval expression for metric filter
span	INTEGER		Baseline window width (seconds)
lag	INTEGER		Lag before current time (seconds)
last_baseline	INTEGER	DEFAULT 0	Unix epoch of last run
ttl_baseline	INTEGER		Min seconds between auto-runs
last_changed_by	TEXT		Audit column
last_changed_date	TEXT		Audit column

Unique index baseline_configs_ux1 on (name, title).

11.6 Table: baseline_detail_links

Join table linking each baselines row to the individual detail_id values in InfluxDB3 that contributed to it. get_baseline_detail uses these IDs to build an IN-list query against InfluxDB3 rather than re-scanning the entire time window.

Column	Type	Constraints	Description
baseline_id	INTEGER	PK (part), FK	FK → baselines.baseline_id
detail_id	INTEGER	PK (part)	InfluxDB3 detail_id value
last_changed_by	TEXT		Audit column
last_changed_date	TEXT		Audit column

Primary key is (baseline_id, detail_id).

11.7 Table: test_results

One row per hypothesis test outcome.

Column	Type	Description
test_result_id	INTEGER	Surrogate primary key
baseline_id	INTEGER	FK → baselines
baseline_config_id	INTEGER	FK → baseline_configs
metric_detail_id	INTEGER	FK → metric_details
start_detail_id	INTEGER	Start of sample in InfluxDB3
end_detail_id	INTEGER	End of sample in InfluxDB3
label	TEXT	Metric label
test_name	TEXT	Test configuration name
timestamp	TEXT	Test execution time
consecutive_errors	INTEGER	Failure run length
testnumber	INTEGER	Global test sequence number
result	TEXT	P / F / I / N
notified	TEXT	Y / N
pvalue	REAL	Test p-value
report_path	TEXT	Path to report directory

11.8 Table: notifications

Tracks the running consecutive-failure counter for each metric/label/test combination to implement the tolerance gate. A row exists only while failures are accumulating; it is deleted on both notification-fire and on PASS.

Column	Type	Constraints	Description
notification_id	INTEGER	PK	Surrogate primary key
metric_detail_id	INTEGER	FK	FK → metric_details
label	TEXT		Time-window label
test_name	TEXT		Test config name
timestamp	TEXT	DEFAULT now	Last update time
consecutive_errors	INTEGER		Running failure count
last_changed_by	TEXT		Audit column
last_changed_date	TEXT		Audit column

Unique index notifications_ux1 on (metric_detail_id, label, test_name).

11.9 Table: cases

Open anomaly cases. A case is opened when notified='Y' and closed when the metric returns to PASS. Metric identity is stored as a foreign key into metric_details.

Column	Type	Constraints	Description
case_id	INTEGER	PK	Surrogate primary key
test_result_id	INTEGER		FK → test_results
case_number	INTEGER		From cntwmcase sequence
status	TEXT		O=open, C=closed
metric_detail_id	INTEGER		FK → metric_details
label	TEXT		Time-window label
test_name	TEXT		Test configuration name
assignable_cause_id	INTEGER	DEFAULT 0	FK → assignable_causes
last_changed_by	TEXT		Audit column
last_changed_date	TEXT		Audit column

Index cases_ix1 on (metric_detail_id, label, test_name). Source, group and name are recovered by joining metric_details on metric_detail_id.

11.10 Table: users

User accounts managed by cntwmsrv.

Column	Type	Constraints	Description
user_id	INTEGER	PK	Surrogate primary key
email	TEXT	UNIQUE	Login email address
fname	TEXT		First name
lname	TEXT		Last name
secret_value	TEXT		base64(sha1(lower(email)+passwo
permissions	INTEGER	DEFAULT 1	Permission level (0, 1, 2, or 9)
registered	TEXT		Registration timestamp
last_logon	TEXT		Last successful login timestamp
last_changed_by	TEXT		Audit column
last_changed_date	TEXT		Audit column

11.11 Table: notify_history

Throttle table: records the last time a notification was sent for each source/mgroup/name/label/test_name combination, preventing notification floods.

Column	Type	Constraints	Description
source	TEXT	PK (part)	Metric source identifier
mgroup	TEXT	PK (part)	Metric group identifier
name	TEXT	PK (part)	Metric name
label	TEXT	PK (part)	Time-window label
test_name	TEXT	PK (part)	Test configuration name
last_notified	INTEGER	DEFAULT 0	Unix epoch of last notification sent

Primary key is (source, mgroup, name, label, test_name). No audit columns: this is a transient throttle record, not an audit trail.

