NYSE +0.31% · NASDAQ +0.47% · LSE -0.12% · Eurex +0.18% · JPX +0.22% · HKEX -0.08% · CME +0.55% · SGX +0.11%



EQUITY TRADING INFRASTRUCTURE

REFERENCE MANUAL

The Technical Reference Manual for EduMatcher With Complete Technical Details

MARKET DEPTH · AAPL +1.24 +0.83%

BID QTY		PRICE	ASK QTY
8,340		150.35	3,920
7,210		150.34	5,100
5,820		150.33	6,400
12,100	LAST	150.32	7,840
4,420		150.31	10,220
3,200		150.30	12,800

EduMatcher Reference Manual
For EduMatcher Version: v0.77.0
Copyright © Johan Persson
Generated on November 8, 2026
Prepared with LaTeX from Markdown sources for duplex printing. Sources for documentation available at https://github.com/johan162/EduMatcher.

Reference Manual

`edumatcher`, Version: v0.77.0

Commands, configuration, terms, and known limitations

Johan Persson <johan162@gmail.com>

Generated on November 8, 2026

Chapters

1 How to Use This Book 1

2 Processes, Environment and Ports 3

3 Configuration Schema and Process Blocks 115

4 Engine Configuration Specification 159

5 Glossary 185

6 Known Limitations & Bugs 207

Contents

1 How to Use This Book **1**
1.1 How the book is organized . . . 1
1.2 Finding things fast . . . 1

2 Processes, Environment and Ports **3**
2.1 Background — Why Separate Processes? . . . 3
2.1.1 The monolith approach . . . 3
2.1.2 The message-bus approach . . . 3
2.1.3 The two bus patterns used here . . . 6
2.2 Process Overview . . . 6
2.3 Environment variables . . . 6
2.3.1 Network variables . . . 7
2.3.2 How the default is selected . . . 9
2.4 Port Reference . . . 12
2.5 pm-engine — Matching Engine . . . 19
2.6 pm-alf-console — User Gateway . . . 22
2.7 pm-alf-gwy — ALF TCP Gateway . . . 24
2.8 pm-viewer — Order Book Viewer . . . 26
2.9 pm-orders — Order Status Monitor . . . 28
2.10 pm-audit - Event Logger . . . 29
2.11 pm-audit-cli — Audit Log Query CLI . . . 30
2.12 pm-audit-replay — Audit Replay . . . 32
2.13 pm-clearing — Clearing & P&L . . . 34
2.14 pm-clearing-cli - Clearing Query CLI . . . 36
2.15 pm-stats — Statistics Recorder . . . 39
2.15.1 How statistics are computed . . . 41
2.15.2 Statistics Database Schema . . . 41
2.16 pm-stats-cli — Statistics Query CLI . . . 44
2.17 pm-scheduler — Session Scheduler . . . 46
2.18 pm-eod — End-of-Day Settlement . . . 47
2.18.1 `roster.yaml` . . . 49
2.18.2 Files written to `settlement/<date>/` . . . 49
2.18.3 The settlement ledger `eod.db` . . . 50
2.18.4 Settlement rules (reference) . . . 50
2.19 pm-ticker — Scrolling Market Ticker . . . 50
2.20 pm-board — Market Board . . . 52
2.21 pm-ai-trader — Autonomous Trader Bot . . . 53

2.22 pm-ai-swarm — Multi-Agent Trading Swarm . . . 55
2.23 pm-market-sim — Market Model . . . 56
2.24 pm-news — News Desk . . . 57
2.25 pm-compdb-cli — Company Database CLI . . . 58
2.25.1 Reading a company profile . . . 60
2.26 pm-new-symbol / pm-ipo — List a New Symbol . . . 66
2.27 pm-mm-bot — Autonomous Market-Maker Bot . . . 67
2.28 pm-valuation — IPO Valuation . . . 72
2.29 pm-ralf-gwy — Post-Trade Dissemination Gateway . . . 73
2.30 pm-ralf-spy (RALF Protocol Spy) . . . 74
2.31 pm-dc-gwy — Drop-Copy TCP Gateway . . . 76
2.32 pm-dc-spy (Drop-Copy Spy) . . . 77
2.33 pm-index — Index Calculation Process . . . 78
2.34 pm-api-gwy — REST/WebSocket API Gateway . . . 80
2.35 pm-admin — Interactive Admin Console . . . 83
2.36 pm-admin-cli — CLI Admin Commands . . . 85
2.37 pm-cverifier — Config Verifier . . . 86
2.38 pm-config-show — Config Viewer . . . 87
2.39 pm-setup — Session Bootstrap Tool . . . 88
2.40 pm-config-deploy — Compile and Install a Configuration . . . 89
2.41 pm-config-gen — Engine Config Generator . . . 90
2.42 pm-index-cli — Index Structural/Audit History Query Tool . . . 99
2.43 pm-index-admin-cli — Index Corporate Action / Constituent Change CLI . . . 100
2.44 pm-opctl-cli — Operational Process Control . . . 101
2.45 pm-help / pm-man — Command Index and Man-Page Reference . . . 104
2.46 Order Lifecycle Message Flow . . . 105
2.47 pm-balf-gwy — BALF TCP Gateway . . . 106
2.48 pm-md-gwy (CALF Market-Data Gateway) . . . 106
2.49 pm-calf-spy (CALF Protocol Spy) . . . 107
2.50 pm-log-srv — Centralized Log Server . . . 108
2.51 pm-log-cli — Log Server Query/Troubleshooting CLI . . . 110
2.52 pm-monitor — Health Collector . . . 110
2.53 pm-monitor-cli — Health Query CLI . . . 111
2.54 pm-msgen — Message Binding Generator . . . 111
2.55 See also . . . 112

3 Configuration Schema and Process Blocks 115
3.1 Current Schema . . . 115
3.2 Which Process Reads What . . . 116
3.3 Configuring `pm-alf-gwy` . . . 119
3.3.1 Fields . . . 120
3.3.2 How the settings work together . . . 120
3.3.3 Overrides and generation . . . 121
3.4 Configuring `pm-ralf-gwy` . . . 121
3.5 Configuring `pm-md-gwy` . . . 122
3.6 Configuring `pm-balf-gwy` . . . 123
3.7 Configuring `pm-dc-gwy` . . . 124
3.8 Configuring `pm-api-gwy` . . . 125
3.9 Configuring `pm-index` . . . 127
3.10 Configuring `pm-log-srv` . . . 128

3.10.1 LALF-PS fields — the ZeroMQ log-distribution interface . . . 129
3.11 Configuring pm-monitor . . . 130
3.12 Engine Behavior Flags . . . 132
3.12.1 sessions_enabled . . . 132
3.12.2 enforce_collars . . . 132
3.12.3 enforce_circuit_breakers . . . 132
3.12.4 auction_indicative_interval_sec . . . 132
3.12.5 engine_tuning . . . 132
3.12.6 engine_tuning.snapshot_interval_sec . . . 133
3.12.7 engine_tuning.quote_history_maxlen . . . 133
3.12.8 engine_tuning.drop_copy_buffer_size . . . 133
3.12.9 engine_tuning.recent_trades_maxlen . . . 133
3.12.10 engine_tuning.depth_snapshot_tolerance_ticks . . . 133
3.13 Participant Defaults . . . 134
3.14 Participants . . . 134
3.14.1 Participant Fields . . . 135
3.14.2 Role Privileges . . . 136
3.15 Market-Maker Obligation Defaults . . . 136
3.16 Symbol Universe . . . 137
3.16.1 Symbol Fields . . . 137
3.16.2 Mandatory Fields . . . 138
3.16.3 Collar Reference Price Selection . . . 138
3.17 Risk Controls and Collars . . . 139
3.17.1 Per-symbol risk-level assignment . . . 139
3.17.2 Risk-control Fields . . . 140
3.17.3 Collar Fields . . . 140
3.18 Circuit Breakers . . . 141
3.18.1 Circuit-breaker Fields . . . 142
3.19 Market-Maker Quote Seeds . . . 142
3.20 Startup Market-Maker Combo Seeds . . . 143
3.21 Session Schedule . . . 144
3.21.1 locale . . . 145
3.22 Formal Specification . . . 146
3.22.1 Top-level fields . . . 146
3.22.2 engine_tuning fields . . . 147
3.22.3 locale fields . . . 148
3.22.4 participants[] — gateway entry fields . . . 148
3.22.5 participants[].mm_obligations.<SYMBOL> fields . . . 149
3.22.6 mm_obligation_defaults fields . . . 149
3.22.7 mm_obligation_defaults.symbols.<SYMBOL> fields . . . 150
3.22.8 risk_controls fields . . . 150
3.22.9 risk_controls.levels.<LEVEL> fields . . . 150
3.22.10 Collar fields — in risk_controls.levels.<LEVEL>.collar or symbols.<SYMBOL>.collar 150
3.22.11 circuit_breaker_defaults fields . . . 151
3.22.12 circuit_breaker_defaults.reopening and symbols.<SYMBOL>.circuit_breaker.reopening fields . . . 151
3.22.13 circuit_breaker_defaults.levels.<LEVEL> and symbols.<SYMBOL>.circuit_breaker.levels.<L fields . . . 152
3.22.14 symbols.<SYMBOL> fields . . . 153
3.22.15 symbols.<SYMBOL>.order_limits fields . . . 154

3.22.16 `symbols.<SYMBOL>.market_maker_quotes[]` fields . . . 155
3.22.17 `market_maker_combos[]` fields . . . 155
3.22.18 `market_maker_combos[].legs[]` fields . . . 156
3.22.19 `schedule` fields . . . 157
3.22.20 Cross-field validation rules . . . 158

4 Engine Configuration Specification 159
4.1 1. Conventions . . . 159
4.1.1 1.1 Requirement keywords . . . 159
4.1.2 1.2 The file . . . 159
4.1.3 1.3 Scalar type notation . . . 159
4.1.4 1.4 Field-table columns . . . 160
4.1.5 1.5 Unknown keys . . . 161
4.1.6 1.6 Case normalisation . . . 161
4.2 2. Enumerations . . . 161
4.3 3. Formal schema tree . . . 162
4.4 4. Domain sub-structures . . . 164
4.4.1 4.1 `MMQuoteSeedSpec` — `symbols.<S>.market_maker_quotes[]` . . . 164
4.4.2 4.2 `CollarSpec` — `symbols.<S>.collar` and `risk_controls.levels.<L>.collar` . 164
4.4.3 4.3 `OrderLimitsSpec` — `symbols.<S>.order_limits` . . . 164
4.4.4 4.4 `MMObligationSpec` — `participants[].mm_obligations.<S>` . . . 165
4.4.5 4.5 `ComboSeedSpec` — `market_maker_combos[]` . . . 165
4.4.6 4.6 `IndexSpec` — `indices[]` . . . 166
4.4.7 4.7 `WeeklyScheduleSpec` — `schedule`, and `DayScheduleSpec` — one day block . . 166
4.5 5. Engine sections . . . 168
4.5.1 5.1 `symbols` (REQUIRED) . . . 168
4.5.2 5.2 `participants` (REQUIRED) . . . 168
4.5.3 5.2a `participant_defaults` (OPTIONAL) — `ParticipantDefaultsSpec` . . . 168
4.5.4 5.3 Engine behaviour flags . . . 168
4.5.5 5.3a `auction_indicative_interval_sec` and `engine_tuning` (OPTIONAL) . . . 169
4.5.6 5.4 `mm_obligation_defaults` (OPTIONAL) — `MMObligationDefaultsSpec` . . . 169
4.5.7 5.5 `risk_controls` (OPTIONAL) — `RiskControlsSpec` . . . 170
4.5.8 5.6 `circuit_breaker_defaults` and `symbols.<S>.circuit_breaker` — `CircuitBreakerSpec` 170
4.5.9 5.7 `market_maker_combos` (OPTIONAL) . . . 171
4.5.10 5.8 `indices` (OPTIONAL) . . . 171
4.5.11 5.9 `schedule` (OPTIONAL) . . . 171
4.5.12 5.10 `locale` (OPTIONAL) . . . 171
4.6 6. Auxiliary gateway blocks . . . 172
4.6.1 6.1 `alf_gateway` — `pm-alf-gwy` . . . 173
4.6.2 6.2 `balf_gateway` — `pm-balf-gwy` . . . 173
4.6.3 6.3 `market_data_gateway` — `pm-md-gwy` (CALF) . . . 174
4.6.4 6.4 `post_trade_gateway` — `pm-ralf-gwy` (RALF) . . . 174
4.6.5 6.5 `api_gateways` — `pm-api-gwy` (REST / WebSocket) . . . 174
4.6.6 6.6 `dc_gateway` — `pm-dc-gwy` (drop-copy TCP relay) . . . 176
4.6.7 6.7 `log_server` — `pm-log-srv` (centralized LALF log collector) — `LogSrvProcSpec` 177
4.6.8 6.8 `monitor` — `pm-monitor` and every beaconing process — `MonitorSpec` . . . 180
4.7 7. Cross-field and semantic constraints . . . 181
4.8 8. Processing model . . . 183
4.9 9. Minimal conformant document . . . 183
4.10 See also . . . 184

5 Glossary **185**
5.1 EduMatcher terms . . . 185
5.2 General terms . . . 187
5.2.1 A . . . 187
5.2.2 B . . . 188
5.2.3 C . . . 189
5.2.4 D . . . 191
5.2.5 E . . . 192
5.2.6 F . . . 192
5.2.7 G . . . 193
5.2.8 H . . . 194
5.2.9 I . . . 194
5.2.10 K . . . 195
5.2.11 L . . . 196
5.2.12 M . . . 197
5.2.13 N . . . 198
5.2.14 O . . . 199
5.2.15 P . . . 199
5.2.16 Q . . . 201
5.2.17 R . . . 201
5.2.18 S . . . 202
5.2.19 T . . . 204
5.2.20 U . . . 205
5.2.21 V . . . 205
5.2.22 W . . . 205
5.2.23 Z . . . 206

6 Known Limitations & Bugs **207**
6.1 Current limitations and design boundaries . . . 207
6.2 What this page covers . . . 209
6.3 Background: how a fill used to be reported . . . 210
6.4 How the gateway turns a fill into a position . . . 210
6.5 Worked example . . . 211
6.5.1 Direction of the (former) error . . . 211
6.6 Visualising the fix . . . 212
6.7 How it was fixed . . . 213

CHAPTER 1

How to Use This Book

The Reference Manual is the place to **look something up**. You arrive from another book with one question — what does this option do, what values may this field take, what does this word mean — and leave with the answer. It contains no tutorials; the other books explain and link here for detail.

Who it is for	Everyone, in the middle of a task
What it assumes	You know what you are looking for. If you do not yet, start with the Quick Start Guide
What it leaves to other books	How-to and explanation (the Participant and Operator's Guides); wire messages (Protocols and Clients)

1.1 How the book is organized

Part	Look here for
I. Command line	Every `pm-*` process and tool: what it does, its options, the environment variable that decides where data lives, and every network port
II. Configuration	Every top-level key and field of `engine_config.yaml`, the per-process configuration blocks, and the formal specification the verifier enforces
Appendices	The Glossary and the Known Limitations

1.2 Finding things fast

- Every command also documents itself: `pm-help` lists all `pm-*` commands with a one-line summary, and `pm-help <command>` prints its full manual page.
- `pm-config-show` prints the configuration the running exchange actually uses; `pm-cverifier` explains any problem in a configuration file.
- In the online documentation, the search box at the top finds any command, option or field name.

CHAPTER 2

Processes, Environment and Ports

Learning objectives

After reading this page you will understand:

- Why EduMatcher is built as a set of independent processes connected by a message bus rather than a single monolithic program
- The trade-offs that architecture introduces: scalability and observability vs. deployment complexity and latency
  - The role and responsibilities of each of the core runtime processes plus the optional AI trader, market-maker bot, and admin tools
- Which processes are mandatory and which are optional observers
- How to read the message-flow tables to trace an order from submission to fill

2.1 Background — Why Separate Processes?

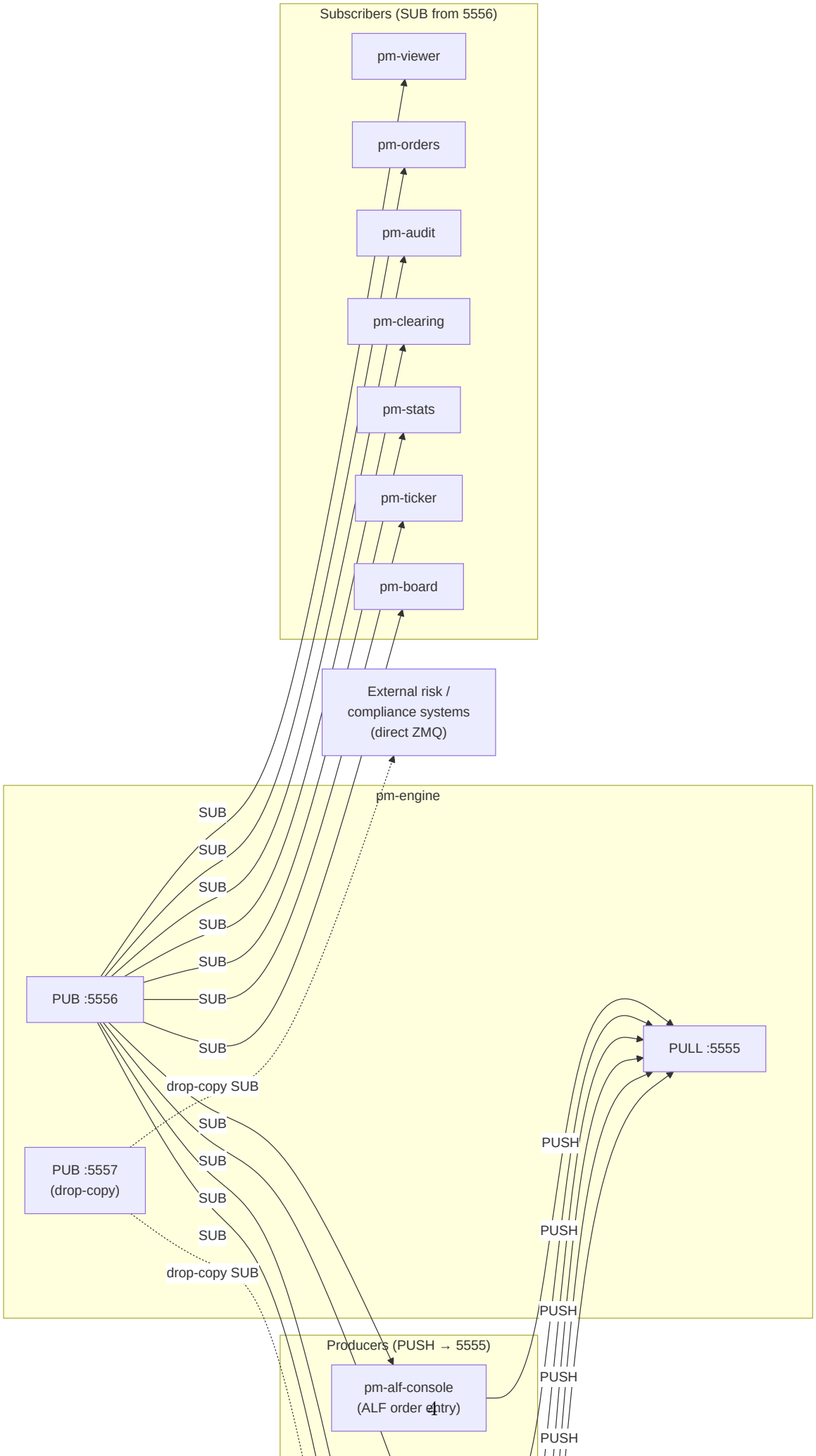
2.1.1 The monolith approach

The simplest trading system is a single program: the user types an order, the matching logic runs, and the result is printed — all in the same process. This is easy to build, easy to debug, and fast. It is also a dead end as soon as you need any of the following:

- More than one trader connected simultaneously
- Observers (audit, statistics, P&L) that see every trade without modifying the matching logic
- The ability to restart one component without restarting everything
- A viewer that can run on a different machine from the engine

2.1.2 The message-bus approach

EduMatcher distributes work across processes connected by a **ZeroMQ message bus**. The engine owns three sockets:


Subscribers (SUB from 5556)
pm-viewer
pm-orders
pm-audit
pm-clearing
pm-stats
pm-ticker
pm-board
External risk / compliance systems (direct ZMQ)
pm-engine
SUB
SUB
SUB
SUB
SUB
SUB
SUB
PUB :5556
drop-copy SUB
SUB
SUB
SUB
SUB
PUB :5557 (drop-copy)
drop-copy SUB
PULL :5555
PUSH
PUSH
PUSH
PUSH
PUSH
Producers (PUSH → 5555)
pm-alf-console (ALF order entry)

The engine is the only process that writes to the order book. Every other process is either a **producer** (sends commands via PUSH) or a **subscriber** (reads events via SUB), or both. No process shares memory with any other.

Port	Socket type	Bound by	Purpose
5555	PULL	pm-engine	Receives all inbound commands (orders, cancels, admin)
5556	PUB	pm-engine	Broadcasts all market events (fills, book updates, session changes)
5557	PUB	pm-engine	Drop-copy feed, fill events tagged per gateway, with sequence numbers and replay support

This design has concrete advantages:

Advantage	How it helps
Isolation	A crashing viewer cannot corrupt the order book
Observability	Any subscriber can see every event without changing the engine
Horizontal scale	Multiple gateways connect simultaneously; each is independent
Language freedom	A subscriber can be written in any language that speaks ZeroMQ
Testability	The engine can be tested with a fake gateway; subscribers can be tested with recorded message streams

And honest disadvantages:

Disadvantage	Consequence
Latency	Each message crosses a socket; a round-trip (order → engine → ACK) takes microseconds rather than nanoseconds on localhost
Startup order	The engine must bind before any other process connects
Partial failure	If the engine crashes mid-session, subscribers hold stale state until it restarts
No shared clock	Timestamps are set by the sender; two processes on different machines can disagree by milliseconds
Message loss	ZeroMQ PUB/SUB drops messages if a subscriber is slow; fast publishers can outrun slow consumers

For an educational system running on a laptop, none of the disadvantages are serious. For a production exchange, each one would require careful engineering (persistent queues, hardware timestamping, consensus protocols).

2.1.3 The two bus patterns used here

Pattern	Socket pair	Direction	Used for
PUSH/PULL	Gateway PUSH → Engine PULL	One-to-one, reliable delivery	Sending commands (orders, cancel requests) to the engine
PUB/SUB	Engine PUB → many SUBs	One-to-many, topic-filtered	Broadcasting events (fills, book updates, session changes)

PUSH/PULL guarantees delivery to exactly one receiver (the engine). PUB/SUB does not guarantee delivery — a subscriber that is not yet connected will miss messages sent before it subscribes. This is why the engine publishes an initial book snapshot on startup: late-joining viewers can request the current state rather than waiting for the next change.

2.2 Process Overview

A complete EduMatcher session is built from the processes below: the core engine, its recorders, and the external protocol gateways (tabulated as **Core processes**); a set of query, admin and spy tools used as needed (tabulated as **Monitoring & Admin tools**); and a handful of AI-trader, setup and config entrypoints that each get their own section further down instead of a table row.

Protocol landscape

EduMatcher supports multiple external protocol families with different roles: ALF for order entry, BALF for binary order entry, CALF for market-data dissemination, and RALF for post-trade dissemination. For a single map of purpose, runtime status, and links to detailed chapters and formal specs, see External Protocols Overview.

Why pm-?

All CLI commands share the `pm-` prefix, short for **Process for Matching**. The prefix avoids name collisions with system utilities and makes it easy to identify EduMatcher processes in a process list (`ps aux | grep pm-`).

Installation modes

Commands are shown without a prefix. If you are running from a **source checkout** (developer mode), prepend `poetry run`:

```
poetry run pm-engine --verbose
```

If you installed with **pipx** (end-user mode), the commands are on your PATH and need no prefix:

```
pm-engine --verbose
```

See the Quick Start's Install and Start for a first installation, and the Operator's Guide Installation chapter for every installation mode and the `pm-setup` bootstrap command.

2.3 Environment variables

One variable, `EDUMATCHER_DATA_DIR`, controls where EduMatcher finds and stores everything at runtime. Set it once in your shell profile (`~/.zshrc` or `~/.bashrc`) and every `pm-*` command picks it up automatically. A second family of variables, listed under Network variables below, moves the *network* endpoints; all of them are optional and the defaults keep everything on the loopback interface.

Variable	Default (installed)	Default (source checkout)	Purpose
`EDUMATCHER_DATA_DIR`	`~/.local/share/edumatcher`	`<repo>/src/data/`	Directory for all persistent data files (`gtc_orders.json`, `stats.db`, `audit.log`, etc.) **and** the deployed engine configuration

The engine configuration is **not** separately configurable. Every process reads `<EDUMATCHER_DATA_DIR>/ref_data/` — there is no `--config` flag and no `EDUMATCHER_CONFIG` variable. This is deliberate: while each process took its own path, starting one of them against the wrong file produced an exchange whose parts silently disagreed, and the symptom (a gateway reporting no instruments) pointed nowhere near the cause.

One data directory is therefore one exchange instance. Getting it wrong now detaches a process from its statistics and logs as well, so the mistake announces itself immediately.

```
# Example: per-session isolation
export EDUMATCHER_DATA_DIR="$HOME/sessions/morning"
pm-config-deploy ~/configs/morning.yaml
pm-engine --verbose
```

See `pm-config-deploy` for installing a configuration.

2.3.1 Network variables

These are read once, when a process starts, by `edumatcher/config.py` (the scheduler pair by `pm-scheduler` itself). Unset, every default keeps the core bus on `127.0.0.1`, so a plain install is not reachable from other machines. "Bind" variables say where a *server* listens; `EDUMATCHER_ENGINE_HOST` says where every *client* connects.

Variable	Default	Read by	Purpose
`EDUMATCHER_ENGINE_BIND_HOST`	`127.0.0.1`	`pm-engine`	Interface the engine binds `5555`, `5556` and `5557` on. Set to `0.0.0.0` (or a private address) to let processes on other hosts reach the bus
`EDUMATCHER_ENGINE_HOST`	`127.0.0.1`	every process that connects	Host to connect to for the engine ports and for the index, sim and health endpoints below. Also the default for `--alf-host` on the AI bots
`EDUMATCHER_INDEX_BIND_HOST`	`127.0.0.1`	`pm-index`	Interface for `5558`, `5559` and the index health socket

Variable	Default	Read by	Purpose
`EDUMATCHER_INDEX_PUB_PORT`	`5558`	`pm-index` and its clients	Index `PUB` port (`index.update`)
`EDUMATCHER_INDEX_PULL_PORT`	`5559`	`pm-index` and its clients	Index `PULL` port (operator commands, history requests)
`EDUMATCHER_INDEX_HEALTH_PORT`	`5564`	`pm-index`, `pm-index-cli health`	Health `REP` socket
`EDUMATCHER_SIM_BIND_HOST`	`127.0.0.1`	`pm-market-sim`	Interface for the sim ports. Loopback by default on purpose: nothing a student's client can reach carries a true value
`EDUMATCHER_SIM_PUB_PORT`	`5553`	`pm-market-sim` and its clients	`sim.value`, `sim.state`, `news.event`
`EDUMATCHER_SIM_PULL_PORT`	`5554`	`pm-market-sim` and its clients	`sim.command` from ADMIN participants
`EDUMATCHER_STATS_BIND_HOST` / `EDUMATCHER_STATS_HEALTH_PORT`	`127.0.0.1` / `5561`	`pm-stats`, `pm-stats-cli health`	Health `REP` socket
`EDUMATCHER_AUDIT_BIND_HOST` / `EDUMATCHER_AUDIT_HEALTH_PORT`	`127.0.0.1` / `5562`	`pm-audit`, `pm-audit-cli health`	Health `REP` socket
`EDUMATCHER_CLEARING_BIND_HOST` / `EDUMATCHER_CLEARING_HEALTH_PORT`	`127.0.0.1` / `5563`	`pm-clearing`, `pm-clearing-cli health`	Health `REP` socket
`EDUMATCHER_GATEWAY_BIND_HOST`	unset	`pm-alf-gwy`, `pm-balf-gwy`, `pm-md-gwy`, `pm-ralf-gwy`, `pm-dc-gwy`, `pm-log-srv`, `pm-api-gwy`, `pm-monitor`	Overrides the bind address of every service-layer listener at once. Precedence: an explicit `--host` flag, then this variable, then the section's `bind_address` in `engine_config.yaml`, then `0.0.0.0` (`127.0.0.1` for `pm-monitor`)
`EDUMATCHER_SCHEDULER_SPEED`	unset	`pm-scheduler`	Default for `--speed`: run the `--daily` clock F times faster than the wall clock

Variable	Default	Read by	Purpose
`EDUMATCHER_SCHEDULER_START`	unset	`pm-scheduler`	Default for `--start`: the session-local time (`YYYY-MM-DDTHH:MM`) the compressed clock starts at

Two planes, two ways to steer them

The **core plane** (`5553`–`5559`, and the health sockets) is steered only by environment variables, because those ports are fixed source constants or env-overridable and appear nowhere in `engine_config.yaml`. The **service layer** (the TCP gateways, `pm-log-srv`, `pm-monitor`) is steered by its own section of the YAML, and `EDUMATCHER_GATEWAY_BIND_HOST` is the single override on top. `pm-config-show` prints both, with the source of each value.

2.3.2 How the default is selected

The data directory is selected when the EduMatcher Python package is imported, not from the process's current working directory:

1. If `EDUMATCHER_DATA_DIR` is set, its expanded, absolute path wins — in a source checkout and in an installed deployment alike.
2. Otherwise EduMatcher checks where `edumatcher/config.py` is installed. If its package parent is named `src`, the process is treated as running from a source checkout and uses `<repo>/src/data/`.
3. Otherwise the package is treated as installed and uses `~/.local/share/edumatcher` (for example `/Users/<user>/.local/share/edumatcher` on macOS). This path is fixed; it does not follow `XDG_DATA_HOME`.
4. If the directory chosen above does not exist, but `./data` exists in the current working directory, that is used instead, with a warning on stderr. The chosen directory is used anyway, also with a warning, when neither exists.

So running an installed command from inside a repository does not turn it into a source checkout, and running a Poetry command from another directory does not move the source-checkout data location. When several processes must share one exchange, give them all the same explicit `EDUMATCHER_DATA_DIR`.

The authored YAML may live anywhere, but deployment always installs the compiled artifact, with a copy of its source, under the selected directory:

```
<DATA_DIR>/ref_data/engine_config.json
<DATA_DIR>/ref_data/engine_config.yaml
```

Relative runtime paths in the configuration, such as `data/stats.db`, are resolved under `<DATA_DIR>` too, so they name the same files whatever the command's working directory. Absolute paths remain explicit overrides.

A container sets the variable for you: `EDUMATCHER_DATA_DIR` is `/data` inside it, bind-mounted from `~/.edumatcher/data` (or `deployment/docker/data` in a checkout), so the databases and logs are ordinary files on your disk.

The container's data directory is deliberately **not** the installed default `~/.local/share/edumatcher`. Keeping it inside the install directory makes an install self-contained (one folder to back up or delete, separate data for each `--dir` install). It also stops a container and a `pipx` install from overwriting each other's deployed configuration and databases, and keeps the container's root-owned files (Docker on Linux) out of your own data directory. A host-side `pm-*` command therefore

needs `EDUMATCHER_DATA_DIR=~/.edumatcher/data` to read the container's data, or must be run from `~/.edumatcher`, where rule 4 finds `./data`.

Core processes:

Process	Command	Role	Required?
pm-engine	`pm-engine`	Matching engine — the single writer	Yes
pm-alf-console	`pm-alf-console --id GW01`	ALF order entry terminal (one per trader)	At least one
pm-scheduler	`pm-scheduler`	Drives session phase transitions	No
pm-viewer	`pm-viewer --symbol AAPL`	Live order book display	No
pm-orders	`pm-orders`	Cross-gateway order status monitor	No
pm-board	`pm-board`	Full-screen multi-symbol display	No
pm-ticker	`pm-ticker`	Scrolling market data ticker	No (needs pm-stats)
pm-stats	`pm-stats`	OHLCV, trade, and index-level statistics to SQLite	No — but recommended immediately after pm-engine (see below)
pm-clearing	`pm-clearing`	P&L and trade settlement	No
pm-audit	`pm-audit`	Full event log to disk	No — but recommended immediately after pm-engine (see below)
pm-alf-gwy	`pm-alf-gwy`	ALF TCP gateway — order entry for external bots over TCP :5565	No
pm-ralf-gwy	`pm-ralf-gwy`	External post-trade dissemination gateway (RALF)	No
pm-md-gwy	`pm-md-gwy`	External market-data gateway (CALF) over TCP :5570	No
pm-balf-gwy	`pm-balf-gwy`	External binary order-entry gateway (BALF) over TCP :5560	No
pm-dc-gwy	`pm-dc-gwy`	External drop-copy gateway (DC1) — relays :5557 fills over TCP :5590 for non-ZeroMQ clients	No
pm-api-gwy	`pm-api-gwy`	REST/WebSocket order-entry and market-data API gateway	No

Process	Command	Role	Required?
pm-index	`pm-index`	Real-time cap-weighted index calculation and dissemination	No
pm-log-srv	`pm-log-srv`	Centralized LALF log collector — logging over TCP :5600 to `log.db`	No

Monitoring & Admin tools:

Process	Command	Role	Required?
pm-admin	`pm-admin`	Interactive admin console	No
pm-admin-cli	`pm-admin-cli <command>`	One-shot CLI admin commands	No
pm-opctl-cli	`pm-opctl-cli <command>`	Start/stop/monitor named process profiles (recommended way to run the stack)	Recommended
pm-help / pm-man	`pm-help [pm-command]`	Command index and man-page reference for every pm-* command	No
pm-cverifier	`pm-cverifier [options] <config>`	Validate `engine_config.yaml` (YAML, schema, semantic, completeness checks)	No
pm-clearing-cli	`pm-clearing-cli <command> [options]`	Read/query interface for `clearing.db` (includes prune)	Optional
pm-eod	`pm-eod run | status | balances`	Close a trading day: settlement, statements, housekeeping	Optional
pm-stats-cli	`pm-stats-cli <command> [options]`	Read-only query interface for `stats.db`	Optional
pm-audit-cli	`pm-audit-cli <command> [options]`	Read-only query interface for audit JSONL log files	Optional
pm-audit-replay	`pm-audit-replay [options] <command>`	Read-only reconstruction of the audit trail into causal episodes and stories	Optional
pm-index-cli	`pm-index-cli <command> [options]`	Read-only query interface for index history JSONL files	Optional
pm-index-admin-cli	`pm-index-admin-cli --id <GW_ID> <command> [options]`	One-shot CLI for index corporate actions and constituent changes	Optional
pm-log-cli	`pm-log-cli <command> [options]`	Read-only query/troubleshooting interface for `log.db` (includes `diagnose`, `prune`)	Optional
pm-calf-spy	`pm-calf-spy [--channels CH] [--symbols SYM] [--ping-interval SEC] [--format human|json]`	Read-only CALF protocol spy — connects to `pm-md-gwy` and prints every line it sends	Optional

Process	Command	Role	Required?
pm-ralf-spy	`pm-ralf-spy --role ROLE [--channels CH] [--symbols SYM] [--ping-interval SEC] [--format human|json]`	Read-only RALF protocol spy — connects to `pm-ralf-gwy` and prints every line it sends	Optional
pm-dc-spy	`pm-dc-spy [--gateway GW_ID] [--replay-of ID] [--format human|json]`	Read-only drop-copy spy — connects to the engine's drop-copy `PUB` socket (:5557) and prints every fill event	Optional

2.4 Port Reference

Port numbers are scattered across the protocol, gateway, and configuration chapters (for example Protocols Overview and Examples). The table below is the single place that collects every default port for every long-running process in one view, split into the internal ZeroMQ bus (bound only on `127.0.0.1` by default, spoken only between EduMatcher's own processes) and the external TCP gateways (bound on `0.0.0.0` by default, spoken by outside clients). All ports are configurable in `engine_config.yaml`; the values below are the shipped defaults, cross-checked against `src/edumatcher/config.py` and each process's own `config.py`.

Core processes:

Process	Internal ZMQ port(s)	External port(s)	Protocol	Purpose
`pm-engine`	`5555` PULL (bind) · `5556` PUB (bind) · `5557` PUB (bind)	–	ZeroMQ (PULL/PUB)	Matching engine. `5555` receives all inbound commands; `5556` broadcasts all market/session events; `5557` is the dedicated drop-copy fill feed.

Process	Internal ZMQ port(s)	External port(s)	Protocol	Purpose
`pm-log-srv`	`5601` PUB (bind) · `5602` PULL (bind)	`5600`	LALF over TCP (collection) · LALF-PS over ZeroMQ (distribution)	Centralized log collector. `5600` accepts LALF log lines from every other `pm-*` process; `5601`/`5602` are the LALF-PS PUB/PULL pair that pushes live rows to log viewers/subscribers without polling `log.db`.
`pm-audit`	`5562` REP (bind); connects out to `5556`	–	ZeroMQ (SUB/REP)	Subscribes to every engine event and writes the full audit trail to disk. `5562` only answers `pm-audit-cli health` pings (`EDUMATCHER_AUDIT_BIND_HO` `EDUMATCHER_AUDIT_HEALTH_P`
`pm-stats`	`5561` REP (bind); connects out to `5555`, `5556`, `5558`	–	ZeroMQ (SUB/PUSH/REP)	Records market and index statistics to `stats.db`. `5561` only answers `pm-stats-cli health` pings (`EDUMATCHER_STATS_BIND_HO` `EDUMATCHER_STATS_HEALTH_P`
`pm-clearing`	`5563` REP (bind); connects out to `5556`	–	ZeroMQ (SUB/REP)	Subscribes to trade events for P&L and settlement. `5563` only answers `pm-clearing-cli health` pings (`EDUMATCHER_CLEARING_BIND` `EDUMATCHER_CLEARING_HEALT`

Process	Internal ZMQ port(s)	External port(s)	Protocol	Purpose
`pm-scheduler`	connects out to `5555`	–	ZeroMQ (PUSH)	Drives session-phase transitions by pushing commands to the engine. Binds no port of its own.
`pm-md-gwy`	connects out to `5556`, `5558`	`5570`	CALF over TCP	External market-data gateway. Consumes engine and index PUB feeds internally and republishes top/book/trade/state channels to external CALF clients on `5570`.
`pm-ralf-gwy`	connects out to `5556`	`5580`	RALF over TCP	External post-trade dissemination gateway for clearing, drop-copy, and audit consumers (role-gated).
`pm-dc-gwy`	connects out to `5557`	`5590`	DC1 over TCP	Relays the engine's `5557` drop-copy feed as DC1 text lines to plain-TCP clients that don't speak ZeroMQ.
`pm-api-gwy`	connects out to `5555`, `5556`	`8080` (code default; example configs may assign additional instances other ports, e.g. `8081`)	HTTP/REST + WebSocket	REST/WebSocket gateway; translates HTTP/WS requests into the same ZMQ order flow used by `pm-alf-console`. Each named `api_gateways` instance in config gets its own port.

Process	Internal ZMQ port(s)	External port(s)	Protocol	Purpose
`pm-alf-gwy`	connects out to `5555`, `5556`, `5557`	`5565`	ALF over TCP	External ALF order-entry gateway for bots/remote clients (same protocol as `pm-alf-console`, over TCP instead of stdin/stdout).
`pm-balf-gwy`	connects out to `5555`, `5556`	`5560`	BALF (binary) over TCP	External binary order-entry gateway for low-latency programmatic clients.
`pm-index`	`5558` PUB (bind) · `5559` PULL (bind) · `5564` REP (bind); connects out to `5556`	–	ZeroMQ (PUB/PULL/REP)	Real-time cap-weighted index calculation. `5558` broadcasts `index.update`; `5559` receives operator commands and history requests; `5564` only answers `pm-index-cli health` pings (`EDUMATCHER_INDEX_BIND_HO` `EDUMATCHER_INDEX_HEALTH_P` No external protocol of its own — external consumers reach index data via `pm-md-gwy`/CALF.

Process	Internal ZMQ port(s)	External port(s)	Protocol	Purpose
`pm-monitor`	`5610` PULL (bind) · `5611` PUB (bind); connects out to `5556`	–	ZeroMQ (PULL/PUB/SUB)	Health collector. `5610` receives the `health.beacon` every long-running process pushes every 2 s; `5611` publishes `health.snapshot` (1 Hz) and `health.event` to `pm-monitor-cli` and the Operator Dashboard. Loopback by default (`monitor.bind_address`, `EDUMATCHER_GATEWAY_BIND_HOST`). See Health Monitoring.
`pm-market-sim`	`5553` PUB (bind) · `5554` PULL (bind); connects out to `5555`, `5556`	–	ZeroMQ (PUB/PULL)	The market model: every symbol's true value, and news. `5553` publishes `sim.value`, the `sim.state` heartbeat and `news.event`; `5554` takes `sim.command` from ADMIN participants. Loopback only by default (`EDUMATCHER_SIM_BIND_HOST`). See The Market Model.

Besides the connections listed, every process in the table (and the bots `pm-mm-bot` and `pm-ai-swarm`) connects a ZeroMQ PUSH socket to `pm-monitor` on `5610` (`monitor.host/pull_port`) and sends its health beacon there every two seconds. The send never blocks: when `pm-monitor` is not running the beacons are dropped and counted.

Internal-only utilities (ZeroMQ clients; no listening port of their own):

Process	Internal ZMQ port(s) used	Protocol	Purpose
`pm-alf-console`	connects out to `5555`, `5556`	ZeroMQ (PUSH/SUB)	Interactive ALF order-entry terminal for a human trader; the same PUSH/SUB pattern as a gateway, without a TCP front end.
`pm-viewer`	connects out to `5556`	ZeroMQ (SUB)	Live single/multi-symbol order book display.
`pm-board`	connects out to `5555`, `5556`	ZeroMQ (PUSH/SUB)	Full-screen multi-symbol market board.
`pm-ticker`	connects out to `5556`, `5553`	ZeroMQ (SUB)	Scrolling market-data ticker (reads recent trades from `pm-stats`'s `stats.db` as well); headlines from `pm-market-sim`.

Reading the table

"Internal ZMQ port(s)" are ZeroMQ endpoints intended for EduMatcher's own processes on the same host (or a trusted internal network) — by default bound to `127.0.0.1`. "External port(s)" are TCP/HTTP listeners meant for outside clients — by default bound to `0.0.0.0`. Every port shown is a default; production or multi-host deployments should bind internal ports to a private interface and firewall external ports as appropriate. See The Configuration Workflow for how to override any of these.

The same table, for the config you actually deployed

This table lists the shipped defaults. To see the ports *one particular configuration* resolves to — including which were set explicitly and which fell back to a default — run `pm-config-show`. It derives its port map from `edumatcher/gateway_ports.py`, the same table `pm-cverifier` uses for its `M018` collision check.

Web applications (`web-apps/`; each a Node bridge serving a browser app, none part of `engine_config.yaml`). The container port is what the stacks publish (`deployment/docker make up-all`, `deployment/curl`); `make dev` uses the web and bridge ports. They follow one scheme: `809N`, `819N`, `519N`.