11.12 Table: assignable_causes

A catalogue of root-cause categories assignable to open cases via the web UI or the /continwm/cases/assignable_cause API. The system always contains the default row assignable_cause_id=0 ("Unknown / No cause determined").

Column	Type	Constraints	Description
assignable_cause_id	INTEGER	PK	Surrogate key (0 = built-in)
name	TEXT	NOT NULL	Short identifier
title	TEXT	NOT NULL	Human-readable title
description	TEXT	NOT NULL	Detailed description
control	TEXT	IN(I,O)	I=in-control, O=out-of-control
last_changed_by	TEXT	NOT NULL	Audit column
last_changed_date	TEXT	NOT NULL	Audit column

The control flag indicates whether the assigned cause represents normal in-control variation (I) or a genuine out-of-control condition (O) requiring corrective action.

11.13 Table: suspect_ranges

Records suspect metric observation ranges identified during case investigation. Each row links a failing test result to the exact detail_id range in InfluxDB3 considered suspect.

Column	Type	Constraints	Description
suspect_range_id	INTEGER	PK	Surrogate primary key
test_result_id	INTEGER	NOT NULL	FK → test_results
baseline_config_id	INTEGER	NOT NULL	FK → baseline_configs
case_id	INTEGER	NOT NULL	FK → cases
metric_detail_id	INTEGER	NOT NULL	FK → metric_details
start_detail_id	INTEGER	NOT NULL	First InfluxDB3 detail_id
end_detail_id	INTEGER	NOT NULL	Last InfluxDB3 detail_id
last_changed_by	TEXT	NOT NULL	Audit column
last_changed_date	TEXT	NOT NULL	Audit column

Indexes: suspect_ranges_ix1 on (baseline_config_id); suspect_ranges_ix2 on (case_id); suspect_ranges_ix3 on (metric_detail_id).

11.14 Table: headlines

Short human-readable summary rows generated each time a notified test result is written. Used by the web UI Headlines tab to display a live event feed.

Column	Type	Constraints	Description
headline_id	INTEGER	PK	Surrogate primary key
test_result_id	INTEGER	NOT NULL DEFAULT 0	FK → test_results
timestamp	TEXT	NOT NULL DEFAULT now	When headline was created
headline	TEXT	NOT NULL	Human-readable headline text
last_changed_by	TEXT	NOT NULL	Audit column
last_changed_date	TEXT	NOT NULL DEFAULT now	Audit column

Indexes: headlines_ix1 on (test_result_id); headlines_ix2 on (timestamp).

Headlines are inserted by cntwmtd after writing a notified (notified='Y') test result. The text is auto-formatted as:

```
<name> <label>: <title>. Baseline: <baseline>.
Test: <test_name> <test_title> p-value = <p> alpha
= <a>.
```

Test child processes may also push custom headline text via a `headline` message in the `cntwmifc` pipe protocol; `cntwmtd` forwards these to `cntwmdb`.

12 InfluxDB3 Database

`InfluxDB3` stores the raw metric detail rows in a measurement called `detail`:

Field	Kind	Description
`detail_id`	tag	Unique row identifier
`source`	tag	Metric source
`mgroup`	tag	Metric group
`name`	tag	Metric name
`time`	column	ISO 8601 timestamp (nanoseconds)
`epoch`	field	Unix epoch seconds
`metric_type`	field	Metric type code
`sequence`	field	Monotonic sequence
`delta_usec`	field	Interval duration (μs)
`delta_count`	field	Event count
`delta_s_value`	field	Sum of values
`delta_sq_value`	field	Sum of squared values

13 Cross-Database Relationships

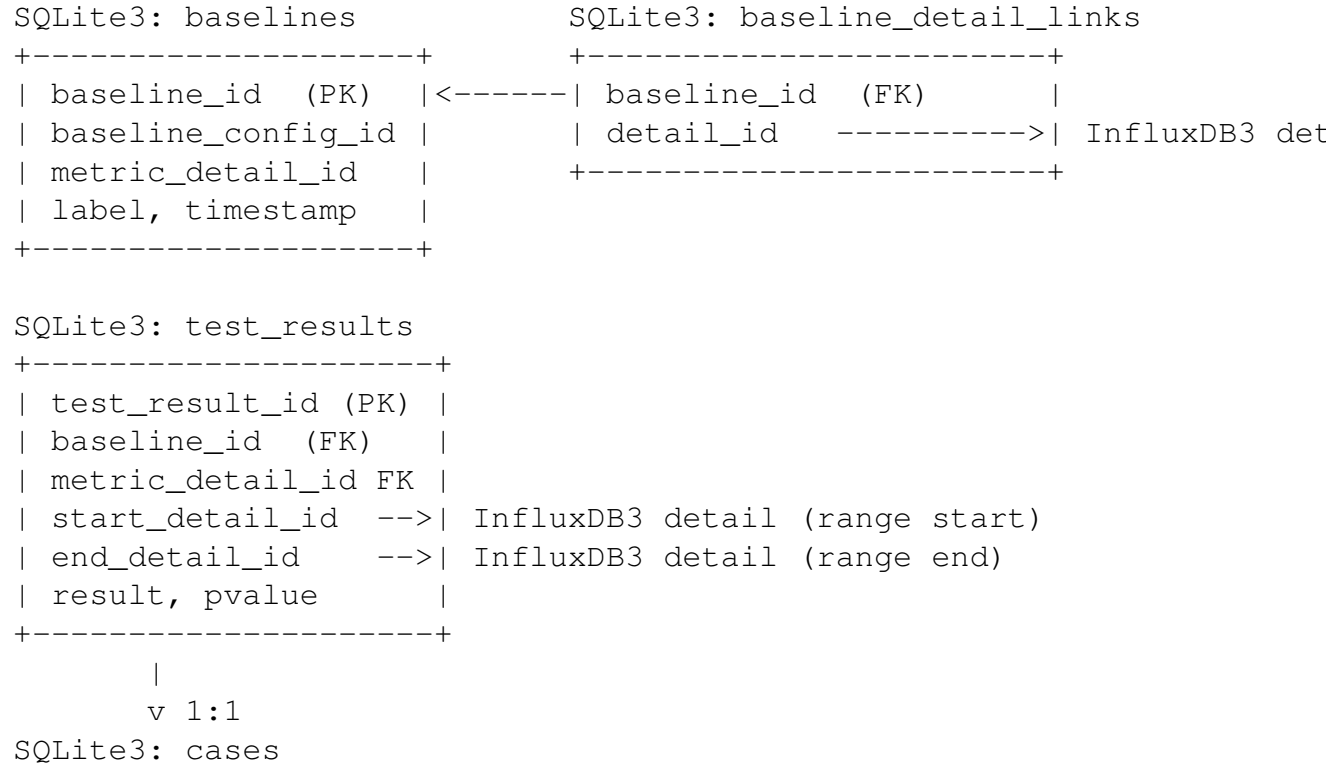
`SQLite3` holds all operational state; `InfluxDB3` holds the raw time-series detail rows. The two stores are linked by integer `detail_id` values (tags in `InfluxDB3`, integer columns/indexes in `SQLite3`).

13.1 Data Lifecycle

1. **Metric collection.** A metric broadcaster sends detail rows to `continwm` on port 60005. `continwm` forwards each row to `cntwmdb` (`write_detail`), which writes it to `InfluxDB3` and assigns a monotonically increasing `detail_id` from the `cntwmseq` sequence.
2. **Baseline computation (`cntwmbl`).** `cntwmbl` posts `run_baseline` to `cntwmdb`. `cntwmdb` queries `InfluxDB3` for all detail rows in the configured time window, aggregates them into per-metric/label summary statistics, inserts one row into `baselines`, and inserts one row per contributing `detail_id` into `baseline_detail_links`.

3. **Hypothesis test dispatch (`continwm`).** Once `test_size` detail rows have accumulated for a metric, `continwm` serialises a `hypothesis_test_t` and sends a `MESSAGE_HTEST` to a `cntwmtd` worker. The message carries the boundary `detail_id` values and wall-clock timestamps for the sample window.
4. **Data retrieval (`cntwmtd` → `cntwmdb`).** `cntwmtd` calls `get_baseline_summary` (returns aggregate statistics from `baselines`), `get_baseline_detail` (returns raw rows from `InfluxDB3` via `baseline_detail_links`), and `get_sample_detai` (returns raw rows from `InfluxDB3` using the `detail_id` range from the boundary IDs).
5. **Test execution.** The test child receives data via `cntwmifc` pipes, computes a p-value, and posts it back to `cntwmtd`.
6. **Result storage (`cntwmtd` → `cntwmdb`).** `cntwmtd` calls `write_result`: `cntwmdb` applies the tolerance gate against the `notifications` table, inserts a `test_results` row, and returns `notified` and `consecutive_errors`. If `notified='Y'`: `cntwmdb` opens a case (`add_open_case`) and checks the repeat-delay gate (`check_notify` against `notify_history`). If the gate allows it, `cntwmtd` invokes the `notify_process` via `popen()`. On PASS: the case is closed (`delete_open_case`).