Application	Container port	Dev web / bridge	Talks to	Guide
TapeDeck (`pm-terminal`)	`8090`	`8190` / `5190`	`pm-md-gwy 5570`, `pm-api-gwy 8081`	TapeDeck
Log Operator Console (`pm-log-ui`)	`8091`	`8191` / `5191`	`pm-log-srv 5601/5602`, `log.db` read-only	Log Operator Console

Configuration GUI (`config-gui`)	`8092`	`8192` / `5192`	nothing	Configuration GUI
Trading GUI (`pm-trading-ui`)	`8093`	`8193` / –	`pm-api-gwy 8080`	Trading Platform GUI
Order Book Viewer (`pm-book`)	`8094`	`8194` / `5194`	`pm-api-gwy 8081`	Order Book Viewer
Operator Dashboard (`pm-dash`)	`8095`	`8195` / `5195`	`pm-monitor 5611`, `metrics.db` read-only	Operator Dashboard

Setup and configuration tools:

Process	Command	Role	Required?
pm-setup	`pm-setup`	Bootstrap working directory and runtime files	Recommended first run
pm-config-gen	`pm-config-gen [options]`	Generate `engine_config.yaml` from CLI options	Optional
pm-cverifier	`pm-cverifier CONFIG_FILE`	Validate a config across four layers	Recommended
pm-config-deploy	`pm-config-deploy SOURCE`	Compile and install the config every process reads	Required to run
pm-config-show	`pm-config-show [options]`	Display the deployed config, or render it as a PDF	Optional
pm-new-symbol / pm-ipo	`pm-new-symbol --symbol SYM ...`	List a new symbol (an IPO) in the configuration and redeploy it	Optional
pm-valuation	`pm-valuation [options]`	Interview-driven DCF valuation and IPO simulation for a fictive company	Optional

Optional AI trader tools:

Process	Command	Role	Required?
pm-ai-trader	`pm-ai-trader`	Single AI trading bot gateway	No
pm-ai-swarm	`pm-ai-swarm`	Coordinated multi-agent AI trading swarm	No
pm-market-sim	`pm-market-sim`	Market model: true values and news for the AI traders	No
pm-news	`pm-news --id ADMIN_ID …`	Instructor's news desk for `pm-market-sim`	No
pm-mm-bot	`pm-mm-bot --symbol SYM`	Autonomous market-maker bot for a single symbol	No

Start the engine first

The engine binds the ZeroMQ sockets. All other processes connect to those sockets. If a process starts before the engine is ready, it will either fail immediately or silently lose its first messages.

Recommended startup sequence

For a deterministic first run, use this order:

1. Bootstrap session files: `bash pm-setup`
2. Generate a config (or reuse/edit the sample config): `bash pm-config-gen --symbols AAPL MSFT --participants TRADER01 TRADER02 OPS01:ADMIN --sessions-enabled --output engine_config.yaml`
3. Use the operational control `pm-opctl-cli` to start the system in the correct order with: `pm-opctl-cli start` Then use `pm-opctl-cli list` to list the status of each process, and `pm-opctl-cli stop` to shut the profile down again. See Running the Exchange.
4. Start remaining optional observers/tools as needed (`pm-board`, `pm-viewer`, `pm-ticker`).

2.5 pm-engine — Matching Engine

The heart of the system — receives orders, matches them, publishes events.

```
pm-engine [-v|-vv] [--log-level LEVEL] [-q] [--log-target TARGET] [--log-file PATH] [--log-failover-timeout S
```

Startup options:

Flag	Default	Description
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase log verbosity (`-v` → `INFO`: startup/lifecycle messages; `-vv` → `DEBUG`: per-order/per-trade detail)
`-q` / `--quiet`	off	Explicit `WARNING` (same as the default)
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	30 (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

See Running the Engine → Logging levels for the full flag reference and example output.

Expected runtime input arguments:

None. `pm-engine` is a long-running background process after startup.

Startup behaviour: 1. Reads the deployed `<DATA_DIR>/ref_data/engine_config.json` — registers the symbol and gateway allowlists, session settings, collars, circuit breakers and tuning 2. Binds the main ZMQ PULL :5555 and PUB :5556 sockets during engine initialisation 3. On `run()`, bumps the run sequence (`engine_run_seq.json`) so trade ids are unique across restarts 4. Publishes `system.startup` first: engine version, git commit, the SHA-256 and full text of the configuration, and the persisted files it is about to load. An audit reader takes it as the start of a run 5. Restores persisted resting orders (`gtc_orders.json`), combos (`gtc_combos.json`) and book statistics (`book_stats.json`). Both `GTC` and `DAY` orders are restored, in each book's saved priority order;

a `DAY` order whose business day has passed is discarded. Publishes `system.startup_recovery` with the counts 6. Applies config-driven seeds (book-statistics gaps, market-maker quotes, market-maker combos) 7. Publishes what restoring left in the books' trace, the collar bands (`risk.band.<SYM>`) and a `system.book_checkpoint.<SYM>` for every book (reason `STARTUP`) 8. Binds the dedicated drop-copy PUB :5557 socket (if that fails, it logs a warning and runs without drop copy), then enters the poll loop

The loop waits up to 200 ms for a message, handles it completely, then runs a maintenance tick (snapshots, timers, persistence checkpoint every 5 s). Every decoded inbound *command* (not the `*_request` queries) is re-published once, unchanged, so that `pm-audit` records what the exchange was *asked* to do, not just what it did, including a cancel the engine ignored.

Shutdown (Ctrl-C / SIGTERM): 1. Serialises every resting `GTC` **and** `DAY` order to `<DATA_DIR>/gtc_orders.json`. A process exit is not a day boundary, so `DAY` orders are *not* expired here; real end-of-day expiry happens when the scheduler moves the session to `CLOSED` 2. Saves GTC combos (`gtc_combos.json`) and book statistics (`book_stats.json`) 3. Publishes a `system.book_checkpoint.<SYM>` for every book (reason `EOD`) 4. Publishes `system.eod` with the final book snapshot for every symbol, so subscribers record closing prices 5. Closes the sockets

A message that cannot be decoded at all is discarded and counted rather than ending the loop, so a malformed packet cannot skip the shutdown persistence.

Messages received (PULL :5555). The engine dispatches 38 inbound topics; a topic it does not recognise is dropped, counted and answered with a `system.diagnostic` (`UNROUTED_TOPIC`). A handler that raises is answered with a `system.diagnostic` (`DISPATCH_ERROR`) and a reject to the submitting gateway.

Group	Topics	Source
Orders	`order.new`, `order.cancel`, `order.amend`, `order.oco`, `order.oco_cancel`, `order.combo`, `order.combo_cancel`	Gateways
Quotes	`quote.new`, `quote.cancel`	Market-maker gateways
Session of a gateway	`system.gateway_connect`, `system.gateway_disconnect`, `system.gateway_heartbeat`	Gateways
Queries	`system.symbols_request`, `system.reference_request`, `book.snapshot_request`, `order.orders_request`, `order.price_level_orders_request`, `system.quote_bootstrap_request`, `system.quote_legs_request`, `system.position_request`	Gateways, viewers, `pm-stats`
Admin queries	`system.session_state_request`, `system.session_schedule_request`, `system.gateways_request`, `system.volume_request`, `system.halt_status_request`, `system.risk_state_request`	`pm-admin`, `pm-admin-cli`, `pm-api-gwy`

Group	Topics	Source
Risk controls	`risk.kill_switch`, `risk.kill_switch_gateway`, `risk.kill_switch_global`, `risk.circuit_breaker_halt_all`, `risk.circuit_breaker_resume_all`, `risk.symbol_halt`, `risk.symbol_resume`, `risk.cancel_symbol`, `risk.force_uncross`	Admin gateways
Session control	`session.transition`, `session.next`	Scheduler, admin
Reference data	`system.reference_reload`	`pm-api-gwy`

Messages published (PUB :5556). Per-gateway topics end in `.{GW_ID}`; the payload of every topic is defined in the Message Reference.

Group	Topics	Purpose
Order events	`order.ack`, `order.fill`, `order.cancelled`, `order.expired`, `order.amended`, `order.triggered`, `order.orders`, `order.price_level_orders` (all `.{GW_ID}`)	Private to the owning gateway. `order.fill` carries one VWAP fill per order per dispatch; `order.triggered` reports a stop that fired
Combo, OCO, quote	`combo.ack/status/cancel_ack`, `oco.ack/cancelled/cancel_ack`, `quote.ack/status/cancel_ack` (all `.{GW_ID}`)	Lifecycle of structured orders
Market data	`trade.executed`, `book.{SYMBOL}`, `depth.{SYMBOL}`, `auction.indicative.{SYMBOL}`, `auction.result.{SYMBOL}`	Public tape and book. Snapshots are throttled to `snapshot_interval_sec`
Session and risk	`session.state`, `circuit_breaker.halt/extend/resume.{SYMBOL}`, `risk.band.{SYMBOL}`, acknowledgements `risk.*_ack.{GW_ID}`, `session.transition_ack.{GW_ID}`	Phase changes, halts, collars, amend acknowledgements
Admin	`admin.action.{GW_ID}`	What one admin command acted on

Group	Topics	Purpose
System replies	`system.gateway_auth`, `system.gateway_bye`, `system.symbols`, `system.reference`, `system.gateways`, `system.volume`, `system.halt_status`, `system.risk_state`, `system.session_status`, `system.session_schedule`, `system.position_snapshot`, `system.quote_bootstrap`, `system.quote_legs`, `system.reference_reload_ack` (all `.{GW_ID}`)	Replies to the queries above
Lifecycle and audit	`system.startup`, `system.startup_recovery`, `system.recovery_item`, `system.book_checkpoint.{SYMBOL}`, `engine.trace.{SYMBOL}`, `system.diagnostic`, `system.eod`	Run start and recovery, book checkpoints, queue-position trace, engine faults, end of day

Messages published (drop-copy PUB :5557):

Topic	Purpose
`drop_copy.event.{GW_ID}`	Fill event with sequence number and nanosecond timestamp, filtered per gateway
`drop_copy.replay.{RECIPIENT_ID}`	Replay of buffered events to a consumer that asks for them

The drop-copy socket is bound after startup (see step 8). See Drop-Copy Feed for the subscription protocol and replay support.

Warning

Start the engine **first** — gateways and subscribers will fail to connect otherwise.

2.6 pm-alf-console — User Gateway

One instance per user. Accepts ALF commands on stdin. See ALF Protocol Reference.

```
pm-alf-console --id <GW_ID>
```

Startup options:

Flag	Required	Description
`--id`	Yes	Unique gateway identifier (e.g. `GW01`, `ALICE`)
`--drop-copy`	No	Enable the drop-copy relay on startup (equivalent to sending `DC|STATE=ON` immediately after connecting); default off — see ALF Console → DC
`--log-level`	No	Explicit level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	No	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	No	Reduce output to warnings/errors
`--log-target`	No	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file` (default `server`)
`--log-file`	No	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	No	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable (default `30` from config)
`--version`	No	Print version and exit

Expected runtime input arguments:

- `NEW|...` order-entry commands (including `LIMIT`, `MARKET`, `STOP`, `STOP_LIMIT`, `FOK`, `ICEBERG`, `IOC`, `TRAILING_STOP`)
- `AMEND|...`, `CANCEL|...`
- `QUOTE|...`, `QUOTE_CANCEL|...`
- `QLEGS|...` (market-maker quote-leg monitor with fill flags)
- `ORDERS`, `SYMBOLS`, `HELP`, `EXIT`, `QUIT`
- `KILL|...` for gateway-scoped kill-switch actions

Connection behaviour: 1. Sends `system.gateway_connect` to the engine 2. Waits for `system.gateway_auth.<G` 3. Enters command loop only if accepted

If the ID is not listed in `engine_config.yaml` under `participants`, connection is refused and the gateway exits.

Messages sent (PUSH → :5555):

Topic	Purpose
`system.gateway_connect`	Authentication request
`order.new`	Submit order
`order.cancel`	Cancel order
`order.combo`	Submit combo
`order.combo_cancel`	Cancel combo
`order.orders_request`	Request order list
`system.symbols_request`	Request symbol list

Messages subscribed (SUB from :5556):

Topic	Purpose
`system.gateway_auth.{own GW_ID}`	Authentication reply
`order.ack.{own GW_ID}`	Order acknowledgements
`order.fill.{own GW_ID}`	Fill notifications
`order.amended.{own GW_ID}`	Successful amend notifications
`order.cancelled.{own GW_ID}`	Cancel confirmations
`order.expired.{own GW_ID}`	Expiry notifications
`order.orders.{own GW_ID}`	Order list replies
`combo.ack.{own GW_ID}`	Combo acknowledgements
`combo.status.{own GW_ID}`	Combo status updates
`oco.ack.{own GW_ID}`	OCO creation acknowledgements
`oco.cancelled.{own GW_ID}`	OCO sibling-cancel notifications
`quote.ack.{own GW_ID}`	Quote acknowledgements
`quote.status.{own GW_ID}`	Quote lifecycle updates
`risk.kill_switch_ack.{own GW_ID}`	Kill-switch acknowledgement
`system.symbols.{own GW_ID}`	Symbol list reply
`trade.executed`	Global trade feed for last-price / P&L display

See the ALF Console for the full command list.

2.7 pm-alf-gwy — ALF TCP Gateway

Accepts ALF order-entry commands from external bots and remote processes over a plain TCP connection. Uses the same ALF command vocabulary as `pm-alf-console` but is designed for programmatic clients, not interactive terminals. One connection per gateway ID; all configured `participants` IDs may connect.

```
pm-alf-gwy [--bind 0.0.0.0] [--port 5565] [--engine-host HOST] [--log-level LEVEL] [-v|-vv] [-q]
```

Startup options:

Flag	Default	Description
`--bind`	from config / `0.0.0.0`	TCP bind address for external clients
`--port`	from config / `5565`	TCP listen port for ALF clients
`--engine-host`	from config	Override engine host for ZMQ ports `5555` / `5556`
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`

Flag	Default	Description
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	30 (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

No terminal input. External clients connect over TCP and send ALF lines:

- `HELLO|CLIENT=...|PROTO=ALF1|ID=<gateway-id>` (must be first)
- `NEW|...`, `AMEND|...`, `CANCEL|...`
- `QUOTE|...`, `QUOTE_CANCEL|...`
- `KILL[|SYM=...]`
- `SYMBOLS`, `ORDERS`, `QBOOT[|SYM=...]`
- `PING`, `EXIT` / `QUIT`

Messages sent (PUSH → :5555):

Topic	Purpose
`system.gateway_connect`	Authentication request
`system.gateway_disconnect`	Graceful disconnect notice
`order.new`	Submit order
`order.cancel`	Cancel order
`order.amend`	Amend order
`order.combo`	Submit combo
`order.combo_cancel`	Cancel combo
`order.oco`	Submit OCO pair
`order.oco_cancel`	Cancel OCO pair
`order.orders_request`	Request order list
`quote.new`	Submit / replace MM quote
`quote.cancel`	Cancel MM quote
`risk.kill_switch`	Gateway kill-switch
`system.symbols_request`	Request symbol list
`system.quote_bootstrap_request`	Request quote bootstrap state

Messages subscribed (SUB from :5556):

Topic	Purpose
`system.gateway_auth.{GW_ID}`	Engine authentication reply
`order.ack.{GW_ID}`	Order accepted or rejected

Topic	Purpose
`order.fill.{GW_ID}`	Partial or full fill
`order.amended.{GW_ID}`	Amend confirmation
`order.cancelled.{GW_ID}`	Cancel confirmation
`order.expired.{GW_ID}`	TIF expiry
`order.orders.{GW_ID}`	Order list reply
`combo.ack.{GW_ID}`	Combo accepted or rejected
`combo.status.{GW_ID}`	Combo lifecycle change
`oco.ack.{GW_ID}`	OCO accepted or rejected
`oco.cancelled.{GW_ID}`	OCO sibling-cancel notification
`quote.ack.{GW_ID}`	Quote accepted or rejected
`quote.status.{GW_ID}`	Quote lifecycle change
`risk.kill_switch_ack.{GW_ID}`	Kill-switch acknowledgement
`system.symbols.{GW_ID}`	Symbol list reply
`system.quote_bootstrap.{GW_ID}`	Quote bootstrap state reply
`session.state`	Session phase changes (broadcast)
`trade.executed`	Trade events (broadcast)
`circuit_breaker.halt.*`	Symbol halt events (broadcast)
`circuit_breaker.resume.*`	Symbol resume events (broadcast)

See ALF TCP Gateway for operational usage, command reference, and client examples.

2.8 pm-viewer — Order Book Viewer

Live terminal view of a single symbol's order book. Draws a full-screen bordered box on the alternate terminal buffer that repaints cleanly on resize.

```
pm-viewer --symbol AAPL [--depth N] [--db data/stats.db] \
  [--text-color COLOR] [--zebra-lines] [--zebra-lines-color COLOR]
```

Startup options:

Flag	Default	Description
`--symbol` / `-s`	required	Symbol to watch
`--depth` / `-d`	fit to terminal	Max price levels to display per side (default: fills available height)
`--db`	`data/stats.db`	Stats SQLite DB — seeds session OHLC and previous-close at startup
`--text-color`	`white`	Body text color for the order book tables — `#rrggbb` hex or a Rich color name
`--zebra-lines`	off	Shade every second row of the Bids/Asks/Trades tables with a background tint

Flag	Default	Description
`--zebra-lines-color`	`grey19`	Background tint used by `--zebra-lines` — `#rrggbb` hex or a Rich color name; has no effect unless `--zebra-lines` is given
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	30 (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

Key	Action
`s` or `F1`	Open the symbol picker and switch to another symbol
`Ctrl-C`	Quit

Inside the picker: type to narrow the list by prefix, ↑/↓ to move, `Enter` to switch, `Esc` to go back to the book unchanged. Both keys are advertised in the footer that crosses the bottom border.

Display: - Two-line header: last price with directional colour/arrow, change / %, session OHLC seeded from `pm-stats` DB, previous-close, bid/ask, spread, volume, and a live clock - Three equal-height columns filling the terminal: **Bids** (with proportional depth bars), **Asks**, and **Trades** (newest first, uptick/downtick coloured) - Bids and asks show proportional depth micro-bars per price level - The frame border colour reflects the day's direction (green/red) - Repaints the full frame on every terminal resize — no leftover garbage

Iceberg orders

Iceberg orders only show their `displayed_qty` (the visible peak) in the viewer. The hidden quantity is completely invisible — this is by design and demonstrates the privacy feature of iceberg orders.

Colors on a dark terminal

Body text in the Bids/Asks/Trades tables defaults to `white`; override it with `--text-color` if that doesn't fit your terminal's theme (e.g. a light-background profile). Alternating-row shading is off by default and opt-in via `--zebra-lines`, which tints every second row with a background color (default `grey19`, a dark grey) without changing that row's text color — customize it with `--zebra-lines-color`. Both color options accept a `#rrggbb` hex code or any Rich color name.

Switching symbol without restarting

Press `s` (or `F1`) to pick a different symbol from a popup list. The viewer asks the engine for the current symbol universe each time the picker opens, resubscribes to the chosen symbol's book, and reloads that symbol's session statistics — so one viewer can follow a whole class around the

market. The picker needs a real terminal; when stdin is piped or redirected the viewer behaves exactly as it always did.
Run multiple viewers simultaneously for different symbols:

```
pm-viewer --symbol AAPL &
pm-viewer --symbol MSFT &
pm-viewer --symbol TSLA
```

Messages subscribed (SUB from :5556):

Topic	Purpose
`book.{SYMBOL}`	Book updates for the watched symbol
`system.symbols.{GW_ID}`	Symbol list backing the interactive picker
`session.state`	Session phase changes (displayed in header)

Messages sent (PUSH → :5555):

Topic	Purpose
`book.snapshot_request`	Requests initial book state on startup, and again after a symbol switch
`system.symbols_request`	Requests the symbol list when the picker is opened

2.9 pm-orders — Order Status Monitor

Live cross-gateway view of all orders in the system.

```
pm-orders [--gateway GW01] [--log-level LEVEL] [-v|-vv] [-q]
```

Startup options:

Flag	Default	Description
`--gateway` / `-g`	(all)	Filter to a single gateway
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

None.

Displays a live table with columns: `ID | Gateway | Symbol | Side | Type | TIF | Qty | Remaining | Price | Status | Updated`

Status colours: green=NEW, yellow=PARTIAL, bright green=FILLED, red=REJECTED/CANCELLED, dim=EXPIRED.

Messages subscribed (SUB from :5556):

Topic	Purpose
`order.` (prefix)	All order events (ack, fill, cancelled, expired)
`combo.` (prefix)	Combo ack and status events
`session.state`	Session phase changes

2.10 pm-audit - Event Logger

Records every message on the bus to a day-closing log file; closed days are compressed to one zstd archive each.

```
pm-audit [--audit-log-file data/audit.log] [--terminal] [--buffer-size 100] [--flush-interval 10] [--max-file
```

Startup options:

Flag	Default	Description
`--audit-log-file`	`data/audit.log`	Audit-trail output log file path
`--terminal` / `-t`	off	Also print each entry to stdout
`--buffer-size`	100	Number of messages to buffer in memory before writing to disk
`--flush-interval`	10.0	Maximum seconds to wait before flushing buffer to disk
`--max-file-mb`	10	Close the audit log as a day part when it reaches this size in MB
`--keep-days`	0	Delete daily archives older than N days (0 keeps all)
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	30 (from config)	Seconds to wait before falling back to a local log file if `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

None.

Buffering behaviour:

To reduce disk wear, `pm-audit` buffers messages in memory before writing them to disk in batches. The buffer is flushed to disk when either:

- The buffer reaches `--buffer-size` messages (e.g., 100 messages), or
- `--flush-interval` seconds have elapsed since the last flush (e.g., 10 seconds)

On shutdown (Ctrl-C, SIGINT, or SIGTERM), any remaining buffered messages are automatically flushed to disk before the process exits, ensuring no data is lost.

Log format (one entry per line):

```
[2026-04-29T14:32:01.123] [trade.executed] {"id": "...", "symbol": "AAPL", ...}
[2026-04-29T14:32:01.125] [book.AAPL] {"bids": [...], "asks": [...], ...}
```

Log files rotate at 10 MB with 5 backups kept.

Use `--terminal` during demos so the class can see every event in real time. Use `-vv` or `--log-level DEBUG` when you want aggregated operational summaries about buffering, flushes, decode errors, and broad topic-family mix without changing the raw event log format.

Examples:

```
# Default: buffer 100 messages, flush every 10 seconds
pm-audit

# High-traffic scenario: larger buffer, longer flush interval
pm-audit --buffer-size 500 --flush-interval 30

# Low-latency: smaller buffer, aggressive flushing
pm-audit --buffer-size 10 --flush-interval 1

# Disable buffering entirely: flush every message immediately
pm-audit --buffer-size 1 --flush-interval 0.1
```

Messages subscribed (SUB from :5556):

Topic	Purpose
(empty prefix — receives all messages)	Records everything published on the PUB socket

2.11 pm-audit-cli — Audit Log Query CLI

`pm-audit-cli` is a read-only offline tool: it reads JSONL files directly from disk, runs one query, prints output, and exits. No ZeroMQ connection required — except for `health`, which pings the running `pm-audit` on port `5562`.

```
pm-audit-cli [--log-file data/audit.log] [--log-dir PATH] [--format table|json|csv] [--no-header] [--use-index PATH]
```

Global options:

Flag	Default	Description
`--log-file PATH`	`data/audit.log`	Primary audit log file; reads the live trail (unarchived day parts, then this file). An archive or day part named here is read alone
`--log-dir PATH`	directory of `--log-file`	Directory holding the `audit/` day parts and archives
`--format`	`table`	Output format: `table`, `json`, or `csv`
`--no-header`	off	Suppress header row in CSV output
`--use-index PATH`	auto-detected	Path to SQLite index file (auto-detected from `--log-dir` when not specified; ignored for commands that don't support it)
`--version`	—	Print version and exit

Subcommands:

Subcommand	Default limit	Purpose	Key filters
`events`	100	Search log entries by topic, gateway, symbol, and time range	`--topic`, `--gateway`, `--symbol`, `--date`, `--from`, `--to`, `--reverse`
`orders`	100	Order lifecycle events for specific order IDs or filters	`--id` (repeatable), `--gateway`, `--symbol`, `--date`, `--from`, `--to`
`trades`	100	Trade executions	`--symbol`, `--gateway`, `--buy-gateway`, `--sell-gateway`, `--min-price`, `--max-price`, `--min-qty`, `--date`, `--from`, `--to`, `--reverse`
`topics`	—	List topics present in logs with event counts	`--date`, `--from`, `--to`, `--prefix`, `--sort` (count|alpha)
`gateways`	—	Gateway activity summary	`--date`, `--from`, `--to`, `--min-events`
`timeline`	500	Raw chronological event stream for session replay	`--from`, `--to`, `--topic`, `--gateway`, `--symbol`

Subcommand	Default limit	Purpose	Key filters
`stats`	—	Summary statistics about audit log files	`--verbose`
`index`	—	Build or update the optional SQLite index for faster queries	`--output PATH`
`archive`	—	Compress closed days' parts into one verified `audit-D.log.zst` per day	`--date D` or `--pending`, `--level N` (default 9), `--keep-days N`
`health`	—	Check the `pm-audit` process and ping its health socket; `--extended` adds the audit log read/write and last-entry check	`--extended`, `-v`/`--verbose`, `-q`/`--quiet` (print nothing; exit `0` when healthy, `-1` otherwise)

No-row behaviour:

- `table`: prints `No rows found.`
- `json`: prints `[]`
- `csv`: prints only header row unless `--no-header` is set

2.12 pm-audit-replay — Audit Replay

Reconstructs the causal structure of the audit trail and narrates it: which order caused which trade, which kill switch cancelled what, where the log disagrees with itself. It reads the same JSONL files as `pm-audit-cli`, read-only, and keeps a derived *episode index* in its own SQLite file. It joins no ZeroMQ socket.

```
pm-audit-replay [global options] COMMAND [command options]
pm-audit-replay --last 15m stream
pm-audit-replay story --order 7f3a2c
pm-audit-replay anomalies --severity warn
```

Commands:

Command	Purpose	Command options
`stream`	Narrate a window chronologically (the default view)	—

Command	Purpose	Command options
`story`	One entity and everything causally connected to it (needs the index)	exactly one selector: `--order`, `--trade`, `--quote`, `--oco`, `--combo`, `--command`, `--client-tag`, `--chain ULID`, `--msg ULID`; `--depth N` (default 2; not with `--chain`), `--strict-causality`
`digest`	One paragraph per episode, most significant first	`--top N`, `--significance {anomalies,notional,qty,duration}`
`episodes`	The index as a table, one row per episode	`--outcome OUTCOME` (repeatable)
`anomalies`	Every finding, worst first, with the `story` command to run next	`--severity {error,warn,info}` (a floor; default `info`)
`stats`	Link confidence and envelope coverage over the window	—
`index`	Build or refresh the episode index	`--stats`

Global options (accepted before or after the command):

Flag	Default	Description
`--log-file PATH`	`<DATA_DIR>/audit.log`	Audit log to read: the live trail (unarchived day parts, then this file). An archive or day part named here is read alone
`--db PATH`	`<DATA_DIR>/audit_replay.db`	Episode index
`--no-index`	off	Stream without building or reading an index
`--rebuild`	off	Rebuild the index before rendering. There is no incremental mode; the index rebuilds itself whenever the reconstruction rules, the detection settings or the source logs change
`--from ISO_TS`, `--to ISO_TS`	—	Window bounds (ISO-8601 or `YYYY-MM-DD`)
`--date YYYY-MM-DD`	—	One trading day, in the session timezone. Cannot be combined with `--from`/`--to` or with an archive named by `--log-file`

Flag	Default	Description
`--last DURATION`	—	Relative window, e.g. `15m`, `2h`, `1d`. Cannot be combined with `--from`, `--to` or `--date`
`--symbol`, `--gateway`, `--kind`	—	Restrict to symbols, gateways or episode kinds; each repeatable
`-q` / `-v` / `-vv` / `-vvv`	level 1	Detail level: `-q` is level 0 (episode outcomes only); each `-v` adds one level, up to `-vvv`
`--format {text,ndjson,json,markdown,csv}`	`text`	Output format (a command that cannot produce a format exits `2`)
`--show-source`	off	Append `audit.log:LINE` to every narrated line
`--show-units`	off	Append unit provenance to every price
`--explain`	off	Show link evidence and confidence inline
`--id-len N|full`	`6`	Order-id abbreviation
`--actor-style {id,descriptive}`	`id`	Actor naming
`--tz TZ`	`UTC`	Zone for rendered timestamps
`--reorder-window SPEC`	`2000/5s`	Reorder buffer, a message count or a duration
`--no-color` / `--force-color`	auto	Never / always emit ANSI colour (`--force-color` for `... | less -R`)
`--strict`	off	Also report findings that are expected at a window edge (`ARRIVAL_SEQ_GAP`)
`--clock-skew-warn SECONDS`	`0.1`	Engine-versus-receipt clock difference worth reporting
`--version`	—	Print version and exit

Exit codes: `0` success; `1` nothing to narrate, or no audit log at the given path; `2` argument error.

See Audit Replay for how causality is recovered, what each anomaly means and worked examples.

2.13 pm-clearing — Clearing & P&L

SQLite-backed clearing writer for P&L and position state.

```
pm-clearing [--datapath PATH] [--db-name NAME] [--flush-size N] [--flush-interval SEC] [--print-every N] [--retentio
  [--log-target TARGET] [--log-file PATH] [--log-failover-timeout SEC]
```

Startup options:

Flag	Default	Description
`--datapath`	data dir from `EDUMATCHER_DATA_DIR`	Data directory or explicit `.db` path
`--db-name`	`clearing.db`	SQLite filename when `--datapath` is a directory
`--flush-size`	`100`	Flush immediately when buffered trades reaches N
`--flush-interval`	`5.0`	Flush interval in seconds when buffer is non-empty
`--print-every`	`100`	Print in-memory P&L snapshot every N trades (`0` disables)
`--retention-days`	`90`	Prune `trade_events` rows older than N days on startup and at every close of the market (`0` disables pruning)
`--sql-trace`	off	Log executed SQLite SQL statements from the clearing writer connection
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

`pm-clearing-cli` global options include `--raw-output` to disable display normalization and print raw tick-unit values for price-derived fields.

Expected runtime input arguments:

None.

On startup, `pm-clearing` applies schema migrations and prunes `trade_events` older than the retention window (default: 90 days). Writes are batched into single SQLite transactions to update:

- `trade_events`
- `gateway_symbol_positions`
- `gateway_daily_summary`

`trade.executed.tick_decimals` is persisted so downstream output can normalize price-derived values for table/JSON/CSV rendering.

Messages subscribed (SUB from :5556):

Topic	Purpose
`trade.executed`	Every matched trade pair (including `tick_decimals`) — drives P&L calculations
`system.eod`	End-of-day marks and the session-close `session_events` row
`system.gateway_auth.`	Gateway connect (accepted) — opens a `gateway_sessions` row
`system.gateway_bye.`	Gateway disconnect — closes the matching `gateway_sessions` row

See Positions and P&L for the full accounting model.

2.14 pm-clearing-cli - Clearing Query CLI

`pm-clearing-cli` is a command-line interface for querying `clearing.db` without writing SQL manually. It supports both read-style reporting commands and a maintenance command (`prune`) that deletes old raw trade rows.

```
pm-clearing-cli [--datapath PATH] [--db-name clearing.db] [--format table|json|csv] [--no-header] [--raw-output] COMM
```

Unlike `pm-clearing`, this is a one-shot tool: it runs one command, prints output, and exits. Every verb works on `clearing.db` directly except `health`, which also pings the running `pm-clearing` on port `5563`.

Global options:

Flag	Default	Description
`--datapath PATH`	resolved from `EDUMATCHER_DATA_DIR`	Data directory or explicit `.db` file path
`--db-name NAME`	`clearing.db`	SQLite filename when `--datapath` is a directory
`--format`	`table`	Output format: `table`, `json`, or `csv`
`--no-header`	off	Suppress header row in CSV output
`--raw-output`	off	Disable tick-decimal normalization and emit raw tick-unit values
`--version`	—	Print version and exit

Subcommands:

Subcommand	Default limit	Purpose	Typical filters
`gateways`	1000	Gateway-level realized/unrealized/total P&L totals	`--gateway`
`positions`	10000	Current open position state by gateway and symbol	`--gateway`, `--symbol`
`pnl`	10000	Realized/unrealized/total P&L rows per gateway and symbol	`--gateway`, `--symbol`
`daily`	1000	Daily rollup summary rows	`--gateway`, `--symbol`, `--date`, `--from`, `--to`
`trades`	200	Raw trade-event rows	`--gateway`, `--symbol`, `--date`, `--from`, `--to`
`exposure`	1000	Net/gross notional exposure and P&L	`--gateway`, `--symbol`, `--sort`
`symbols`	1000	Symbol-level totals and open exposure snapshot	`--date`, `--from`, `--to`, `--sort`
`dates`	1000	Available trade dates (optionally with totals)	`--gateway`, `--symbol`, `--from`, `--to`, `--with-totals`
`health`	n/a	Check the `pm-clearing` process and ping its health socket; `--extended` adds the clearing DB integrity and read/write check, and `--extended -v` the row counts, flush metadata and journal mode	`--extended`, `-v`/`--verbose`, `-q`/`--quiet` (print nothing; exit `0` when healthy, `-1` otherwise)
`reconcile`	n/a	Compares raw `trade_events` vs daily summary (both buy and sell sides; also reports summary-only keys)	`--gateway`, `--symbol`, `--from`, `--to`, `--retention-days`
`sessions`	—	Gateway connection and disconnection history	`--gateway`, `--from`, `--to`, `--connected-only`, `--limit` (default 500)

Subcommand	Default limit	Purpose	Typical filters
`eod`	100	End-of-day sentinel events written by `pm-clearing` on `system.eod`	`--from`, `--to`, `--limit`
`prune`	n/a	Deletes old `trade_events` rows by retention window	`--days`, `--dry-run`

`daily` rows are per gateway, while `symbols` sums both counterparties' participant-side activity. A single 800-share trade therefore contributes 800 to each gateway's `daily` row and 1,600 to the symbol's `traded_qty`; use `pm-stats-cli daily` for market volume counted once per executed trade.

Sort options:

Command	Allowed `--sort` values
`exposure`	`gross_notional`, `net_notional`, `realized_pnl`, `unrealized_pnl`, `total_pnl`
`symbols`	`symbol`, `traded_qty`, `traded_notional`, `realized_pnl`, `open_net_qty`

Examples:

```
# Gateway totals (table)
pm-clearing-cli gateways

# One gateway in JSON
pm-clearing-cli --format json gateways --gateway GW_A

# Current positions as CSV (normalized display values)
pm-clearing-cli --format csv positions --gateway MM01

# Emit raw tick-unit values instead of normalized display values
pm-clearing-cli --format json --raw-output positions --gateway MM01

# Daily summary for one date range
pm-clearing-cli daily --from 2026-07-01 --to 2026-07-05

# Raw trades for one symbol
pm-clearing-cli trades --symbol AAPL --limit 50

# Risk view by total P&L
pm-clearing-cli exposure --sort total_pnl

# Date discovery with aggregate totals
pm-clearing-cli dates --with-totals

# Data-consistency check
pm-clearing-cli reconcile --from 2026-07-01 --to 2026-07-05

# Retention maintenance (dry run then execute)
pm-clearing-cli prune --days 90 --dry-run
pm-clearing-cli prune --days 90
```

No-row behaviour:

- `table`: prints `No rows found.` (except `reconcile`, which prints `OK — no discrepancies found.`)
- `json`: prints `[]`
- `csv`: prints only header row unless `--no-header` is set

See Positions and P&L for the accounting model and schema-level details.

2.15 pm-stats — Statistics Recorder

Records market statistics for every symbol, plus index level history, to a SQLite database (`data/stats.db`). This is the queryable time-series home for both trading statistics and index levels — `pm-index`'s own JSONL file only retains structural/corporate-action audit records, not level ticks.

```
pm-stats [--db data/stats.db] [--snapshot-interval SEC] \
  [--keep-days N] [--keep-order-events-days N] \
  [--human-gateways GLOB,...] [--keep-human-order-events-days N] \
  [--sql-trace] \
  [--log-level LEVEL] [-v|-vv] [-q]
```

Startup options:

Flag	Default	Description
`--db`	`data/stats.db`	SQLite database file path

Flag	Default	Description
`--snapshot-interval`	`900` (15 min)	Seconds between `price_snapshots` rows per symbol. Use a smaller value for finer intraday resolution, e.g. `60` for one-minute snapshots.
`--keep-days`	`0` (keep all)	Keep N days of `trade_log`, `order_events`, `price_snapshots` and `feed_gaps`; older rows are deleted at startup and once an hour. Daily rollups are always kept. The file stops growing at N days (SQLite reuses the freed pages)
`--keep-order-events-days`	`--keep-days`	Keep N days of `order_events` alone, overriding `--keep-days` for them. Fractions are fine (`0.05` is about an hour); under an AI swarm order events are most of the file
`--human-gateways`	none	Comma-separated gateway id globs, e.g. `TRADER*`, whose `order_events` prune on `--keep-human-order-events-days` instead. A person places a handful of orders a day and should not lose them on a clock set for a hundred bots
`--keep-human-order-events-days`	`--keep-order-events-days`	Keep N days of `order_events` for gateways matching `--human-gateways`; `0` keeps them for good. Requires `--human-gateways` — on its own it would match nobody and change nothing
`--sql-trace`	off	Log executed SQLite SQL statements from the stats writer connection
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

None.

Subscriptions:

`pm-stats` opens two independent PUB/SUB connections: one to `pm-engine` (`ENGINE_PUB_ADDR`, port 5556) and one to `pm-index` (`INDEX_PUB_CONNECT_ADDR`, port 5558).

Source	Topic	Purpose
pm-engine	`trade.*`	Updates OHLCV, VWAP, min/max, volume, trade log
pm-engine	`book.*`	Records opening bid/ask prices; drives 15-minute snapshots
pm-engine	`system.eod`	Records end-of-day closing bid/ask prices
pm-index	`index.update`	Records every index level tick to `index_level_snapshots` and rolls up daily OHLC into `index_daily_stats`

On engine shutdown, the engine broadcasts `system.eod` before closing sockets. `pm-stats` receives it and immediately flushes the closing bid/ask for every active symbol into `daily_stats`. The `index.update` subscription is independent of the engine socket and keeps working even if `pm-index` starts or stops at a different time than `pm-engine`.

2.15.1 How statistics are computed

OHLCV and VWAP accumulate in memory, one `SymbolStats` object per symbol, and are flushed to `daily_stats` after each trade.

Statistic	Trigger	Rule
`open_price`	First `trade.*` of the day	Set once; never overwritten
`close_price`	Every `trade.*`	Always overwritten with the latest price
`high_price`	Every `trade.*`	Running `max(current, trade_price)`
`low_price`	Every `trade.*`	Running `min(current, trade_price)`
`volume`	Every `trade.*`	Cumulative sum of matched quantities
`trade_count`	Every `trade.*`	Incremented by 1
`vwap`	Every `trade.*`	$\sum(price \times qty) / \sum(qty)$; maintained as two running accumulators
`largest_trade_qty/price`	Every `trade.*`	Replaced when `qty > current_largest`
`open_bid/ask`	First `book.*` update of the day	Set once from the book snapshot
`close_bid/ask`	`system.eod`	Overwritten with the final bid/ask from the book

15-minute price snapshots are written to `price_snapshots` when a `book.*` message arrives and at least `--snapshot-interval` seconds have elapsed since the last snapshot for that symbol (default: `SNAPSHOT_INTERVAL_SEC = 900`, i.e. 15 minutes). The mid-price is computed as:

$$mid = \frac{best_bid + best_ask}{2}$$

If one side of the book is empty the available side is used as the mid-price; if both sides are empty the last known trade price is used instead.

Trade log rows are appended on every `trade.*` event — no aggregation, one row per matched trade pair, using the trade UUID from the engine as the primary key.

Startup behaviour — on startup `pm-stats` sends a `book.snapshot_request` via PUSH `:5555` for every symbol it discovers from `system.symbols.STATS`. This primes the `open_bid/open_ask` values and the first snapshot interval even before the first trade.

2.15.2 Statistics Database Schema

The database lives at `data/stats.db` (SQLite 3). Nine tables are maintained in total: the three below (the ones queried directly by `pm-stats-cli` for day-to-day reporting), plus `instruments` and `stats_meta` (reference/meta), `feed_gaps` and `order_events` (private-event and gap tracking), and `index_daily_stats` / `index_level_snapshots` (index history, fed by `pm-index`). See Statistics and Reporting — The Statistics Database Schema for the complete reference, including those five tables.

All three tables below store prices as **integer ticks**, with a per-row `tick_decimals` column giving the conversion: `display price = price / 10^tick_decimals`. Always use the row's own `tick_decimals` — symbols do not share a tick size.

`daily_stats`

One row per `(date, symbol)`. Upserted on every trade and again when `system.eod` arrives.

Column	Type	Description
`date`	TEXT (PK)	Calendar date `YYYY-MM-DD`
`symbol`	TEXT (PK)	Instrument ticker
`open_price`	INTEGER	Price of the first trade of the day, in ticks
`high_price`	INTEGER	Highest trade price of the day, in ticks
`low_price`	INTEGER	Lowest trade price of the day, in ticks
`close_price`	INTEGER	Price of the last trade of the day, in ticks
`open_bid`	INTEGER	Best bid at the time of the first book update, in ticks
`open_ask`	INTEGER	Best ask at the time of the first book update, in ticks
`close_bid`	INTEGER	Best bid recorded at engine shutdown (`system.eod`), in ticks
`close_ask`	INTEGER	Best ask recorded at engine shutdown (`system.eod`), in ticks
`volume`	INTEGER	Total traded quantity for the day
`trade_count`	INTEGER	Number of individual trades
`turnover`	INTEGER	$\sum(price_ticks \times qty)$ — exact; the VWAP numerator
`vwap`	REAL	`turnover / volume`, in ticks. Derived, so REAL — divide by `10^tick_decimals` for display money
`largest_trade_qty`	INTEGER	Quantity of the single largest trade
`largest_trade_price`	INTEGER	Price of the single largest trade, in ticks
`tick_decimals`	INTEGER	Decimal scale for this row's tick columns

Example query — end-of-day summary, converted to display money:

```
SELECT date, symbol,
       open_price / POWER(10, tick_decimals) AS open_price,
       high_price / POWER(10, tick_decimals) AS high_price,
       low_price / POWER(10, tick_decimals) AS low_price,
       close_price / POWER(10, tick_decimals) AS close_price,
       volume, trade_count,
       ROUND(vwap / POWER(10, tick_decimals), 4) AS vwap
FROM daily_stats
ORDER BY date DESC, symbol;
```

`price_snapshots`

One row per `(ts, symbol)` written every **`--snapshot-interval`** seconds (default: 15 minutes) when a book update arrives.

Column	Type	Description
`ts`	TEXT (PK)	ISO-8601 timestamp (UTC, second precision)
`symbol`	TEXT (PK)	Instrument ticker
`mid_price`	REAL	`(best_bid + best_ask) / 2`, in ticks; falls back to whichever side is present, then the last trade price. Derived, so REAL — a midpoint between two ticks can land on a half-tick
`best_bid`	INTEGER	Best bid price at snapshot time, in ticks (null if empty book)
`best_ask`	INTEGER	Best ask price at snapshot time, in ticks (null if empty book)
`pct_change`	REAL	Percentage change of `mid_price` vs. previous snapshot: $100 \times (mid_t - mid_{t-1})/mid_{t-1}$ (a ratio, not a price — never converted)
`tick_decimals`	INTEGER	Decimal scale for this row's tick columns

Example query — intraday price path for MSFT, converted to display money:

```
SELECT ts,
       mid_price / POWER(10, tick_decimals) AS mid_price,
       best_bid / POWER(10, tick_decimals) AS best_bid,
       best_ask / POWER(10, tick_decimals) AS best_ask,
       ROUND(pct_change, 4) || '%' AS change
FROM price_snapshots
WHERE symbol = 'MSFT'
ORDER BY ts;
```

`trade_log`

Append-only record of every individual matched trade.

Column	Type	Description
`ts`	TEXT	ISO-8601 timestamp (UTC, millisecond precision)
`trade_id`	TEXT (PK)	Durable engine trade id (`run_seq-counter`); duplicate delivery is idempotent on this key
`symbol`	TEXT	Instrument ticker
`price`	INTEGER	Execution price, in ticks
`quantity`	INTEGER	Matched quantity
`tick_decimals`	INTEGER	Decimal scale for `price`
`buy_gateway_id`	TEXT	Gateway that submitted the buy order
`sell_gateway_id`	TEXT	Gateway that submitted the sell order
`aggressor_side`	TEXT	`BUY` or `SELL` for a continuous match; `AUCTION` for an uncross print where both sides were resting

Example query — all trades for a symbol sorted by time, converted to display money:

```
SELECT ts, price / POWER(10, tick_decimals) AS price,
       quantity, aggressor_side, buy_gateway_id, sell_gateway_id
FROM trade_log
WHERE symbol = 'AAPL'
ORDER BY ts;
```

Querying the database

You can open `data/stats.db` with any SQLite client: `bash sqlite3 data/stats.db .headers on .mode column SELECT * FROM daily_stats;`

2.16 pm-stats-cli — Statistics Query CLI

`pm-stats-cli` is a read-only command-line interface for querying `data/stats.db` without writing SQL manually.

```
pm-stats-cli [--db data/stats.db] [--format table|json|csv] [--no-header] [--timezone TZ] COMMAND [options]
```

Unlike `pm-stats`, this is not a subscriber process. It runs one query, prints output, and exits.

Global options:

Flag	Default	Description
`--db`	`data/stats.db`	SQLite database file path
`--format`	`table`	Output format: `table`, `json`, or `csv`
`--no-header`	off	Suppress header row in `csv` output
`--timezone`	DB-recorded timezone	Override the session timezone that `--date` resolves in; rarely needed, and a mismatched value triggers a warning since `--date` would then select a different trading day than `pm-stats` used. `pm-stats` itself takes no `--timezone` flag — it reads `locale.session_timezone` from the deployed config
`--version`	—	Print version and exit

Subcommands:

Subcommand	Purpose	Typical filters
`daily`	Daily OHLCV summary from `daily_stats`	`--date`, `--symbol`, `--limit`, `--after`, `--wide`
`snapshots`	Intraday snapshots from `price_snapshots`	`--symbol` (required), `--date`, `--from`, `--to`, `--limit`, `--after`
`trades`	Trade history from `trade_log`	`--symbol`, `--date`, `--from`, `--to`, `--limit`, `--after`
`order-events`	Private order lifecycle events from the order_events table	`--gateway` (required), `--symbol`, `--event-type`, `--date`, `--from`, `--to`, `--limit` (default 500), `--after`
`order-lifecycle`	All events for a single order ID	`--gateway` (required), `--order-id` (required)

Subcommand	Purpose	Typical filters
`symbols`	Discover symbols available in stats data	`--date`
`dates`	Discover trading dates available in `daily_stats`	`--symbol`
`index-daily`	Daily index OHLC rollup from `index_daily_stats`	`--date`, `--index-id`, `--limit`, `--after`, `--wide`
`index-snapshots`	Every recorded index level tick from `index_level_snapshots`	`--index-id` (required), `--date`, `--from`, `--to`, `--limit`, `--after`
`index-ids`	Discover index IDs with recorded data	`--date`
`instruments`	Show instrument reference data (tick scale per symbol)	`--symbol`
`gaps`	Show detected feed gaps (trades the recorder never received)	`--date`, `--from`, `--to`, `--limit` (default 500)
`health`	Check the `pm-stats` process and ping its health socket; `--extended` adds the stats DB integrity and read/write check	`--extended`, `-v`/`--verbose`, `-q`/`--quiet` (print nothing; exit `0` when healthy, `-1` otherwise)

`--after CURSOR` pages through results returned by a previous call to the same subcommand — pass the cursor value from that call's last row to fetch the next page.

`index-daily`, `index-snapshots`, and `index-ids` query the level/EOD history that `pm-index`'s own JSONL file no longer stores — see Statistics and Reporting for the full reference and Market Index for why this data moved here.

Examples:

```
# Latest available daily summary date
pm-stats-cli daily

# Daily summary for one date and one symbol
pm-stats-cli daily --date 2026-06-14 --symbol AAPL

# Include bid/ask and largest-trade fields
pm-stats-cli daily --date 2026-06-14 --wide

# Intraday snapshots for one symbol in a time window
pm-stats-cli snapshots --symbol MSFT --from 2026-06-14T09:00:00+00:00 --to 2026-06-14T16:30:00+00:00

# Trades as CSV for scripting/export
pm-stats-cli --format csv trades --symbol AAPL --date 2026-06-14

# Trades as JSON for automation
pm-stats-cli --format json trades --symbol AAPL --limit 50

# Discovery helpers
pm-stats-cli symbols
pm-stats-cli dates --symbol AAPL
```

No-row behaviour:

- `table`: prints `No rows found.`
- `json`: prints `[]`
- `csv`: prints only header row unless `--no-header` is set

For schema details, see the `pm-stats` section above and Statistics and Reporting.

2.17 pm-scheduler — Session Scheduler

Drives session-phase transitions (PRE_OPEN → OPENING_AUCTION → CONTINUOUS → CLOSING_AUCTION → CLOSED) by sending `session.transition` messages to the engine at configured wall-clock times.

```
pm-scheduler [--now] [--delay 3] [--daily] [--speed F] [--start YYYY-MM-DDTHH:MM] [--no-confirm] [--on-close COMMAND]
  [--log-target TARGET] [--log-file PATH] [--log-failover-timeout SEC]
```

Startup options:

Flag	Default	Description
`--now`	off	Skip wall-clock waiting; send all transitions immediately with a delay between each
`--delay`	`None` (effective default 3.0, only used with `--now`)	Seconds between transitions in `--now` mode (ignored, with a warning, outside `--now`)
`--daily`	off	Run continuously, repeating the schedule every working day (weekends and the configured country's bank holidays are skipped)
`--speed F`	`$EDUMATCHER_SCHEDULER_SPEED`, else off	Compressed mode (with `--daily`): simulated time runs F times faster than the wall clock, so a whole trading day fits in minutes
`--start YYYY-MM-DDTHH:MM`	`$EDUMATCHER_SCHEDULER_START`, else now	Compressed mode: the session-local time the simulated clock starts at
`--no-confirm`	off	Do not query/confirm session state via the engine before sending a transition
`--on-close COMMAND`	off	Start this command after each day's `CLOSED` transition and once at start-up (for example `pm-eod run --pending`). It runs detached, one at a time, with its output in the scheduler's log; ignored in `--now` mode. See Closing the Day
`--on-close-delay SEC`	30	Seconds to wait after `CLOSED` before starting `--on-close`, so `pm-clearing` can save its close snapshot first
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`--verbose` / `-v`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`

Flag	Default	Description
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

None.

Messages sent (PUSH → :5555):

Topic	Purpose
`session.transition`	Requests the engine to move to the next session phase

The scheduler does not subscribe to any PUB messages — it is fire-and-forget.

See Auctions & Scheduling for the full schedule configuration and session-phase documentation.

2.18 pm-eod — End-of-Day Settlement

Settles a closed trading day and tidies up after it. A **one-shot** command: it runs, prints what it did and exits. For each trading date it checks that the day really closed and the books reconcile (**gate**), turns each trader's change in mark-to-market equity into a cash amount against the exchange (**settle**), writes statements and a summary (**report**), and runs the housekeeping steps (**audit**, **logs**, **backup**). The concepts, setup and troubleshooting are in Closing the Day.

```
pm-eod [--datapath DIR] [--format table|json|csv] [--no-header] run \
  (--date YYYY-MM-DD | --pending) [--only STEP] [--force] [--redo] [--dry-run] \
  [--log-keep-days N] [--backup-keep-days N] [--backup-skip NAME ...]
pm-eod [--datapath DIR] [--format table|json|csv] status [-q] [--limit N]
pm-eod [--datapath DIR] [--format table|json|csv] balances [--date YYYY-MM-DD] [--gateway GW_ID] [--check]
```

Global options (must come *before* the subcommand):

Flag	Default	Description
`--datapath DIR`	the configured data directory	Data directory holding `clearing.db`, `eod.db`, `ref_data/roster.yaml` and the output folders
`--format`	`table`	Output format for `status` and `balances`: `table`, `json` or `csv`
`--no-header`	off	Omit the header row (table and csv)
`--version`	—	Print version and exit

`run` options:

Flag	Default	Description
`--date YYYY-MM-DD`	—	The trading date to close. Exactly one of `--date` and `--pending` is required.
`--pending`	off	Every closed date that has a close snapshot and no `ok` run, oldest first. Stops at the first failing date. With more than one date, each date's `backup` step is deferred and one backup is taken at the end, under the last date that got through.
`--only STEP`	all steps	Re-run just `report`, `audit`, `logs` or `backup` for an already settled date. Not recorded in the run history.
`--force`	off	Do not require the `CLOSED` session event (for exchanges without a scheduler). The close snapshot and reconcile checks still apply.
`--redo`	off	Recompute the date: deletes its settlement **and the settlement of every later date**, then settles it again, all in one transaction (if the recomputation fails, nothing is deleted).
`--dry-run`	off	Run the gate and compute the amounts, print them, write nothing. Cannot be combined with `--only`.
`--log-keep-days N`	`14`	Delete compressed fallback logs older than N days
`--backup-keep-days N`	`7`	Delete backup folders older than N days
`--backup-skip NAME`	none	Leave one database out of the backup: `clearing.db`, `stats.db` or `eod.db`. Repeatable. The `backup` step's message says what was skipped.

`status` options: `-q` / `--quiet` prints nothing and exits 0 when no closed date is waiting (a closed date whose run failed counts as waiting) and 1 otherwise; `--limit N` (default 30) caps the number of runs shown.

`balances` options: `--date` shows balances as of that date; `--gateway` filters to one trader (case-

insensitive); `--check` verifies that every settled date sums to exactly zero and exits 1 if one does not.

Exit codes of `run`: `0` clean · `1` the gate, the settlement or the report failed · `2` only housekeeping warnings. A second `pm-eod` started while one is running exits 1 (lock file `<data>/.pm-eod.lock`).

Reads: `clearing.db` (trades, positions, `position_close_snapshot`, `session_events`), `stats.db` (optional, for the summary's market table), `ref_data/roster.yaml` (optional). **Writes:** `eod.db` (the ledger), `settlement/<date>/`, `backup/<date>/`, compressed audit days under `audit/`, and compressed fallback logs under `logs/`. It binds no port and sends no message.

2.18.1 `roster.yaml`

Key	Required	Description
`traders`	yes	List of trader entries
`traders[].gateway`	yes	Gateway identifier; upper-cased on load; unique; not `EXCHANGE`
`traders[].name`	no	Display name (default: the gateway identifier)
`traders[].email`	no	Printed on the statement and in `instructions.csv`; never used to send mail
`traders[].opening_balance`	no	Starting balance, at most 2 decimals (default 0)

A missing file is an empty roster. Errors name the line (`roster.yaml: line 7: …`).

2.18.2 Files written to `settlement/<date>/`

File	Content
`statements/<gateway>.md`	The trader's statement (the headline, result by symbol, open positions at the close, trades). Result by symbol lists a symbol only if it is held at the close, traded that day or has a non-zero result
`statements/<gateway>.json`	The same, with `amount_minor` / `balance_minor` as integers in minor units
`statements/<gateway>.csv`	One row per symbol: `symbol,day_pnl,net_qty,mark`
`instructions.csv`	`gateway,name,email,amount,direction,balance`; `direction` is `receive`, `pay` or `none`
`leaderboard.csv`	`rank,gateway,name,balance`; equal balances share a rank
`summary.md`, `summary.json`	Accounts settled, the zero-sum proof, rounding, biggest winners and losers, market table, warnings

2.18.3 The settlement ledger `eod.db`

Table	Columns	Description
`eod_settlement`	`trade_date`, `account`, `amount_minor`, `detail_json`	One row per account per settled date. `account` is a gateway identifier or `EXCHANGE` (the rounding line). The amounts of a date sum to exactly 0.
`eod_runs`	`trade_date`, `status`, `started_at`, `finished_at`, `steps_json`	One row per date run. `status` is `ok`, `failed` or `report-failed`.

Balances are not stored: they are the sum of `amount_minor` plus the roster's opening balance. The full DDL is in the generated SQLite schema reference.

2.18.4 Settlement rules (reference)

- A date's amount for a gateway is the sum over its symbols of `(realized_pnl + unrealized_pnl)` at this date's `position_close_snapshot` minus the same at the snapshot of the latest earlier **settled** date (0 if there is none, or if `clearing.db` no longer has that date's snapshot), each converted from ticks to currency with the symbol's own `tick_decimals`. Any unsettled snapshot dates in between are part of the one interval.
- A date **has closed** when its latest `session_events` `PHASE` row is `CLOSED` and an earlier `PHASE` row of the same date is not, so the market was open that day. The `CLOSED` an engine announces when it starts, before the open, is therefore not a close.
- Dates settle in order: a date is refused while an earlier date has closed (with a snapshot) but is not settled.
- Rounded to 2 decimals with the largest-remainder method; the exact total must be within `0.01` per position of zero, otherwise the settlement fails.
- A position present in the earlier snapshot but missing from this date's is ignored.
- The gate requires: the date has closed (skipped by `--force`); a close snapshot; and an empty `pm-clearing-cli reconcile` for the date.

See Closing the Day for the worked example, the troubleshooting table and the housekeeping details.

2.19 pm-ticker — Scrolling Market Ticker

Draws a bordered ticker box across the top rows of the terminal. The box has a header (the `EduMatcher` brand, today's total trade volume, and the current date/time), a ticker line listing all active symbols with live prices, OHLCV, and bid/ask spreads, and a news line. The symbol line scrolls leftward like a classic ticker tape. The news line shows the latest `pm-market-sim` headline: its time, `[RUMOUR 60%]` (with the credibility) or `[CONFIRMED]`/`[RETRACTED]` when a rumour resolves, and the headline in green (good news) or red (bad), struck through when retracted; `no news yet` until the first.

```
pm-ticker [--db data/stats.db] [--db-interval 900] [--timezone TZ] \
  [--log-level LEVEL] [-v|-vv] [-q] [--log-target TARGET] [--log-file PATH] [--log-failover-timeout SEC]
```

Startup options:

Flag	Default	Description
`--db`	`data/stats.db`	Path to the statistics SQLite database
`--db-interval`	900	Seconds between daily_stats DB re-queries (15 min)
`--timezone`	DB-recorded timezone	Exchange session timezone (IANA name) used to bucket data into a trading day
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	30 (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

None.

Display behaviour:

- The box redraws to the terminal width on resize.
- If every symbol fits within the current width the line stays static; scrolling only starts once the symbols are wider than the box.
- Below a minimum width the ticker line is truncated with an ellipsis (…).

Ticker line format:

```
AAPL  415.00  +0.48%  H:418.00  L:412.00  Vol:52,400 (8T)  414.50/415.50  ◆  MSFT …
```

Each symbol segment shows:

Field	Description
Symbol	Instrument name (bold cyan)
Last price	Most recent trade price or closing price from DB
Change %	Percentage change vs. today's open price (green if up, red if down)
H: / L:	Intraday high (green) and low (red) from the daily_stats table
Vol	Cumulative traded volume for the day
(nT)	Number of trades today
Bid/Ask	Current best bid (green) / best ask (red) from live book

The ticker combines **live ZMQ data** (last price, bid/ask from `book.*` messages) with **historical DB data** (OHLCV, trade count from `pm-stats`'s SQLite database). The DB is re-queried every `--db-interval` seconds to pick up updated daily statistics.

Messages subscribed (SUB from :5556):

Topic	Purpose
`book.` (prefix)	Live last price, best bid/ask per symbol

Depends on pm-stats

The ticker reads from `data/stats.db` which is populated by `pm-stats`. Without `pm-stats` running, the ticker still works but only shows live bid/ask and last price — OHLCV, volume, and trade count will be missing.

2.20 pm-board — Market Board

Full-screen multi-symbol display designed for large monitors or projection screens. Shows all active symbols in a single paged table with exchange-style colouring.

```
pm-board [--rows 8] [--interval 10]
```

Startup options:

Flag	Default	Description
`--rows` / `-r`	8	Maximum number of symbols (rows) displayed per page
`--interval` / `-i`	10	Seconds before auto-rotating to the next page
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

- ++enter++ to advance to the next page immediately
- ++ctrl+c++ to exit

Controls:

Key	Action
++enter++	Advance to next page immediately
++ctrl+c++	Exit

Display columns:

Column	Description
Symbol	Instrument ticker
Last	Last traded price (coloured green if up from first trade, red if down)
Chg %	Percentage change from the first trade of the session: $100 \times (last - first)/first$
Bid	Best (highest) bid price currently in the book
Ask	Best (lowest) ask price currently in the book
Spread	Difference between best ask and best bid: $ask - bid$
Last Buy	Last trade price where this symbol was bought (green)
Last Sell	Last trade price where this symbol was sold (red)
Vol	Cumulative traded volume for the session
Updated	Timestamp of the most recent book update for this symbol

The header bar shows: page number, total pages, number of active symbols, the auto-rotate interval, and the current clock time.

Paging behaviour:

- Symbols are sorted alphabetically and divided into pages of `--rows` symbols each.
- The display auto-rotates to the next page every `--interval` seconds.
- When the last page is reached, rotation wraps back to page 1.
- Pressing ENTER advances immediately and resets the auto-rotate timer.

Colour conventions (matching standard exchange displays):

- **Green** — price increase, buy trades, positive change %
- **Red** — price decrease, sell trades, negative change %
- **White** — unchanged or neutral values

Messages subscribed (SUB from :5556):

Topic	Purpose
`book.` (prefix)	Book updates for all symbols — discovers symbols automatically
`trade.executed`	Every matched trade — drives volume and last-price tracking

Large-screen demo

For classroom or conference demos, maximise the terminal and use large rows: `bash pm-board --rows 15 --interval 8` The board auto-discovers symbols as they become active — no configuration needed.

2.21 pm-ai-trader — Autonomous Trader Bot

Runs one AI trader: a participant driven by a *preset* (strategy × execution style × tempo × risk limits). It logs on to `pm-alf-gwy` as its own participant, sends its orders over that ALF session and reads market data from the engine's PUB socket (and the true values and news from `pm-market-sim`'s). See AI Traders.

```
pm-ai-trader --id AI001 [--preset NAME | --preset-file PATH] [options]
pm-ai-trader --list-presets
```

Startup options:

Flag	Default	Description
`--id`	required	Participant ID to log on as (e.g. `AI001`); must be a `TRADER` in the deployed configuration
`--preset`	`noise-retail`	Built-in preset: `auction-player`, `block-taker`, `contrarian`, `iceberg-seller`, `institutional`, `market-taker`, `news-trader`, `noise-retail`, `patient-liquidity`, `scalper`, `trend-follower`, `value-investor`
`--preset-file`	none	Preset YAML file instead of a built-in
`--list-presets`	—	Print the built-in presets (strategy, execution, protection, description) and exit
`--symbols`	every deployed symbol	Comma-separated symbols to trade
`--seed`	`1`	Random seed; same seed and same market, same decisions
`--alf-host`	`EDUMATCHER_ENGINE_HOST` or `127.0.0.1`	Host of `pm-alf-gwy`
`--alf-port`	`alf_gateway.port` of the deployed config	Port of `pm-alf-gwy`
`--duration`	`0`	Runtime in seconds; `0` means run until stopped
`--log-level`	`WARNING`	Explicit level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	`-v` → `INFO` (a status line a minute, session changes), `-vv` → `DEBUG` (every reject)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`

Flag	Default	Description
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

None.

Connect and reconnect:

On logon, and after every reconnect, the trader reads the symbols, the session, the halts and its own positions from the engine before it sends an order. If the ALF session drops it retries with jittered back-off (at most 10 s apart); when the engine restarts, it logs on again and re-reads everything.

2.22 pm-ai-swarm — Multi-Agent Trading Swarm

Runs many AI traders: a supervisor process splits them into contiguous ID blocks and runs each block in one worker process, restarting a worker that dies. Each trader is its own ALF session. Settings come from a `swarm.yaml` (`--swarm`), overridden by flags. See Running the AI-Trader Swarm.

```
pm-ai-swarm --swarm swarm.yaml [options]
```

Startup options:

Flag	Default	Description
`--swarm`	none	`swarm.yaml` file; flags override it
`--count`	`10`	Number of traders
`--prefix`	`AI`	Participant ID prefix
`--start-index`	`1`	First ID number (`AI001`)
`--presets`	every built-in, equal weights	Composition: `a,b` (equal weights) or `a:3,b:1`
`--symbols`	every deployed symbol	Comma-separated symbol universe
`--symbols-per-agent`	just enough to cover the universe	Consecutive symbols each trader trades
`--seed-base`	`1000`	Trader i gets seed `seed-base + i`
`--duration`	`0`	Runtime in seconds; `0` means run until stopped
`--workers`	`min(cpus - 1, ceil(count / 100))`	Worker processes

Flag	Default	Description
`--budget`	`0`	Order actions per second for the whole swarm; `0` lets the presets decide
`--alf-host`	`EDUMATCHER_ENGINE_HOST` or `127.0.0.1`	Host of `pm-alf-gwy`
`--alf-port`	`alf_gateway.port` of the deployed config	Port of `pm-alf-gwy`
`--log-level`	`WARNING`	Logging level of the supervisor and the workers
`-v` / `--verbose`	off	`-v` → `INFO`, `-vv` → `DEBUG`; applies to the workers too
`-q` / `--quiet`	off	Reduce output to warnings/errors; applies to the workers too
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

None. `SIGTERM` or Ctrl+C stops every worker; workers get 10 s before they are killed.

Output files:

`<DATA_DIR>/ai_swarm/<date>-<worker>.jsonl` — one JSON line per trader, five seconds after each close.

2.23 pm-market-sim — Market Model

Publishes every symbol's true value from a seeded market model, and news. See The Market Model.

```
pm-market-sim [options]
pm-market-sim --init [--seed N] > market_sim.yaml
pm-market-sim --status --id ADMIN_ID [--top N]
```

Startup options:

Flag	Default	Description
`--init`	off	Print a `market_sim.yaml` for the deployed configuration's symbols and exit
`--seed`	`1`	Seed written by `--init`
`--status`	off	Ask the running process for its state and the widest market-vs-value gaps, and exit
`--id`	none	ADMIN participant ID that `--status` asks as
`--top`	`10`	Gaps `--status` lists
`--log-level`, `-v`, `-q`, `--log-target`, `--log-file`, `--log-failover-timeout`, `--version`	—	As for the other processes

Expected runtime input arguments:

None. It reads `market_sim.yaml` from the deployed configuration directory and keeps its state in `<DATA_DIR>/market_sim_state.json`.

2.24 pm-news — News Desk

The instructor's news desk: sends commands to the running `pm-market-sim` as an ADMIN participant. See The Market Model — News.

```
pm-news --id ADMIN_ID inject (--symbol SYM | --sector SECTOR | --market) \
    --kind KIND --sentiment S --impact I [--headline TEXT] [--rumour [--credibility C]]
pm-news --id ADMIN_ID confirm NEWS_ID
pm-news --id ADMIN_ID retract NEWS_ID
pm-news --id ADMIN_ID play scenario.yaml
pm-news list [--duration SECONDS]
```

Subcommand	Does
`inject`	Publish a headline; prints its ID. `--symbol` and `--sector` repeat; `--kind` is one of `CONTRACT`, `EARNINGS`, `ESG`, `GUIDANCE`, `LEGAL`, `MACRO`, `MANAGEMENT`, `MNA`, `OPERATIONS`, `PRODUCT`, `REGULATORY`, `RESEARCH`; `--sentiment` −1 to 1 is public; `--impact` (log move of the value when confirmed) is not
`confirm`	Confirm a rumour: the values move now
`retract`	Retract a rumour: nothing moves
`play`	Run a timed scenario YAML
`list`	Print headlines as they are published; `--duration 0` (default) until Ctrl+C

Expected runtime input arguments:

None.

2.25 pm-compdb-cli — Company Database CLI

A SQLite database of company profiles for the 300 tickers of the s300 simulation (`docs/examples/ref_data/s300-sim-se`) and the tool to query and edit it. It also holds 100 Swedish and other Nordic companies whose tickers `pm-market-sim` does not trade. The companies are made up: each ticker keeps its sector and industry from `pm-market-sim` (and so the news it gets), but every name, person and number is invented. No running process is needed.

```
pm-compdb-cli [--db PATH] [--format table|grid|markdown|json|csv] [--no-header] COMMAND [options]
pm-compdb-cli list [-c COLS|all] [-w FILTER]... [-s COL [--desc]] [-n N]
pm-compdb-cli add|update SYMBOL FIELD=VALUE...
```

Subcommand	Does
`init`	Build the database of the 300 bundled companies; `--force` replaces an existing one
`columns`	List the columns, their types and what each holds
`show SYMBOL...`	Everything about one or more companies
`list`	List companies: `-c` comma-separated columns or `all`; `-w COLUMN OP VALUE` with `OP` one of `= != < <= > >= ~` (contains, any case), repeat to narrow; `-s` sort column, `--desc`; `-n` at most N rows
`add SYMBOL FIELD=VALUE...`	Add a company; `name`, `sector` and `industry` are required
`import FILE...`	Add the companies of YAML files (a mapping of `SYMBOL:` to fields, the format of the bundled ones); all or nothing
`update SYMBOL FIELD=VALUE...`	Change fields; an empty value clears one
`delete SYMBOL`	Remove a company

Output formats. `--format` applies to `show` and `list`:

Format	Output
`table` (default)	Plain text columns separated by `|`; long cells are cut short with `...`
`grid`	A table drawn with UTF-8 box characters (┌┬┐ │ ├┼┤ └┴┘); numbers right-aligned, long cells wrapped over several lines instead of cut. `show` gives a two-column table of column and value

Format	Output
`markdown`	Markdown for notes, wikis and pull requests. `list` gives a pipe table with column names and tickers in `code` and numbers right-aligned. `show` gives a heading with the name and ticker, then a table of column, value and meaning for each of Identity, Business, History, Management and Financials. A `|` in a value is escaped. `--no-header` is ignored, as a Markdown table needs its header
`json`, `csv`	For programs; `--no-header` leaves out the CSV header

```
pm-compdb-cli --format grid list -w sector=TECH -s market_cap_musd --desc -n 5
pm-compdb-cli --format markdown show NVDA > nvda.md
```

Where the companies come from. The bundled companies are hand-written YAML files, one per sector, installed as part of the `edumatcher` package in its `compdb/companies/` folder. `init` always reads them from the installed package, and no option points it elsewhere. To load your own files, use `import`. Where the folder is depends on how EduMatcher was installed:

Install	Source YAML (`init` reads)	Database (default `--db`)
Source checkout (`poetry install`, editable)	`src/edumatcher/compdb/companies/` in the checkout; edits take effect at the next `init`	`<DATA_DIR>/companies.db`
`pip` / `pipx` install of the wheel	`<site-packages>/edumatcher/compdb/companies/` inside that environment (pipx: `~/.local/share/pipx/venvs/edumatcher/lib/python3.*/site-packages/…`); replaced by every upgrade	`<DATA_DIR>/companies.db`
Simulation container (`edumatcher-ai-sim`)	`/opt/edumatcher/.venv/lib/python3.14/site-packages/edumatcher/compdb/companies/` in the image	`/data/companies.db`, which is `~/.edumatcher-sim/data/companies.db` (installed) or `./data-sim/companies.db` (checkout) on the host

To see the folder of the installation you are running:

```
python -c "from edumatcher.compdb.store import SEED_DIR; print(SEED_DIR)"
```

`init` never changes the YAML files. Adding, changing or deleting a company changes only the database. The source checkout also has a prebuilt copy, `docs/examples/ref_data/s300-sim-setup/companies.db`. A test keeps it equal to the YAML, so after editing the YAML rebuild it with `pm-compdb-cli --db docs/examples/ref_data/s300-sim-setup/companies.db init --force`. The prebuilt copy is not in the wheel or the container image.

In the simulation container. The container's entrypoint (`deployment/docker/entrypoint.sh`) builds the database when it prepares the data directory: if the bundled `s300-sim` example is configured (`EM_CONFIG=s300-sim`, the image's default) and `/data/companies.db` does not exist, it runs `pm-compdb-cli --db /data/companies.db init`. It does nothing in the following cases:

- the database already exists, so companies you added survive restarts and upgrades. Delete the file and restart to get the bundled set of the installed version back.
- another example is configured, or your own configuration is given with `EM_CONFIG_FILE`. Its tickers need not be the s300 ones; run `init` yourself if you want the database anyway.

A failed build is logged (`warning: could not build /data/companies.db`) and the exchange starts regardless.

Columns. Identity: `symbol`, `name`, `sector`, `industry`, `headquarters`. Business: `purpose`, `key_market`, `market_position`, `competitors` (tickers), `key_risk`. History: `origin` (`FOUNDER`, `CO_FOUNDERS`, `FAMILY`, `MERGER`, `SPIN_OFF`, `PRIVATISATION`, `COOPERATIVE`), `founders`, `founder_story`, `founded_year`, `ipo_year`, `ipo_valuation_musd`. Management: `ceo_name`, `ceo_since`, `ceo_industry_years`, `employees`. Financials for `fiscal_year` (money in USD millions): `revenue_musd`, `net_income_musd`, `revenue_growth_pct`, `operating_margin_pct`, `free_cash_flow_musd`, `market_cap_musd`, `dividend_yield_pct`, `debt_to_equity`, `credit_rating`. Derived by the database: `years_in_business`, `ceo_tenure_years`, `pe_ratio` (market cap / net income, empty for a loss), `ps_ratio`, `net_margin_pct`.

Checks on every write: a ticker the simulation knows must keep its sector; the industries the simulation gives news of their own (`Semiconductors`, `Software`, `Pharmaceuticals`, `Automobiles`, `Airlines`, `Aerospace & Defense`) belong to exactly their members; years must run in order (founded, IPO, fiscal year; the CEO no earlier than the founding).

```
pm-compdb-cli show NVDA
pm-compdb-cli list -c symbol,name,pe_ratio,dividend_yield_pct -w sector=UTILITIES -s dividend_yield_pct --desc
pm-compdb-cli list -w "origin=MERGER" -w "founded_year<1950" -c symbol,name,founders
pm-compdb-cli list -w "net_income_musd<0" -c symbol,name,revenue_growth_pct,key_risk
pm-compdb-cli --format csv list -c all > companies.csv
```

Expected runtime input arguments:

None.

2.25.1 Reading a company profile

The Company tab of `pm-trader` and `pm-terminal`, and `pm-compdb-cli show`, present the same profile. This section explains every value in it, in the order the tab shows them: what the value is, how it is calculated, how to read it, and how it affects the company, its valuation and its market position. The companies and figures are invented, but they are built to behave as the real ones do, so the habits learned here carry over. Column names are in `code`.

Three ideas run through everything below.

- **A figure means little alone.** Compare it with the company's own past, with its competitors (the `competitors` column lists their tickers), and with its industry. A 2 % net margin is excellent for a supermarket and a disaster for a software company.
- **A price is a view of the future.** What the market pays for a company (`market_cap_musd`) depends less on what it earned last year than on what investors expect it to earn from now on. Most of the figures below are either a measure of that expectation or evidence for it.
- **Everything is connected.** Growth needs investment, investment uses cash or debt, debt costs interest, interest lowers profit, and profit and cash decide what the company is worth. Reading a profile is following these links.

Money is in millions of US dollars (shown as `M$`), for the fiscal year in `fiscal_year`. A blank value means the database has no figure, or, for a derived ratio, that the ratio has no meaning (a P/E for a company making a loss).

Valuation

Market cap (`market_cap_musd`). The market capitalisation: what the market says the whole company is worth, equal to the share price times the number of shares. A company with 100 million shares at 50 $ has a market cap of 5,000 M$. It is the price of the *equity*, the owners' part, and does not count the company's debts.

- *Reading it.* It is a measure of size. Large companies are called large-caps, and are usually older, more stable and easier to trade. Small-caps can grow faster but are more volatile and have wider spreads.
- *Effect.* It is the base of every ratio in this group: P/E and P/S divide it by profit and sales. A high market cap gives a company cheap access to money (it can issue shares at a good price, or pay for acquisitions with them) and often makes it a member of the indices that funds buy by rule, which adds steady demand for its shares. A falling market cap makes everything more expensive for it, and can make it a takeover target.

P/E (`pe_ratio`). Price / earnings: `market_cap_musd` divided by `net_income_musd`. It says how many years of the current profit you pay for the company if the profit stayed the same. A P/E of 20 means 20 years of earnings. It is left empty when the net income is zero or negative, because a price over a loss has no meaning.

- *Reading it.* A high P/E means investors expect the profit to grow, or find the company safe, or both. A low P/E means the company is cheap, or that investors expect the profit to fall (a "value trap"). Compare it with the industry: utilities trade on low single digits to the low teens, fast-growing software companies on several times that.
- *Effect.* The P/E is the market's report card on the company's future. A high one is an asset: the company can raise money cheaply and buy others with its shares. It is also a burden, because a company with a high P/E must keep delivering, and a disappointing quarter can cut the share price by a large fraction in a day, since the multiple *and* the profit both fall.

P/S (`ps_ratio`). Price / sales: `market_cap_musd` divided by `revenue_musd`. It says what you pay for each dollar of yearly sales.

- *Reading it.* It works where P/E does not: for a young company that spends its profit on growth, or for one that has a loss. It does not see costs, so a low P/S is only attractive if the company can turn sales into profit. Compare within an industry, where margins are alike: a P/S of 0.3 is normal for a retailer, 8 for a software company.
- *Effect.* A high P/S is a bet on future *margins*: the company must one day earn a lot of profit on each dollar of sales to justify the price. P/S and net margin together give the P/E: `P/E = P/S / net margin`.

Dividend yield (`dividend_yield_pct`). The yearly dividend per share as a percentage of the share price. A company paying 2 $ a year on a 50 $ share yields 4 %. A dividend is a part of the profit paid out to owners in cash.

- *Reading it.* A yield of 0 means the company keeps its profit to reinvest, which is normal for companies that grow fast. A steady, moderate yield is the sign of a mature business with more cash than it can use. A *very* high yield is a warning: it is often the result of a share price that has fallen because the market expects the dividend to be cut.
- *Effect.* The yield draws a certain kind of owner, pension funds and income investors, who hold shares for years and so make the price steadier. A company that cuts its dividend sends a strong negative signal, and the price usually falls. A dividend also uses cash that could have gone to growth or to paying down debt, so the choice is a statement about the company's prospects.

Debt / equity (`debt_to_equity`). Total debt divided by shareholders' equity (the owners' part of

the balance sheet). A value of 1.5 means that for each dollar the owners have put in, the company has borrowed 1.5.

- *Reading it.* Above 1 the company is financed mostly by lenders. That is normal for banks, utilities and real estate companies, whose income is stable and can carry debt, and a warning for a company whose sales rise and fall.
- *Effect.* Debt is *leverage*. It makes the owners' return larger in good years, since the profit is shared by fewer owners' dollars, and the loss larger in bad ones. Interest must be paid whatever the sales are, so a highly indebted company has little room for a bad year, and may have to sell assets or issue shares at a poor price to survive. A heavy debt also weakens the credit rating, and lenders then demand more interest. A company with little debt has the opposite strength: it can buy a weakened rival or invest through a downturn.

Credit rating (`credit_rating`). A rating agency's opinion of the chance that the company repays its debt in full and on time. The scale runs from `AAA` (safest) down through `AA`, `A`, `BBB`, `BB`, `B` and `CCC` to default, with `+` and `-` for fine steps. `BBB-` and above is *investment grade*; `BB+` and below is *speculative grade*, or "high yield".

- *Reading it.* Many pension funds and insurers are only allowed to hold investment-grade debt, so the line between `BBB-` and `BB+` is the most important one on the scale.
- *Effect.* The rating sets the interest a company pays when it borrows: a `AAA` company pays little over the government's rate, a `B` company several percentage points more. That cost reaches the profit and so the P/E. A downgrade, especially below investment grade, forces some holders to sell, raises borrowing costs and can lower the share price at once. A strong rating is a competitive weapon: the company can borrow cheaply to build, buy or wait out a price war that its weaker rivals cannot.

Financials

The figures are for the last fiscal year, the year named in the section's title. They follow the order of the income statement: sales at the top, profit at the bottom, and the cash that really comes out of it.

Revenue (`revenue_musd`). The money the company took in from customers during the year, before any cost is deducted. Also called sales or turnover. It is the "top line" of the income statement.

- *Reading it.* It measures the size of the business, independent of how well it is run. Its value is in its *trend* and in what it is compared with: revenue per employee (`revenue_musd` against `employees`) shows how productive the staff are.
- *Effect.* Revenue is the ceiling for everything else: no cost cutting makes a profit bigger than the sales. Large revenue gives scale, which spreads fixed costs over more units, and buying power over suppliers. It is a main part of a company's market position: a leader is usually the largest seller in its market.

Revenue growth (`revenue_growth_pct`). This year's revenue against last year's, in percent. Growth of 8 means sales are 8 % higher.

- *Reading it.* Compare it with the industry's growth. Faster than the market means the company is taking market share; slower means it is losing it, even if its sales still rise. Negative means the business is shrinking.
- *Effect.* Growth is the main reason one company deserves a higher P/E than another: the same profit is worth more if it is growing. It is also the most important driver of the share price, since the market prices the *change*. A fast-growing company, however, spends cash on staff, stock and equipment before the sales come, so it often has low free cash flow or needs to borrow.

Growth bought by selling at a loss is not worth the same as growth that earns a profit.

Net income (`net_income_musd`). The profit: what remains of revenue after every cost, interest and tax. The "bottom line", and the *E* in P/E. A negative value is a loss.

- *Reading it.* It is the number the owners can ultimately claim. It is shaped by accounting choices, such as how fast equipment is written off, so it can be flattered. This is why it is read together with free cash flow.
- *Effect.* Profit funds dividends, buy-backs and the investment that makes growth, so it feeds the future. Losses use up the company's reserves; a company in loss with a high debt is in the most danger. The valuation follows it directly: with the market cap fixed, a profit that rises by a tenth lowers the P/E by a tenth, and the share price tends to rise to restore the usual multiple.

Net margin (`net_margin_pct`). Net income divided by revenue, in percent: how many cents of profit each dollar of sales leaves.

- *Reading it.* Margins are set by the structure of an industry. Selling food in a shop earns a few percent, software and drugs earn 20 % or more because a copy costs little to make. Compare only like with like.
- *Effect.* A higher margin than competitors usually means a real advantage: a brand that lets the company ask more, a cost advantage from scale, or a protected product. It also gives a *cushion*: a company with a 25 % margin survives a price war that would bankrupt one with 3 %. A margin that is falling while sales rise is a warning of rising costs or falling prices.

Operating margin (`operating_margin_pct`). Operating income divided by revenue, in percent. Operating income is the profit from running the business itself, *before* interest and tax.

- *Reading it.* It isolates the quality of the operation from the way the company is financed and taxed. Compare it with the net margin: if the gap is large, interest or tax takes much of the profit. A highly indebted company can have a good operating margin and a poor net margin.
- *Effect.* It shows pricing power and cost control. Management can improve it by raising prices, cutting costs or selling more of the high-margin products. A company with a rising operating margin is getting more out of each dollar of sales, and the market usually rewards that with a higher multiple. Because it comes before interest, it also shows how much room there is to pay the debt: operating income that is a few times the interest is safe.

Free cash flow (`free_cash_flow_musd`). The cash the business generates after it has paid for the investment needed to keep it going: equipment, buildings, software. Roughly, the cash from operations less capital spending.

- *Reading it.* Profit is an accounting figure and cash is a fact. A company can show a profit and run out of cash, if customers pay late or if it must keep buying equipment. Free cash flow well below net income, year after year, is a warning. A negative value is normal for a young company that invests ahead of growth, and a concern for a mature one.
- *Effect.* Free cash flow is what can actually be paid to the owners or the lenders: dividends, buy-backs of shares, debt repayment, or the purchase of another company. Many analysts value a company by the cash it will produce in the future, so it is at the heart of the valuation. A company with strong free cash flow is independent: it does not need the markets to survive a bad period, which is a real strength against rivals that do.

Position

These four lines are text. They describe the competitive situation that the figures above are the result of, and the one that will decide them next.

Key market (`key_market`). Where most of the revenue comes from: a country, a customer group or a kind of product.

- *Reading it.* It tells you what the company's results depend on. A company with one market is exposed to that market's cycle, regulation and competition; one that sells in many markets is steadier.
- *Effect.* Concentration is a risk that lowers the multiple investors will pay, and also an advantage when the market is large and growing, since the company can specialise. News about the market, such as a new law or a change in demand, moves every company that depends on it, which is why the news of the simulation is often given by industry.

Market position (`market_position`). What sets the company apart from its rivals: its unique position.

- *Reading it.* Look for a lasting advantage that others cannot easily copy: a leading brand, the lowest cost, a patent, a network of users that grows more valuable with each new one, or high costs for a customer to switch. Economists call such an advantage a *moat*.
- *Effect.* A wide moat is what makes high margins last, and so what justifies a high P/E. A company without one competes on price and its margin tends to fall towards the industry's average. Leaders can set prices; followers must accept them.

Competitors (`competitors`). The tickers of the main listed rivals, separated by commas.

- *Reading it.* Use it to find the group to compare with. Look the tickers up (`pm-compdb-cli show` or `list`) and put the margins, growth and P/E side by side. A company that is better on all three than its rivals either deserves a higher P/E or is already priced for it.
- *Effect.* A competitor's news is the company's news. A rival's price cut, new product or bankruptcy changes the company's own prospects, and in the simulation the shares of an industry tend to move together for this reason.

Key risk (`key_risk`). The biggest single threat to the business.

- *Reading it.* It names what to watch. Typical risks are the loss of a large customer, a change in regulation, a new technology, a dependence on a single supplier or country, or a heavy debt that falls due.
- *Effect.* The market prices risk. When the risk is realised, the share price falls at once and often more than the direct damage, because the market also doubts everything else. A company that is open about a risk and has a plan for it is valued higher than one that is surprised by it.

Management

CEO (`ceo_name`, `ceo_since`; the tenure is `ceo_tenure_years`). The chief executive, who sets the strategy and runs the company day to day, and the year they took office.

- *Reading it.* A new CEO often brings a change of direction: new products, cost cuts, a sale of parts of the business. A long tenure means stability and a leader who is trusted, or one who has stayed past the point of being useful.
- *Effect.* A change of CEO is one of the events that most often moves a share price, up when the market thought the old one was failing, down when the departing one was highly regarded. For a company whose value depends on a founder's vision, the succession is a risk in itself.

Years in the industry (`ceo_industry_years`). How long the CEO has worked in this industry, in this company or others.

- *Reading it.* A long career means the CEO has lived through several cycles of the industry and knows its customers, suppliers and rivals. A short one means a fresh view, and a risk of

mistakes that experience would avoid.
- *Effect.* It is a soft figure, and no guarantee: it is one input to the market's trust in management. It matters most in industries with long cycles (energy, mining, shipbuilding), where decisions made today pay off or fail years later.

Employees (`employees`). The number of people who work for the company.