13.2 SQLite3 / InfluxDB3 Linkage



```
+----------------------+
| case_id (PK)         |
| test_result_id (FK)  |
| assignable_cause_id  |---> assignable_causes
| status (O/C)         |
+----------------------+
```

14 apply_to_metrics Expression Evaluation

The `apply_to_metrics` field in both `test_configs` and `baseline_configs` is evaluated by the **expeval** expression evaluation library [1] to determine whether a given test or baseline configuration applies to an incoming metric.

The expression context provides:

Variable	Type	Description
`source`	string	Metric source identifier
`mgroup`	string	Metric group string
`name`	string	Metric name

Example expressions:

```
source like ".*"
```

Matches all metrics.

```
source == "posipmon"
```

Applies only to metrics from source `posipmon`.

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

Applies to all response-time metrics from posipmon sources.

The `like` operator performs POSIX regular expression matching. Full expression syntax is described in the expeval documentation [1, 2].

15 Notification Process

Notifications are raised when a test configuration's `tolerance` threshold is reached (i.e. the metric has failed consecutively for at least `tolerance` test cycles).

15.1 Notification Flow

1. Test child returns p-value to `cntwmtd`.
2. `cntwmtd` classifies: PASS / FAIL / INCONCLUSIVE.
3. `cntwmtd` POSTs `write_result` to `cntwmdb`:
   - **FAIL path:** look up / create `notifications` row; increment `consecutive_errors`; if $\geq$ `tolerance`: DELETE row, set `notified='Y'`.
   - **PASS path:** DELETE `notifications` row, set `notified='N'`.
4. INSERT `test_results` row.
5. If `notified='Y'`:
   - POST `add_open_case` $\rightarrow$ `cases` table.
   - POST `check_notify` $\rightarrow$ `notify_history`: if elapsed time since last notification $>$ `notify_min_delay` then allowed; else suppressed.
   - If allowed: `popen(test_config.notify_process)`, write alert message to stdin.
6. If result is PASS: POST `delete_open_case`.

15.2 Email Notification

The supplied `continwm_email_notify.sh` script can be used as the `notify_process`. It reads the notification message from stdin and dispatches it via the `cntwmen` SMTP binary.

Configure in `test_configs`:

```
notify_process       = /path/to/continwm_email_notify.sh
notify_process_parms = recipient@example.com
```

SMTP settings are stored in `$HOME/.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-password-here
CNTWMEN_SMTP_SSL=0
```

Note: `CNTWMEN_SMTP_USER` and `CNTWMEN_SMTP_PASSWORD` are optional when `CNTWMEN_SMTP_SSL` is `0` (plain SMTP). Omit them when connecting to an unauthenticated local MTA or internal relay.

16 Shell Scripts

All shell scripts are installed alongside the compiled binaries in `$HOME/software/build/bin/`.

16.1 continwm.sh — System Startup and Control

Purpose: Start, stop, restart, and query the status of the complete continwm stack (`cntwmdb`, `continwm`, and N `cntwmtd` workers).