- *Reading it.* It is a second measure of size, and, set against revenue, of efficiency. Revenue per employee varies enormously: it is high in software and trading, and low in retail and services.
- *Effect.* Wages are usually a company's largest cost, so the head count decides how easily it can cut costs, and how exposed it is to rising wages. A company that grows its revenue faster than its head count is gaining productivity, which feeds directly into the operating margin.

History

Founded (`founded_year`; the age is `years_in_business`). The year the company was started.

- *Reading it.* Age is a rough guide to maturity. An old company has survived several cycles and often holds a large share of a slow market. A young one has less proof, and more room to grow.
- *Effect.* Longevity builds a brand and relations with customers and suppliers, which are parts of the moat. It also brings habits and old products that a young rival does not have, and the young rival can attack with a new technology.

IPO (`ipo_year`, `ipo_valuation_musd`). The initial public offering: the year the company first sold its shares to the public on an exchange, and the market value put on it then.

- *Reading it.* Set the value at the IPO against today's `market_cap_musd`: it shows how much value the company has created since it went public. A company that has grown many times over has delivered on its promise. One that is worth less than at the IPO has destroyed value for the investors who bought then.
- *Effect.* An IPO turns a private company into one with a price that anyone can see, and gives it a way to raise money by selling more shares and to pay its staff and acquisitions with them. In return it must publish its accounts and answer to the market every quarter.

Origin (`origin`, with `founders` and `founder_story`). How the company began: `FOUNDER`, `CO_FOUNDERS`, `FAMILY`, `MERGER`, `SPIN_OFF`, `PRIVATISATION` or `COOPERATIVE`.

- *Reading it.* The start shapes the company's character. A founder's company tends to be driven by one vision. A family firm thinks in generations, and may be slow to take outside advice. A merger must join two cultures and cut the overlap. A spin-off starts with a clear business but has to stand without its parent. A privatised company comes from a state monopoly and often carries its habits. A cooperative is owned by its members, who may care more about price than profit.
- *Effect.* Origin does not change the figures directly, but it explains them and warns of the risks: control that stays with a family or founder can protect the company from short-term pressure and also from change, and a merger is only worth what the integration achieves.

Putting the figures together

The same company can be read from three sides, and the profile gives you a figure for each.

Question	Where to look	A strong company shows
What is it worth, and is that fair?	Market cap, P/E, P/S, dividend yield	A multiple that its growth and margins justify

Question	Where to look	A strong company shows
Can it carry on and pay its way?	Debt / equity, credit rating, free cash flow, net income	Moderate debt, a good rating, cash above its needs
Does it have an edge?	Revenue growth, margins, market position, competitors	Faster growth and higher margins than its rivals, for a reason that lasts

A few combinations tell more than any single figure:

- *High P/E with low growth and falling margins:* the price is not supported. A fall in the share price is likely when the market notices.
- *Low P/E with high debt:* it may be cheap for a reason. Check the credit rating and free cash flow before calling it a bargain.
- *Profit but a negative free cash flow:* the profit is not turning into cash. Look for growing receivables, stock or heavy investment.
- *High margins, falling growth:* the moat is at work but the market is saturated. Look for a rising dividend as the company has less to invest.
- *Falling margins, rising sales:* the company is buying growth, or competitors are forcing the price down.

To try these on the simulation's companies, use `list` with filters and sorting:

```
# Cheap and profitable: low P/E, high net margin
pm-compdb-cli list -c symbol,name,pe_ratio,net_margin_pct -w "pe_ratio<12" -w "net_margin_pct>15" -s pe_ratio

# Highly leveraged companies and what they are rated
pm-compdb-cli list -c symbol,name,debt_to_equity,credit_rating,free_cash_flow_musd -w "debt_to_equity>2" -s debt_to_

# Growth at a price: fast growers ranked by P/S
pm-compdb-cli list -c symbol,name,revenue_growth_pct,ps_ratio -w "revenue_growth_pct>20" -s ps_ratio --desc

# A company against its competitors
pm-compdb-cli show NVDA
pm-compdb-cli list -c symbol,pe_ratio,operating_margin_pct,revenue_growth_pct -w "industry=Semiconductors" -s pe_rati
```

2.26 pm-new-symbol / pm-ipo — List a New Symbol

Lists a new symbol (an IPO) in an engine configuration: appends it to `symbols:`, gives it its reference price and, normally, a seed quote from the market maker, validates the whole edited file and, when that file is the deployed source, redeploys it. `pm-ipo` is an alias for the same command.

```
pm-new-symbol --symbol NEWCO --ipo-price 25.00 --outstanding-shares 40000000
pm-new-symbol --config my_config.yaml --symbol NEWCO --ipo-price 25.00 --outstanding-shares 40000000
```

Startup options:

Flag	Default	Description
`--config PATH`	the deployed configuration's source	Authored YAML to edit. A file that is not the deployed source is edited only; run `pm-config-deploy` on it yourself

Flag	Default	Description
`--symbol NAME`	required	New symbol, 1–8 of `A-Z 0-9 . _`
`--ipo-price PRICE`	required	Offer price; the book's reference price for collars and circuit breakers
`--outstanding-shares N`	required	Shares outstanding (for index market caps)
`--tick-decimals N`	`2`	Price grid, `0`–`8`
`--mm-gateway-id ID`	the only `MARKET_MAKER` gateway	Gateway that posts the seed quote
`--mm-bid-price`, `--mm-ask-price`	maximum allowed spread around the price	Explicit seed prices; give both or neither
`--mm-bid-qty`, `--mm-ask-qty`	`1000`	Seed quote sizes
`--mm-tif {DAY,GTC}`	`DAY`	Time-in-force of the seed quote
`--mm-seed-once` / `--no-mm-seed-once`	on	Skip the seed when a restored quote already exists
`--field KEY=YAML_VALUE`	none	Optional symbol section: `level`, `collar`, `order_limits` or `circuit_breaker`; repeatable, e.g. `--field 'collar={static_band_pct: 0.2}'`
`--version`	—	Print version and exit

Nothing is written until the edited file passes the same four verification layers as `pm-config-deploy`; a refusal leaves both the YAML and the deployed artifact untouched. It refuses while `pm-engine` runs (when editing the deployed source), when the deployed source has undeployed edits, when saved engine state already exists for the symbol, and when the file is only a copy or a bundled example.

See Listing a New Symbol for the full list of refusals, how the seed quote is priced and the IPO-day checklist.

2.27 pm-mm-bot — Autonomous Market-Maker Bot

Runs one autonomous market-maker gateway. Connects as a `MARKET_MAKER` participant, posts a two-sided quote per symbol, and automatically reprices on fills, mid-price drift, and session transitions.

One process is one gateway and can quote several symbols: repeat `--symbol`, give `--symbols A,B,C`, or use `--all-symbols`. Each symbol is isolated from the others' failures. To compete on the same symbol, or to keep symbols under separate gateway IDs, run several instances with different `--id-suffix` values.

```
pm-mm-bot --symbol AAPL [options]
pm-mm-bot --symbol AAPL --gap 0.10 --symbol TSLA --gap 0.50 --strategy inventory_skew --max-position 5000
pm-mm-bot --all-symbols
```

Per-symbol flags. `--symbol` may be repeated. A per-symbol flag (`--strategy`, `--gap`, `--qty`,

`--max-position`, `--drift-ticks`, `--tif`, `--anchor-sim`, the passive-strategy flags, …) applies to the symbol named *before* it, until the next `--symbol`. Flags given before the first `--symbol` are shared defaults for all symbols. A flag on the command line overrides the same key in a `--config` file in the same scope.

Startup options:

Flag	Default	Description
`--symbol`	one of `--symbol`, `--symbols`, `--all-symbols` or a config `symbols:` block is required	Instrument to make a market in (e.g. `AAPL`). Repeatable; per-symbol flags after it apply to it (see above)
`--symbols`	unset	Comma-separated symbols (e.g. `AAPL,MSFT`) that all share the same settings; shorthand for repeating `--symbol` with no per-symbol flags; mutually exclusive with `--symbol`
`--all-symbols`	off	Quote every symbol of the currently deployed configuration with the same settings; gateway ID defaults to `MM_ALL_<nn>`. Mutually exclusive with `--symbol`, `--symbols` and a config `symbols:` block
`--label`	derived from symbol(s)	Override the gateway-ID symbol segment (default: the single symbol, or `SYM1_SYM2_...`); mainly to keep a multi-symbol gateway ID short
`--config`	unset	YAML file with gateway-wide settings, shared `defaults` and per-symbol overrides (bot-level, not the engine config). See *Config file* below
`--strategy`	`symmetric`	Pricing strategy: `symmetric`, `inventory_skew` or `passive`
`--max-position`	unset	Net position (either direction) at which inventory skewing saturates; required when `--strategy inventory_skew`, an error with any other strategy
`--retreat-ticks`	`5`	`passive` only: how many ticks beyond its home price each side may step back to stay behind other traders
`--behind-ticks`	`1`	`passive` only: how many ticks behind other traders' best price a covered side quotes (must be 1)
`--min-cover-qty`	`1`	`passive` only: quantity other traders must show inside the retreat band before a side counts as covered
`--fade-ticks`	`2`	`passive` only: extra ticks a side steps back after it is filled; `0` disables fading
`--fade-sec`	`3.0`	`passive` only: seconds a fade lasts after a fill; `0` disables fading

Flag	Default	Description
`--gap`	`0.10`	Total spread in price units (bid at mid−gap/2, ask at mid+gap/2)
`--qty`	`500`	Quote size on each leg
`--id-suffix`	`01`	Running number for gateway ID (`MM_AAPL_01`)
`--gateway-id`	*derived*	Exact gateway ID (e.g. `MM01`); overrides `MM_<symbols>_<id-suffix>`
`--drift-ticks`	`3`	Reprice when mid moves by more than this many ticks (`passive`: when a side's target price does)
`--anchor-sim`	`0` (off)	Quote around `pm-market-sim`'s true value; share of the gap closed per model step (0–1)
`--reissue-delay-ms`	`200`	Milliseconds to wait after fill before re-issuing
`--tif`	`DAY`	Time-in-force for quote legs (`DAY` or `GTC`)
`--heartbeat-interval-sec`	`5.0`	Periodic live-quote check interval
`--startup-session-timeout-sec`	`5.0`	Max wait to learn the session phase (asked for, or a `session.state`)
`--bootstrap-timeout-sec`	`1.0`	Max wait for QBOOT reply
`--cancel-timeout-sec`	`1.0`	Max wait for cancel confirmation
`--shutdown-timeout-sec`	`2.0`	Max wait for cancel on SIGINT/SIGTERM
`--qlegs-reconcile-interval-sec`	`15.0`	Periodic QLEGS reconciliation interval
`--initial_min`	unset	Lower bound for random bootstrap price
`--initial_max`	unset	Upper bound for random bootstrap price
`--engine-pull`	`tcp://127.0.0.1:5555`	Engine PUSH/PULL address
`--engine-pub`	`tcp://127.0.0.1:5556`	Engine PUB address
`--log-level`	`WARNING`	Explicit level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` enables bot debug prints, `-vv` sets DEBUG)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

None. The bot is fully autonomous after startup.

Gateway ID convention:

The gateway ID is constructed as `MM_<label>_<suffix>`, where the label is the single symbol, `SYM1_SYM2_...` for several, or `ALL` with `--all-symbols` — for example, `pm-mm-bot --symbol AAPL --id-suffix 02` connects as `MM_AAPL_02`. `--gateway-id` overrides the whole name. Whichever way it is built, the ID must be pre-registered in `engine_config.yaml` with `role: MARKET_MAKER`.

Strategies:

Strategy	Quotes	Parameters
`symmetric`	bid at mid − gap/2, ask at mid + gap/2	`gap`, `drift_ticks`
`inventory_skew`	like `symmetric`, but shifts both prices against the current position so a long bot sells more eagerly; skew saturates at `max_position`	`max_position` (required)
`passive`	a backstop: steps `behind_ticks` behind other traders' best price, never tighter than its home price nor outside the `retreat_ticks` band, and fades `fade_ticks` for `fade_sec` after a fill. The mid comes from *other* traders' book only, so the bot never chases its own quote. If the symbol has a market-maker spread obligation, the band is narrowed so the widest quote still meets it	the five `passive` flags

Config file (`--config`): a YAML file with `version: 1` separates gateway-wide settings from per-symbol ones:

```
version: 1
gateway:
  label: TECH
  id_suffix: "01"
defaults:
  gap: 0.10
  qty: 500
symbols:
  AAPL: {}
  TSLA:
    gap: 0.50
    strategy: inventory_skew
    max_position: 5000
```

Unknown keys are rejected with a suggestion for the likely typo. A `logging:` block sets `level`, `target`, `file` and `failover_timeout_sec`.

Startup behaviour:

1. Opens ZMQ PUSH and SUB sockets
2. Sends `system.gateway_connect` and waits for `system.gateway_auth` ACK

3. Requests the symbol list and verifies the assigned symbol exists
4. Sends `QBOOT` (quote bootstrap) request — if an active quote already exists for `(gateway_id, symbol)`, adopts it instead of creating a duplicate
5. Sends `QLEGS` request to reconcile quote-leg mapping
6. Waits for `session.state` (exits if not received within timeout)
7. Resolves initial reference price and enters the quoting loop

Reference price resolution (in priority order):

1. Active quote from `QBOOT` (restart recovery)
2. Book mid-price (if other participants are already quoting)
3. Last trade price from `trade.executed` events
4. Bootstrap quote prices from `QBOOT` snapshot (inactive)
5. Random price from `[--initial_min, --initial_max]` range

State machine:

State	Meaning
`CONNECTING`	Opening ZMQ sockets
`AUTHENTICATING`	Waiting for gateway auth reply
`WAITING_FOR_SESSION`	Waiting for session to enter CONTINUOUS
`QUOTING`	Active two-sided quote is resting in the book
`REPRICING`	Mid-price drift detected; preparing new quote
`REISSUING`	Sending replacement quote after fill or drift
`PAUSED`	Session is not CONTINUOUS (auction, halt, closed)

Shutdown (Ctrl-C or SIGTERM):

1. Sends `quote.cancel` to remove the resting quote
2. Waits up to `--shutdown-timeout-sec` for cancel confirmation
3. Closes ZMQ sockets and exits

Messages sent (PUSH → :5555):

Topic	Purpose
`system.gateway_connect`	Authentication request
`system.symbols_request`	Request symbol list
`system.quote_bootstrap_request`	Request active quote state for adopt-or-create (QBOOT)
`system.quote_legs_request`	Request quote-leg mapping for reconciliation (QLEGS)
`quote.new`	Submit or replace a two-sided quote
`quote.cancel`	Cancel the active quote (shutdown or session change)

Messages subscribed (SUB from :5556):

Topic	Purpose
`system.gateway_auth.{GW_ID}`	Authentication reply

Topic	Purpose
`system.symbols.{GW_ID}`	Symbol list reply
`system.quote_bootstrap.{GW_ID}`	Bootstrap snapshot with active quote state
`system.quote_legs.{GW_ID}`	Quote-leg reconciliation reply
`book.{SYMBOL}`	Book updates — drives mid-price tracking and drift detection
`depth.{SYMBOL}`	Depth / imbalance metrics (microprice, mid_price) for the assigned symbol
`trade.executed`	Trade events — fallback reference price source
`order.fill.{GW_ID}`	Fill notifications on quote legs
`order.cancelled.{GW_ID}`	Cancel confirmations for quote legs
`quote.ack.{GW_ID}`	Quote accepted/rejected acknowledgements
`quote.status.{GW_ID}`	Quote lifecycle changes (inactivated, cancelled)
`session.state`	Session phase transitions (pause/resume quoting)
`circuit_breaker.halt.{SYMBOL}`	Circuit breaker halt for the assigned symbol
`circuit_breaker.resume.{SYMBOL}`	Circuit breaker resume for the assigned symbol

See Market-Maker Bot for usage examples, configuration guide, bootstrap details, and troubleshooting.

2.28 pm-valuation — IPO Valuation

Interviews the student about a fictive company, values it with a two-stage discounted cash flow, and simulates the IPO that prices it. The result can be listed with `pm-new-symbol`. It is a full-screen terminal application and joins no ZeroMQ socket.

```
pm-valuation
pm-valuation --case halvard --level intermediate
pm-valuation --load my_scenario.yaml --no-tui --export report.md
```

Startup options:

Flag	Default	Description
`--load FILE`	—	Scenario file to start from (mutually exclusive with `--case`)
`--case {halvard,tornfalk}`	—	Classroom case to start from
`--no-tui`	off	Print the report instead of interviewing
`--level {beginner,intermediate,advanced,expert}`	`beginner`	Interview detail; `F3` changes it in the application
`--market {se,us}`	scenario's, else `se`	Where the company lists: Sweden or the US; overrides the scenario

Flag	Default	Description
`--mode {deterministic,montecarlo,both}`	scenario's, else `both`	Scenarios, Monte Carlo, or both
`--draws N`	—	Monte Carlo draws
`--seed N`	—	Monte Carlo seed
`--save FILE`	—	Write the scenario to this file
`--with-defaults`	off	With `--save`: write every value, commented with its source
`--export FILE`	—	Write the report as Markdown
`--pdf FILE`	—	Write the report as a printable PDF (also `p` in the application)
`--paper {a4,letter}`	—	PDF page size
`--presets FILE`	bundled	Alternative presets file
`--list`	off	With `--no-tui`: list the priced IPO with `pm-new-symbol`
`--config FILE`	deployed source	With `--list`: the engine YAML that `pm-new-symbol` edits
`--version`	—	Print version and exit

See IPO Valuation for the finance behind the model and the interview, and the Training Guide's IPO chapter for the classroom exercise.

2.29 pm-ralf-gwy — Post-Trade Dissemination Gateway

Runs the external machine-facing post-trade dissemination gateway that publishes RALF over TCP for clearing, drop-copy, and audit consumers.

```
pm-ralf-gwy [--bind 0.0.0.0] [--port 5580] [--engine-pub tcp://127.0.0.1:5556] [--log-level LEVEL] [-v|-vv] [
  [--log-target TARGET] [--log-file PATH] [--log-failover-timeout SEC]
```

Startup options:

Flag	Default	Description
`--bind`	from config / `0.0.0.0`	TCP bind address for external clients
`--port`	from config / `5580`	TCP listen port for RALF clients
`--engine-pub`	`tcp://127.0.0.1:5556`	Engine PUB address consumed by the gateway
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`

Flag	Default	Description
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	30 (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

No terminal input. External parties connect over TCP and send RALF lines:

- `HELLO|...`
- `SUB|...`
- `UNSUB|...`
- `PING`
- `EXIT`

Messages subscribed (SUB from :5556):

Topic	Purpose
`trade.executed`	Source for `EXEC` post-trade events
`system.eod`	Source for `EOD` summary markers

Messages published (TCP RALF feed):

Message type	Purpose
`WELCOME`	Session acceptance and capabilities
`SNAP`	Subscription/recovery baseline
`EXEC`	Live execution dissemination
`EOD`	End-of-day summary marker
`HB`	Heartbeat
`PONG`	Ping reply
`ERR`	Protocol/entitlement/replay errors
`EXIT`	Session termination reason

For external usage and examples see Post-Trade Dissemination and RALF Protocol Reference.

2.30 pm-ralf-spy (RALF Protocol Spy)

`pm-ralf-spy` is a read-only CLI client for `pm-ralf-gwy`: it opens a RALF TCP session under a chosen `--role` (`CLEARING`, `DROP_COPY`, or `AUDIT`), subscribes to whatever channels/symbols you ask for, and prints every line it receives — human-readable or as JSON — so you can see exactly what the protocol sends without writing a client. It never mutates exchange state, and any number of instances can run at once (e.g. one per terminal, each with a different `--role`/`--channels`/`--symbols` filter).

Since it never sends anything on its own after the initial subscription, it sends a periodic `PING` (`--ping-interval`, default 60s) to keep the gateway from dropping it as an idle client.

Startup options:

Flag	Default	Description
`--host`	`127.0.0.1`	`pm-ralf-gwy` TCP host
`--port`	`5580`	`pm-ralf-gwy` TCP port
`--client-name`	`ralf-spy-<pid>`	`HELLO|CLIENT=` identifier reported in gateway logs
`--role`	`AUDIT`	`HELLO|ROLE=` to authenticate as: `CLEARING`, `DROP_COPY`, or `AUDIT`
`--channels`	*	Comma-separated channels; * = every channel `--role` is entitled to
`--symbols`	*	Comma-separated symbols; * = every symbol
`--lastseq`	`0`	Requests replay on connect via `HELLO|LASTSEQ=N` (`0` = no replay)
`--ping-interval`	`60`	Seconds between `PING` keepalives sent to the gateway; `0` disables
`--format`	`human`	`human` or `json` output
`--count`	`0`	Exit after N data-carrying lines (`0` = run until Ctrl-C)
`--raw`	off	Print raw protocol bytes instead of decoded fields
`--no-color`	off	Disable ANSI colour in output
`--show-heartbeats`	off	Include `HB` heartbeat lines in output (hidden by default)
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`

Flag	Default	Description
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

See RALF Protocol Spy (pm-ralf-spy) for full usage.

2.31 pm-dc-gwy — Drop-Copy TCP Gateway

Runs a small external-facing TCP gateway that relays the engine's drop-copy feed (`DropCopyPublisher`, `PUB :5557`) to plain-TCP clients using the DC1 text protocol — no ZeroMQ dependency required on the client side. It is the TCP counterpart to `pm-dc-spy`: both subscribe internally to `drop_copy.event.<gateway_id>`, but `pm-dc-gwy` fans events out to any number of concurrently connected external clients instead of printing to a terminal. There is no authentication, entitlement model, or replay-by-sequence — any client may `HELLO` with any gateway `ID` and receive that gateway's live fills for as long as it stays connected.

```
pm-dc-gwy [--bind 0.0.0.0] [--port 5590] [--engine-dc-pub tcp://127.0.0.1:5557] [--log-level LEVEL] [-v|-vv] [-q] \
  [--log-target TARGET] [--log-file PATH] [--log-failover-timeout SEC]
```

Startup options:

Flag	Default	Description
`--bind`	from config / `0.0.0.0`	TCP bind address for external clients
`--port`	from config / `5590`	TCP listen port for DC1 clients
`--engine-dc-pub`	`tcp://127.0.0.1:5557`	Engine drop-copy PUB address consumed by the gateway
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

No terminal input. External parties connect over TCP and send DC1 lines:

- `HELLO|CLIENT=..|PROTO=DC1|ID=..`
- `PING`

- `EXIT`

Messages subscribed (internal SUB from :5557, refcounted per gateway ID requested by connected clients):

Topic	Purpose
`drop_copy.event.<gateway_id>`	Source for `DC_FILL` relay lines

Messages published (TCP DC1 feed):

Message type	Purpose
`WELCOME`	Session acceptance
`DC_FILL`	Unsolicited fill relay
`HB`	Heartbeat
`PONG`	Ping reply
`ERR`	Protocol errors

For external usage and examples see Drop-Copy TCP Gateway.

2.32 pm-dc-spy (Drop-Copy Spy)

`pm-dc-spy` is a read-only CLI client for the engine's drop-copy feed (`DropCopyPublisher`, `PUB :5557`): it opens a plain `zmq.SUB` connection — no handshake, no heartbeat, unlike CALF/RALF — subscribes to fill events for one gateway or all gateways, and prints every message it receives, human-readable or as JSON. It never mutates exchange state, and any number of instances can run at once (e.g. one per terminal, each with a different `--gateway` filter).

Startup options:

Flag	Default	Description
`--host`	`127.0.0.1`	Drop-copy `PUB` socket host
`--port`	`5557`	Drop-copy `PUB` socket port
`--gateway`	*(none)*	Only show fills for this gateway; default subscribes to all gateways
`--replay-of`	*(none)*	Also subscribe to `drop_copy.replay.<RECIPIENT_ID>` to observe `DropCopyPublisher.replay()` calls
`--format`	`human`	`human` or `json` output
`--count`	`0`	Exit after N messages (`0` = run until Ctrl-C)

Flag	Default	Description
`--raw`	off	Print raw bytes instead of decoded fields
`--no-color`	off	Disable ANSI colour in output
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

See Drop-Copy Spy (pm-dc-spy) for full usage.

2.33 pm-index — Index Calculation Process

Subscribes to trade events from the engine and maintains one or more configurable cap-weighted market indices in real time. Publishes live index values on a dedicated ZMQ PUB socket so `pm-md-gwy` can forward them to external subscribers over the CALF `INDEX` channel.

```
pm-index [--reset] [--log-level LEVEL] [-v|-vv] [-q] [--log-target TARGET] [--log-file PATH] [--log-failover-timeout
```

Startup options:

Flag	Default	Description
`--reset`	off	Delete persisted state files and reinitialise all indices from config
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`

Flag	Default	Description
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

None. `pm-index` is a long-running background process after startup.

Startup behaviour: 1. Reads all `indices:` entries from `engine_config.yaml` 2. For each index: loads divisor and last prices from the state file if present, or initialises fresh from constituent reference prices and `base_value` 3. Validates that every constituent symbol appears in `symbols:` with a positive `outstanding_shares` 4. Writes an `INIT` structural/audit history record for each index that starts without prior state 5. Binds ZMQ PULL :5559 for operator commands and history requests 6. Binds ZMQ PUB :5558 for broadcasting `index.update` messages 7. Subscribes to engine PUB :5556 for `trade.executed`, `session.state`, and `system.eod` 8. Enters the poll loop

Shutdown (Ctrl-C): 1. Flushes all pending structural/audit JSONL history writes 2. Closes ZMQ sockets

Messages subscribed (SUB from :5556):

Topic	Purpose
`trade.executed`	Drives index recalculation for constituent symbols
`session.state`	Tracks intraday OHLC reset points and triggers EOD finalisation on `CLOSED`
`system.eod`	Alternate EOD trigger — publishes the final index.update for the day

Messages received (PULL :5559):

Topic	Purpose
`index.history_request`	Returns structural/audit records (`INIT`, `CORP_ACTION`, `ADD_CONSTITUENT`, `DELIST`) from the JSONL history file — not level/EOD data, which lives in pm-stats
`index.corp_action`	Applies a stock split, cash dividend, or shares-issuance adjustment
`index.constituent_change`	Adds or removes a constituent symbol from an index basket

Messages published (PUB :5558):

Topic	Purpose
`index.update`	Current index level, OHLC, aggregate cap, divisor, and session state — also consumed by `pm-stats` for level/EOD history
`index.history.{GW_ID}`	Structural/audit history query response addressed to the requesting gateway
`index.corp_action_ack.{GW_ID}`	Corporate action applied / rejected acknowledgement

Topic	Purpose
`index.constituent_change_ack.{GW_ID}`	Constituent add or delist accepted / rejected acknowledgement
`index.error.{GW_ID}`	Generic error for malformed or rejected requests

Persistence:

File	Purpose
`data/indexes/<ID>_state.json`	Divisor, last prices, and intraday OHLC checkpoint — rewritten after each EOD and corporate action
`data/indexes/<ID>_history.jsonl`	Append-only JSONL **structural/audit trail** with `INIT`, `CORP_ACTION`, `ADD_CONSTITUENT`, and `DELIST` records only — level and EOD history is recorded by `pm-stats` in `data/stats.db` instead (see the pm-stats section)

See Market Index (pm-index) for configuration details, calculation explanation, and corporate action procedures.

2.34 pm-api-gwy — REST/WebSocket API Gateway

Exposes EduMatcher order entry, order management, reference data, history, and market data over REST/JSON and WebSocket. It is not a second matching engine — it translates HTTP and WebSocket requests into the same ZMQ messages used by the interactive `pm-alf-console` terminal. Reads configuration from the `api_gateways:` section of `engine_config.yaml`.

```
pm-api-gwy [--instance NAME] [--host ADDR] [--port PORT] [--engine-host HOST] [--stats-db PATH] \
  [--clearing-db PATH] \
  [--log-level LEVEL] [-v|-vv] [-q] [--log-target TARGET] [--log-file PATH] [--log-failover-timeout SEC]
```

Startup options:

Flag	Default	Description
`--instance NAME`	auto-selected when only one entry exists	Named `api_gateways` entry to run; required when more than one entry is configured
`--host ADDR`	config value	Override HTTP bind address
`--port PORT`	config value	Override HTTP listen port
`--engine-host HOST`	config value	Override engine host for ZMQ connections (cross-host deployments)
`--stats-db PATH`	config value	Path to `data/stats.db` for `/history/*` endpoints
`--clearing-db PATH`	config value	Path to `data/clearing.db`, opened read-only for the session/MTD/YTD realized P&L on `/positions`. Optional: without it those three fields report `null`
`--log-level LEVEL`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`

Flag	Default	Description
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	30 (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

Expected runtime input arguments:

No terminal input. External clients connect via HTTP or WebSocket.

Startup behaviour: 1. Reads the named `api_gateways.<NAME>` entry from `engine_config.yaml` 2. Exits immediately if `enabled: false` 3. Sends `system.gateway_connect` to the engine and waits for `system.gateway_auth` ACK 4. Starts the uvicorn HTTP server and enters the request-serving loop

REST endpoints (base path `/api/v1`):

Method	Path	Key type	Purpose
`POST`	`/orders`	trading	Submit an order
`DELETE`	`/orders/{id}`	trading	Cancel an order
`PATCH`	`/orders/{id}`	trading	Amend price and/or quantity
`GET`	`/orders`	trading	List live orders
`POST`	`/oco`	trading	Submit OCO pair
`POST`	`/combos`	trading	Submit combo order
`POST`	`/quotes`	trading	Submit two-sided quote
`DELETE`	`/quotes/{symbol}`	trading	Cancel active quote
`POST`	`/mass-cancel`	trading	Cancel all or symbol-scoped exposure
`GET`	`/symbols`	trading	Instrument metadata for the caller
`GET`	`/session`	any	Session phase, timezone, trading date
`GET`	`/positions`	trading	Net positions by symbol
`GET`	`/history/orders`	trading	Historical order lifecycle events
`GET`	`/history/fills`	trading	Historical fills
`GET`	`/history/trades`	any	Public trade log from `pm-stats`
`GET`	`/history/daily`	any	Daily OHLCV from `pm-stats`

WebSocket endpoints:

Path	Key type	Stream
`/ws/private`	trading	Private fills, acks, expiries — filtered to the connected gateway
`/ws/market-data`	any	Public order book snapshots and trade events
`/ws/admin/monitor`	ADMIN	Cross-gateway monitor feed
`/ws/admin/sim`	ADMIN	`pm-market-sim` true values and heartbeat

This is a summary. The full contract is the REST API Reference and the WebSocket API Reference.

Authentication:

- REST: `Authorization: Bearer <api_key>` header
- WebSocket: send `{"api_key": "<key>"}` as the first JSON message

Trading keys (`gateway_id` set to a gateway ID) can reach order-entry endpoints and private event streams. Read-only keys (`gateway_id: null`) are limited to the public endpoints: reference data, session, halts, public history and market data.

Tip

When `swagger_enabled: true` in config, browse `http://127.0.0.1:<PORT>/docs` for interactive endpoint documentation and a live try-it UI.
Messages sent (PUSH → :5555, the engine):

Group	Topics
Session	`system.gateway_connect`, `system.gateway_heartbeat`, `system.gateway_disconnect`
Order entry	`order.new`, `order.cancel`, `order.amend`, `order.oco`, `order.oco_cancel`, `order.combo`, `order.combo_cancel`, `quote.new`, `quote.cancel`, `risk.kill_switch` (`/mass-cancel`)
Queries	`order.orders_request`, `system.symbols_request`, `system.reference_request`, `system.session_state_request`, `system.session_schedule_request`, `system.position_request`, `system.halt_status_request`, `system.quote_bootstrap_request`, `system.quote_legs_request`, `system.gateways_request`, `system.risk_state_request`, `system.volume_request`, `book.snapshot_request`, `order.price_level_orders_request`

Admin	`session.transition`, `system.reference_reload`, `risk.symbol_halt`, `risk.symbol_resume`, `risk.cancel_symbol`, `risk.kill_switch_gateway`, `risk.kill_switch_global`, `risk.circuit_breaker_halt_all`, `risk.circuit_breaker_resume_all`, `risk.force_uncross`

To `pm-index` (its own PULL socket): `index.rebalance`, `index.history_request`.
Messages subscribed: every topic the engine publishes (SUB from :5556 with an empty filter). The gateway routes each event itself: one addressed to a gateway goes to that gateway's private sockets, the rest to market data, and all of it to the admin monitor (see the WebSocket API Reference). It also subscribes to `pm-index` and `pm-market-sim` for index history and the simulator's state, values and news.
See API Gateway (REST/WebSocket) for configuration, keys, caches and operation, and Protocol Support Library Examples for client code.

2.35 pm-admin — Interactive Admin Console

An interactive REPL for sending operational commands to a running engine without needing a full gateway session.

```
pm-admin --id <ADMIN_GW_ID> [--log-level LEVEL] [-v|-vv] [-q] [--log-target TARGET] [--log-file PATH] [--log-
```

Startup options:

Flag	Required	Description
`--id`	Yes	ADMIN gateway ID configured in `engine_config.yaml` (e.g. `GW_ADMIN`)
`--log-level`	No	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG` (default `WARNING`)
`-v` / `--verbose`	No	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	No	Reduce output to warnings/errors
`--log-target`	No	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file` (default `server`)
`--log-file`	No	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	No	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable (default `30` from config)
`--version`	No	Print version and exit

Expected runtime input arguments:

- `HALT`, `RESUME`
- `HALT_SYM|SYM=<sym>`, `RESUME_SYM|SYM=<sym>`, `CANCEL_SYM|SYM=<sym>`
- `REOPEN|SYM=<sym>[|PRICE=<px>][|DRY_RUN=1][|NOTE=<text>]`
- `KILL|GW=<gw>[|SYM=<sym>]`
- `KICK|GW=<gw>[|REASON=<text>]`

- `QCANCEL|GW=<gw>|SYM=<sym>`
- `BOOK|SYM=<sym>`
- `ORDERS|GW=<gw>`
- `LEVEL|SYM=<sym>[|PRICE=<px>]`
- `SYMBOLS`
- `SESSION|STATE=<PRE_OPEN|OPENING_AUCTION|CONTINUOUS|CLOSING_AUCTION|CLOSED>`
- `SESSION_STATUS`, `SCHEDULE`, `GATEWAYS`, `VOLUME`
- `POSITION|GW=<gw>[|SYM=<sym>,..]`
- `HELP`, `EXIT`, `QUIT`

Messages sent (PUSH → :5555):

Topic	Purpose
`system.gateway_connect`	Authentication request (once at startup)
`risk.circuit_breaker_halt_all`	`HALT` — halt all symbols (requires ADMIN gateway role)
`risk.circuit_breaker_resume_all`	`RESUME` — resume all halted symbols (requires ADMIN gateway role)
`risk.symbol_halt`	`HALT_SYM` — halt one symbol
`risk.symbol_resume`	`RESUME_SYM` — resume one symbol
`risk.cancel_symbol`	`CANCEL_SYM` — cancel all resting orders on one symbol
`risk.force_uncross`	`REOPEN` — force a symbol to uncross now and clear any halt
`risk.kill_switch`	`KILL` — cancel all (or symbol-scoped) orders/quotes for a gateway
`system.gateway_disconnect`	`KICK` — forcefully disconnect a gateway (no ack published)
`quote.cancel`	`QCANCEL` — cancel a gateway's active two-sided quote
`book.snapshot_request`	`BOOK` — request a book snapshot
`order.orders_request`	`ORDERS` — request a gateway's resting orders
`order.price_level_orders_request`	`LEVEL` — request every resting order at a symbol (optionally one price level)
`system.symbols_request`	`SYMBOLS` — request the instrument list
`session.transition`	`SESSION` — request a session-phase transition
`system.session_state_request`	`SESSION_STATUS` — request the current session state
`system.session_schedule_request`	`SCHEDULE` — request the configured schedule
`system.gateways_request`	`GATEWAYS` — request gateway connection states
`system.volume_request`	`VOLUME` — request the daily volume summary
`system.position_request`	`POSITION` — request a gateway's net position snapshot

Messages subscribed (SUB from :5556):

Topic	Purpose
`system.gateway_auth.{GW_ID}`	Authentication reply
`risk.circuit_breaker_halt_all_ack.{GW_ID}`	`HALT` acknowledgement

Topic	Purpose
`risk.circuit_breaker_resume_all_ack.{GW_ID}`	`RESUME` acknowledgement
`risk.symbol_halt_ack.{GW_ID}`	`HALT_SYM` acknowledgement
`risk.symbol_resume_ack.{GW_ID}`	`RESUME_SYM` acknowledgement
`risk.cancel_symbol_ack.{GW_ID}`	`CANCEL_SYM` acknowledgement
`risk.force_uncross_ack.{GW_ID}`	`REOPEN` acknowledgement, with indicative/printed price
`risk.kill_switch_ack.{GW_ID}`	`KILL` acknowledgement
`quote.ack.{GW_ID}`	`QCANCEL` acknowledgement
`book.{SYMBOL}`	`BOOK` reply — book snapshot
`order.orders.{GW_ID}`	`ORDERS` reply — resting order list
`order.price_level_orders.{GW_ID}`	`LEVEL` reply — per-order book composition
`system.symbols.{GW_ID}`	`SYMBOLS` reply
`system.session_status.{GW_ID}`	`SESSION_STATUS` reply
`system.session_schedule.{GW_ID}`	`SCHEDULE` reply
`system.gateways.{GW_ID}`	`GATEWAYS` reply
`system.volume.{GW_ID}`	`VOLUME` reply
`system.position_snapshot.{GW_ID}`	`POSITION` reply
`session.state`	Session phase changes (broadcast; also the `SESSION` reply)

2.36 pm-admin-cli — CLI Admin Commands

A non-interactive alternative to `pm-admin` for scripting or single-shot operational commands.

```
pm-admin-cli <command> [options]
```

Each invocation sends one command to the engine, waits for an acknowledgement, prints the result, and exits. Suitable for shell scripts and CI automation.

Startup options:

Flag	Required	Description
`--id <GW_ID>`	Yes	ADMIN gateway ID
`--push <ADDR>`	No	Engine PULL address (default from config)
`--sub <ADDR>`	No	Engine PUB address (default from config)
`--timeout <MS>`	No	Ack timeout in milliseconds (default: `3000`)
`--format {text,json}`	No	`text` (default) prints a human-readable result; `json` prints it machine-readable to stdout, for scripts
`--version`	No	Print version and exit

Expected runtime input arguments:

Subcommands and required arguments:

Command	Required arguments	Optional arguments	Purpose
`halt`	none	none	Exchange-wide halt
`resume`	none	none	Exchange-wide resume
`halt-sym`	`--sym <SYMBOL>`	none	Halt one symbol
`resume-sym`	`--sym <SYMBOL>`	none	Resume one symbol
`cancel-sym`	`--sym <SYMBOL>`	none	Cancel all resting orders on one symbol
`reopen`	`--sym <SYMBOL>`	`--price <PRICE>`, `--dry-run`, `--note <TEXT>`	Force a symbol to uncross now and clear any halt
`kill`	`--gw <GW_ID>`	`--sym <SYMBOL>`	Cancel all (or symbol-scoped) orders/quotes for gateway
`kick`	`--gw <GW_ID>`	`--reason <TEXT>`	Disconnect a gateway
`qcancel`	`--gw <GW_ID> --sym <SYMBOL>`	none	Cancel active quote for gateway on symbol
`book`	`--sym <SYMBOL>`	none	Fetch book snapshot
`orders`	`--gw <GW_ID>`	none	List resting orders for gateway
`level`	`--sym <SYMBOL>`	`--price <PRICE>`	Show every resting order for a symbol (or one price level), across all gateways
`symbols`	none	none	List configured instruments
`session`	`--state <STATE>`	none	Request session transition
`session-status`	none	none	Read current session state
`schedule`	none	none	Read configured schedule
`gateways`	none	none	List gateway states
`volume`	none	none	Show daily volume summary
`position`	`--gw <GW_ID>`	none	Show net position, avg cost, and live bid/ask/spread/mid per symbol for gateway

2.37 pm-cverifier — Config Verifier

Validates `engine_config.yaml` across four layers: YAML syntax, JSON schema, semantic checks, and completeness checks. Produces a human-readable or JSON report and exits with a non-zero code if any findings meet or exceed the minimum severity threshold.

```
pm-cverifier [--format text|json] [--level info|warn|error] [--no-color] [--strict] CONFIG_FILE
```

Startup options:

Flag	Default	Description
`CONFIG_FILE`	required	Path to `engine_config.yaml`

Flag	Default	Description
`--format`	`text`	Output format: `text` or `json`
`--level`	`info`	Minimum severity to show: `info`, `warn`, or `error`
`--no-color`	off	Disable ANSI colour in text output
`--strict`	off	Treat warnings as errors for CI exit-code purposes
`--version`	—	Print version and exit

Expected runtime input arguments:

None. `pm-cverifier` reads the config file, prints its report, and exits.

Exit codes:

- `0` — no findings at or above the threshold (or `--strict` and no warnings/errors)
- `1` — one or more findings at or above the threshold

2.38 pm-config-show — Config Viewer

Prints the effective configuration as a terminal dashboard, or renders it as a multi-page A4 PDF, and exits. Where `pm-cverifier` answers *is this file correct*, `pm-config-show` answers *what does this file say* — including the port map, which cannot be read off the YAML at all because the engine and index sockets live in `config.py` and a gateway section without a `port:` key still binds on its runtime default.

```
pm-config-show
pm-config-show -m 2
pm-config-show --all
pm-config-show -f my_config.yaml
pm-config-show --format pdf -o exchange.pdf
```

Startup options:

Flag	Default	Description
`-f`, `--file <YAML>`	`<DATA_DIR>/ref_data/engine_config.yaml`	Config file to read
`-m`, `--density [1|2]`	`0` (bare `-m` means `1`)	Pack more information in; `1` adds risk and gateway detail, `2` adds every knob
`-a`, `--all`	off	Show everything: implies `-m 2`, unmasks API keys, lists unrecognised keys
`--format <FMT>`	`terminal`	Output format: `terminal` or `pdf`
`-o`, `--output <FILE>`	`engine-config-<stem>.pdf`	Destination file for `--format pdf`
`--no-color`	off	Disable ANSI colour; also implied when stdout is not a TTY or `NO_COLOR` is set
`--ascii`	off	ASCII box drawing; auto-enabled on non-UTF-8 terminals

Flag	Default	Description
`--width <N>`	terminal width (`100` when piped)	Force render width — for scripted capture and piping
`--height <N>`	terminal height	Force render height — affects default-density height trimming only
`--version`	off	Print version and exit

Expected runtime input arguments:

None. `pm-config-show` reads the config file, prints its dashboard (or writes the PDF), and exits.

Exit codes:

- `0` — the configuration was displayed, or the PDF was written
- `2` — the config file does not exist or cannot be read
- `3` — the file is not valid YAML (the message points at `pm-cverifier`)

With no `--file` it reads the deployed configuration from `<DATA_DIR>/ref_data/engine_config.yaml`, so what it shows is what the exchange was configured from. It is strictly **read-only**: it never writes to the configuration or the data directory, and `--output` is the only path it creates.

The layout is computed from the terminal actually in use rather than a fixed column count — panels pack side by side where they fit, short panels stack beside tall ones, and tables shed optional columns as they narrow. Below 72 columns or 18 rows it falls back to a plain summary of counts, flags, and the port map. At the default density the output is also trimmed to the window height, dropping optional panels and then shortening the symbol list; both trims say so and name the flag that restores the content. Passing `-m` or `-a` disables height trimming.