```
continwm.sh start
continwm.sh stop
continwm.sh restart
continwm.sh status
```

The `start` subcommand creates all required runtime directories, then starts processes in dependency order: `cntwmdb` first (pausing 2 seconds), then `cntwmsrv` (user management REST API, port 7070), then `continwm` (pausing 2 seconds), then `CNTWMTD_COUNT` (default 5) `cntwmtd` instances. Each process has stdout and stderr redirected to named files under `logs/`. PID files are written to `pids/`.

The `stop` subcommand shuts down in reverse order, sending SIGTERM and escalating to SIGKILL after 15 seconds if necessary.

Key configuration variables in the script:

Variable	Default	Description
`CONTINWM_HOME`	`$HOME/continwm`	Runtime root directory
`CNTWMDB_PORT`	60999	`cntwmdb` HTTP port
`INFLUXDB3_HOST`	network.codemagus.com	`InfluxDB3` server
`INFLUXDB3_AUTH_TOKEN`	(env var)	`InfluxDB3` API token
`CNTWMTD_COUNT`	5	Number of `cntwmtd` workers
`METRIC_PORT`	60005	Metric reception port
`TESTDRIVER_PORT`	60095	Test-driver port
`REPORT_ROOT_DIR`	`$CONTINWM_HOME/reports`	Test report root
`CNTWMSRV_HOST`	0.0.0.0	`cntwmsrv` bind address
`CNTWMSRV_PORT`	7070	`cntwmsrv` HTTP port
`CNTWMSRV_REGISTER_SCRIPT`	`.../cntwmsrv_register.sh`	Registration script
`CNTWMSRV_DEFAULT_PERMISSIONS`	1	Default permissions for new users

16.2 cntwmbl.sh — Baseline Computation Cron Wrapper

Purpose: Run `cntwmbl` for all baseline configurations whose TTL has expired. Suitable as a cron job.

```
# Via cron (every hour)
0 * * * * /home/stephen/software/build/bin/cntwmbl.sh
```

Output is logged to `$CONTINWM_HOME/logs/cntwmbl_stdout.txt` and `cntwmbl_stderr.txt`

16.3 continwm_email_notify.sh — Email Notification Handler

Purpose: Deliver alert messages via SMTP. Configured as `notify_process` in a `test_configs` row. Reads the alert message from stdin and passes it to `cntwmen`. SMTP settings (and optional credentials) are read from `$HOME/.continwm_email.conf`. Credentials are not required when using plain SMTP with an unauthenticated local MTA or relay; see Section 3.10 for details.

16.4 cntwmtc.sh — Test Configuration Manager Wrapper

Purpose: Wrapper script for `cntwmtc`. Supplies the default `cntwmdb` host and port and locates the `cntwmtc` binary, then passes all arguments through unchanged. The `cntwmdb` location can be overridden:

```
CNTWMDB_HOST=192.168.1.10 CNTWMDB_PORT=60999 cntwmtc.sh --list-configs
```

16.5 config_test.sh — Test Configuration Integration Test

Purpose: Regression test for test configuration management. Calls `cntwmtc` to perform a sequence of list, add, edit, and delete operations against the live `cntwmdb` service.

16.6 fake_notify.sh — Test Notification Sink

Purpose: Lightweight notification sink for development and testing. Appends the notification message (with a UTC timestamp header) to `/tmp/continwm_fake_notify.log` rather than sending email. Configure in `test_configs` as:

```
notify_process = /path/to/fake_notify.sh
```

Monitor with `tail -f /tmp/continwm_fake_notify.log`.

16.7 cntwmsrv_register.sh — Registration Notification Script

Purpose: Sample script invoked by `cntwmsrv` whenever a new user registers or requests a password reset. Referenced by the mandatory `--register-script` option.

The script receives five positional arguments:

```
$1  email address
$2  event type: "register" or "reset"
$3  generated password
$4  first name
$5  last name
```

The sample implementation logs the event (including the generated password) to `$CONTINWM_HOME/logs/cntwmsrv_register.log`. In a production deployment this script should be replaced with one that delivers the password to the new user by email or another secure channel.

16.8 cntwmrks_config.sh — R KS-Test Configuration Helper

Purpose: Registers the `cntwmrks` test configuration in the `test_configs` table via `cntwmtc`. Sets `test_process` to `/usr/bin/Rscript` and `test_process_parms` to the path of `cntwmrks.R`.

16.9 test_cntwmbl_config.sh — Baseline Configuration Integration Test

Purpose: Ten-step integration test for baseline configuration management. Exercises all `cntwmbl` config actions and triggers a baseline computation run.

17 Live Instance Directory and File Structure

A live continwm deployment is rooted at `$CONTINWM_HOME` (default `$HOME/continwm`). The directory structure is created automatically by `continwm.sh start`.

```
$HOME/continwm/
|-- database/
|   `-- continwm.db          SQLite3 database (cntwmdb only)
|-- logs/
|   |-- cntwmdb_stdout.txt
|   |-- cntwmdb_stderr.txt
|   |-- cntwmdb_NNN_D{date}_T{time}_{pid}_{seq}_log.txt
```

```
|   |-- cntwmsrv_stdout.txt
|   |-- cntwmsrv_stderr.txt
|   |-- cntwmsrv_NNN_D{date}_T{time}_{pid}_{seq}_log.txt
|   |-- cntwmsrv_register.log
|   |-- continwm_stdout.txt
|   |-- continwm_stderr.txt
|   |-- continwm_NNN_D{date}_T{time}_{pid}_{seq}_log.txt
|   |-- cntwmtd_001_stdout.txt
|   |-- cntwmtd_001_stderr.txt
|   |-- cntwmtd_001_NNN_D{...}_log.txt
|   |-- cntwmbl_stdout.txt
|   `-- cntwmbl_stderr.txt
|-- pids/
|   |-- cntwmdb.pid
|   |-- cntwmsrv.pid
|   |-- continwm.pid
|   |-- cntwmtd_001.pid
|   `-- cntwmtd_005.pid
|-- configs/
`-- reports/
    `-- D{date}_T{time}_{testnum}_{source}_{mgroup}_{name}_{label}_{tes
```

17.1 Log File Naming Convention

Rotating log files follow the pattern:

`{prefix}_{NNN}_D{YYYYMMDD}_T{HHMMSS}_{PID}_{SEQNO}_log.txt`

Example:

`cntwmdb_000_D20260731_T095612_03261203_00000001_log.txt`

17.2 Report Directory Naming Convention

Per-test report directories are named:

`D{YYYYMMDD}_T{HHMMSS}_{testnum}_{source}_{mgroup}_{name}_{label}_{testr`

Example:

`D20260731_T101350_2263743_posipmon_ecosys_modelapp1_posipmon_01_possler`

The test child process creates this directory and writes report files into it. The path is recorded in `test_results.report_path`.

18 Dynamic Configuration Reload

When a `test_configs` row is inserted, deleted, or updated via `cntwmtc`, `continwm` picks up the change without requiring a restart. The mechanism consists of two co-operating levels.

18.1 SIGUSR1 Signal Path

1. On startup `continwm` POSTs a `register_continwm_pid` action to `/continwm/server` with its process identifier:

   `{"action":"register_continwm_pid","pid":<pid>}`

2. Any `handle_testconfig` mutation in `cntwmdb` (insert, delete, or update — committed to `SQLite3`) calls `notify_continwm()`, which issues `kill(continwm_pid, SIGUSR1)`.

3. The `SIGUSR1` handler in `continwm` sets a `volatile sig_atomic_t` flag `g_reload_testconfigs = 1`.

4. At the *top* of every main-loop iteration `continwm` checks this flag. When set it calls `reload_test_configs()` which closes and re-opens both the `testconfigs` and `metrics_to_test` in-memory hash tables, reloads all rows from `cntwmdb`, and updates the `last_config_load` timestamp.

Because the reload check precedes the `pselect()` call, an `EINTR` from the signal causes an immediate loop re-entry and reload.

18.2 Periodic Backstop

In addition to the signal path, `continwm` automatically reloads test configurations when `CONFIG_RELOAD_INTERVAL` seconds (300 by default) have elapsed since the last load. This backstop fires regardless of signal delivery, ensuring eventual consistency.

18.3 What Gets Reloaded

Cache	Description
`control->testconfigs`	Compiled test configuration objects, including parsed `expeval` expressions from `apply_to_metrics`
`control->metrics_to_test`	Per-metric hash of matching test configurations (built lazily at metric dispatch time)

The `hmattr` (seen-metrics set), `testbatches` (per-metric batch state), and `lastmetrics` (per-metric last value) are **not** cleared, so in-progress metric accumulation is not disturbed.

18.4 Operational Impact

- A `test_size` change takes effect within at most one main-loop cycle — typically under one second when signalled, or up to 300 seconds via the backstop.
- A metric that has already accumulated samples continues from where it left off; only the dispatch threshold changes.
- Multiple rapid `cntwmtc` operations each send a `SIGUSR1`; `continwm` performs at most one reload per main-loop cycle because the flag is cleared at the start of `reload_test_configs()`.

References

[1] Stephen R. Donaldson, Patrick N. Hayward, Patrick C. Power, and Jan Vlok. expeval: Expression Evaluation API Reference. CML Document CML00052-01, Code Magus Limited, July 2024. PDF.

[2] Stephen R. Donaldson, Patrick N. Hayward, Patrick C. Power, and Jan Vlok. expeval: Expression Evaluation User Guide. CML Document CML00091-01, Code Magus Limited, July 2024. PDF.