Two details matter operationally:

- **Ports.** Every listener appears in one table with its process, function, bind address, and where the value came from — `fixed` (a `config.py` constant), `env` (an environment override), `set` (explicit in the file), or `default` (the section is present but omits `port:`). A port claimed twice is flagged in red, the same condition `pm-cverifier` reports as `M018`.
- **API keys.** Masked by default and revealed with `--all`. A key is never wrapped or truncated at any width, and no styling is applied inside the token, so a terminal double-click selects the whole thing. Masked and revealed keys are the same length, so `--all` never moves the layout.

Density is a *layout* control and `--all` is a *disclosure* control, which is why they are separate flags: `-m 2` shows every setting while keeping keys masked, so it stays safe to run on a projector. Note that a PDF produced with `--all` contains live credentials — masking is per-run, not per-file.

For the panel-by-panel walkthrough and a worked example, see Inspect Configs with `pm-config-show`.

This tool is local inspection logic and does not participate in the ZeroMQ runtime message bus.

2.39 pm-setup — Session Bootstrap Tool

Bootstraps a runnable EduMatcher session directory with sensible defaults.

```
pm-setup [--data-dir PATH] [--show] [--config NAME] [--force] [--no-config]
```

Startup options:

Flag	Default	Description
`--data-dir <PATH>`	`$EDUMATCHER_DATA_DIR`, else `<repo>/src/data` in a source checkout, else `~/.local/share/edumatcher`	Data directory for persistent files
`--show`	off	Print the data directory that would be used and exit
`--config <NAME>`	`s10-basic`	Bundled example to deploy, e.g. `s1-basic`, `s3-nominal`, `s10-complex`, `s150-basic`. Resolved under `docs/examples/ref_data/`; append `-nomm` for the variant without market-maker quotes (`s1-basic-nomm`)
`--force`	off	Replace an already-deployed config
`--no-config`	off	Create the data dir only; deploy nothing
`--version`	—	Print version and exit

Expected runtime input arguments:

None.

It creates the data directory and deploys a bundled sample configuration (`s10-basic` unless `--config` names another) to `<DATA_DIR>/ref_data/engine_config.json`, so a fresh environment can start the exchange immediately.

Use this once per new environment (VM, container volume, or fresh local workspace) before starting runtime processes.

This tool is local bootstrap logic and does not participate in the ZeroMQ runtime message bus.

2.40 pm-config-deploy — Compile and Install a Configuration

Validates an authored `engine_config.yaml`, compiles it, and installs the result as the one file every process reads.

```
pm-config-deploy my_config.yaml
pm-config-deploy --check my_config.yaml
pm-config-deploy --example s3-basic
pm-config-deploy --show
```

Startup options:

Flag	Default	Description
`SOURCE`	—	Authored `engine_config.yaml` to validate, compile and install
`--example NAME`	—	Use a bundled example instead of `SOURCE`, e.g. `s3-basic` (from `docs/examples/ref_data/s3-basic-setup/engine_config.yaml`); append `-nomm` for the variant without market-maker quotes
`--check`	off	Validate and compile, but install nothing — for CI

Flag	Default	Description
`--show`	off	Print the deployed paths and exit
`--version`	—	Print version and exit

Expected runtime input arguments:

None. It reads `SOURCE`, writes `<DATA_DIR>/ref_data/engine_config.json` plus a copy of the source beside it, and exits.

Validation runs all four `pm-cverifier` layers. That is stricter than starting used to be: a configuration with, say, a `MARKET_MAKER` gateway and no `market_maker_quotes` (`M001`), or an API credential naming a `gateway_id` absent from `participants` (`M022`), will now refuse to deploy where it previously ran. Warnings do not block — a command that refused on advice would push people back towards editing the deployed copy by hand.

Compiling also resolves every default exactly once. That is what allows the runtime loaders to deserialise rather than each keep its own copy of the defaults for its section.

A configuration that fails is not installed and the previous artifact is left untouched — the write is staged and renamed, so a process starting during a deploy sees either the old file or the new one, never half of either.

The command prints the symbol and gateway counts it installed. Those numbers are worth reading: an unexpected zero is the difference between an exchange and a gateway that accepts connections but has no instruments to offer.

Deploying does not disturb running processes. Restart them to pick up the new configuration; until you do, each will warn at startup that its source has moved on.

This tool is local bootstrap logic and does not participate in the ZeroMQ runtime message bus.

2.41 pm-config-gen — Engine Config Generator

Generates an `engine_config.yaml` from explicit CLI parameters so environments can be recreated without manual YAML editing.

```
pm-config-gen --symbols AAPL MSFT --participants TRADER01 TRADER02 OPS01:ADMIN --sessions-enabled --output engine_co
```

Startup options:

Required

Flag	Description
`--symbols SYM [SYM ...]`	One or more symbols
`--participants GW_SPEC [...]`	One or more gateway specs (`ID[:ROLE[:DISCONNECT]]`); `ID` is `A-Z`, `0-9`, `_`, starting with a letter

Gateway / symbol overrides

Flag	Description
`--participant-smp GW_ID:SMP_ACTION`	Per-gateway self-match-prevention default (`NONE`, `CANCEL_AGGRESSOR`, `CANCEL_RESTING`, `CANCEL_BOTH`); repeatable
`--participant-desc GW_ID:TEXT`	Per-gateway `description`; quote it when `TEXT` has spaces; repeatable
`--participant-default-smp SMP_ACTION`	Writes `participant_defaults.smp_action`, inherited by gateways without their own
`--participant-default-disconnect DISCONNECT`	Writes `participant_defaults.disconnect_behaviour`, inherited by gateways whose spec names no disconnect behaviour
`--symbol-opts SYMBOL:KEY=VALUE[,...]`	Per-symbol overrides; repeatable
`--symbol-static-band SYM:PCT`	Per-symbol collar static band in (0,1); repeatable
`--symbol-dynamic-band SYM:PCT`	Per-symbol collar dynamic band in (0,1); repeatable
`--symbol-risk-level SYM:LEVEL`	Per-symbol risk level key; repeatable
`--symbol-max-order-qty SYM:N`	Per-symbol `order_limits.max_order_qty` in shares (> 0); repeatable
`--symbol-max-order-value SYM:AMOUNT`	Per-symbol `order_limits.max_order_value` as notional (> 0); repeatable
`--outstanding-shares SYM:N`	Per-symbol outstanding shares; repeatable

Session & engine tuning

Flag	Default	Description
`--sessions-enabled` / `--no-sessions-enabled`	disabled	Enable/disable scheduler-driven sessions
`--snapshot-interval SECS`	`0.5`	Snapshot interval in seconds
`--quote-history-maxlen N`	`30`	Quote history max length
`--drop-copy-buffer-size N`	`10000`	Drop-copy buffer size
`--recent-trades-maxlen N`	`20`	Recent trades max length
`--depth-snapshot-tolerance-ticks N`	`100`	Depth snapshot tolerance in ticks

Collars & circuit breakers

Flag	Default	Description
`--no-collars`	off	Set `enforce_collars: false`

Flag	Default	Description
`--no-circuit-breakers`	off	Set `enforce_circuit_breakers: false`
`--static-band PCT`	unset	Default static collar band
`--dynamic-band PCT`	unset	Default dynamic collar band
`--risk-level NAME:STATIC_PCT[:DYNAMIC_PCT]`	—	Risk level definition; repeatable
`--cb-levels NAME:SHIFT_PCT[:HALT_MINS[:RESUMPTION_MODE]]`	—	Circuit-breaker level definition; repeatable
`--cb-window-ns NS`	`300000000000`	Circuit-breaker reference window in nanoseconds

Reopening corridor (ACE)

Automated Corridor Expansion governs how a halted symbol reopens. It is on by default; these flags tune or disable it.

Flag	Default	Description
`--no-ace`	off	Disable ACE: a halt then reopens at the equilibrium price whatever it is, with no corridor and no extensions
`--ace-initial-band PCT`	`0.1`	Reopening corridor half-width as a fraction of the reference price
`--ace-expansions WIDEN_PCT:MINUTES [...]`	`0.10:2 0.20:5`	Rungs of the expansion ladder, in order. The last rung repeats indefinitely, which is what lets the corridor eventually contain any price (the default is Nasdaq's ladder)
`--ace-random-end-ns NS`	`30000000000` (30 s)	Upper bound of the uniform random tail added to every call phase; `0` disables the random end and makes reopen times predictable
`--ace-random-seed N`	OS entropy	Seed the random-end generator for reproducible demos. Omit it on a real venue

Market-maker obligations

Flag	Default	Description
`--mm-spread-ticks N`	`20`	Global MM max spread ticks

Flag	Default	Description
`--mm-min-qty N`	`100`	Global MM minimum quote quantity
`--enforce-mm-obligations` / `--no-enforce-mm-obligations`	disabled	Enable/disable MM obligation enforcement

Symbols / seeding

Flag	Default	Description
`--tick-decimals N`	`2`	Default tick decimals (`0..8`)
`--seed-last-prices`	off	Emit null last-price placeholders
`--seed N`	—	RNG seed for generated training values
`--seed-mm-mid-range MIN:MAX`	—	Seed MM quotes from a random midpoint
`--mm-seed-spread-ticks N`	`10`	Half-spread, in ticks, either side of the seeded MM midpoint
`--seed-last-prices-from-mm`	off	Set last prices from the seeded MM midpoint
`--no-mm-seed-quotes`	off	Write `require_mm_seed_quotes: false` and no `market_maker_quotes`, so the book starts genuinely empty even though a `MARKET_MAKER` gateway exists. `--seed-mm-mid-range` then only seeds last prices (with `--seed-last-prices-from-mm`)

Gateway sections

Each flag below enables a top-level YAML section for the corresponding gateway process.

Flag	Emits	Key sub-flags
`--post-trade-gateway`	`post_trade_gateway:` for `pm-ralf-gwy`	`--post-trade-name`, `--post-trade-bind-address`, `--post-trade-port`, `--post-trade-replay-retention-sec`, `--post-trade-heartbeat-interval-sec`, `--post-trade-idle-timeout-sec`, `--post-trade-max-connections`, `--post-trade-max-client-queue`, `--post-trade-allowed-roles`

Flag	Emits	Key sub-flags
`--market-data-gateway`	`market_data_gateway:` for `pm-md-gwy`	`--market-data-name`, `--market-data-bind-address`, `--market-data-port`, `--market-data-heartbeat-interval-sec`, `--market-data-idle-timeout-sec`, `--market-data-replay-window-sec`, `--market-data-max-connections`, `--market-data-max-symbols-per-client`, `--market-data-max-client-queue`, `--market-data-depth-levels`, `--market-data-enabled` / `--market-data-disabled`
`--alf-gateway`	`alf_gateway:` for `pm-alf-gwy`	`--alf-enabled`, `--alf-disabled`, `--alf-name`, `--alf-bind-address`, `--alf-port`, `--alf-heartbeat-interval-sec`, `--alf-handshake-timeout-sec`, `--alf-idle-timeout-sec`, `--alf-max-connections`, `--alf-max-client-queue`, `--alf-max-commands-per-second`, `--alf-max-errors-before-disconnect`, `--alf-error-window-sec`
`--balf-gateway`	`balf_gateway:` for `pm-balf-gwy`	`--balf-name`, `--balf-bind-address`, `--balf-port`, `--balf-heartbeat-interval-sec`, `--balf-heartbeat-timeout-sec`, `--balf-idle-timeout-sec`, `--balf-auth-timeout-sec`, `--balf-max-connections`, `--balf-max-client-queue`, `--balf-max-messages-per-second`, `--balf-max-errors-before-disconnect`, `--balf-error-window-sec`, `--balf-duplicate-session-policy`

Flag	Emits	Key sub-flags
`--api-gateway`	`api_gateways:` for `pm-api-gwy`	`--api-gateway-name`, `--api-gateway-instance NAME:GATEWAY[,GATEWAY...][:PORT]` (repeatable), `--api-gateway-host`, `--api-gateway-port`, `--api-gateway-log-level`, `--api-gateway-stats-db`, `--api-gateway-swagger-enabled` / `--api-gateway-swagger-disabled`, `--api-gateway-enabled` / `--api-gateway-disabled`, `--api-gateway-rate-limit-writes-per-se` `--api-gateway-rate-limit-burst`, `--api-gateway-engine-auth-sec`, `--api-gateway-engine-reply-sec`, `--api-gateway-wait-ack-sec`, `--api-gateway-order-retention-sec`; credential flags: `--api-key KEY:GATEWAY_ID[:DESCRIPTION]` (repeatable), `--api-gateway-generate-keys` / `--no-api-gateway-generate-keys`, `--api-gateway-readonly-key`
`--dc-gateway`	`dc_gateway:` for `pm-dc-gwy`	`--dc-name`, `--dc-bind-address`, `--dc-port`, `--dc-heartbeat-interval-sec`, `--dc-idle-timeout-sec`, `--dc-max-connections`, `--dc-max-client-queue`

Flag	Emits	Key sub-flags
`--log-server`	`log_server:` for `pm-log-srv`	`--log-server-enabled`/`-disabled`, `--log-server-name`, `-bind-address`, `-port`, `-db-path`, `-retention-days`, `-max-message-bytes`, `-max-client-queue`, `-write-batch-size`, `-write-batch-interval-ms`, `-heartbeat-interval-sec`, `-pubsub-enabled`/`-disabled`, `-pub-port`, `-pull-port`, `-lease-sec`, `-max-lease-sec`, `-max-subscribers`, `-notify-interval-ms`, `-backfill-chunk-rows`, `-max-backfill-minutes`, `-max-backfill-rows`, `-max-pending-rows`, `-pub-sndhwm` (each prefixed `--log-server`)
`--monitor`	`monitor:` for `pm-monitor` and every health beacon, with every field	`--monitor-enabled`/`-disabled`, `--monitor-bind-address`, `--monitor-host`, `--monitor-pull-port`, `--monitor-pub-port`, `--monitor-beacon-interval-sec`, `--monitor-hung-after-sec`, `--monitor-silent-after-beacons`, `--monitor-book-metrics-interval-sec`, `--monitor-raw-hours`, `--monitor-minute-days`, `--monitor-threshold` `RULE:WARN:CRIT` (repeatable; rules `host_cpu_pct`, `host_mem_pct`, `disk_free_pct` (warn *above* crit), `engine_handle_p99_us`, `reject_pct_5m`, `rss_growth_pct_1h`, `heap_bloat_ratio`)

Index sections

Flag	Description
`--index ID[:DESCRIPTION]`	Add an index (up to 5); `ID` is letters and digits only

Flag	Description
`--index-constituents ID:SYM[,SYM,...]`	Constituents of index `ID`
`--index-base-value ID:VALUE`	Base value of index `ID` (default `1000.0`)
`--index-interval ID:SECS`	Publish interval of index `ID` (default `1.0`)
`--index-history-file ID:PATH`	History file of index `ID` (default `data/indexes/<ID>_history.jsonl`)
`--index-state-file ID:PATH`	State file of index `ID` (default `data/indexes/<ID>_state.json`)

Each `--index-*` flag takes one value per index: give it at most once per `ID`, and only for an `ID` defined with `--index`. Every constituent needs `--outstanding-shares` and a reference price, which `pm-config-gen` only writes with a `MARKET_MAKER` participant and `--seed-mm-mid-range MIN:MAX --seed-last-prices-from-mm`; otherwise it refuses with an error.

Schedule

Flag	Default	Description
`--schedule` / `--no-schedule`	auto	Force include/suppress schedule section
`--pre-open HH:MM`	`09:00`	Pre-open schedule time
`--opening-auction HH:MM`	`09:25`	Opening auction start time
`--continuous HH:MM`	`09:30`	Continuous trading start time
`--closing-auction HH:MM`	`16:00`	Closing auction start time
`--closing-end HH:MM`	`16:05`	Closing auction end time
`--weekend`	off	Also emit a `weekend:` block (Sat+Sun) using the `--weekend-*` times below; without this flag weekends are `CLOSED`
`--weekend-pre-open HH:MM`	`10:00`	Weekend pre-open schedule time
`--weekend-opening-auction HH:MM`	`10:25`	Weekend opening auction start time
`--weekend-continuous HH:MM`	`10:30`	Weekend continuous trading start time
`--weekend-closing-auction HH:MM`	`14:00`	Weekend closing auction start time
`--weekend-closing-end HH:MM`	`14:05`	Weekend closing auction end time

Flag	Default	Description
`--holidays` / `--no-holidays`	off	Also emit a `holidays:` block using the `--holidays-*` times below; without this flag bank holidays are `CLOSED`
`--holidays-pre-open HH:MM`	`10:00`	Holiday pre-open schedule time
`--holidays-opening-auction HH:MM`	`10:25`	Holiday opening auction start time
`--holidays-continuous HH:MM`	`10:30`	Holiday continuous trading start time
`--holidays-closing-auction HH:MM`	`14:00`	Holiday closing auction start time
`--holidays-closing-end HH:MM`	`14:05`	Holiday closing auction end time
`--country CODE`	—	Holiday calendar country code (ISO 3166-1 alpha-2) — which dates the `holidays:` block applies to
`--session-timezone TZ`	—	Session timezone, an IANA name (e.g. `Europe/Stockholm`)

Combos

Flag	Description
`--combo SPEC`	Add a pre-configured combo definition; repeatable

Output

Flag	Default	Description
`--output FILE`	stdout	Output YAML file path
`--force`	off	Overwrite existing output file
`--dry-run`	off	Print generated YAML; do not write file
`--comment-default-config-fields`	off	Add a commented-out header with every optional section (engine settings, gateways, `api_gateways`, `log_server`, `monitor`, `indices`, `schedule`) at its default values, ready to uncomment

Expected runtime input arguments:

None.

Use cases:

- Generate consistent configs across multiple machines.
- Create deterministic demo/test environments quickly.
- Avoid hand-editing errors in gateway/symbol/session configuration.

Like `pm-setup`, this is a local tooling command and does not subscribe/publish on the runtime ZeroMQ sockets.

For more details on the options see Configuration Chapter

2.42 pm-index-cli — Index Structural/Audit History Query Tool

Reads the structural/corporate-action audit JSONL files written by `pm-index` directly from disk and renders them as a table, JSON, or CSV. No running process is required, except for `health`. It does not expose level or EOD history — use `pm-stats-cli index-daily` / `index-snapshots` for that instead (see the pm-stats-cli section above).

```
pm-index-cli [--config engine_config.yaml] [--format table|json|csv] COMMAND [options]
```

Startup options:

Flag	Default	Description
`--config PATH` / `-c PATH`	unset	Path to `engine_config.yaml`; auto-discovers history file paths and index IDs
`--data-dir DIR`	`data/indexes`	Directory containing history files; used when `--config` is absent or an index is not in config
`--format table|json|csv`	`table`	Output format
`--no-header`	off	Suppress header row (CSV only)
`--version`	—	Print version and exit

Expected runtime input arguments:

None.

Subcommands:

Subcommand	Default limit	Purpose
`events`	1,000	Structural events: `INIT`, `CORP_ACTION`, `ADD_CONSTITUENT`, `DELIST`
`indices`	—	List configured indices from `engine_config.yaml`
`health`	—	Check the `pm-index` process and ping its health socket; `--extended` adds every index's state/history file check. Options: `--extended`, `-v`/`--verbose`, `-q`/`--quiet` (print nothing; exit `0` when healthy, `-1` otherwise)

The `events` subcommand accepts `--index ID` (repeatable), `--days N`, `--from DATE_OR_TS`, `--to DATE_OR_TS`, `--limit N`, and `--type TYPE` to filter to a specific structural event kind.

Like `pm-stats-cli`, this is a read-only tool that works offline. The only ZeroMQ connection it ever makes is `health`'s ping to `pm-index` on port `5564`.

See pm-index-cli in the commands reference for the full option reference, output column descriptions, and examples including CSV export and Python plotting.

2.43 pm-index-admin-cli — Index Corporate Action / Constituent Change CLI

One-shot CLI for applying corporate actions (splits, cash dividends, share issuances and buy-backs) and constituent changes (add/delist) to a running `pm-index` process. Unlike `pm-index-cli` above, this tool is a **write** tool and talks directly to `pm-index`'s live PUSH/PULL socket pair — it does not read JSONL files and does not require `--config`.

```
pm-index-admin-cli --id <GW_ID> [--push ADDR] [--sub ADDR] [--timeout MS] [--dry-run] [-y] [--format table|json] COMM
```

Startup options:

Flag	Default	Description
`--id`	required	Ack-routing label — not authenticated by `pm-index` (its PULL socket has no auth of any kind)
`--push`	`tcp://127.0.0.1:5559`	`pm-index` PULL socket address
`--sub`	`tcp://127.0.0.1:5558`	`pm-index` PUB socket address
`--timeout`	`3000`	Ack timeout in milliseconds
`--dry-run`	off	Print the outbound payload instead of sending it
`-y` / `--yes`	off	Skip the confirmation prompt
`--format`	`table`	Output format: `table` or `json`
`--version`	—	Print version and exit

Expected runtime input arguments:

None (one-shot; runs a single subcommand and exits).

Subcommands:

Subcommand	Purpose
`split`	Apply a stock split or reverse split
`dividend`	Apply a cash dividend adjustment
`shares`	Set shares outstanding — covers both issuances and buy-backs
`add`	Add a new constituent
`delist`	Remove a constituent
`history`	Show recent structural/corp-action history for an index

Subcommand arguments (all take `--index INDEX_ID`, which is required):

Subcommand	Arguments	Notes
`split`	`--sym SYMBOL`, `--ratio N:M`	`4:1` is a 4-for-1 split, `1:10` a 1-for-10 reverse split
`dividend`	`--sym SYMBOL`, `--amount AMOUNT`	Dividend per share in price units; must be > 0
`shares`	`--sym SYMBOL`, and either `--new-shares N` or `--delta N`	`--new-shares` is the new absolute total; `--delta` is a signed change to the last known count (negative for a buy-back), resolved via `history` and shown in the prompt
`add`	`--sym SYMBOL`, `--shares N`, `--price PRICE`	Initial shares outstanding and reference price; both must be > 0
`delist`	`--sym SYMBOL`	
`history`	`--from UNIX_TS`, `--to UNIX_TS`, `--types T,T,...`, `--limit N`	Defaults: last 24 h, all types (`INIT`, `CORP_ACTION`, `DELIST`, `ADD_CONSTITUENT`), 50 rows

Every mutating subcommand (`split`, `dividend`, `shares`, `add`, `delist`) prints the action and prompts `Continue? [y/N]` before sending, unless `-y`/`--yes` is given. `pm-index` has no dedicated buy-back action type — the `shares` subcommand's `--delta` flag models a buy-back as a negative delta against the constituent's last known share count, resolved via `history` before sending an absolute `SHARES_ISSUANCE` value.

This tool has no `connect()`/authentication step, unlike `pm-admin-cli`: `pm-index`'s PULL socket accepts any non-empty `gateway_id`, using it only as an ack-routing key.

See Index Administration (`pm-index-admin-cli`) for the full subcommand reference, confirmation-prompt behaviour, and worked examples.

2.44 pm-opctl-cli — Operational Process Control

Start, stop, and monitor a named group of `pm-*` processes together. This is the recommended way to bring up (or tear down) a full EduMatcher stack, instead of launching each process by hand in its own terminal — see Running the Exchange → Starting the stack with pm-opctl-cli.

```
pm-opctl-cli start [PROFILE] [-d]
pm-opctl-cli list [-y | --no-restart]
pm-opctl-cli health [-q]
pm-opctl-cli stop
pm-opctl-cli kill
pm-opctl-cli init
pm-opctl-cli show [--json]
pm-opctl-cli clear (--state | --all) [--yes]
```

Profiles:

A *profile* is a named list of processes to run together, defined in `<DATA_DIR>/emo-config.yaml`. Four profiles are built in and used as-is until that file exists:

Profile	Contents
`micro`	Centralized logging plus the matching engine only
`mini`	A trading-capable subset: logging, stats, engine, scheduler, market data, the desk API gateway, and the ALF/post-trade/drop-copy gateways
`default`	The full nominal exchange stack, including audit, clearing, both API gateway instances, the BALF gateway, and the index calculation process (`pm-index`)
`mm-demo`	`default` plus one `pm-mm-bot`: `passive` strategy, quoting as gateway `MM01` on every symbol of the deployed configuration (`--all-symbols`), deliberately slow (`--gap 0.20 --qty 200 --reissue-delay-ms 5000 --fade-ticks 3 --fade-sec 20 --drift-ticks 15`) so students can follow what a market maker does. All settings are command-line arguments; no bot config file is used. Needs a config that registers `MM01` and those symbols, such as `s10-basic`
`ai-swarm`	The `default` set, `mm-demo`'s market-maker bot with `--anchor-sim 0.3` (quoting around the true values; an unanchored passive maker pins prices), `pm-market-sim --verbose` (the market model and its news) and `pm-ai-swarm --count 20 --budget 40 --symbols-per-agent 1000 --verbose`: twenty AI traders (`AI001`–`AI020`, every built-in preset in equal shares, each on every symbol) in one worker process. Needs a config that registers them and ships a `market_sim.yaml`, such as the `s150` examples or `s300-load`
`ai-sim`	`ai-swarm` set up to run for weeks on the `s300-sim` example: `pm-ai-swarm --swarm <examples>/s300-sim-setup/swarm.yaml --verbose` (`AI001`–`AI100`, 120 order actions a second), the market maker with the `symmetric` strategy (the passive one re-quotes on every change in other traders' orders, which with a hundred traders was most of the exchange's load), `pm-engine` without `--verbose`, `pm-stats --keep-days 7 --keep-order-events-days 0.5 --human-gateways 'TRADER*' --keep-human-order-events-days 7` and `pm-clearing --print-every 0 --retention-days 7`. What the `edumatcher-ai-sim` image runs; needs `s300-sim` deployed. See AI Traders and Market Simulation

Run `pm-opctl-cli init` to write the built-ins to `emo-config.yaml` for editing; once that file exists its profiles replace the built-ins entirely (a missing `default` profile is backfilled from the built-in nominal stack).

Subcommands:

Subcommand	Aliases	Options	Purpose
`start [PROFILE]`	`up`	`-d`/`--debug`/`-vv` (append `--log-level DEBUG` to every process's command)	Start a profile (`default` when omitted), skipping entries already running
`list`	—	`-y`/`--restart` (restart dead entries without asking), `--no-restart` (never offer)	Status table for the active profile: uptime (`HH:MM`) and resident memory (MiB) per process

Subcommand	Aliases	Options	Purpose
`health`	—	`-q`/`--quiet` (print nothing)	Same checks as `list`; exits `0` only when every process is running — for monitoring scripts
`stop`	`down`	—	Send `SIGTERM` to processes this tool started (recorded in the PID directory)
`kill`	—	—	Emergency stop: `pkill -15 -f -i -l pm-` — signals every process whose command line contains `pm-`, including ones this tool did not start
`init`	—	—	Write the built-in profiles to `<DATA_DIR>/emo-config.yaml`; refuses to overwrite an existing file
`show`	—	`--json`	Print version, resolved data directory, and deployed config paths — same info as `pm-config-deploy --show`
`clear`	—	`--state` (engine/session state only) or `--all` (also logs and audit trail), `--yes` (skip prompt)	Delete persisted data under the data directory; `ref_data/` (configuration) is never touched

Exactly one of `--state`/`--all` must be given to `clear`.

`start --debug` appends `--log-level DEBUG` to every process's command line before it is launched, overriding whatever level a profile's `--verbose` (or lack of one) would otherwise select — see Developer → Logging levels and pm-log-srv for what that changes and why every process still needs to be *started* this way (an already-running process must be stopped and restarted with `--debug` to pick up DEBUG logging; there is no way to raise a running process's log level in place).

Expected runtime input arguments:

None. Each subcommand runs once and exits (`list`/`health` print a snapshot, they do not stay running).

Health checking:

There is no general way to prove an arbitrary process is not internally hung, so health is reported

at three levels of confidence: `dead` (no live PID found), `not responding` (the PID is alive but its `healthcheck` failed or its `tcp` probe refused/timed out), or `running` (the PID is alive and any configured check passed). A `healthcheck` command takes precedence over a `tcp` probe because it exercises the application itself, not just its socket.

Process tracking:

Started processes are detached into their own session and recorded as `<DATA_DIR>/emo/<name>.pid`, with combined stdout/stderr appended to `<DATA_DIR>/emo/<name>.log`. The active profile name is remembered in `<DATA_DIR>/emo/active-profile`. A process restarted outside this tool is re-adopted on the next `list`/`health`/`stop` by matching its command line via `pgrep`, so a stale PID file is never treated as final.

This tool is local orchestration logic — it starts other `pm-*` processes as subprocesses and reads their PID/log files, but binds no ZeroMQ or TCP port of its own and does not participate in the runtime message bus.

See Running the Exchange and the Quick Start's Install and Start for the recommended startup sequence.

2.45 pm-help / pm-man — Command Index and Man-Page Reference

Lists every `pm-*` command with a one-sentence explanation, or prints a full man page for one of them. `pm-man` is a plain alias for the exact same tool – use whichever name is more familiar.

```
pm-help [--format table|text] [-v]
pm-help [--format table|text] [-v] <pm-command>
pm-help --completion bash|zsh
pm-man <pm-command>
```

With no argument, prints the version and data-file locations (the same information as `pm-opctl-cli show`), then a table of every `pm-*` command grouped by category: Core Runtime, External Gateways, Protocol Spies, AI & Bots, Index, Query & Reporting CLIs, Admin & Operations, Setup & Configuration, Logging, Developer Tools, and Help.

With a command name, prints a full man page: NAME, SYNOPSIS, DESCRIPTION, OPTIONS, SUBCOMMANDS (where applicable), PORTS, MESSAGE BUS involvement, RELATED COMMANDS, EXAMPLES, NOTES, and SEE ALSO. The name is looked up with or without its `pm-` prefix, so `pm-help viewer` finds `pm-viewer`.

Startup options:

Flag	Default	Description
`--format table|text`	`table`	`table` uses UTF-8 box-drawing characters; `text` is plain aligned columns with no box-drawing – friendly to `grep`, redirects, and non-UTF-8 terminals
`-v`, `--verbose`	off	In the command-index view, show extra detail (and a worked example, where one exists) under each command

Flag	Default	Description
`--no-color`	off	Disable ANSI colour
`--completion bash|zsh`	—	Print the tab-completion script for every `pm-*` command and exit – see Shell completion
`--version`	—	Print version and exit

Expected runtime input arguments:

None. Each invocation prints its output once and exits.

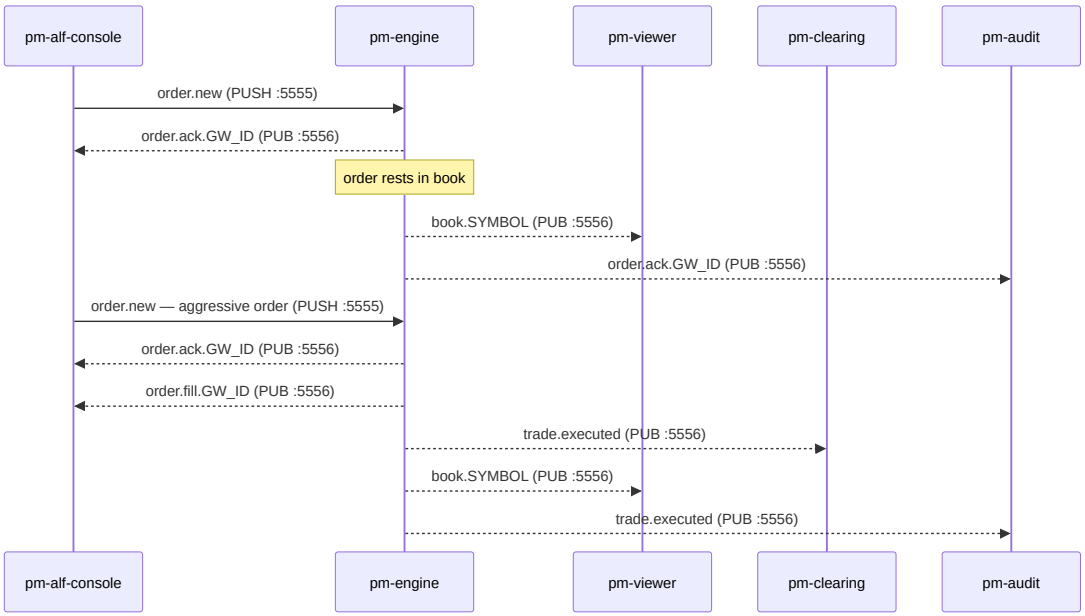
This tool is local reference data – hand-curated, not scraped from each command's own `--help` output – so it can show information no `argparse` parser exposes on its own, such as ports and ZeroMQ message-bus involvement. It does not participate in the runtime message bus.

Examples:

```
pm-help                    # version, data dirs, and the full command table
pm-help --format text -v   # plain-text table with per-command examples
pm-help pm-viewer          # full man page for pm-viewer
pm-man pm-alf-console      # pm-man is an alias for pm-help
pm-help --completion bash  # print the bash tab-completion script and exit
```

2.46 Order Lifecycle Message Flow

The following diagram traces a single limit order from submission to full fill, showing which process sends each message and on which socket:



All subscribers (`pm-viewer`, `pm-clearing`, `pm-audit`, `pm-stats`, `pm-board`, etc.) receive the same `trade.executed` and `book.{SYMBOL}` events concurrently from the single PUB socket — none of them coordinate with each other.

2.47 pm-balf-gwy — BALF TCP Gateway

`pm-balf-gwy` is implemented and provides external binary order-entry over TCP. It accepts BALF binary frames from low-latency programmatic clients and translates them into the same engine order flow as the ALF gateways.

See BALF TCP Gateway for operational usage and BALF Protocol Reference for the wire-level contract.

```
pm-balf-gwy [--bind 0.0.0.0] [--port 5560] [--engine-host HOST] [--log-level LEVEL] [-v|-vv] [-q] \
  [--log-target TARGET] [--log-file PATH] [--log-failover-timeout SEC]
```

Startup options:

Flag	Default	Description
`--bind`	from config / `0.0.0.0`	TCP bind address for BALF clients
`--port`	from config / `5560`	TCP listen port for BALF clients
`--engine-host`	from config	Override engine host for ZMQ ports `5555` / `5556`
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

2.48 pm-md-gwy (CALF Market-Data Gateway)

`pm-md-gwy` is implemented and provides external market-data distribution over CALF/TCP. It consumes engine (and index) PUB topics and exposes seven sequence-aware channels: `TOP` (top-of-book), `TRADE` (executed trades), `STATE` (session and halt/resume state), `INDEX` (index level updates), `DEPTH` (full book depth), `AUCTION` (auction uncross results), and `CB` (circuit-breaker halt/resume events). A client discovers which channels a given gateway build supports via `WELCOME|CH_SUPPORTED=`; only `TOP`, `TRADE`, `STATE`, and `AUCTION` accept the `SYM=*` wildcard.

```
pm-md-gwy [--bind 0.0.0.0] [--port 5570] [--engine-pub tcp://127.0.0.1:5556] [--index-pub tcp://127.0.0.1:5558] [--lo
  [--log-target TARGET] [--log-file PATH] [--log-failover-timeout SEC]
```

Startup options:

Flag	Default	Description
`--bind`	from config / `0.0.0.0`	TCP bind address for external clients
`--port`	from config / `5570`	TCP listen port for CALF clients

Flag	Default	Description
`--engine-pub`	`tcp://127.0.0.1:5556`	Engine PUB address consumed by the gateway
`--index-pub`	`tcp://127.0.0.1:5558`	Index PUB socket address (overrides config)
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	30 (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

See Market Data Feed (CALF) for operational usage and CALF Protocol Reference for the wire-level contract.

2.49 pm-calf-spy (CALF Protocol Spy)

`pm-calf-spy` is a read-only CLI client for `pm-md-gwy`: it opens a CALF TCP session, subscribes to whatever channels/symbols you ask for, and prints every line it receives — human-readable or as JSON — so you can see exactly what the protocol sends without writing a client. It never mutates exchange state, and any number of instances can run at once (e.g. one per terminal, each with a different `--channels`/`--symbols` filter) since `pm-md-gwy` accepts unlimited concurrent connections up to `max_connections`. Since it never sends anything on its own after the initial subscription, it sends a periodic `PING` (`--ping-interval`, default 60s) to keep the gateway from dropping it as an idle client.

Startup options:

Flag	Default	Description
`--host`	`127.0.0.1`	`pm-md-gwy` TCP host
`--port`	`5570`	`pm-md-gwy` TCP port
`--client-name`	`calf-spy-<pid>`	`HELLO|CLIENT=` identifier reported in gateway logs
`--channels`	*	Comma-separated channels; * = every channel in `WELCOME|CH_SUPPORTED=`
`--symbols`	*	Comma-separated symbols; * = wildcard for channels that allow it
`--resume`	*(none)*	One-shot `CH:SYM:LASTSEQ` — requests single-stream replay on connect

Flag	Default	Description
`--ping-interval`	`60`	Seconds between `PING` keepalives sent to the gateway; `0` disables
`--format`	`human`	`human` or `json` output
`--count`	`0`	Exit after N data-carrying lines (`0` = run until Ctrl-C)
`--raw`	off	Print raw protocol bytes instead of decoded fields
`--no-color`	off	Disable ANSI colour in output
`--show-heartbeats`	off	Include `HB` heartbeat lines in output (hidden by default)
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`server`	Where operational log records go: `server` (auto-detected `pm-log-srv`), `stdout`, or `file`
`--log-file`	none	Operational log file path — required when `--log-target file`
`--log-failover-timeout`	`30` (from config)	Grace window in seconds before falling back to a local log file once `pm-log-srv` becomes unreachable
`--version`	—	Print version and exit

See CALF Protocol Spy (pm-calf-spy) for full usage.

2.50 pm-log-srv — Centralized Log Server

`pm-log-srv` is a dedicated collector process, unrelated to the ZeroMQ bus: it accepts LALF ("Logging ALF") TCP connections from any number of `pm-*` processes and appends every received log record to a queryable SQLite database (`log.db`), the same "dedicated collector + SQLite + read-only query CLI" shape `pm-stats`/`pm-audit` already use for market data and trading events, applied here to operational `logging`-module output.

Phased rollout

`pm-log-srv`/`pm-log-cli` are fully implemented and usable today. Wiring every existing `pm-*` process's own logging into a LALF client that auto-detects a running `pm-log-srv` at startup is a follow-up phase and has not yet been rolled out — until then, `--log-level`/`-v`/`-q` still control each

process's own stdout logging as before.
pm-log-srv [–host ADDR] [–port PORT] [–db PATH] [–retention-days N] [–max-message-bytes N] [–pub-port PORT] [–pull-port PORT] [–no-pubsub] [–lease-sec N] [–log-level LEVEL] [-v|-vv] [-q] [–log-target stdout|file] [–log-file PATH]

Flag	Default	Description
`--host`	`0.0.0.0` (from config)	TCP bind address
`--port`	`5600`	TCP listen port for LALF clients
`--db`	`data/log.db`	SQLite database path
`--retention-days`	`30`	Prune `log_events` rows older than N days; `0` = unbounded
`--max-message-bytes`	`65536`	Truncation ceiling per `LOG` payload (never dropped, only truncated)
`--pub-port`	`5601`	LALF-PS ZeroMQ PUB bind port for log distribution
`--pull-port`	`5602`	LALF-PS ZeroMQ PULL bind port for subscriber control
`--no-pubsub`	off	Disable the LALF-PS interface entirely (bind no ZeroMQ sockets)
`--lease-sec`	`30`	Subscription lease TTL in seconds; a subscriber that stops sending `log.renew` is reaped after this long
`--log-level`	`WARNING`	Explicit log level: `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`
`-v` / `--verbose`	off	Increase verbosity (`-v` → `INFO`, `-vv` → `DEBUG`)
`-q` / `--quiet`	off	Reduce output to warnings/errors
`--log-target`	`stdout`	Where this process's own operational log records go: `stdout` or `file` — never `server`, since `pm-log-srv` must not depend on itself over the network
`--log-file`	none	Operational log file path — required when `--log-target file`
`--version`	—	Print version and exit

2.51 pm-log-cli — Log Server Query/Troubleshooting CLI

`pm-log-cli` is a read-only offline tool: it queries `log.db` directly, never over the network, so a busy or even-stopped `pm-log-srv` never blocks troubleshooting with data already collected.

```
pm-log-cli [--db PATH] [--format human|json] COMMAND [options]
```

Subcommands: `tail`, `query`, `processes`, `stats`, `diagnose` (rule-based troubleshooting heuristics with concrete recommendations), and `prune` (manual retention maintenance).

See Centralized Log Server for the full operational guide — starting the server, every `pm-log-cli` subcommand, and a workflow cookbook — and LALF Protocol Reference for the normative wire specification.

2.52 pm-monitor — Health Collector

`pm-monitor` collects the health beacon every long-running `pm-*` process sends, judges each process (HEALTHY, DEGRADED, HUNG, SILENT, DEAD, MISSING, STOPPED) and the exchange as a whole (OPERATIONAL, DEGRADED, IMPAIRED, DOWN), turns counters into rates, evaluates thresholds, keeps history in `metrics.db`, and publishes `health.snapshot` once a second and `health.event` as things happen. Every built-in `pm-opctl-cli` profile except `micro` starts it.

```
pm-monitor [--log-level LEVEL] [-v|-vv] [-q]
           [--log-target server|stdout|file] [--log-file PATH]
           [--log-failover-timeout SEC] [--version]
```

All its settings come from the `monitor` section of the deployed configuration (see Configuring `pm-monitor`); the flags are the common logging options.

Port	Socket	Purpose
`5610`	PULL (bind)	`health.beacon` from every process
`5611`	PUB (bind)	`health.snapshot` (1 Hz), `health.event`
`5556`	SUB (connect)	Engine lifecycle events: connects, disconnects, diagnostics, session, halts

File	Purpose
`<data dir>/metrics.db`	Samples, 1-minute roll-ups, daily totals, counter state, events (WAL)
`<data dir>/metrics.db.lock`	Held while running: one `pm-monitor` per data directory

Exit codes: `0` stopped normally (also after idling with `monitor.enabled: false`); `1` cannot bind its ports or open `metrics.db` (held by another `pm-monitor`, or recorded with another session timezone); `2` usage error or an unknown session timezone.

See Health Monitoring for how health is judged, the thresholds, `metrics.db` and troubleshooting.

2.53 pm-monitor-cli — Health Query CLI

`pm-monitor-cli` is read-only: it subscribes to `pm-monitor`'s PUB port for the live view and reads `metrics.db` for the event log, never writing either.

```
pm-monitor-cli [--host HOST] [--port PORT] [--db PATH]
               [--format table|json] [--timeout SEC] COMMAND [options]
```

Command	Purpose
`health [-q] [--extended] [-v]`	`pm-monitor` publishing snapshots (its loop runs); `--extended` also checks `metrics.db` integrity, schema, pinned timezone and write access. `-q`: exit 0 / 255. Used as the profile's `healthcheck`
`status`	Exchange status, trading summary, every process, active breaches. Exit `0` OPERATIONAL, `1` DEGRADED, `3` IMPAIRED, `4` DOWN, `5` `pm-monitor` unreachable
`events [--since TS] [--kind K] [--subject TEXT] [--limit N] [--follow [--count N]]`	The event log, then live events with `--follow`

See `pm-monitor-cli` for every option and a cookbook.

2.54 pm-msgen — Message Binding Generator

Developer/build tool that generates the Python message-binding modules (`src/edumatcher/models/generated/`), C header artifacts, and the Message Reference documentation table from the canonical specification in `spec/` (`spec/transports.yaml` and `spec/messages/`). It is not part of the running exchange — no `pm-*` runtime process imports or invokes it — it exists so that the spec is the single source of truth for every topic name and payload shape, and so a spec change without regeneration (or a hand-edit to a generated file) cannot silently drift from what the engine and gateways actually send.

```
pm-msgen generate [--spec DIR] [--out-python DIR] [--out-c DIR]
pm-msgen check    [--spec DIR] [--out-python DIR] [--out-c DIR]
pm-msgen lint     [--spec DIR]
pm-msgen grep-literals [--spec DIR] [--src DIR]
```

Subcommands:

Subcommand	Purpose
`generate`	Render Python bindings, C artifacts, and docs from the spec and write them to disk
`check`	Re-render from the spec and fail (exit `1`) if the committed output differs — the CI gate; run via `make msgen-check`
`lint`	Validate the spec only (YAML structure, message/family consistency); prints a family/message count

Subcommand	Purpose
`grep-literals`	Scan a source tree for topic string literals that a generated constant should replace instead

Startup options:

Flag	Default	Applies to	Description
`--spec DIR`	`spec`	all	Spec root holding `transports.yaml` and `messages/`
`--out-python DIR`	`src/edumatcher/models/generated`	`generate`, `check`	Python output directory
`--out-c DIR`	`docs/examples/generated`	`generate`, `check`	C output directory
`--src DIR`	`src`	`grep-literals`	Source tree to scan
`--version`	—	all	Print version and exit

Expected runtime input arguments:

None. Each subcommand runs once and exits.

Exit codes:

- `0` — success (`generate` wrote or confirmed up-to-date output; `check`/`lint` found no problems)
- `1` — `check` found drift between the spec and the committed generated output
- `2` — spec error (invalid YAML/schema) or the `--spec` directory was not found (most often caused by running from the wrong working directory — run from the repository root, or pass `--spec` explicitly)

Regenerate after editing a spec file with `make msgen`; `make msgen-check` (part of `make check`/`make pre-commit`) fails the build if generated output was not committed alongside a spec change.

This tool is local build tooling — it edits source files under version control and does not participate in the ZeroMQ runtime message bus.

See Messages for the generated message catalog itself.

2.55 See also

Participant Guide:

- Market-Maker Bot — autonomous quoting process for a single symbol

Operator's Guide:

- Running the Exchange — startup order, launch scripts, and verification
- The Configuration Workflow — how each process is configured via `engine_config.yaml`
- Inspect Configs (pm-config-show) — read-only viewer for the deployed config and its port map
- Persistence — which process writes which data file
- RALF Gateway — Post-Trade Dissemination — external RALF gateway usage
- CALF Gateway — Market Data — external CALF gateway usage
- Centralized Log Server — pm-log-srv/pm-log-cli operational guide

- ALF Gateway — Order Entry — external ALF order-entry gateway usage
- API Gateway (REST/WebSocket) — running and operating the REST/WebSocket gateway; the endpoint contract is the REST API Reference and the WebSocket API Reference
- Market Index (pm-index) — index configuration, calculation, and corporate actions

Protocols and Clients:

- Message Reference — the full ZeroMQ message catalog all processes share
- Drop Copy — the engine's built-in :5557 drop-copy feed
- RALF Protocol Reference — official protocol appendix
- CALF Protocol Spy (pm-calf-spy) — read-only CALF inspection CLI
- RALF Protocol Spy (pm-ralf-spy) — read-only RALF inspection CLI
- LALF Protocol Reference — official protocol appendix

CHAPTER 3

Configuration Schema and Process Blocks

This chapter is the field-by-field reference for `engine_config.yaml`: every top-level key, which process reads it, and the settings of each section. For how to write, check and deploy a configuration, read the Operator's Guide chapter The Configuration Workflow; for the formal grammar and validation rules, see Engine Configuration Specification.

3.1 Current Schema

The current parser recognizes these top-level keys:

Key	Required when file exists?	Used by	Purpose
`include`	No	Every tool that reads the authored file	List of files merged into the configuration; see Splitting the file
`symbols`	Yes	Engine	Accepted symbols and per-symbol settings
`participants`	Yes	Engine	Participant identities allowed to connect, with per-participant settings
`participant_defaults`	No	Engine	Values a `participants` entry inherits when it omits them
`alf_gateway`	No	`pm-alf-gwy`	External ALF text TCP gateway settings
`sessions_enabled`	No	Engine	Enable scheduler-driven session states
`enforce_collars`	No	Engine	Global collar enforcement toggle
`enforce_circuit_breakers`	No	Engine	Global circuit-breaker enforcement toggle
`require_mm_seed_quotes`	No	Engine	Whether every symbol must seed `market_maker_quotes` when a `MARKET_MAKER` gateway is configured
`auction_indicative_interval_sec`	No	Engine	Throttle (seconds) for indicative-uncross republishing during auction call phases
`engine_tuning`	No	Engine	Runtime retention and throttling knobs
`mm_obligation_defaults`	No	Engine	Default market-maker quote obligation policy

Key	Required when file exists?	Used by	Purpose
`risk_controls`	No	Engine	Named collar and order-limit profiles
`circuit_breaker_defaults`	No	Engine	Default circuit-breaker ladder
`market_maker_combos`	No	Engine	Startup multi-symbol combo seeds
`schedule`	No	Scheduler, parsed by engine too	Session transition times
`locale`	No	`pm-scheduler`, Engine	Country (bank-holiday/weekend calendar) and session timezone (trading-day bucketing)
`post_trade_gateway`	No	`pm-ralf-gwy`	External RALF dissemination gateway settings
`market_data_gateway`	No	`pm-md-gwy`	External CALF dissemination gateway settings
`balf_gateway`	No	`pm-balf-gwy`	External BALF binary TCP gateway settings
`dc_gateway`	No	`pm-dc-gwy`	Drop-copy TCP relay gateway settings
`log_server`	No	`pm-log-srv`	Centralized LALF log-collector settings
`monitor`	No	`pm-monitor`, every beaconing process	Health beacons and the health collector
`api_gateways`	No	`pm-api-gwy`	Named REST/WebSocket order-entry and market-data gateway process settings
`indices`	No	`pm-index`	Index calculation process configurations

The nested sections below document every field currently parsed under these top-level keys. Unknown keys in a mapping are generally ignored by the loader, but they should not be relied on for runtime behavior — `pm-config-show -a` lists any it finds, which is the cheapest way to catch a mistyped section name.

3.2 Which Process Reads What

`engine_config.yaml` is one shared ref-data file, but no single process reads all of it. Each auxiliary gateway or service opens the file independently at its own startup, ignores every top-level key it doesn't recognize, and parses only the section(s) it owns. `pm-engine` is the only process that reads most of the file — the gateway-specific blocks (`market_data_gateway`, `balf_gateway`, `post_trade_gateway`, `dc_gateway`, `log_server`, `alf_gateway`, `api_gateways`) are never touched by the engine itself.

Process	Loader module	Top-level section(s) read	What it needs it for
`pm-engine`	`engine/config_loader.py`	`symbols`, `participants`, `participant_defaults`, `sessions_enabled`, `enforce_collars`, `enforce_circuit_breakers`, `require_mm_seed_quotes`, `auction_indicative_interval_sec`, `engine_tuning`, `mm_obligation_defaults`, `risk_controls`, `circuit_breaker_defaults`, `market_maker_combos`, `schedule`, `indices`	Symbol universe, allowed order-entry gateways, session/collar/order-limit/circuit-breaker policy, runtime tuning, MM obligations, startup combo seeds, session schedule, and index definitions
`pm-alf-gwy`	`alf_gwy/config.py`	`alf_gateway`, `participants`	Own bind address/port/timeouts, plus the gateway ID allowlist and roles for ALF client sessions
`pm-balf-gwy`	`balf_gwy/config.py`	`balf_gateway`, `participants`	Own bind address/port/timeouts, plus the gateway ID allowlist, roles, and `disconnect_behaviour` for BALF sessions
`pm-ralf-gwy`	`ralf_gateway/config.py`	`post_trade_gateway`	Own bind address/port/timeouts and `allowed_roles` for RALF (post-trade) subscribers
`pm-md-gwy`	`md_gateway/config.py`	`market_data_gateway`	Own bind address/port/timeouts, replay window, and `depth_levels` for CALF subscribers
`pm-dc-gwy`	`dc_gateway/config.py`	`dc_gateway`	Own bind address/port/timeouts and per-client queue limit for the drop-copy TCP relay

Process	Loader module	Top-level section(s) read	What it needs it for
`pm-log-srv`	`log_srv/config.py`	`log_server`	Own bind address/port/retention/throughput knobs for the centralized LALF log collector, plus the LALF-PS ZeroMQ log-distribution ports and subscription limits
`pm-monitor`	`monitor/config.py`	`monitor`, `locale.session_timezone`, `log_server.db_path`	Bind address and ports, health-judgement timings, retention and thresholds; the session timezone that defines "today"; the log database whose size it reports. Every beaconing process also reads `monitor` (`enabled`, `host`, `pull_port`, `beacon_interval_sec`) to know where and how often to send
`pm-api-gwy`	`api_gateway/config.py`	`api_gateways`	Named REST/WebSocket gateway instances, API credentials, rate limits, and timeouts
`pm-index`	`index/config_loader.py` (wraps `engine/config_loader.py`)	`indices`, `symbols.<SYM>.outstanding_shares`, `symbols.<SYM>.last_buy_price` / `last_sell_price`	Index definitions (constituents, base value, publish interval) and the per-constituent share counts / reference prices needed to seed each index at startup

Process	Loader module	Top-level section(s) read	What it needs it for
`pm-scheduler`	`scheduler/main.py`	`schedule`, `locale.country`	Session-phase transition times — re-reads the same block `pm-engine` reads, but as an independent process so the schedule can be driven or tested externally — plus the country used to skip weekends and bank holidays

Tooling reads everything

`pm-cverifier` and `pm-config-show` are the exceptions: the linter parses and cross-validates the whole file, and the viewer displays all of it, including sections no runtime process consumes on its own. Neither is a "reader" in the operational sense above — see Verify Configs with `pm-cverifier` and Inspect Configs with `pm-config-show`.

Two practical consequences follow from this split:

- A typo in, say, `balf_gateway` will not be caught by `pm-engine` at all — only `pm-balf-gwy` (or `pm-cverifier`) will reject it. Run `pm-cverifier` before starting a full stack to catch cross-section mistakes early.
- Each gateway process can be restarted independently with a changed config section (for example, bumping `market_data_gateway.depth_levels`) without restarting `pm-engine`, since the engine never reads that section.

3.3 Configuring `pm-alf-gwy`

`pm-alf-gwy` is the TCP front door for the ALF text protocol. It reads an optional top-level `alf_gateway` block from the same `engine_config.yaml`. The block is not consumed by `pm-engine`; it only configures the gateway process itself. Who may log in, and with which role, is **not** set here: the gateway takes that from the `participants` list (see Participants). Omit the block entirely to run on the defaults shown below.

Minimal example (every value shown is the default):

```
alf_gateway:
  enabled: true
  name: alf-gwy01
  bind_address: 0.0.0.0
  port: 5565
  heartbeat_interval_sec: 5
  handshake_timeout_sec: 10
  idle_timeout_sec: 30
  max_connections: 1024
  max_client_queue: 10000
  max_commands_per_second: 100
  max_errors_before_disconnect: 50
  error_window_sec: 60
```

3.3.1 Fields

Field	Default	Rule	What it does
`enabled`	`true`	boolean	When `false`, `pm-alf-gwy` logs a warning and refuses to start
`name`	`alf-gwy01`	string	Reported to every client as `GW=` in the `WELCOME` line
`bind_address`	`0.0.0.0`	string	Interface the listener binds. Use `127.0.0.1` for loopback-only. The `EDUMATCHER_GATEWAY_BIND_HOST` environment variable overrides this value
`port`	`5565`	`1..65535`	TCP port clients connect to
`heartbeat_interval_sec`	`5`	integer `> 0`	How often the gateway sends `HB` to an authenticated client that has received nothing else in that time. Advertised as `HBINT=` in `WELCOME`
`handshake_timeout_sec`	`10`	integer `> 0`	How long a new connection has to complete the `HELLO` login
`idle_timeout_sec`	`30`	integer `> 0`	How long a client may stay silent before it is disconnected. Advertised as `IDLE=` in `WELCOME`
`max_connections`	`1024`	integer `> 0`	Cap on simultaneous client connections
`max_client_queue`	`10000`	integer `> 0`	Cap on lines queued for one client that is not reading
`max_commands_per_second`	`100`	integer `> 0`	Per-client command rate limit
`max_errors_before_disconnect`	`50`	integer `> 0`	Number of protocol errors tolerated within `error_window_sec`
`error_window_sec`	`60`	integer `> 0`	Length of the sliding window over which errors are counted

3.3.2 How the settings work together

A connection goes through these stages, and each setting acts at one of them.

1. **Accept.** A new TCP connection is accepted unless `max_connections` clients are already connected; in that case the socket is closed immediately and a warning is logged. The client receives no `ERR` line, only a closed connection.
2. **Login.** The first message must be `HELLO`. If login is not complete within `handshake_timeout_sec`, the gateway sends `ERR|CODE=AUTH_TIMEOUT` and closes the connection. On success the client gets `WELCOME|PROTO=ALF1|GW=<name>|ID=<gateway id>|HBINT=<n>|IDLE=<n>`, so a client can read the timers it must respect instead of hard-coding them.
3. **Keep-alive.** Once logged in, the gateway sends `HB` whenever `heartbeat_interval_sec` has passed without any other line going out to that client. In the other direction, *any* bytes received from the client count as activity; if nothing arrives for `idle_timeout_sec` the gateway sends

`ERR|CODE=IDLE_TIMEOUT` and closes. A client that is only watching and sending nothing must therefore send a `PING` more often than the idle timeout, so keep `idle_timeout_sec` comfortably above the interval at which your clients ping.

4. **Commands.** Each client has its own token bucket that holds up to `max_commands_per_second` tokens and refills at that rate; every command spends one token. A command arriving with the bucket empty is rejected with `ERR|CODE=RATE_LIMITED` and is **not** forwarded to the engine; the connection stays open. `PING`, `EXIT` and `QUIT` after login do not spend tokens.
5. **Errors.** Every `ERR` the gateway sends for a client's mistake (bad syntax, unknown command, rejected request, `RATE_LIMITED`, and so on) is recorded with a timestamp. Timestamps older than `error_window_sec` drop out. When `max_errors_before_disconnect` errors are inside the window, the gateway sends `ERR|CODE=MAX_ERRORS` and disconnects. A client that is rate-limited all the time is therefore eventually disconnected; raise `max_commands_per_second` or lower the client's send rate.
6. **Slow consumers.** Lines for a client wait in an outbound queue until its socket accepts them. If that queue reaches `max_client_queue` lines, the gateway drops the queued `TRADE` prints and heartbeats, keeps the client's own events (fills, acks, cancels), queues `ERR|CODE=SLOW_CLIENT` after them and closes the connection once they are sent, or after 5 seconds if the client still does not read. One stuck client cannot consume the gateway's memory.

Choosing values for a classroom: the defaults suit a few dozen desks. Raise `max_connections` if more clients connect, raise `max_commands_per_second` for bots that send quote refreshes in bursts, and lower `idle_timeout_sec` if you want abandoned sessions released quickly.

3.3.3 Overrides and generation

- `pm-alf-gwy --bind ADDR` and `--port N` override `bind_address` and `port` for one run; `--engine-host HOST` points the gateway at an engine on another machine. The other fields can only be changed in the file.
- Changing the block takes effect when `pm-alf-gwy` is restarted; `pm-engine` does not need to be restarted, since it never reads this block.
- `pm-config-gen` emits the block with `--alf-gateway` and the `--alf-*` overrides listed under ALF gateway options.
- `pm-cverifier` validates every field above and reports an unrecognised key in the block (`S121`), which would otherwise be ignored and leave the default in force.

See ALF Gateway for operating the gateway and ALF Protocol for the message formats.

3.4 Configuring `pm-ralf-gwy`

`pm-ralf-gwy` reads an optional top-level `post_trade_gateway` block from the same `engine_config.yaml` file used by the engine. This block is not consumed by `pm-engine`; it is consumed by the RALF dissemination gateway process itself.

Minimal example:

```
post_trade_gateway:
  name: ralf-gwy01
  bind_address: 0.0.0.0
  port: 5580
  replay_retention_sec: 86400
  heartbeat_interval_sec: 2
  idle_timeout_sec: 20
  max_connections: 64
  max_client_queue: 10000
  allowed_roles:
    - CLEARING
    - DROP_COPY
    - AUDIT
```

Use this block to control where the RALF gateway listens and which external client roles it will accept. In the current implementation:

- `name` is the gateway id reported in `WELCOME`
- `bind_address` and `port` define the TCP listener for external subscribers
- `replay_retention_sec` controls the in-memory replay window
- `heartbeat_interval_sec` controls `HB` cadence
- `idle_timeout_sec` controls inactive-session disconnect timing
- `max_connections` caps simultaneous TCP connections
- `max_client_queue` caps slow-client buffering before `SLOW_CLIENT`
- `allowed_roles` limits accepted `HELLO|ROLE=...` values

If you prefer to generate this block instead of writing it by hand, `pm-config-gen` can emit it with `--post-trade-gateway` and optional `--post-trade-*` overrides.

3.5 Configuring `pm-md-gwy`

`pm-md-gwy` reads an optional top-level `market_data_gateway` block from the same `engine_config.yaml` file. This block is not consumed by `pm-engine`; it is consumed by the CALF market-data gateway process itself.

Minimal example:

```
market_data_gateway:
  enabled: true
  name: md-gwy01
  bind_address: 0.0.0.0
  port: 5570
  heartbeat_interval_sec: 1
  idle_timeout_sec: 5
  replay_window_sec: 30
  max_connections: 64
  max_messages_per_second: 200
  max_symbols_per_client: 200
  max_client_queue: 10000
  depth_levels: 10
```

Use this block to control whether the CALF gateway starts and how it serves subscribers. In the current implementation:

- `enabled` controls whether `pm-md-gwy` starts serving clients
- `name` is the gateway id reported in welcome/session payloads
- `bind_address` and `port` define the TCP listener for CALF subscribers

- `heartbeat_interval_sec` controls heartbeat cadence
- `idle_timeout_sec` controls inactive-session disconnect timing
- `replay_window_sec` controls the in-memory replay history window
- `max_connections` caps simultaneous TCP connections
- `max_messages_per_second` is the inbound per-client token-bucket rate limit
- `max_symbols_per_client` caps per-client subscription fanout
- `max_client_queue` caps slow-client buffering
- `depth_levels` sets how many aggregated price levels per side are sent on the CALF `DEPTH` channel (see 920-app-calf-protocol.md)

If you prefer to generate this block instead of writing it by hand, `pm-config-gen` can emit it with `--market-data-gateway` and optional `--market-data-*` overrides.

3.6 Configuring `pm-balf-gwy`

`pm-balf-gwy` reads an optional top-level `balf_gateway` block from the same `engine_config.yaml` file used by the engine. This block is not consumed by `pm-engine`; it is consumed by the BALF binary TCP gateway process.

Gateway identities and disconnect behaviour are read from the existing `participants` list — no separate credentials block is needed.

Minimal example:

```
balf_gateway:
  enabled: true
  name: balf-gwy01
  bind_address: 0.0.0.0
  port: 5560
  heartbeat_interval_sec: 1
  heartbeat_timeout_sec: 5
  idle_timeout_sec: 30
  auth_timeout_sec: 10
  max_connections: 64
  max_client_queue: 10000
  max_messages_per_second: 100
  max_errors_before_disconnect: 10
  error_window_sec: 60
  duplicate_session_policy: REJECT_NEW
```

Field	Default	Description
`enabled`	`true`	When `false`, `pm-balf-gwy` logs a warning and refuses to start
`name`	`balf-gwy01`	Gateway name echoed in the `LOGON_ACK` message field
`bind_address`	`0.0.0.0`	TCP listen interface (`127.0.0.1` for loopback-only)
`port`	`5560`	TCP listen port
`heartbeat_interval_sec`	`1`	Seconds between server-initiated `HEARTBEAT` frames when no other outbound traffic

Field	Default	Description
`heartbeat_timeout_sec`	5	Disconnect session if no inbound traffic arrives within this window
`idle_timeout_sec`	30	Accepted but currently unused by the gateway runtime — only `heartbeat_timeout_sec` drives idle-session disconnects. Reserved for future use.
`auth_timeout_sec`	10	Hard-close unauthenticated connections if `LOGON` is not completed within this window
`max_connections`	64	Maximum simultaneous TCP connections
`max_client_queue`	10000	Per-client outbound frame buffer depth before `SLOW_CLIENT` disconnect
`max_messages_per_second`	100	Token-bucket inbound rate limit per client
`max_errors_before_disconnect`	10	Error threshold in the sliding `error_window_sec` before forced disconnect
`error_window_sec`	60	Sliding window length (seconds) for the error counter
`duplicate_session_policy`	`REJECT_NEW`	What to do when a second `LOGON` arrives for an already-connected gateway ID: `REJECT_NEW` or `EVICT_OLD`

If you prefer to generate this block instead of writing it by hand, `pm-config-gen` can emit it with `--balf-gateway` and optional `--balf-*` overrides. See BALF TCP Gateway for the full client-facing documentation.

3.7 Configuring `pm-dc-gwy`

`pm-dc-gwy` reads an optional top-level `dc_gateway` block from the same `engine_config.yaml` file. This block is not consumed by `pm-engine`; it is consumed by the drop-copy TCP relay process itself. Gateway identities and roles are read from the existing `participants` list — no separate credentials block is needed. This block has no `enabled` key.

Minimal example (every value shown is the default):

```
dc_gateway:
  name: dc-gwy01
  bind_address: 0.0.0.0
  port: 5590
  heartbeat_interval_sec: 2
  idle_timeout_sec: 20
  max_connections: 64
  max_client_queue: 10000
```

Field	Default	Description
`name`	`dc-gwy01`	Gateway name echoed in `WELCOME`

Field	Default	Description
`bind_address`	`0.0.0.0`	TCP listen interface (`127.0.0.1` for loopback-only)
`port`	`5590`	TCP listen port
`heartbeat_interval_sec`	`2`	Seconds between server-initiated `HB` lines
`idle_timeout_sec`	`20`	Disconnect a session when no traffic flows in either direction within this window
`max_connections`	`64`	Maximum simultaneous TCP connections
`max_client_queue`	`10000`	Per-client outbound buffer depth before a slow-client disconnect

The drop-copy source address the gateway relays from (`tcp://127.0.0.1:5557`, the engine's drop-copy `PUB` socket) is **not** configurable through this block — use `pm-dc-gwy --engine-dc-pub` to point at a non-default engine address for one run.

If you prefer to generate this block instead of writing it by hand, `pm-config-gen` can emit it with `--dc-gateway` and optional `--dc-*` overrides. See Drop-Copy Gateway for the full client-facing documentation.

3.8 Configuring `pm-api-gwy`

`pm-api-gwy` reads an optional top-level `api_gateways` block from the same `engine_config.yaml` file. This block is not consumed by `pm-engine`; it is consumed by the REST/WebSocket API gateway process.

Minimal generated example:

```yaml
api_gateways:
  desk:
    enabled: true
    host: 0.0.0.0
    port: 8080
    swagger_enabled: true
    log_level: info
    stats_db: data/stats.db
    audit_db: data/audit_index.db
    clearing_db: data/clearing.db
    order_retention_sec: 3600
    market_data_cache_sec: 60
    credentials:
      - api_key: key-trader01-example
        gateway_id: TRADER01
        description: Generated key for TRADER01
      - api_key: key-dashboard-example
        gateway_id: null
        description: Read-only dashboard client
    rate_limit:
      writes_per_second: 10
      burst: 20
    timeouts:
      engine_auth_sec: 3.0
      engine_reply_sec: 3.0
      wait_ack_sec: 3.0
```

Use this block to control where the REST API listens, whether Swagger is available, which bearer tokens are accepted, and how write rate limits and engine reply waits are applied. In the current implementation:

- `enabled` lets `pm-api-gwy` refuse startup when set to `false`
- `host` and `port` define the uvicorn HTTP listener
- `swagger_enabled` controls `/docs` and `/openapi.json`
- `stats_db` points history endpoints at the `pm-stats` SQLite database
- `audit_db` is `pm-audit`'s read-only index, used only by `GET /admin/orders/{order_id}/audit`; optional, and that endpoint returns `503` when the file is absent
- `clearing_db` is `pm-clearing`'s read-only ledger, used only for the session/MTD/YTD realized P&L on `GET /positions`; optional, and those three fields report `null` when the file is absent
- `order_retention_sec` bounds how long a terminal order stays in the in-memory order cache; `0` disables eviction
- `market_data_cache_sec` is the TTL for the per-symbol market-data cache backing WebSocket snapshot/resume
- `credentials[].api_key` is the bearer token used by REST and WebSocket clients
- `credentials[].gateway_id` maps a key to an ALF gateway; `null` is read-only
- a non-null `credentials[].gateway_id` may appear in only one `api_gateways` entry
- `credentials_file` names a YAML list of credentials, relative to the file that names it, in place of an inline `credentials` (not both)
- `rate_limit` applies per API key to write endpoints only
- `timeouts.engine_reply_sec` and `timeouts.wait_ack_sec` control request/reply and `?wait=ack` waits

If you prefer to generate this block instead of writing it by hand, `pm-config-gen` can emit it with `--api-gateway`. By default it generates one credential per configured ALF gateway. Add `--api-gateway-readonly-key` for a dashboard-style key with `gateway_id: null`, or pass explicit `--api-key` entries when you need

known token values.

When more than one named API gateway is configured, start each process with its entry name, for example `pm-api-gwy --instance desk`.

3.9 Configuring `pm-index`

`pm-index` reads an optional top-level `indices` block from the same `engine_config.yaml` file. This block is not consumed by `pm-engine`; it is consumed by the index calculation process.

Example with two indices:

```
indices:
  - id: EDU100
    description: EduMatcher broad index
    base_value: 1000.0
    publish_interval_sec: 1.0
    history_file: data/indexes/EDU100_history.jsonl
    state_file: data/indexes/EDU100_state.json
    constituents:
      - AAPL
      - MSFT
      - TSLA
  - id: TECH2
    description: Technology pair
    base_value: 500.0
    publish_interval_sec: 2.0
    history_file: data/indexes/TECH2_history.jsonl
    state_file: data/indexes/TECH2_state.json
    constituents:
      - AAPL
      - MSFT
```

Use this block to define one or more weighted-average price indices that `pm-index` tracks as trades print through the engine. In the current implementation:

- `id` is the unique index identifier used in events, history records, and derived file names
- `description` is a human-readable label emitted in `INDEX_OPEN` and `INDEX_UPDATED` events
- `base_value` is the divisor-normalized starting level; `1000.0` is the standard convention
- `publish_interval_sec` throttles how frequently index updates are published to subscribers
- `history_file` is a line-delimited JSON file where corporate-action and constituent events are persisted
- `state_file` is a JSON file where the current divisor and constituent shares are checkpointed
- `constituents` is an ordered list of symbols; each must have `outstanding_shares` set in `symbols:`

Constraints enforced at startup:

- Maximum 5 indices per config file
- Every constituent symbol must appear in `symbols:` with a positive `outstanding_shares`
- `id` values must be unique across the `indices` list

If you prefer to generate this block instead of writing it by hand, `pm-config-gen` can emit it with `--index` and the associated `--index-*` options. File paths are automatically derived from the index ID when not specified.

3.10 Configuring `pm-log-srv`

`pm-log-srv` reads an optional top-level `log_server` block from the same `engine_config.yaml` file. This block is not consumed by `pm-engine`; it is consumed by the centralized LALF log-collector process. See Centralized Log Server for the operational guide (starting the server, using `pm-log-cli`) and LALF Protocol Reference for the normative wire specification.

Minimal example:

```
log_server:
  enabled: true
  name: log-srv01
  bind_address: 0.0.0.0
  port: 5600
  db_path: data/log.db
  retention_days: 30
  max_message_bytes: 65536
  max_client_queue: 10000
  write_batch_size: 50
  write_batch_interval_ms: 100
  heartbeat_interval_sec: 5
```

Field	Default	Description
`enabled`	`true`	Master switch — lets `pm-log-srv` refuse to accept connections when set to `false`
`name`	`log-srv01`	Server name echoed in the LALF `WELCOME.SRV` field
`bind_address`	`0.0.0.0`	TCP listen interface (`127.0.0.1` for loopback-only)
`port`	`5600`	TCP listen port for LALF clients
`db_path`	`data/log.db`	SQLite database path where `log_events`/`processes`/`server_stats` are stored
`retention_days`	`30`	Prune `log_events` rows older than this many days, once per hour; `null` (or `--retention-days 0`) means unbounded retention
`max_message_bytes`	`65536`	Maximum `LOG` payload size before truncation — oversized messages are truncated and stored, never dropped
`max_client_queue`	`10000`	Uncommitted-row backlog above which the server stops reading from clients (TCP backpressure); also caps one client's unread replies
`write_batch_size`	`50`	Maximum rows per SQLite transaction in the background writer thread
`write_batch_interval_ms`	`100`	Maximum time between writer-thread flushes, whichever comes first with `write_batch_size`

Field	Default	Description
`heartbeat_interval_sec`	`5`	How often a connected client must send something (`LOG` or `HB`) to stay alive; the server disconnects after 2× this interval of silence and advertises the value to clients in `WELCOME.HBINT` (the server itself never sends `HB`). Doubles as the publish interval for the LALF-PS `log.server_state` tick

3.10.1 LALF-PS fields — the ZeroMQ log-distribution interface

Everything above configures how logging gets *in* to `pm-log-srv`. The fields below configure how it gets back *out*: `pm-log-srv` also binds a ZeroMQ `PUB`/`PULL` pair so live log viewers can be pushed rows as they are committed, rather than polling `log.db` on a timer. See LALF-PS for the full interface.

```
log_server:
  # ... the collector fields above ...
  pubsub_enabled: true
  pub_port: 5601
  pull_port: 5602
  lease_sec: 30
  max_lease_sec: 300
  max_subscribers: 32
  notify_interval_ms: 250
  backfill_chunk_rows: 500
  max_backfill_minutes: 1440
  max_backfill_rows: 100000
  max_pending_rows: 20000
  pub_sndhwm: 10000
```

Field	Default	Description
`pubsub_enabled`	`true`	Master switch for LALF-PS. When `false`, no ZeroMQ socket is bound at all and `pm-log-srv` runs as a pure TCP collector
`pub_port`	`5601`	ZeroMQ `PUB` port carrying live rows, notification ticks, backfill chunks, control acks and errors
`pull_port`	`5602`	ZeroMQ `PULL` port receiving subscriber control requests
`lease_sec`	`30`	Subscription lease TTL. A `PUB` socket cannot see that a peer died, so every subscription carries a TTL the subscriber must refresh with `log.renew`; one that goes silent is reaped and its buffers discarded
`max_lease_sec`	`300`	Ceiling on a subscriber's requested lease. A request above it is *clamped*, not rejected. Must be `>= lease_sec`

Field	Default	Description
`max_subscribers`	`32`	Maximum concurrent leased subscriptions; further `log.subscribe` requests are answered with `TOO_MANY_SUBS`
`notify_interval_ms`	`250`	Coalescing window for `NOTIFY`-mode ticks, and the floor on a subscriber's own requested interval
`backfill_chunk_rows`	`500`	Rows per backfill chunk, and the maximum rows per live stream message
`max_backfill_minutes`	`1440`	Largest "last n minutes" window a subscriber may request (24 h)
`max_backfill_rows`	`100000`	Hard cap on rows returned by one backfill; the final chunk sets `truncated` when it bites
`max_pending_rows`	`20000`	Per-subscription stream buffer cap. A subscriber that is alive but too slow loses its oldest buffered rows rather than growing the server without bound
`pub_sndhwm`	`10000`	ZeroMQ send high-water mark on the `PUB` socket

`pm-log-srv` therefore occupies a contiguous three-port block — `5600`/`5601`/`5602` by default — and all three must be different. It refuses to start otherwise, and `pm-cverifier` reports the condition as `S102` before you ever get there. `pm-cverifier` also cross-checks the two LALF-PS ports against every other configured listener (`M018`) and rejects a `max_lease_sec` below `lease_sec` (`S103`).

Every CLI flag on `pm-log-srv` (`--host`, `--port`, `--db`, `--retention-days`, `--max-message-bytes`, `--pub-port`, `--pull-port`, `--lease-sec`, `--no-pubsub`) overrides the corresponding config field for that invocation only — the same CLI-flag-over-config precedence every other `pm-*` process uses.

If you prefer to generate this block instead of writing it by hand, `pm-config-gen` can emit it with `--log-server` and the associated `--log-server-*` options — see "Log server options" and "Log server LALF-PS options" in Generate Configs with pm-config-gen above.

3.11 Configuring `pm-monitor`

`pm-monitor` and every process that sends a health beacon read an optional top-level `monitor` block. An absent block means every default below, with monitoring on. See Health Monitoring for the operational guide.

```
monitor:
  enabled: true
  bind_address: 127.0.0.1
  host: 127.0.0.1
  pull_port: 5610
  pub_port: 5611
  beacon_interval_sec: 2
  hung_after_sec: 5
  silent_after_beacons: 3
  book_metrics_interval_sec: 10
  retention:
    raw_hours: 6
    minute_days: 7
  thresholds:
    host_cpu_pct:         {warn: 80,   crit: 95}
    host_mem_pct:         {warn: 80,   crit: 92}
    disk_free_pct:        {warn: 15,   crit: 5}
    engine_handle_p99_us: {warn: 5000, crit: 20000}
    reject_pct_5m:        {warn: 10,   crit: 30}
    rss_growth_pct_1h:    {warn: 50,   crit: 100}
    heap_bloat_ratio:     {warn: 1.5,  crit: 3}
```

Field	Default	Description
`enabled`	`true`	Master switch. When `false` no process sends beacons and `pm-monitor` idles until stopped
`bind_address`	`127.0.0.1`	Interface `pm-monitor` binds. Stored as written; `EDUMATCHER_GATEWAY_BIND_HOST` overrides it where `pm-monitor` runs
`host`	`127.0.0.1`	Where every process sends its beacons and where `pm-monitor-cli` and the dashboard connect; must reach `bind_address`
`pull_port`	`5610`	Beacons in (PULL)
`pub_port`	`5611`	`health.snapshot` and `health.event` out (PUB)
`beacon_interval_sec`	`2`	How often each process reports
`hung_after_sec`	`5`	Main-loop age above which a process is HUNG; must be greater than `beacon_interval_sec`
`silent_after_beacons`	`3`	Missed beacons before a process is SILENT (process alive) or DEAD
`book_metrics_interval_sec`	`10`	How often the engine recomputes its per-book figures and memory estimates
`retention.raw_hours`	`6`	Hours of 10-second samples in `metrics.db`
`retention.minute_days`	`7`	Days of 1-minute buckets, events and counter state; daily totals are kept for good

Field	Default	Description
`thresholds.<rule>.warn` / `.crit`	as above	Breach levels. `warn` must be below `crit`, except `disk_free_pct`, where a lower value is worse. A rule left out keeps its default

`pm-cverifier` checks the shape (`S124`–`S131`), the cross-field rules (`M029` band order, `M030` `hung_after_sec` above the interval, `M031` `host` reaching `bind_address`) and the two ports against every other listener (`M018`, counted even when the block is absent). `pm-config-gen` writes the block with `--monitor` and the `--monitor-*` options — see "Monitor options" in Generate Configs with pm-config-gen.

3.12 Engine Behavior Flags

3.12.1 `sessions_enabled`

```
sessions_enabled: true
```

When `true`, the engine starts in `CLOSED` and accepts scheduler transitions. When `false`, the engine starts in `CONTINUOUS` and ignores scheduler transitions.

Scenario	Effective value
Config file present, field absent	`true`
No config file (unrestricted mode)	`false`

3.12.2 `enforce_collars`

```
enforce_collars: true
```

Controls whether configured price collars reject incoming orders. This defaults to `true` and should normally remain enabled outside tests.

3.12.3 `enforce_circuit_breakers`

```
enforce_circuit_breakers: true
```

Controls whether configured circuit breakers can halt symbols. This defaults to `true` and should normally remain enabled outside tests.

3.12.4 `auction_indicative_interval_sec`

```
auction_indicative_interval_sec: 1.0
```

Throttle, in seconds, between republications of the indicative uncross price and quantity while a symbol is in an auction call phase (`PRE_OPEN` or `PRE_CLOSE`). Must be numeric and greater than zero; defaults to `1.0` when omitted.

3.12.5 `engine_tuning`

```
engine_tuning:
  snapshot_interval_sec: 0.5
  quote_history_maxlen: 30
  drop_copy_buffer_size: 10000
  recent_trades_maxlen: 20
  depth_snapshot_tolerance_ticks: 100
```

`engine_tuning` groups low-level runtime retention and throttling knobs that affect memory usage, snapshot cost, and replay depth. All of them are optional; omitted fields fall back to built-in defaults.

3.12.6 `engine_tuning.snapshot_interval_sec`

```
engine_tuning:
  snapshot_interval_sec: 0.5
```

Controls the per-symbol throttle window for `book.<SYMBOL>` publications from dirty books.

Rules:

- must be numeric
- must be greater than zero
- defaults to `0.5` seconds when omitted

3.12.7 `engine_tuning.quote_history_maxlen`

```
engine_tuning:
  quote_history_maxlen: 30
```

Controls how many recently inactivated quotes per gateway are retained in memory for `QLEGS SHOW=RECENT` / `SHOW=ALL`.

Rules:

- must be an integer
- must be greater than zero
- defaults to `30` when omitted

3.12.8 `engine_tuning.drop_copy_buffer_size`

```
engine_tuning:
  drop_copy_buffer_size: 10000
```

Controls how many drop-copy events are retained in memory for replay after a subscriber reconnects.

Rules:

- must be an integer
- must be greater than zero
- defaults to `10000` when omitted

3.12.9 `engine_tuning.recent_trades_maxlen`

```
engine_tuning:
  recent_trades_maxlen: 20
```

Controls how many recent trade rows each order book keeps for snapshots and diagnostics.

Rules:

- must be an integer
- must be greater than zero
- defaults to `20` when omitted

3.12.10 `engine_tuning.depth_snapshot_tolerance_ticks`

```
engine_tuning:
  depth_snapshot_tolerance_ticks: 100
```

Controls the depth window around the last trade, measured in ticks, when the engine publishes aggregated depth snapshots.

Rules:

- must be an integer
- must be greater than zero
- defaults to `100` when omitted

3.13 Participant Defaults

The optional top-level `participant_defaults` block holds values that every `participants` entry **inherits when it omits the key**. Use it to state a uniform policy once instead of repeating it on each gateway.

```
participant_defaults:
  smp_action: CANCEL_AGGRESSOR
  disconnect_behaviour: CANCEL_ALL

participants:
  - id: TRADER01            # inherits both defaults
  - id: TRADER02
    smp_action: NONE        # explicit: allows self-trades on this gateway
  - id: OPS01
    role: ADMIN
    disconnect_behaviour: LEAVE_ALL
```

Field	Required	Accepted values	Default when omitted
`smp_action`	No	`NONE`, `CANCEL_AGGRESSOR`, `CANCEL_RESTING`, `CANCEL_BOTH`	`NONE`
`disconnect_behaviour`	No	`CANCEL_QUOTES_ONLY`, `CANCEL_ALL`, `LEAVE_ALL`	`CANCEL_QUOTES_ONLY`

Rules:

- a gateway's effective value is its own key, then `participant_defaults`, then the built-in default shown above
- an explicit gateway value always wins, including `smp_action: NONE` when the default is something else
- the block applies to every role; a `disconnect_behaviour` default of `CANCEL_ALL` also reaches `ADMIN` and `MARKET_MAKER` entries that omit the key, so give those entries their own value when they need a different behaviour
- `participant_defaults` must be a mapping, values are case-insensitive, and any key other than the two above is rejected at load (a mistyped name is not silently ignored)
- per-order `SMP=` still outranks both (see the `smp_action` note below)

`pm-config-gen` writes the block with `--participant-default-smp` and `--participant-default-disconnect`, `pm-cverifier` checks it (`S118`–`S120`), and the config GUI edits it under **Basics → Participants**.

3.14 Participants

Only gateway IDs listed under `participants` may connect and submit orders when a config file exists.

```
participants:
  - id: TRADER01
    description: Student workstation 1
    role: TRADER
    disconnect_behaviour: CANCEL_ALL
```

3.14.1 Participant Fields

Field	Required	Accepted values / type	Default
`id`	Yes	Non-empty string, uppercased by parser	None
`description`	No	String or null	Empty string
`role`	No	`TRADER`, `MARKET_MAKER`, `ADMIN`	`TRADER`
`disconnect_behaviour`	No	`CANCEL_QUOTES_ONLY`, `CANCEL_ALL`, `LEAVE_ALL`	`participant_defaults`, then `CANCEL_QUOTES_ONLY`
`quote_refresh_policy`	No	`INACTIVATE_ON_ANY_FILL`, `INACTIVATE_ON_FULL_FILL`, `NEVER_INACTIVATE`	`INACTIVATE_ON_ANY_FILL`
`enforce_mm_obligation`	No	Boolean	Global MM default
`mm_max_spread_ticks`	No	Positive integer	Global MM default, then `20`
`mm_min_qty`	No	Positive integer	Global MM default, then `100`
`mm_obligations`	No	Per-symbol mapping	Empty mapping
`smp_action`	No	`NONE`, `CANCEL_AGGRESSOR`, `CANCEL_RESTING`, `CANCEL_BOTH`	`participant_defaults`, then `NONE`

smp_action is a fallback default, not an override

`participants[].smp_action` is the self-match-prevention action the engine applies to this gateway's orders **when the order itself doesn't specify one**:

- `QUOTE` legs (the bid/ask orders a `QUOTE` command generates via `pm-alf-gwy` or the ALF console) have no per-request SMP concept of their own, so they **always** use this gateway default.
- `NEW` orders and combo orders/legs submitted through `pm-alf-gwy`, the ALF console, or the REST API gateway each carry their own optional per-order `SMP=` field (see the `NEW` command in ALF Protocol). If the client sends an explicit `SMP=` — including `SMP=NONE`, a deliberate request to allow self-trades — that value is always honoured as-is. Only when the client omits `SMP=` entirely does the engine fall back to this gateway's `smp_action` (or the inherited `participant_defaults.smp_action`), and finally to `NONE` if neither is configured.

In short: an explicit per-order `SMP=` always wins; `participants[].smp_action` only fills the gap when the client didn't say anything.

Nested `mm_obligations.<SYMBOL>` entries support these fields:

Field	Required	Accepted values / type	Default
`enforce_mm_obligation`	No	Boolean	Gateway `enforce_mm_obligation`

`max_spread_ticks`	No	Integer; use positive values for a valid spread limit	Gateway `mm_max_spread_ticks`
`min_qty`	No	Integer; use positive values for a valid quantity floor	Gateway `mm_min_qty`

Inside `mm_obligations`, use this shape:

```
participants:
  - id: MM01
    role: MARKET_MAKER
    mm_obligations:
      AAPL:
        enforce_mm_obligation: true
        max_spread_ticks: 6
        min_qty: 300
```

Use `max_spread_ticks` and `min_qty` inside `mm_obligations`; do not use the flat-field names `mm_max_spread_ticks` and `mm_min_qty` there.

3.14.2 Role Privileges

Role	Regular orders	Quotes	Admin circuit-breaker halt/resume	Typical use
`TRADER`	Yes	No	No	Students, manual participants, AI traders
`MARKET_MAKER`	Yes	Yes	No	Quote providers
`ADMIN`	Yes	No	Yes	Instructor/operator console

`MARKET_MAKER` gateways are the only gateways allowed to submit quotes. `ADMIN` gateways can send exchange-wide circuit-breaker halt/resume commands.

3.15 Market-Maker Obligation Defaults

`mm_obligation_defaults` defines quote-quality policy inherited by market-maker gateways.

```
mm_obligation_defaults:
  enforce_mm_obligation: true
  mm_max_spread_ticks: 20
  mm_min_qty: 100
  symbols:
    AAPL:
      enforce_mm_obligation: true
      mm_max_spread_ticks: 8
      mm_min_qty: 200
```

Field	Required	Description
`enforce_mm_obligation`	No	Enable quote obligation checks
`mm_max_spread_ticks`	No	Maximum allowed bid/ask spread in ticks
`mm_min_qty`	No	Minimum bid and ask quantity

Field	Required	Description
`symbols`	No	Per-symbol overrides using the same three fields

Defaults and validation:

Field	Accepted values / type	Default
`enforce_mm_obligation`	Boolean	`false`
`mm_max_spread_ticks`	Positive integer	`20`
`mm_min_qty`	Positive integer	`100`
`symbols.<SYMBOL>.enforce_mm_obligation`	Boolean	Top-level `enforce_mm_obligation`
`symbols.<SYMBOL>.mm_max_spread_ticks`	Positive integer	Top-level `mm_max_spread_ticks`
`symbols.<SYMBOL>.mm_min_qty`	Positive integer	Top-level `mm_min_qty`

The effective policy is resolved from most specific to least specific:

1. `participants[*].mm_obligations.<SYMBOL>`
2. `mm_obligation_defaults.symbols.<SYMBOL>`
3. Gateway flat fields
4. `mm_obligation_defaults` flat fields
5. Built-in defaults

3.16 Symbol Universe

Only symbols declared under `symbols` are accepted by the configured engine.

Adding a symbol is an IPO

Treat each symbol as an **initial listing (IPO)**: you set its opening reference price, issued shares, and (when a market maker is configured) its opening quote up front, and those values seed the book and both risk-control references. The symbol universe is fixed at startup — the engine does **not** support adding symbols intra-day. See Adding or Removing Symbols and Risk Controls - Day one (IPO) behaviour.

```
symbols:
  AAPL:
    tick_decimals: 2
  MSFT: {}
  TSLA:
```

Symbol keys are uppercased. The value for a symbol may be a mapping, `{}`, or null.

3.16.1 Symbol Fields

Field	Required	Type / accepted values	Description
`tick_decimals`	No	Integer `0..8`	Decimal places used to convert display prices to ticks

Field	Required	Type / accepted values	Description
`level`	No	Key from `risk_controls.levels`	Named collar profile
`outstanding_shares`	Conditional	Positive integer	Required only if the symbol is an index constituent
`last_buy_price`	No	Number	Initial last-buy reference when no persisted stat exists
`last_sell_price`	No	Number	Initial last-sell reference when no persisted stat exists
`collar`	No	Mapping	Symbol-level collar override
`circuit_breaker`	No	Mapping	Symbol-level circuit-breaker override
`market_maker_quotes`	Conditional	List of mappings	Required (non-empty) only if a `MARKET_MAKER` gateway exists

3.16.2 Mandatory Fields

None of `tick_decimals`, `level`, `last_buy_price`, `last_sell_price`, `collar`, or `circuit_breaker` are ever mandatory — each has a built-in default or is simply left inactive when omitted. Two fields become mandatory, but only under specific conditions:

- **`market_maker_quotes`** — becomes mandatory (must be a non-empty list) for *every* symbol as soon as any `participants` entry has `role: MARKET_MAKER`, unless the top-level `require_mm_seed_quotes` is explicitly set to `false` (default `true`). With `require_mm_seed_quotes: false`, a `MARKET_MAKER` gateway may exist with no `market_maker_quotes` entries at all. If no `MARKET_MAKER` gateway is configured, this field can be omitted for every symbol regardless of `require_mm_seed_quotes`.
- **`outstanding_shares`** — becomes mandatory (must be a positive integer) only for symbols listed in an `indices[].constituents` entry (see Configuring `pm-index`). A symbol not referenced by any index does not need `outstanding_shares`.

3.16.3 Collar Reference Price Selection

When a symbol ends up with an active collar (via `symbols.<SYMBOL>.collar`, its `level`, or `risk_controls.default_level` — see Risk Controls and Collars), the engine derives the collar's static-band `reference_price` at startup as follows:

1. Persisted `<DATA_DIR>/book_stats.json` values are restored first (see Startup and Persistence Order). If the symbol has a persisted `last_buy_price`, it is used.
2. Otherwise, if the symbol has a persisted `last_sell_price`, it is used.
3. Otherwise, fall back to the config file's `last_buy_price`.
4. Otherwise, fall back to the config file's `last_sell_price`.
5. If none of the above are set, the collar is still parsed but is **not** activated for that symbol — no collar check runs for it at all, even though `enforce_collars: true` and a `collar`/`level` section are present.

In short: **persisted `book_stats.json` prices always take precedence over the `last_buy_price` / `last_sell_price` seed values in `engine_config.yaml`**, so the collar tracks the most recently known trading price instead of a config value that may go stale as the session progresses. This uses the same resolved last-buy/last-sell prices that seed the order book itself, so the collar reference and the book's displayed last prices never disagree.

Orders for unknown symbols are rejected with:

```
Symbol not configured: UNKNOWN
```

3.17 Risk Controls and Collars

`risk_controls` defines reusable collar levels.

```
risk_controls:
  default_level: L2
  levels:
    L1:
      collar:
        static_band_pct: 0.30
        dynamic_band_pct: 0.05
    L2:
      collar:
        static_band_pct: 0.20
        dynamic_band_pct: 0.02
```

3.17.1 Per-symbol risk-level assignment

Use per-symbol risk levels when different symbols should inherit different named collar profiles from `risk_controls.levels`.

You can assign the symbol level directly in YAML:

```
risk_controls:
  default_level: DEFAULT
  levels:
    DEFAULT:
      collar:
        static_band_pct: 0.20
        dynamic_band_pct: 0.02
    CORE:
      collar:
        static_band_pct: 0.18
        dynamic_band_pct: 0.02
    HIGH_BETA:
      collar:
        static_band_pct: 0.12
        dynamic_band_pct: 0.04

symbols:
  AAPL:
    level: CORE
  TSLA:
    level: HIGH_BETA
```

Or generate the same structure from CLI:

```
pm-config-gen \
  --symbols AAPL TSLA \
  --participants TRADER01 \
  --risk-level CORE:0.18:0.02 \
  --risk-level HIGH_BETA:0.12:0.04 \
  --symbol-risk-level AAPL:CORE \
  --symbol-risk-level TSLA:HIGH_BETA
```

Semantics:

- `symbols.<SYMBOL>.level` selects one named profile from `risk_controls.levels`.
- If `level` is omitted, the symbol uses `risk_controls.default_level` when present.
- If neither a symbol level nor `default_level` applies, the symbol has no collar unless `symbols.<SYMBOL>.collar` is defined directly.
- `symbols.<SYMBOL>.collar` remains the highest-priority per-field override over any selected level.

3.17.2 Risk-control Fields

Field	Required	Accepted values / type	Default
`default_level`	No	Non-empty string matching a key in `levels`	None
`levels`	No	Mapping of named level configs	Empty mapping
`levels.<LEVEL>`	No	Mapping; level name is uppercased	None
`levels.<LEVEL>.collar`	No	Mapping	Empty mapping

3.17.3 Collar Fields

Collars may appear under `risk_controls.levels.<LEVEL>.collar` or under `symbols.<SYMBOL>.collar`.

Field	Required	Accepted values / type	Default when a collar is active
`static_band_pct`	No	Number in `(0, 1)`	`0.20`
`dynamic_band_pct`	No	Number in `(0, 1)`	`0.02`

Meaning of collar values:

- `static_band_pct` is an absolute guard around the symbol reference price (for example prior close or seeded last price). A value of `0.20` means allow prices within ±20% of that reference.
- `dynamic_band_pct` is an incremental guard around the latest trade price. A value of `0.02` means allow prices within ±2% of the latest fill.

This is the same behavior described in Risk Controls and implemented in `src/edumatcher/engine/collar.py`.

Validation rules:

- `risk_controls` must be a mapping
- `risk_controls.default_level` must reference a key under `risk_controls.levels`
- each `levels.<LEVEL>.collar` must be a mapping when present
- `risk_controls.levels.<LEVEL>.circuit_breaker` is not supported; use top-level `circuit_breaker_defaults`
- collar percentages must be in `(0, 1)` after level and symbol overrides are merged

A symbol only gets a collar if at least one of these is present: `symbols.<SYMBOL>.collar`, the symbol's `level` collar, or the `risk_controls.default_level` collar. If none of them apply, the symbol has **no collar at all**, even when `enforce_collars: true`.

When a collar *is* active, its two fields are resolved most-specific first:

1. `symbols.<SYMBOL>.collar` (per-field override)
2. `symbols.<SYMBOL>.level` collar

3. `risk_controls.default_level` collar
4. built-in field defaults (`static_band_pct: 0.20`, `dynamic_band_pct: 0.02`)

The built-in defaults in step 4 only fill in fields that none of the higher tiers set; they never create a collar on their own.

3.18 Circuit Breakers

`circuit_breaker_defaults` defines the default threshold ladder. Symbol-level `circuit_breaker` sections merge over it field by field.

For an operational comparison of symbol-level circuit breakers versus symbol price collars, see Risk Controls - Price collars vs circuit breakers.

```
circuit_breaker_defaults:
  reference_window_ns: 300000000000
  levels:
    L1:
      price_shift_pct: 0.07
      halt_duration_ns: 300000000000
    L2:
      price_shift_pct: 0.13
      halt_duration_ns: 900000000000
    L3:
      price_shift_pct: 0.20
      halt_duration_ns:

symbols:
  TSLA:
    circuit_breaker:
      levels:
        L1:
          halt_duration_ns: 600000000000
```

Validation rules:

- `circuit_breaker_defaults` must be a mapping when present
- `levels` must be a non-empty mapping after defaults and symbol overrides merge
- each level requires `price_shift_pct` in `(0, 1)`
- `halt_duration_ns` must be a positive integer or null
- `reference_window_ns` is converted to integer nanoseconds

A halt has no resumption setting, and deliberately so. The halt period *is* a reopening auction's call phase — LIMIT orders are accepted and rest, market and immediate-or-cancel orders are rejected, and no matching runs — so every halt ends in an uncross at the equilibrium price. Resuming without one would restart continuous matching on a book that had been accumulating crossed interest for the whole halt.

A symbol only gets a circuit breaker if `circuit_breaker_defaults` or its own `symbols.<SYMBOL>.circuit_breaker` section is present. If neither exists, the symbol has **no circuit breaker at all**, even when `enforce_circuit_brea` `true`.

When a breaker *is* active, configuration is resolved as follows:

1. `symbols.<SYMBOL>.circuit_breaker` (per-level, per-field override)
2. `circuit_breaker_defaults`
3. built-in ladder fallback (L1 7%/5m, L2 13%/15m, L3 20%/rest-of-day), used **only** when a

circuit-breaker section is present but supplies no `levels`

3.18.1 Circuit-breaker Fields

Circuit breakers may appear under `circuit_breaker_defaults` or under `symbols.<SYMBOL>.circuit_breaker`. Symbol-level fields merge over defaults.

Field	Required	Accepted values / type	Default
`reference_window_ns`	No	Integer nanoseconds	`300000000000`
`levels`	Yes when a breaker is active	Non-empty mapping after merging	Built-in L1/L2/L3 only when no levels are supplied
`levels.<LEVEL>.price_shift_pct`	Yes	Number in `(0, 1)`	None
`levels.<LEVEL>.halt_duration_ns`	No	Positive integer nanoseconds or null	Null

3.19 Market-Maker Quote Seeds

`market_maker_quotes` create linked bid/ask quote legs at engine startup.

```
symbols:
  AAPL:
    market_maker_quotes:
      - gateway_id: MM01
        quote_id: SEED-MM01-AAPL
        bid_price: 209.00
        ask_price: 211.00
        bid_qty: 2000
        ask_qty: 2000
        tif: DAY
        seed_once: true
```

Field	Required	Accepted values / type	Default	Description
`gateway_id`	Yes	Non-empty string, uppercased	None	Configured `MARKET_MAKER` gateway that owns the quote
`quote_id`	No	String; empty string is treated as omitted	Generated	Explicit quote label
`bid_price`	Yes	Number	None	Display price converted to ticks by the engine
`ask_price`	Yes	Number greater than `bid_price`	None	Display price converted to ticks by the engine
`bid_qty`	Yes	Positive integer	None	Bid-side quantity
`ask_qty`	Yes	Positive integer	None	Ask-side quantity
`tif`	No	`DAY`, `GTC`	`DAY`	Time in force (`pm-cverifier` rejects any other value, `S016`)

Field	Required	Accepted values / type	Default	Description
`seed_once`	No	Boolean-like value; use YAML `true`/`false`	`true`	Skip injection when this gateway already has a live quote on the symbol restored from the previous run

Validation rules:

- quote seeds must be mappings inside a list
- `gateway_id` must reference a configured gateway with `role: MARKET_MAKER`
- `bid_price` must be lower than `ask_price`
- quantities must be positive
- if any configured gateway has `role: MARKET_MAKER`, every symbol must define at least one quote seed

Live quote legs are saved with the resting orders in `gtc_orders.json` and restored on the next start. `seed_once: true` skips injection when the gateway already has a restored live quote on the symbol; a quote that was fully traded away or expired is not restored, so its seed fires again. `seed_once: false` injects on every startup, replacing any restored quote. See Persistence.

3.20 Startup Market-Maker Combo Seeds

`market_maker_combos` inject startup combo orders through the same combo path used by live combo entry.

```
market_maker_combos:
  - combo_id: SEED-PAIR-AAPL-MSFT
    combo_type: AON
    tif: DAY
    legs:
      - symbol: AAPL
        side: BUY
        order_type: LIMIT
        quantity: 100
        price: 209.50
        smp_action: NONE
      - symbol: MSFT
        side: SELL
        order_type: LIMIT
        quantity: 50
        price: 415.50
        smp_action: NONE
```

Combo fields:

Field	Required	Accepted values / type
`combo_id`	Yes	Non-empty string
`combo_type`	No	`AON`; defaults to `AON`
`tif`	No	`DAY`, `GTC`, `ATO`, `ATC`; defaults to `DAY`
`legs`	Yes	List with 2 to 10 entries

Leg fields:

Field	Required	Accepted values / type
`symbol`	Yes	Configured symbol, unique inside the combo
`side`	Yes	`BUY`, `SELL`
`order_type`	Yes	`MARKET`, `LIMIT`, `STOP`, `STOP_LIMIT`, `FOK`, `ICEBERG`, `IOC`, `TRAILING_STOP`
`quantity`	Yes	Integer quantity
`price`	Conditional	Display price for priced order types, on the leg symbol's tick grid
`stop_price`	Conditional	Display stop price for stop order types, on the leg symbol's tick grid
`smp_action`	No	`NONE`, `CANCEL_AGGRESSOR`, `CANCEL_RESTING`, `CANCEL_BOTH`; if omitted, falls back to the seeding gateway's `participants[].smp_action` (§below), then `NONE`

Combo leg values are passed to `ComboLeg.from_dict()`, so these are the only leg fields used by current config parsing. Unlike quote seeds, combo legs do not include a `gateway_id`; startup combo ownership is assigned by the engine's combo seed path — that combo-level `gateway_id` is what an omitted `smp_action` falls back through.

Prefer `tif: DAY` for repeatable demo seeds. `GTC` combo seeds can interact with restored `gtc_combos.json` state and duplicate intended startup liquidity if you are not managing persistence deliberately.

3.21 Session Schedule

The scheduler reads `schedule` and sends transitions to the engine. Every weekday (`mon`..`sun`) resolves to its own five-time block, or CLOSED if unconfigured; `weekdays:`/`weekend:` are shortcuts for repeating one block across Monday-Friday or Saturday+Sunday, and `holidays:` gives bank holidays their own schedule instead of defaulting to CLOSED. See Session Scheduling → Configuring the schedule for the full set of keys, precedence rules, and worked examples.

```
schedule:
  weekdays:
    pre_open: "09:00"
    opening_auction_start: "09:25"
    continuous_start: "09:30"
    closing_auction_start: "16:00"
    closing_auction_end: "16:05"
```

Key	Required	Applies to
`weekdays`	No	Monday-Friday, for any day not overridden below
`mon`..`fri`	No	One weekday, overriding `weekdays` for that day
`weekend`	No	Saturday and Sunday — mutually exclusive with `sat`/`sun`
`sat`, `sun`	No	One weekend day, overriding `weekend` for that day
`holidays`	No	Bank holidays for the configured `locale.country`

Every block that is present MUST define all five transition times (`pre_open`, `opening_auction_start`, `continuous_start`, `closing_auction_start`, `closing_auction_end`) — there is no per-key fallback for a partial block. A day with nothing resolved for it (not `weekdays`/`weekend` and not individually specified) is CLOSED. If no usable `schedule` section is present at all, `pm-scheduler` uses built-in defaults (the

times shown above, applied as `weekdays`; weekends and holidays CLOSED). With `pm-scheduler --now`, the wall-clock values are ignored and transitions are sent immediately with short delays.

The default session path is:

```
PRE_OPEN -> OPENING_AUCTION -> CONTINUOUS -> CLOSING_AUCTION -> CLOSED
```

3.21.1 `locale`

```
locale:
  country: Sweden
  session_timezone: Europe/Stockholm
```

`locale` is a top-level key, a sibling of `schedule`, grouping the two facts about where this exchange is: its country and its session timezone. Both sub-fields are optional and independent of each other — setting one does not change the other's default or validation.

`locale.country` — `pm-scheduler` uses it to decide, for a given calendar day, whether the `holidays:` block applies (see the table above), using the `python-holidays` package to resolve the holiday calendar.

Aspect	Value
Accepted forms	Country name (`"Sweden"`) or ISO 3166-1 alpha-2 code (`"SE"`)
Default when omitted	`"Sweden"`
Behavior on an unrecognized value	Falls back to `"Sweden"` and logs a warning
Weekends	No longer forced CLOSED — `weekend:`/`sat:`/`sun:` can give them their own schedule, same as any weekday

Under `--daily`, a CLOSED day is skipped and the scheduler sleeps through to the next day that resolves to a schedule, rather than the next calendar day. In single-shot mode (the default, no `--daily`), the scheduler simply sends no transitions and exits if today is CLOSED. See Session Scheduling → Bank holidays and weekends for the full behavior breakdown by run mode.

`locale.session_timezone` — the IANA timezone (e.g. `"Europe/Stockholm"`, `"UTC"`) used to decide which calendar date, in that timezone, a UTC-timestamped event falls into. `pm-stats` and `pm-clearing` both read it from the *compiled* config to bucket events into trading days; neither takes a `--timezone` CLI flag any more.

Aspect	Value
Accepted forms	`"UTC"` or any IANA timezone name recognized by `zoneinfo`
Default when omitted	`"Europe/Stockholm"`
Behavior on an unrecognized value	Rejected — the process refuses to start

Both `pm-stats` and `pm-clearing` persist the timezone name they first run with and refuse to start if the configured value later contradicts what is already recorded in their database, since re-bucketing an existing database's events under a different trading-day boundary would silently change already-published figures.

3.22 Formal Specification

This section is a complete machine-readable-style reference for every field parsed from `engine_config.yaml`. Types follow Python conventions: `bool`, `int`, `float`, `str`. "Enum" means the field must match one of the listed string values exactly (case-insensitive during loading; stored in uppercase).

For the auxiliary gateway processes (`pm-alf-gwy`, `pm-balf-gwy`, `pm-md-gwy`, `pm-ralf-gwy`, `pm-api-gwy`) and the full normative schema, see App Config Spec, which governs if the two documents ever disagree. Ranges use mathematical interval notation: `(a, b)` is open (exclusive), `[a, b]` is closed (inclusive).

3.22.1 Top-level fields

Field	Type	Required	Default	Allowed values / range	Constraint
`symbols`	mapping	Yes	—	—	Must contain at least one entry
`participants`	list	Yes	—	—	Non-empty list of participant mappings
`sessions_enabled`	bool	No	`true` when file exists, `false` in unrestricted mode	`true`, `false`	Must be a YAML boolean
`enforce_collars`	bool	No	`true`	`true`, `false`	Must be a YAML boolean
`enforce_circuit_breakers`	bool	No	`true`	`true`, `false`	Must be a YAML boolean
`engine_tuning`	mapping	No	—	—	Runtime tuning block
`mm_obligation_defaults`	mapping	No	—	—	—
`risk_controls`	mapping	No	—	—	—
`circuit_breaker_defaults`	mapping	No	—	—	—
`market_maker_combos`	list	No	`[]`	—	Each entry must be a mapping
`schedule`	mapping	No	—	—	Parsed by scheduler and stored by engine
`require_mm_seed_quotes`	bool	No	`true`	`true`, `false`	If `false`, a `MARKET_MAKER` gateway may exist with no `market_maker_quotes` (see Mandatory Fields)
`locale`	mapping	No	—	—	Country and session timezone; see `locale` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`indices`	list	No	`[]`	—	At most 5 entries; see Configuring `pm-index` for the per-index field reference
`auction_indicative_interval_sec`	float	No	`1.0`	Any number	Must be `> 0`
`participant_defaults`	mapping	No	—	`smp_action`, `disconnect_behaviour`	Inherited by `participants` entries that omit them; see Participant Defaults
`alf_gateway`	mapping	No	—	—	`pm-alf-gwy` settings; see Configuring `pm-alf-gwy`
`balf_gateway`	mapping	No	—	—	`pm-balf-gwy` settings; see Configuring `pm-balf-gwy`
`market_data_gateway`	mapping	No	—	—	`pm-md-gwy` settings; see Configuring `pm-md-gwy`
`post_trade_gateway`	mapping	No	—	—	`pm-ralf-gwy` settings; see Configuring `pm-ralf-gwy`
`dc_gateway`	mapping	No	—	—	`pm-dc-gwy` settings; see Drop-Copy Gateway
`log_server`	mapping	No	—	—	`pm-log-srv` settings; see Configuring `pm-log-srv`
`api_gateways`	mapping	No	—	—	Named `pm-api-gwy` instances; see Configuring `pm-api-gwy`
`monitor`	mapping	No	—	—	`pm-monitor` and health-beacon settings; an absent block means defaults, with the monitor enabled; see Configuring `pm-monitor`

3.22.2 `engine_tuning` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`snapshot_interval_sec`	float	No	`0.5`	Any number	Must be `> 0`
`quote_history_maxlen`	int	No	`30`	Positive integer	Must be `> 0`
`drop_copy_buffer_size`	int	No	`10000`	Positive integer	Must be `> 0`
`recent_trades_maxlen`	int	No	`20`	Positive integer	Must be `> 0`
`depth_snapshot_tolerance_ticks`	int	No	`100`	Positive integer	Must be `> 0`

3.22.3 `locale` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`country`	str	No	`"Sweden"`	Country name or ISO 3166-1 alpha-2 code	Unrecognized value falls back to the default (see `M026` in Config Verifier)
`session_timezone`	str	No	`"Europe/Stockholm"`	`"UTC"` or an IANA timezone name	Unrecognized value is rejected

---

3.22.4 `participants[]` — gateway entry fields

Field	Type	Required	Default	Allowed values / range	Constraint
`id`	str	Yes	—	Any non-empty string	Uppercased; must be unique within the list
`description`	str or null	No	`""`	Any string or null	Null is coerced to `""`
`role`	Enum	No	`TRADER`	`TRADER`, `MARKET_MAKER`, `ADMIN`	Case-insensitive

Field	Type	Required	Default	Allowed values / range	Constraint
`disconnect_behaviour`	Enum	No	`CANCEL_QUOTES_ONLY`	`CANCEL_QUOTES_ONLY`, `CANCEL_ALL`, `LEAVE_ALL`	Case-insensitive
`quote_refresh_policy`	Enum	No	`INACTIVATE_ON_ANY_FILL`	`INACTIVATE_ON_ANY_FILL`, `INACTIVATE_ON_FULL_FILL`, `NEVER_INACTIVATE`	Case-insensitive
`enforce_mm_obligation`	bool	No	From `mm_obligation_defaults.enforce_mm_obligation`, else `false`	`true`, `false`	Must be a YAML boolean
`mm_max_spread_ticks`	int	No	From `mm_obligation_defaults.mm_max_spread_ticks`, else `20`	Integer	Must be `> 0`
`mm_min_qty`	int	No	From `mm_obligation_defaults.mm_min_qty`, else `100`	Integer	Must be `> 0`
`mm_obligations`	mapping	No	`{}`	Mapping of symbol → obligation entry	Symbol keys are uppercased
`smp_action`	Enum	No	`NONE`	`NONE`, `CANCEL_AGGRESSOR`, `CANCEL_RESTING`, `CANCEL_BOTH`	Case-insensitive; fallback default used when an order (`NEW`, combo, or `QUOTE`) doesn't specify its own `SMP=`

3.22.5 `participants[].mm_obligations.<SYMBOL>` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`enforce_mm_obligation`	bool	No	Gateway `enforce_mm_obligation`	`true`, `false`	Must be a YAML boolean
`max_spread_ticks`	int	No	Gateway `mm_max_spread_ticks`	Integer	Must be `> 0`
`min_qty`	int	No	Gateway `mm_min_qty`	Integer	Must be `> 0`

3.22.6 `mm_obligation_defaults` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`enforce_mm_obligation`	bool	No	`false`	`true`, `false`	Must be a YAML boolean
`mm_max_spread_ticks`	int	No	`20`	Integer	Must be `> 0`

Field	Type	Required	Default	Allowed values / range	Constraint
`mm_min_qty`	int	No	`100`	Integer	Must be `> 0`
`symbols`	mapping	No	`{}`	Symbol name → override mapping	Symbol keys are uppercased; each must reference a configured symbol

3.22.7 `mm_obligation_defaults.symbols.<SYMBOL>` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`enforce_mm_obligation`	bool	No	Top-level `enforce_mm_obligation`	`true`, `false`	Must be a YAML boolean
`mm_max_spread_ticks`	int	No	Top-level `mm_max_spread_ticks`	Integer	Must be `> 0`
`mm_min_qty`	int	No	Top-level `mm_min_qty`	Integer	Must be `> 0`

3.22.8 `risk_controls` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`default_level`	str	No	`null`	Any non-empty string	Must match a key in `risk_controls.levels` if set
`levels`	mapping	No	`{}`	Level name → level config mapping	Level names are uppercased

3.22.9 `risk_controls.levels.<LEVEL>` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`collar`	mapping	No	`{}`	See collar fields	Must be a mapping; `circuit_breaker` sub-key is rejected

3.22.10 Collar fields — in `risk_controls.levels.<LEVEL>.collar` or `symbols.<SYMBOL>.collar`

Field	Type	Required	Default	Allowed values / range	Constraint
`static_band_pct`	float	No	`0.20`	`(0, 1)` exclusive	Band truncates toward zero; makes range slightly tighter than exact
`dynamic_band_pct`	float	No	`0.02`	`(0, 1)` exclusive	Same truncation rule

3.22.11 `circuit_breaker_defaults` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`reference_window_ns`	int	No	`300000000000` (5 min)	Positive integer nanoseconds	Coerced to `int`
`levels`	mapping	No	Built-in L1/L2/L3 ladder only when a CB section is present but omits `levels`	Level name → level config mapping	Values must be mappings
`reopening`	mapping	No	Built-in ACE defaults	Automated Corridor Expansion settings	Merges field-by-field over defaults

3.22.12 `circuit_breaker_defaults.reopening` and `symbols.<SYMBOL>.circuit_breaker.reopening` fields

Governs how a circuit-breaker halt ends — see Risk Controls - Automated Corridor Expansion.

Field	Type	Required	Default	Allowed values / range	Constraint
`enabled`	bool	No	`true`	`true` / `false`	When `false` a halt reopens at the equilibrium price with no corridor
`initial_band_pct`	float	No	`0.10`	`(0, 1)` exclusive	Corridor half-width as a fraction of the CB reference price

Field	Type	Required	Default	Allowed values / range	Constraint
`expansions`	list	No	`[{0.10, 2 min}, {0.20, 5 min}]`	Non-empty list of mappings	The final entry repeats indefinitely. **Only valid under `circuit_breaker_defaults`**; per-symbol is an error (`S112`)
`expansions[].widen_pct`	float	Yes within an entry	—	`(0, 1)` exclusive	Added to the half-width; additive on the reference, not compounding
`expansions[].min_duration_ns`	int	Yes within an entry	—	Positive integer nanoseconds	Minimum length of that extension's call phase
`random_end_max_ns`	int	No	`30000000000` (30 s)	`>= 0` nanoseconds	Uniform random tail added to every call phase; `0` disables it
`random_seed`	int or null	No	`null`	Integer or `null`	Engine-wide. **Only valid under `circuit_breaker_defaults`**; per-symbol is an error

3.22.13 `circuit_breaker_defaults.levels.<LEVEL>` and `symbols.<SYMBOL>.circuit_breaker.levels.<LEVEL>` fields

Symbol-level entries merge over the defaults: only the fields you specify are overridden.

Field	Type	Required	Default	Allowed values / range	Constraint
`price_shift_pct`	float	Yes when creating a level	—	`(0, 1)` exclusive	Required in any level that originates from config; inherited from defaults for symbol overrides
`halt_duration_ns`	int or null	No	`null`	Positive integer nanoseconds, or `null`/omitted	`null` means rest-of-day halt; must be `> 0` when provided

Built-in default CB ladder (used only when a circuit-breaker section exists but supplies no `levels`; if no circuit-breaker section is present at all, the symbol has no breaker):

Level	`price_shift_pct`	`halt_duration_ns`
L1	`0.07`	`300000000000` (5 min)
L2	`0.13`	`900000000000` (15 min)
L3	`0.20`	`null` (rest-of-day)

3.22.14 `symbols.<SYMBOL>` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`tick_decimals`	int	No	`2`	`[0, 8]` inclusive	Must be an integer
`level`	str	No	`risk_controls.default_level`	Any non-empty string	Must reference a key in `risk_controls.levels`
`last_buy_price`	float	No	`null`	Any number	Overridden by persisted `book_stats.json`
`last_sell_price`	float	No	`null`	Any number	Overridden by persisted `book_stats.json`

Field	Type	Required	Default	Allowed values / range	Constraint
`collar`	mapping	No	—	See collar fields	Merged over the level's collar; symbol wins on conflicting keys
`circuit_breaker`	mapping	No	—	See circuit-breaker fields	`levels` subkey merged over defaults; other keys replace
`market_maker_quotes`	list	No	`[]`	List of quote seed mappings	Required (non-empty) if any `MARKET_MAKER` gateway is configured, unless `require_mm_seed_quotes: false` (see Mandatory Fields)
`outstanding_shares`	int	No	`null`	Positive integer	Required if the symbol is listed in an `indices[].constituents` entry (see Mandatory Fields)
`order_limits`	mapping	No	—	See below	Rejected at `risk_controls.levels.<LEVEL>` — set per symbol only

3.22.15 `symbols.<SYMBOL>.order_limits` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`max_order_qty`	int	No	`null` (no limit)	Positive integer	Must be `> 0`
`max_order_value`	float	No	`null` (no limit)	Any number	Must be `> 0`

3.22.16 `symbols.<SYMBOL>.market_maker_quotes[]` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`gateway_id`	str	Yes	—	Any non-empty string	Uppercased; must reference a `MARKET_MAKER` gateway
`quote_id`	str	No	Auto-generated	Any string	Empty string treated as absent
`bid_price`	float	Yes	—	Any number	Must be `< ask_price`
`ask_price`	float	Yes	—	Any number	Must be `> bid_price`
`bid_qty`	int	Yes	—	Positive integer	Must be `> 0`
`ask_qty`	int	Yes	—	Positive integer	Must be `> 0`
`tif`	Enum	No	`DAY`	`DAY`, `GTC`	Case-insensitive; `pm-cverifier` rejects any other value (`S016`)
`seed_once`	bool	No	`true`	`true`, `false`	When `true`, skips injection if this gateway already has a live quote on the symbol restored from the previous run

3.22.17 `market_maker_combos[]` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`combo_id`	str	Yes	—	Any non-empty string	Must not be empty after stripping whitespace
`combo_type`	Enum	No	`AON`	`AON`	Case-insensitive
`tif`	Enum	No	`DAY`	`DAY`, `GTC`, `ATO`, `ATC`	Case-insensitive
`legs`	list	Yes	—	List of leg mappings	Must contain 2 to 10 entries

3.22.18 `market_maker_combos[].legs[]` fields

Field	Type	Required	Default	Allowed values / range	Constraint
`symbol`	str	Yes	—	Configured symbol	Uppercased; must be unique within the combo; must be in `symbols`
`side`	Enum	Yes	—	`BUY`, `SELL`	Case-insensitive
`order_type`	Enum	Yes	—	`MARKET`, `LIMIT`, `STOP`, `STOP_LIMIT`, `FOK`, `ICEBERG`, `IOC`, `TRAILING_STOP`	Case-insensitive
`quantity`	int	Yes	—	Positive integer	—
`price`	float	Conditional	`null`	Display price	Required for `LIMIT`, `STOP_LIMIT`, `FOK`, `ICEBERG` (not enforced for `IOC`); must be a multiple of the leg symbol's tick size

Field	Type	Required	Default	Allowed values / range	Constraint
`stop_price`	float	Optional	`null`	Display price	Not currently validated as required for any order type, including `STOP`/`STOP_LIMIT`/`TRAIL` must be a multiple of the leg symbol's tick size
`smp_action`	Enum	No	Seeding gateway's `participants[].smp_action`, else `NONE`	`NONE`, `CANCEL_AGGRESSOR`, `CANCEL_RESTING`, `CANCEL_BOTH`	Case-insensitive

Combo leg prices use the leg symbol's scale

The legs of one combo trade different instruments, which need not share a tick size, so each leg's price is checked against its own symbol's `tick_decimals` rather than the combo's.

3.22.19 `schedule` fields

Top level:

Field	Type	Required	Applies to	Constraint
`weekdays`	day block	No	Mon-Fri, for any day not overridden below	mutually exclusive with nothing; see individual-day override rule below
`mon`, `tue`, `wed`, `thu`, `fri`	day block	No	one weekday	overrides `weekdays` for that day only
`weekend`	day block	No	Sat+Sun, for either day not overridden below	mutually exclusive with `sat`/`sun`

Field	Type	Required	Applies to	Constraint
`sat`, `sun`	day block	No	one weekend day	overrides `weekend` for that day only; mutually exclusive with `weekend`
`holidays`	day block	No	bank holidays for `locale.country`	applies instead of the calendar day's own entry

Each **day block** (the value of `weekdays`, `weekend`, `holidays`, or any individual `mon`..`sun` key) is:

Field	Type	Required	Default	Allowed values / range
`pre_open`	str	Yes, if the block is present	—	`"HH:MM"` (local server time)
`opening_auction_start`	str	Yes, if the block is present	—	`"HH:MM"` (local server time)
`continuous_start`	str	Yes, if the block is present	—	`"HH:MM"` (local server time)
`closing_auction_start`	str	Yes, if the block is present	—	`"HH:MM"` (local server time)
`closing_auction_end`	str	Yes, if the block is present	—	`"HH:MM"` (local server time)

A day block with any of the five keys missing is rejected — there is no per-key default once a block is present at all. A day (or `holidays`) with no block resolved for it is CLOSED. `weekend` and an individual `sat`/`sun` key present together is also rejected.

3.22.20 Cross-field validation rules

These constraints span multiple sections and are checked after all fields are parsed:

1. If any gateway has `role: MARKET_MAKER`, every symbol in `symbols` must have at least one `market_maker_quotes` entry, unless top-level `require_mm_seed_quotes` is set to `false`.
2. Every `market_maker_quotes[].gateway_id` must reference a configured gateway with `role: MARKET_MAKER`.
3. Every `symbols.<SYMBOL>.level` must reference a key in `risk_controls.levels`.
4. `risk_controls.default_level` must reference a key in `risk_controls.levels`.
5. Every `mm_obligation_defaults.symbols.<SYMBOL>` key must reference a symbol in `symbols`.
6. Every `market_maker_combos[].legs[].symbol` must reference a symbol in `symbols`.
7. Symbols within one combo must be unique.
8. `risk_controls.levels.<LEVEL>.circuit_breaker` is explicitly rejected with an error; use top-level `circuit_breaker_defaults` instead.
9. `risk_controls.levels.<LEVEL>.order_limits` is explicitly rejected with an error; set `order_limits` on each symbol instead.

CHAPTER 4

Engine Configuration Specification

Status: Normative. This appendix defines the formal structure of `engine_config.yaml`, the single reference-data file for an EduMatcher exchange. It is the authoritative schema; where it and any tutorial disagree, this document governs. For worked examples, recipes, and rationale, see The Configuration Workflow (informative).

The schema described here is derived from and MUST match the runtime loaders: `engine/config_loader.py`, `alf_gwy/config.py`, `balf_gwy/config.py`, `ralf_gateway/config.py`, `md_gateway/config.py`, `api_gateway/config.py`, `dc_gateway/config.py`, `log_srv/config.py`, and `scheduler/main.py` (`schedule` and `locale.country` only). `pm-cverifier` is the reference validator.

4.1 1. Conventions

4.1.1 1.1 Requirement keywords

The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHOULD**, **MAY**, and **OPTIONAL** are to be interpreted as described in RFC 2119.

4.1.2 1.2 The file

The configuration is a single YAML 1.1 document whose root MUST be a **mapping**. A loader given a non-mapping root MUST reject the file. `load_engine_config()` itself MUST raise `FileNotFoundError` for a missing path — it does not implement a fallback. The *caller*, `pm-engine` (`engine/main.py`), checks for the file's existence before calling the loader: if the deployed config (compiled to `<EDUMATCHER_DATA_DIR>/ref_c` does not exist, `pm-engine` skips loading entirely and starts in *unrestricted mode* (no symbol/gateway allowlist, sessions disabled) rather than failing. That fallback is implemented by `pm-engine`, not by the loader, and is out of scope here — this document specifies the content of a *present* file.

4.1.3 1.3 Scalar type notation

Field tables and the schema tree (§3) use these type names:

Type	YAML form	Definition
`Bool`	boolean	`true` / `false`
`Int`	integer	whole number

Type	YAML form	Definition
`Float`	number	integer or decimal; parsed as floating point
`Str`	string	UTF-8 text
`Path`	string	filesystem path (`~` is expanded where noted)
`Symbol`	string	instrument id; **normalised to upper-case** on load
`GatewayId`	string	participant id; non-empty; **normalised to upper-case**
`IndexId`	string	index id; **alphanumeric**, upper-case, unique
`Price`	number	price in **display units** (e.g. `150.10`); MUST be a whole multiple of the owning symbol's tick size `10^-tick_decimals` (see CV18)
`Qty`	integer	quantity; `> 0` unless stated
`Ticks`	integer	count of minimum price increments; `> 0`
`Pct01`	number	fraction in the **open** interval `(0, 1)`
`Nanos`	integer	duration in nanoseconds; `> 0` (or `null` where noted)
`Secs`	number	duration in seconds; `> 0`
`Port`	integer	TCP port; `> 0`. The ALF, BALF, CALF, RALF and drop-copy gateway ports MUST also be `<= 65535` (§6)
`HHMM`	string	wall-clock local time `"HH:MM"`
`Country`	string	country name (e.g. `"Sweden"`) or ISO 3166-1 alpha-2 code (e.g. `"SE"`); MUST be a country recognised by the `python-holidays` package
`Enum<E>`	string	one member of enum `E` (§2); **case-insensitive**, stored upper-case
`List<T>`	sequence	ordered list of `T`
`Map<K,V>`	mapping	keyed collection; keys of type `K`, values of type `V`

4.1.4 1.4 Field-table columns

Each section table uses: **Field**, **Type**, **Req** (✔ required / – optional), **Default** (value applied when the key is omitted from a *present* file), and **Constraints**. A `Default` of – means the field has no default and, if optional, its absence leaves the feature disabled.

4.1.5 1.5 Unknown keys

Loaders are **permissive**: a key not defined in this specification is **ignored** and MUST NOT be relied upon for behaviour. Conforming producers SHOULD NOT emit unknown keys. `pm-cverifier` reports an unrecognised key anywhere in the file — at the top level or at any depth of an engine section (§5) or gateway and service block (§6) — as an **error** (`S121`), because the loader's silent fallback to a default would otherwise hide the typo. The one exception is the key set of `schedule` itself, whose unknown keys are reported as `M027`. `pm-config-show --all` also lists top-level sections that are not part of this specification.

4.1.6 1.6 Case normalisation

`Symbol`, `GatewayId`, `IndexId`, and every `Enum<E>` value are upper-cased during load. Producers MAY write any case; consumers compare upper-case.

4.2 2. Enumerations

Enum	Members	Used by
`Role`	`TRADER`, `MARKET_MAKER`, `ADMIN`	`participants[].role`
`DisconnectBehaviour`	`CANCEL_QUOTES_ONLY`, `CANCEL_ALL`, `LEAVE_ALL`	`participants[].disconnect_behaviour`, `participant_defaults.disconnect_behaviour`
`QuoteRefreshPolicy`	`INACTIVATE_ON_ANY_FILL`, `INACTIVATE_ON_FULL_FILL`, `NEVER_INACTIVATE`	`participants[].quote_refresh_policy`
`TIF`	`DAY`, `GTC`, `ATO`, `ATC`	quote/combo seeds (`market_maker_quotes[].tif` is further restricted by `pm-cverifier` to `DAY`/`GTC` only). `IOC` and `FOK` are `OrderType` members, not `TIF` members.
`ComboType`	`AON`	`market_maker_combos[].combo_type`
`Side`	`BUY`, `SELL`	combo legs
`OrderType`	`MARKET`, `LIMIT`, `STOP`, `STOP_LIMIT`, `FOK`, `ICEBERG`, `IOC`, `TRAILING_STOP`	combo legs
`SmpAction`	`NONE`, `CANCEL_AGGRESSOR`, `CANCEL_RESTING`, `CANCEL_BOTH`	combo legs, `participants[].smp_action`, `participant_defaults.smp_action`
`DuplicateSessionPolicy`	`REJECT_NEW`, `EVICT_OLD`	`balf_gateway.duplicate_session_policy`

An `Enum<E>` value outside its member set MUST be rejected.

4.3 3. Formal schema tree

The complete structure at a glance. Annotations: `!` = REQUIRED key, `?` = OPTIONAL key, `=` = default, `∈` = domain/constraint. Types are from §1.3–§2. This tree is normative for *shape*; §4–§6 are normative for *field law*.

```
# ── ENGINE (read by pm-engine) ──────────────────────────────────────
symbols:                    ! Map<Symbol, SymbolSpec>          # ≥0 entries; key required
participants:               ! List<ParticipantSpec>            # ≥1 entry
participant_defaults:       ? ParticipantDefaultsSpec          # values a participants entry inherits when it
sessions_enabled:           ? Bool = true
enforce_collars:            ? Bool = true
enforce_circuit_breakers:   ? Bool = true
require_mm_seed_quotes:     ? Bool = true
auction_indicative_interval_sec: ? Float = 1.0  ∈ > 0
engine_tuning:               ? EngineTuningSpec
mm_obligation_defaults:     ? MMObligationDefaultsSpec
risk_controls:              ? RiskControlsSpec
circuit_breaker_defaults:   ? CircuitBreakerSpec
market_maker_combos:        ? List<ComboSeedSpec>              # each: 2..10 legs
indices:                    ? List<IndexSpec>                  # ≤ 5
schedule:                   ? WeeklyScheduleSpec
locale:                      ? LocaleSpec                       # where this exchange is: country + session t

# ── AUXILIARY GATEWAY BLOCKS (each read by its own process) ─────────
alf_gateway:                ? AlfGwyProcSpec        # pm-alf-gwy
balf_gateway:               ? BalfGwyProcSpec       # pm-balf-gwy
market_data_gateway:        ? MdGwyProcSpec         # pm-md-gwy   (CALF)
post_trade_gateway:         ? RalfGwyProcSpec       # pm-ralf-gwy (RALF)
dc_gateway:                 ? DcGwyProcSpec         # pm-dc-gwy   (drop-copy TCP relay)
api_gateways:               ? Map<Str, ApiGwyProcSpec>   # pm-api-gwy (named instances)
log_server:                 ? LogSrvProcSpec        # pm-log-srv  (centralized LALF log collector)
monitor:                    ? MonitorSpec           # pm-monitor + every beaconing process (health)

SymbolSpec:
  level:                    ? Str          ∈ key of risk_controls.levels
  tick_decimals:            ? Int = 2      ∈ 0..8
  outstanding_shares:       ? Int          ∈ > 0        # REQUIRED for index constituents
  last_buy_price:           ? Price
  last_sell_price:          ? Price
  market_maker_quotes:      ? List<MMQuoteSeedSpec>
  collar:                   ? CollarSpec
  order_limits:             ? OrderLimitsSpec            # per-symbol override
  circuit_breaker:          ? CircuitBreakerSpec         # per-symbol override

LocaleSpec:                                 # top-level locale
  country:                  ? Country = "Sweden"         # read by pm-scheduler for holiday gating; pm-engine
  session_timezone:         ? Str = "Europe/Stockholm"   ∈ IANA timezone name

ParticipantSpec:                            # one entry of participants
  id:                       ! GatewayId
  description:              ? Str = ""
  role:                     ? Enum<Role> = TRADER
  disconnect_behaviour:     ? Enum<DisconnectBehaviour> = <participant_defaults.disconnect_behaviour | CANCEL_
  quote_refresh_policy:     ? Enum<QuoteRefreshPolicy> = INACTIVATE_ON_ANY_FILL
  enforce_mm_obligation:    ? Bool  = <mm_obligation_defaults.enforce_mm_obligation | false>
  mm_max_spread_ticks:      ? Ticks = <mm_obligation_defaults.mm_max_spread_ticks | 20>
  mm_min_qty:               ? Qty   = <mm_obligation_defaults.mm_min_qty | 100>
  mm_obligations:           ? Map<Symbol, MMObligationSpec>
  smp_action:               ? Enum<SmpAction> = <participant_defaults.smp_action | NONE>

ParticipantDefaultsSpec:                    # top-level participant_defaults
  disconnect_behaviour:     ? Enum<DisconnectBehaviour>
  smp_action:               ? Enum<SmpAction>
```



```
# ── PROCESS SPECS (field law in §6) ─────────────────────────────────
# bind_address / host: the EDUMATCHER_GATEWAY_BIND_HOST environment variable.
```

4.4 4. Domain sub-structures

4.4.1 4.1 `MMQuoteSeedSpec` — `symbols.<S>.market_maker_quotes[]`

Field	Type	Req	Default	Constraints
`gateway_id`	`GatewayId`		—	MUST reference a configured gateway whose `role` is `MARKET_MAKER`
`bid_price`	`Price`		—	`bid_price < ask_price`
`ask_price`	`Price`		—	`> bid_price`
`bid_qty`	`Qty`		—	`> 0`
`ask_qty`	`Qty`		—	`> 0`
`tif`	`Enum<TIF>`	–	`DAY`	
`quote_id`	`Str`	–	`null`	empty string treated as `null`
`seed_once`	`Bool`	–	`true`	when `true`, skip injection if this gateway already has a live quote on the symbol restored from the previous run

4.4.2 4.2 `CollarSpec` — `symbols.<S>.collar` and `risk_controls.levels.<L>.collar`

Field	Type	Req	Default	Constraints
`static_band_pct`	`Pct01`	–	`0.20`	`0 < x < 1`
`dynamic_band_pct`	`Pct01`	–	`0.02`	`0 < x < 1`

A per-symbol `collar` is merged over the symbol's resolved `level` collar (symbol keys win). When neither is present, no collar applies to the symbol.

4.4.3 4.3 `OrderLimitsSpec` — `symbols.<S>.order_limits`

Field	Type	Req	Default	Constraints
`max_order_qty`	`Qty`	–	—	`> 0`
`max_order_value`	`Float`	–	—	`> 0`; display money

Pre-trade caps on a **single order**, configured **per symbol and nowhere else**. Unlike `collar`, `order_limits` has no risk-level tier and no global default: an appropriate cap depends on the individual instrument's price and typical size, so there is nothing sensible for a shared profile to say. `risk_controls.levels.<L>.order_limits` is **NOT supported** and MUST be rejected (§5.5).

Each cap is independently optional and has **no default**: a cap the symbol does not set is not enforced. There is deliberately no built-in fallback number — a limit that exists only in code and not in the configuration file is not auditable as a requirement.

An order breaching `max_order_qty` is rejected with reject code `MAX_ORDER_QTY`; one breaching `max_order_value` (`quantity × price`, in display money) with `MAX_ORDER_VALUE`. `max_order_value` is **not** evaluated for an order that carries no price on the wire (MARKET, IOC) — the same orders the collar's price bands already skip. Both caps are re-checked on `order.amend` against the amended quantity and price, so the control cannot be bypassed by entering small and amending up.

4.4.4 4.4 `MMObligationSpec` — `participants[].mm_obligations.<S>`

Field	Type	Req	Default	Constraints
`enforce_mm_obligation`	`Bool`	–	gateway's `enforce_mm_obligation`	
`max_spread_ticks`	`Ticks`	–	gateway's `mm_max_spread_ticks`	`> 0`
`min_qty`	`Qty`	–	gateway's `mm_min_qty`	`> 0`

4.4.5 4.5 `ComboSeedSpec` — `market_maker_combos[]`

Field	Type	Req	Default	Constraints
`combo_id`	`Str`		—	non-empty
`combo_type`	`Enum<ComboType>`	–	`AON`	
`tif`	`Enum<TIF>`	–	`DAY`	
`legs`	`List<ComboLegSpec>`		—	length `2..10`; symbols unique within the combo

`ComboLegSpec`:

Field	Type	Req	Default	Constraints
`symbol`	`Symbol`		—	MUST exist in `symbols`
`side`	`Enum<Side>`		—	
`order_type`	`Enum<OrderType>`		—	
`quantity`	`Qty`		—	
`price`	`Price`	–	`null`	required by `LIMIT`, `FOK`, `STOP_LIMIT`, `ICEBERG` legs (not enforced for `IOC` despite carrying a limit price); on **this leg's** symbol's tick grid, which need not be the grid of the combo's other legs; positivity is **not** validated anywhere in code

Field	Type	Req	Default	Constraints
`stop_price`	`Price`	–	`null`	stop price; currently unvalidated for any order type, including `STOP`/`STOP_LIMIT`/`TRAILING_STOP`; on this leg's symbol's tick grid
`smp_action`	`Enum<SmpAction>`	–	seeding gateway's `participants[].smp_action`, else `NONE`	

4.4.6 4.6 `IndexSpec` — `indices[]`

Field	Type	Req	Default	Constraints
`id`	`IndexId`		—	alphanumeric; unique across `indices`
`description`	`Str`		—	non-empty
`base_value`	`Float`	–	`1000.0`	`> 0`
`publish_interval_secs`	`Secs`	–	`1.0`	`> 0`
`history_file`	`Path`	–	`data/indexes/<id>_history.jsonl`	non-empty
`state_file`	`Path`	–	`data/indexes/<id>_state.json`	non-empty
`constituents`	`List<Symbol>`		—	non-empty; each MUST exist in `symbols` **and** define `outstanding_shares`; no duplicates

4.4.7 4.7 `WeeklyScheduleSpec` — `schedule`, and `DayScheduleSpec` — one day block

`WeeklyScheduleSpec` is a mapping of shortcut and day keys, each an optional `DayScheduleSpec`:

Field	Type	Req	Applies to
`weekdays`	`DayScheduleSpec`	–	`mon`..`fri`, for any not given its own key below
`mon`, `tue`, `wed`, `thu`, `fri`	`DayScheduleSpec`	–	one weekday; overrides `weekdays` for that day

Field	Type	Req	Applies to
`weekend`	`DayScheduleSpec`	–	`sat`+`sun`, for either not given its own key below; mutually exclusive with `sat`/`sun` (CV20)
`sat`, `sun`	`DayScheduleSpec`	–	one weekend day; overrides `weekend` for that day; mutually exclusive with `weekend` (CV20)
`holidays`	`DayScheduleSpec`	–	days the configured `locale.country` observes as a bank holiday

A key absent from `schedule` (and, for `mon..sun`, not covered by `weekdays`/`weekend` either) resolves to CLOSED for that day/holiday-set — no transitions are sent and the engine never leaves `CLOSED` on it. No other keys are permitted under `schedule` (CV21).

`DayScheduleSpec` — the value of any key above:

Field	Type	Req	Default
`pre_open`	`HHMM`	, if the block is present	—
`opening_auction_start`	`HHMM`	, if the block is present	—
`continuous_start`	`HHMM`	, if the block is present	—
`closing_auction_start`	`HHMM`	, if the block is present	—
`closing_auction_end`	`HHMM`	, if the block is present	—

Unlike every other REQUIRED marking in this document, “required” here is conditional on the block appearing at all: a `DayScheduleSpec` that is present but omits any of the five fields MUST be rejected (CV21) — there is no per-field default once a block exists.

4.5 5. Engine sections

4.5.1 5.1 `symbols` (REQUIRED)

`Map<Symbol, SymbolSpec>`. The mapping key is the symbol id. A value of `null`/`{}` is a valid empty spec. Symbol fields: see the schema tree (§3, `SymbolSpec`); `market_maker_quotes` §4.1, `collar` §4.2, `order_limits` §4.3, `circuit_breaker` §5.6.

4.5.2 5.2 `participants` (REQUIRED)

`List<ParticipantSpec>` with **at least one** entry (§3, `ParticipantSpec`). Participant ids MUST be unique after upper-casing. This list is the participant allowlist and is **also** consumed by `pm-alf-gwy` and `pm-balf-gwy` for identity and role.

`smp_action` sets the self-match-prevention action the engine applies to this gateway's orders **when the order itself doesn't specify one**:

- Live `QUOTE` legs (the bid/ask orders generated from a `quote.new` request — see Market-Maker Bot and the ALF `QUOTE` command) have no per-request SMP concept of their own, so they always use this default.
- `NEW`/combo order entry carries its own optional per-order `SMP=` field (§4.5, `ComboLegSpec.smp_action`; ALF protocol `NEW|SMP=`). An explicit `SMP=` from the client — including `SMP=NONE` — always takes precedence over this gateway default; only an *omitted* `SMP=` falls back to it.

4.5.3 5.2a `participant_defaults` (OPTIONAL) — `ParticipantDefaultsSpec`

A mapping of values that every `participants` entry **inherits when it omits the key**. It exists so a uniform policy (for example, `CANCEL_AGGRESSOR` on every gateway) is written once instead of on each entry.

Field	Type	Req	Default	Constraints
`smp_action`	`Enum<SmpAction>`	–	`NONE`	inherited by an entry that omits `smp_action`
`disconnect_behaviour`	`Enum<DisconnectBehaviour>`	–	`CANCEL_QUOTES_ONLY`	inherited by an entry that omits `disconnect_behaviour`

Resolution order for a gateway's effective value is: the entry's own key, then `participant_defaults`, then the built-in default. An explicit entry value always wins, including `smp_action: NONE` when the default is something else. The block applies to **every** role: a `participant_defaults.disconnect_behaviour` of `CANCEL_ALL` also reaches `ADMIN` and `MARKET_MAKER` entries that omit the key, so entries that need a different behaviour MUST set it explicitly.

Unlike most sections, an unrecognised key under `participant_defaults` is **rejected** (CV22) rather than ignored, so that a mistyped name is not silently dropped.

4.5.4 5.3 Engine behaviour flags

Field	Type	Req	Default	Constraints
`sessions_enabled`	`Bool`	–	`true`	when `true`, engine starts `CLOSED` and honours `schedule`/session transitions; when `false`, starts and stays `CONTINUOUS`
`enforce_collars`	`Bool`	–	`true`	global collar enforcement toggle
`enforce_circuit_breakers`	`Bool`	–	`true`	global circuit-breaker enforcement toggle
`require_mm_seed_quotes`	`Bool`	–	`true`	when `true`, CV3 applies; when `false`, a `MARKET_MAKER` gateway may exist with no `market_maker_quotes` entries, for a genuinely empty book at startup

NOTE — `sessions_enabled` default: the engine loader applies `true` when the key is omitted from a *present* file. (A completely absent config file runs unrestricted with sessions disabled — see Auctions & Scheduling.)

4.5.5 5.3a `auction_indicative_interval_sec` and `engine_tuning` (OPTIONAL)

Field	Type	Req	Default	Constraints
`auction_indicative_interval_sec`	`Float`	–	`1.0`	`> 0`; throttle for indicative-uncross republishing during auction call phases

`engine_tuning` (`EngineTuningSpec`) groups low-level runtime tuning knobs not expected to need adjustment in normal use:

Field	Type	Req	Default	Constraints
`snapshot_interval_sec`	`Float`	–	`0.5`	`> 0`; per-symbol book snapshot throttle
`quote_history_maxlen`	`Int`	–	`30`	`> 0`
`drop_copy_buffer_size`	`Int`	–	`10000`	`> 0`
`recent_trades_maxlen`	`Int`	–	`20`	`> 0`
`depth_snapshot_tolerance_ticks`	`Int`	–	`100`	`> 0`

4.5.6 5.4 `mm_obligation_defaults` (OPTIONAL) — `MMObligationDefaultsSpec`

Field	Type	Req	Default	Constraints
`enforce_mm_obligation`	`Bool`	–	`false`	
`mm_max_spread_ticks`	`Ticks`	–	`20`	`> 0`
`mm_min_qty`	`Qty`	–	`100`	`> 0`
`symbols`	`Map<Symbol, {enforce_mm_obligation, mm_max_spread_ticks, mm_min_qty}>`	–	`{}`	each key MUST exist in `symbols`; per-symbol fields default to the block-level values above

These values supply the defaults inherited by `participants[]` obligation fields.

4.5.7 5.5 `risk_controls` (OPTIONAL) — `RiskControlsSpec`

Field	Type	Req	Default	Constraints
`default_level`	`Str`	–	`null`	non-empty; MUST be a key of `levels`
`levels`	`Map<Str, {collar: CollarSpec}>`	–	`{}`	level names upper-cased

`risk_controls.levels.<L>.circuit_breaker` is **NOT supported** and MUST be rejected; define circuit breakers under `circuit_breaker_defaults` (§5.6) instead. `risk_controls.levels.<L>.order_limits` is likewise **NOT supported** and MUST be rejected; set `order_limits` on each symbol that needs a cap (§4.3).

4.5.8 5.6 `circuit_breaker_defaults` and `symbols.<S>.circuit_breaker` — `CircuitBreakerSpec`

Field	Type	Req	Default	Constraints
`reference_window_ns`	`Nanos`	–	`300000000000` (5 min)	rolling reference window
`levels`	`Map<Str, CBLevelSpec>`	–	built-in `L1/L2/L3`	non-empty after merge
`reopening`	`ReopeningSpec`	–	built-in ACE defaults	merges field-by-field

`CBLevelSpec`:

Field	Type	Req	Default	Constraints
`price_shift_pct`	`Pct01`		—	`0 < x < 1`
`halt_duration_ns`	`Nanos | null`	–	—	`> 0` when set; `null` = halt for the rest of the trading day

`ReopeningSpec` (Automated Corridor Expansion):

Field	Type	Req	Default	Constraints
`enabled`	`Bool`	–	`true`	–
`initial_band_pct`	`Pct01`	–	`0.10`	`0 < x < 1`
`expansions`	`List<ExpansionSpec>`	–	`[{0.10, 120e9}, {0.20, 300e9}]`	non-empty; final entry repeats indefinitely; only valid under `circuit_breaker_defaults`
`random_end_max_ns`	`Nanos`	–	`30000000000`	`>= 0`; `0` disables the random end
`random_seed`	`Int | null`	–	`null`	engine-wide; only valid under `circuit_breaker_defaults`

`ExpansionSpec`:

Field	Type	Req	Default	Constraints
`widen_pct`	`Pct01`		—	`0 < x < 1`; additive on the reference price
`min_duration_ns`	`Nanos`		—	`> 0`

Merge order: `circuit_breaker_defaults` supplies defaults; a symbol's `circuit_breaker.levels.<L>` overrides by level key. If no levels result from the merge, the built-in ladder applies: `L1 = 0.07 / 5 min`, `L2 = 0.13 / 15 min`, `L3 = 0.20 / rest-of-day`, all `AUCTION`.

4.5.9 5.7 `market_maker_combos` (OPTIONAL)

`List<ComboSeedSpec>` — see §4.5. Leg prices are display money on the leg symbol's own tick grid: the legs of one combo trade different instruments, which need not share a tick size.

4.5.10 5.8 `indices` (OPTIONAL)

`List<IndexSpec>`, **at most 5** entries — see §4.6.

4.5.11 5.9 `schedule` (OPTIONAL)

`WeeklyScheduleSpec` — see §4.7. Resolved once at load time into a plain per-day table (`mon`..`sun`, plus `holidays`) that every consumer reads without re-implementing the `weekdays`/`weekend` shortcut rules: `pm-engine` (when `sessions_enabled`, to publish it on `system.reference` and `system.session_schedule`) and, independently, `pm-scheduler` (to decide which transitions to send today).

4.5.12 5.10 `locale` (OPTIONAL)

`LocaleSpec` — see §3. Groups the two facts about where this exchange is: its country (for bank-holiday gating) and its session timezone (for trading-day bucketing). Every runtime consumer of either field reads the *compiled* artifact's `engine.locale`, never the authored YAML directly.

Field	Type	Req	Default	Constraints
`country`	`Country`	–	`"Sweden"`	MUST be a country name or ISO 3166-1 alpha-2 code recognised by `python-holidays`; an unrecognised value falls back to the default rather than being rejected
`session_timezone`	`Str`	–	`"Europe/Stockholm"`	MUST be `"UTC"` or an IANA timezone name recognised by `zoneinfo`; an unrecognised value is rejected

`locale.country` and `locale.session_timezone` are validated independently and are not otherwise related: changing one does not affect the other's default or validation. Unlike every other field in this document, `locale.country` is also the sole exception to §1.4's general "MUST be rejected" rule for an invalid value — an unrecognised country falls back to the default rather than aborting loading (contrast CV16 below). `locale.session_timezone` has no such exception: an unrecognised value is rejected like any other field.

`locale.country`'s two consumers handle an invalid value differently: `engine/config_loader.py` substitutes `"Sweden"` silently, with no `python-holidays` recognition check and no log message; `pm-scheduler` (`scheduler/main.py`) separately validates the value against `python-holidays`, logs a warning, and substitutes `"Sweden"` if it is unrecognised (CV16).

`pm-engine` reads `locale.country` for two purposes: it derives the ISO alpha-2 country code reported in outbound reference messages, and it uses the same `python-holidays` lookup `pm-scheduler` does to resolve the `today`/ `today_is_holiday` convenience fields on `system.reference` and `system.session_schedule` (§4.7, §5.9) — which of `holidays` or the calendar day's own entry applies right now — and to list the coming year's bank holidays in `upcoming_holidays`. `pm-scheduler` uses `locale.country` for the same bank-holiday lookup to decide which block of the resolved `schedule` (§5.9) — if any — to run today — see Session Scheduling → Bank holidays and weekends.

`locale.session_timezone` is read by `pm-stats` and `pm-clearing` to decide which calendar date, in that timezone, a UTC-timestamped event falls into — the trading-day boundary used for daily rollups and clearing's `trade_date`. Both processes persist the timezone name they first run with (`stats_meta`, `clearing_meta`) and refuse to start if the configured value later contradicts what is already recorded, rather than silently re-bucketing an existing database under a new boundary.

4.6 6. Auxiliary gateway blocks

Each block below is read **only** by its own process; `pm-engine` ignores them. The one exception is `log_server.client` (§6.7), which every `pm-*` process reads. See Configuration → Which Process Reads What. The named spec type of each block (`AlfGwyProcSpec`, …) is defined in the schema tree (§3); the tables below give the field law.

`bind_address` (`host` for `api_gateways`) defaults to `"0.0.0.0"`. When the `EDUMATCHER_GATEWAY_BIND_HOST` environment variable is set it overrides both the YAML value and the default, for every block at once. The ZeroMQ addresses of the engine and index buses are not YAML-configurable, with the exception

of the four `*_addr` fields of `ApiGwyProcSpec` (§6.5).

4.6.1 6.1 `alf_gateway` — `pm-alf-gwy`

Spec type: `AlfGwyProcSpec`.

Field	Type	Req	Default	Constraints
`enabled`	`Bool`	–	`true`	
`name`	`Str`	–	`"alf-gwy01"`	
`bind_address`	`Str`	–	`"0.0.0.0"`	
`port`	`Port`	–	`5565`	`1..65535`
`heartbeat_interval_sec`	`Int`	–	`5`	`> 0`
`handshake_timeout_sec`	`Int`	–	`10`	`> 0`
`idle_timeout_sec`	`Int`	–	`30`	`> 0`
`max_connections`	`Int`	–	`1024`	`> 0`
`max_client_queue`	`Int`	–	`10000`	`> 0`
`max_commands_per_second`	`Int`	–	`100`	`> 0`
`max_errors_before_disconnect`	`Int`	–	`50`	`> 0`
`error_window_sec`	`Int`	–	`60`	`> 0`

Also consumes `participants` for identity/role.

4.6.2 6.2 `balf_gateway` — `pm-balf-gwy`

Spec type: `BalfGwyProcSpec`.

Field	Type	Req	Default	Constraints
`enabled`	`Bool`	–	`true`	
`name`	`Str`	–	`"balf-gwy01"`	
`bind_address`	`Str`	–	`"0.0.0.0"`	
`port`	`Port`	–	`5560`	`1..65535`
`heartbeat_interval_sec`	`Secs`	–	`1.0`	`> 0`
`heartbeat_timeout_sec`	`Secs`	–	`5.0`	`> 0`
`idle_timeout_sec`	`Secs`	–	`30.0`	`> 0`
`auth_timeout_sec`	`Secs`	–	`10.0`	`> 0`
`max_connections`	`Int`	–	`64`	`> 0`
`max_client_queue`	`Int`	–	`10000`	`> 0`
`max_messages_per_second`	`Int`	–	`100`	`> 0`
`max_errors_before_disconnect`	`Int`	–	`10`	`> 0`
`error_window_sec`	`Secs`	–	`60.0`	`> 0`
`duplicate_session_policy`	`DuplicateSessionPolicy`	–	`REJECT_NEW`	

Also consumes `participants` for identity, role, and `disconnect_behaviour`.

4.6.3 6.3 `market_data_gateway` — `pm-md-gwy` (CALF)

Spec type: `MdGwyProcSpec`.

Field	Type	Req	Default	Constraints
`enabled`	`Bool`	–	`true`	
`name`	`Str`	–	`"md-gwy01"`	reported as `WELCOME|GW=`
`bind_address`	`Str`	–	`"0.0.0.0"`	
`port`	`Port`	–	`5570`	`1..65535`
`heartbeat_interval_sec`	`Int`	–	`1`	`> 0`; advertised as `WELCOME|HBINT=`
`idle_timeout_sec`	`Int`	–	`5`	`> 0`
`replay_window_sec`	`Int`	–	`30`	`> 0`; advertised as `WELCOME|REPLAY=`
`max_connections`	`Int`	–	`64`	`> 0`
`max_messages_per_second`	`Int`	–	`200`	`> 0`
`max_symbols_per_client`	`Int`	–	`200`	`> 0`
`max_client_queue`	`Int`	–	`10000`	`> 0`
`depth_levels`	`Int`	–	`10`	`> 0`

4.6.4 6.4 `post_trade_gateway` — `pm-ralf-gwy` (RALF)

Spec type: `RalfGwyProcSpec`.

Field	Type	Req	Default	Constraints
`name`	`Str`	–	`"ralf-gwy01"`	
`bind_address`	`Str`	–	`"0.0.0.0"`	
`port`	`Port`	–	`5580`	`1..65535`
`replay_retention_sec`	`Int`	–	`86400`	`> 0`
`heartbeat_interval_sec`	`Int`	–	`2`	`> 0`
`idle_timeout_sec`	`Int`	–	`20`	`> 0`
`max_connections`	`Int`	–	`64`	`> 0`
`max_client_queue`	`Int`	–	`10000`	`> 0`
`allowed_roles`	`List<Str>`	–	`[CLEARING, DROP_COPY, AUDIT]`	values upper-cased

This block has no `enabled` key.

4.6.5 6.5 `api_gateways` — `pm-api-gwy` (REST / WebSocket)

`api_gateways` is a `Map<Str, ApiGwyProcSpec>` of **named instances** (the map key is the instance name, non-empty). The singular key `api_gateway` is **NOT supported** and MUST be rejected.

`ApiCredentialSpec`, `RateLimitSpec` and `TimeoutSpec` are defined in the schema tree (§3).

`ApiGwyProcSpec`:

Field	Type	Req	Default	Constraints
`enabled`	`Bool`	–	`true`	
`host`	`Str`	–	`"0.0.0.0"`	use `127.0.0.1` for loopback-only deployments
`port`	`Port`	–	`8080`	`> 0`
`log_level`	`Str`	–	`"info"`	
`swagger_enabled`	`Bool`	–	`true`	
`stats_db`	`Path`	–	resolved `stats.db`	~ expanded
`audit_db`	`Path`	–	resolved `audit_index.db`	~ expanded. Read-only; only `GET /admin/orders/{order_id}/audit` uses it, and that endpoint returns 503 when the file is absent
`clearing_db`	`Path`	–	resolved `clearing.db`	~ expanded. Read-only; supplies the session/MTD/YTD realized P&L on `GET /positions`, which report `null` when the file is absent
`order_retention_sec`	`Int`	–	`3600`	`>= 0`. Seconds a terminal order stays in the in-memory cache; `0` disables eviction
`market_data_cache_ttl_sec`	`Int`	–	`60`	`>= 0`. TTL for cached market-data reads served by this instance
`engine_pull_addr`	`Str`	–	`tcp://<engine host>:5555`	ZeroMQ endpoint the gateway sends orders to; `<engine host>` is `EDUMATCHER_ENGINE_HOST` (default `127.0.0.1`). Not validated; normally left unset
`engine_pub_addr`	`Str`	–	`tcp://<engine host>:5556`	ZeroMQ endpoint of the engine event feed. Not validated; normally left unset

Field	Type	Req	Default	Constraints
index_pull_addr	Str	–	tcp://<index bind host>:5559	ZeroMQ endpoint of pm-index control; <index bind host> is EDUMATCHER_INDEX_BIND_HOST (default 127.0.0.1). Not validated; normally left unset
index_pub_addr	Str	–	tcp://<index bind host>:5558	ZeroMQ endpoint of the pm-index feed. Not validated; normally left unset
credentials	List<ApiCredentialSpec>		[]	api keys unique within instance
rate_limit	RateLimitSpec	–	see below	
timeouts	TimeoutSpec	–	see below	

ApiCredentialSpec:

Field	Type	Req	Default	Constraints
api_key	Str		—	non-empty; unique within the instance
gateway_id	GatewayId	–	null	a gateway_id MUST NOT map to two different instances
description	Str	–	""	

RateLimitSpec: writes_per_second Int > 0 = 10, burst Int > 0 = 20. TimeoutSpec: engine_auth_sec Secs > 0 = 3.0, engine_reply_sec Secs > 0 = 3.0, wait_ack_sec Secs > 0 = 3.0.

4.6.6 6.6 dc_gateway — pm-dc-gwy (drop-copy TCP relay)

Spec type: DcGwyProcSpec.

Field	Type	Req	Default	Constraints
name	Str	–	"dc-gwy01"	non-empty; echoed in WELCOME
bind_address	Str	–	"0.0.0.0"	non-empty
port	Port	–	5590	1..65535
heartbeat_interval_sec	Secs	–	2	> 0; interval between HB lines
idle_timeout_sec	Secs	–	20	> 0; disconnect threshold when no traffic flows in either direction

Field	Type	Req	Default	Constraints
`max_connections`	`Int`	–	`64`	`> 0`; simultaneous client connections
`max_client_queue`	`Int`	–	`10000`	`> 0`; per-client outbound buffer before slow-client disconnect

This block has no `enabled` key. The drop-copy source address (`tcp://127.0.0.1:5557`) is **not** configurable via YAML — use the `--engine-dc-pub` CLI flag on `pm-dc-gwy` to point at a non-default engine address.

4.6.7 6.7 `log_server` — `pm-log-srv` (centralized LALF log collector) — `LogSrvProcSpec`

Field	Type	Req	Default	Constraints
`enabled`	`Bool`	–	`true`	master switch; `pm-log-srv` refuses to accept connections when `false`
`name`	`Str`	–	`"log-srv01"`	echoed in the LALF `WELCOME|SRV=` field
`bind_address`	`Str`	–	`"0.0.0.0"`	TCP listen interface (`127.0.0.1` for loopback-only)
`port`	`Port`	–	`5600`	`> 0`
`db_path`	`Path`	–	resolved `data/log.db`	SQLite database path where `log_events`/`processes`/`server_stats` are stored
`retention_days`	`Int | null`	–	`30`	`>= 0` or `null`; `0` is normalised to `null` (both mean unbounded retention — pruning disabled); pruning otherwise runs once per hour
`max_message_bytes`	`Int`	–	`65536`	`> 0`; maximum `LOG` payload size before truncation — oversized messages are truncated and stored, never dropped
`max_client_queue`	`Int`	–	`10000`	`> 0`; per-connection outbound backlog limit before backpressure is applied
`write_batch_size`	`Int`	–	`50`	`> 0`; maximum rows per SQLite transaction in the background writer thread

Field	Type	Req	Default	Constraints
`write_batch_interval_ms`	`Int`	–	`100`	`> 0`; maximum time between writer-thread flushes, whichever comes first with `write_batch_size`
`heartbeat_interval_sec`	`Int`	–	`5`	`> 0`; how often a connected client must send something (`LOG` or `HB`) to stay alive; the server disconnects after 2× this interval of silence and advertises the value in `WELCOME|HBINT=` (the server itself never sends `HB`). Doubles as the publish interval for the LALF-PS `log.server_state` tick
`pubsub_enabled`	`Bool`	–	`true`	master switch for LALF-PS, the ZeroMQ log-distribution interface; when `false` no ZeroMQ socket is bound and `pm-log-srv` runs as a pure TCP collector
`pub_port`	`Port`	–	`5601`	`> 0`; must differ from `port` and `pull_port`; ZeroMQ `PUB` port carrying live rows, notify ticks, backfill chunks and control acks
`pull_port`	`Port`	–	`5602`	`> 0`; must differ from `port` and `pub_port`; ZeroMQ `PULL` port receiving subscriber control requests
`lease_sec`	`Int`	–	`30`	`> 0`; default subscription lease TTL — a subscriber that stops sending `log.renew` within this window is reaped and its buffers discarded
`max_lease_sec`	`Int`	–	`300`	`>= lease_sec`; upper bound on a subscriber's requested `lease_sec`, which is clamped rather than rejected
`max_subscribers`	`Int`	–	`32`	`> 0`; maximum concurrent leased subscriptions before `log.subscribe` is answered with `TOO_MANY_SUBS`

Field	Type	Req	Default	Constraints
`notify_interval_ms`	`Int`	–	`250`	`> 0`; coalescing window for `NOTIFY`-mode ticks, and the floor on a subscriber's requested `notify_interval_ms`
`backfill_chunk_rows`	`Int`	–	`500`	`> 0`; rows per `log.backfill` chunk, and the maximum rows per live `log.event` message
`max_backfill_minutes`	`Int`	–	`1440`	`> 0`; largest "last n minutes" window a subscriber may request; a larger request is rejected with `INVALID_WINDOW`
`max_backfill_rows`	`Int`	–	`100000`	`> 0`; hard cap on the rows returned by one backfill; the final chunk sets `truncated: true` when it bites
`max_pending_rows`	`Int`	–	`20000`	`> 0`; per-subscription `STREAM` buffer cap — a subscriber that is alive but too slow loses its oldest buffered rows, reported back to it as `dropped`
`pub_sndhwm`	`Int`	–	`10000`	`> 0`; ZeroMQ send high-water mark on the `PUB` socket
`client`	`LogClientSpec`	–	see below	read by every `pm-*` process (not by `pm-log-srv`) to configure its `TcpLogHandler`; must be a mapping

`LogClientSpec` (`log_server.client`):

Field	Type	Req	Default	Constraints
`connect_timeout_sec`	`Secs`	–	`0.5`	`> 0`; TCP connect timeout for each attempt to reach the log server
`failover_timeout_sec`	`Float`	–	`30.0`	`>= 0`; how long a process keeps retrying a dropped log-server connection before it switches, permanently for that process, to a local log file

Field	Type	Req	Default	Constraints
`failover_dir`	`Path`	–	resolved `<data dir>/logs`	directory for the local failover log files; ~ expanded

The `pubsub_*`/`pub_*`/`pull_*`/`lease_*`/`backfill_*` fields configure LALF-PS, described in full in Centralized Log Server.

This block has no interaction with `participants` or any other engine section — `pm-log-srv` is a standalone LALF collector, unrelated to the ZeroMQ bus, and does not consume any engine-section fields. The `client` sub-block is the only part read by other processes. See Configuring pm-log-srv for worked examples and Centralized Log Server for the operational guide; the wire protocol it serves is normatively specified in the LALF Protocol Reference.

4.6.8 6.8 `monitor` — `pm-monitor` and every beaconing process — `MonitorSpec`

Unlike the blocks above, `monitor` is read by every long-running process (`enabled`, `host`, `pull_port` and `beacon_interval_sec` say whether, where and how often to send a health beacon) as well as by `pm-monitor`, which reads all of it. Its `bind_address` defaults to `"127.0.0.1"`, not `"0.0.0.0"`; `EDUMATCHER_GATEWAY_BIND_HOST` overrides it as for the other blocks, applied where `pm-monitor` runs (the stored value is the authored one). An absent block means every default below.

Field	Type	Req	Default	Constraints
`enabled`	`Bool`	–	`true`	when `false` no process sends beacons and `pm-monitor` idles until stopped
`bind_address`	`Str`	–	`"127.0.0.1"`	non-empty; interface `pm-monitor` binds
`host`	`Str`	–	`"127.0.0.1"`	non-empty; where every process sends its beacons and clients connect
`pull_port`	`Port`	–	`5610`	ZeroMQ `PULL`, beacons in; MUST differ from `pub_port` and every other listener
`pub_port`	`Port`	–	`5611`	ZeroMQ `PUB`, `health.snapshot` and `health.event` out
`beacon_interval_sec`	`Float`	–	`2.0`	`> 0`
`hung_after_sec`	`Float`	–	`5.0`	`> beacon_interval_sec`
`silent_after_beacons`	`Int`	–	`3`	`> 0`
`book_metrics_interval_sec`	`Float`	–	`10.0`	`> 0`
`retention.raw_hours`	`Int`	–	`6`	`> 0`
`retention.minute_days`	`Int`	–	`7`	`> 0`

Field	Type	Req	Default	Constraints
`thresholds.<rule>`	`ThresholdBandSpec`	–	see §3	rule the seven of `MonitorThresholdsSpec`; `warn`/`crit` `>= 0`, `warn < crit` except `disk_free_pct` (`warn > crit`)

The semantics — how health is judged, what each threshold watches, what is stored — are in Health Monitoring; the messages are `health.beacon`, `health.snapshot` and `health.event` in the Message Reference.

4.7 7. Cross-field and semantic constraints

A conforming document MUST satisfy all of the following. Violations MUST be rejected at load.

#	Rule
CV1	`symbols` is present and a mapping; `participants` is present and a list with 1 entry.
CV2	`participants[].id` values are unique (after upper-casing).
CV3	If **any** gateway has `role: MARKET_MAKER`, then **every** symbol MUST define at least one `market_maker_quotes` entry, unless `require_mm_seed_quotes: false`.
CV4	Every `market_maker_quotes[].gateway_id` references a configured gateway whose role is `MARKET_MAKER`.
CV5	For each MM quote seed, `bid_price < ask_price` and `bid_qty, ask_qty > 0`.
CV6	A symbol's `level` (explicit, else `risk_controls.default_level`) MUST be a key of `risk_controls.levels`, when any level is in effect.
CV7	`risk_controls.default_level`, if set, MUST be a key of `risk_controls.levels`.
CV8	`risk_controls.levels.<L>.circuit_breaker` MUST NOT appear.
CV9	Each `market_maker_combos[]` has 2–10 legs; leg symbols are unique within the combo; every leg `symbol` exists in `symbols`.
CV10	`indices` has 5 entries; each `id` is alphanumeric and unique; each constituent exists in `symbols` and defines `outstanding_shares`; constituents are non-empty and duplicate-free.
CV11	Every key of `mm_obligation_defaults.symbols` references a symbol that exists in `symbols`.
CV12	`collar.*_band_pct` (0,1); `circuit_breaker.levels.<L>.price_shift_pct` (0,1); `halt_duration_ns` is `> 0` or `null`.

#	Rule
CV13	`circuit_breaker.reopening.initial_band_pct` and every `expansions[].widen_pct` $\in (0,1)$; `expansions` is non-empty; `expansions[].min_duration_ns` is `> 0`; `random_end_max_ns` is `>= 0`; `random_seed` and `expansions` appear only under `circuit_breaker_defaults`.
CV14	(`pm-alf-gwy`, `pm-balf-gwy`) No `participants` id may be a prefix of another id.
CV15	(`pm-api-gwy`) The singular `api_gateway` key is not supported; a `gateway_id` credential MUST NOT be shared across two `api_gateways` instances.
CV16	(`pm-scheduler`) An unrecognised `locale.country` value is the **sole exception** to the "MUST be rejected" rule in this section — the loader substitutes the default (`"Sweden"`) and logs a warning instead of aborting. `pm-scheduler` sends no `schedule` transitions on a calendar day whose resolved entry (§4.7 — a `locale.country` bank holiday resolves to `holidays`, otherwise the day's own `mon`..`sun` entry) is CLOSED; weekends are not treated specially, only whatever their resolved entry says.
CV17	(`pm-log-srv`) `log_server.retention_days`, when present, MUST be `>= 0` or `null`; `port`, `max_message_bytes`, `max_client_queue`, `write_batch_size`, `write_batch_interval_ms`, and `heartbeat_interval_sec` MUST each be `> 0`.
CV18	Every `Price` in the file is a whole multiple of its symbol's tick size — `symbols.<S>.last_buy_price/last_sell_price`, `market_maker_quotes[].bid_price/ask_price`, and `market_maker_combos[].legs[].price/stop_price`. The scale is the owning symbol's `tick_decimals`; for a combo leg that is the **leg's** symbol, not the combo's first. An off-grid price is rejected, not rounded.
CV19	`symbols.<S>.tick_decimals` $\in 0..8$; `outstanding_shares`, when present, `> 0`.
CV20	`schedule.weekend` and an individual `schedule.sat`/`schedule.sun` key MUST NOT both be present.
CV21	Every `DayScheduleSpec` present anywhere under `schedule` (`weekdays`, `weekend`, `holidays`, or an individual `mon`..`sun` key) MUST define all five of `pre_open`, `opening_auction_start`, `continuous_start`, `closing_auction_start`, `closing_auction_end`; a partial block is rejected, not filled from a default.
CV22	`participant_defaults`, when present, is a mapping whose only keys are `smp_action` and `disconnect_behaviour`, each a valid member of its enumeration.

#	Rule
CV23	(`monitor`) Every `thresholds` band is ordered (`warn < crit`, except `disk_free_pct` where `warn > crit`); `hung_after_sec > beacon_interval_sec`; `pull_port ≠ pub_port`, and neither collides with another listener (counted even when the block is absent). A `host` that cannot reach `bind_address` (neither equal, nor a wildcard bind, nor both loopback) is reported as a warning, not rejected, because `EDUMATCHER_GATEWAY_BIND_HOST` may change the bind at run time.
CV24	(`post_trade_gateway`, `market_data_gateway`, `dc_gateway`) `idle_timeout_sec` MUST be `> 2 × heartbeat_interval_sec`, each field taken at its default when omitted.

4.8 8. Processing model

1. **Parse.** Load the document as YAML; a non-mapping root is rejected (§1.2).
2. **Section ownership.** `pm-engine` reads the engine sections (§5) plus the `SymbolSpec`/nested structures — with one exception, `locale.country` (§5.10), which is read only by `pm-scheduler`; each auxiliary process reads only its own block (§6). No single process reads the whole file.
3. **Normalisation.** Symbols, gateway ids, index ids, and enum values are upper-cased (§1.6).
4. **Validation.** Field-level constraints (§4–§6) are checked, then the cross-field rules (§7). The first violation MAY abort loading.
5. **Unknown keys** are ignored (§1.5).
6. **Whole-file validation.** `pm-cverifier` validates every section — including blocks no runtime process consumes on its own — across four layers (YAML, schema, semantic, completeness). It is the reference conformance checker.
7. **Whole-file presentation.** `pm-config-show` reads every section for display only. It is read-only and non-normative: it neither validates nor resolves defaults, and it MUST NOT be treated as a conformance signal — a document it renders without complaint may still be rejected by a loader. Where it reports an *effective* value the specification does define — most notably the port a section binds when `port` is omitted (§6) — that value MUST agree with this document.

4.9 9. Minimal conformant document

The smallest document that loads and supports trading (informative):

```
participants:
  - id: TRADER01
symbols:
  AAPL:
    tick_decimals: 2
    last_buy_price: 149.90
    last_sell_price: 150.10
```

If a `MARKET_MAKER` gateway is added, CV3 then requires a `market_maker_quotes` block on every symbol, unless `require_mm_seed_quotes: false` is also set.

4.10 See also

Operator's Guide:

- The Configuration Workflow — informative reference, generator, recipes
- Config Verifier (`pm-cverifier`) — the reference validator
- Inspect Configs (`pm-config-show`) — read-only viewer for a deployed document, including the effective port map
- Risk Controls — collar and circuit-breaker behaviour
- Market Index (`pm-index`) — index calculation using `indices`
- Centralized Log Server — `pm-log-srv`/`pm-log-cli` operational guide

Protocols and Clients:

- External Protocols Overview — the gateways that read the auxiliary blocks
- LALF Protocol Reference — the wire protocol `pm-log-srv` serves

CHAPTER 5

Glossary

The following definitions are written to be concise and standalone, useful as a quick reference. The first section collects the words that are specific to EduMatcher; the alphabetical sections after it cover general exchange and market vocabulary.

5.1 EduMatcher terms

ADMIN / MARKET_MAKER / TRADER (roles): The three participant roles. A `TRADER` sends orders; a `MARKET_MAKER` may also send two-sided quotes and carries quoting obligations; an `ADMIN` operates the exchange — session control, halts, kill switch — through `pm-admin`. The role is set per participant in the configuration.

ALF, BALF, CALF, RALF, LALF, DC1: EduMatcher's protocols: ALF ("Almost FIX") is text order entry, BALF (Binary ALF) binary order entry, CALF (Channel ALF) the market-data feed, RALF (Reconciliation ALF) post-trade dissemination, LALF (Log ALF) centralized logging and DC1 the drop-copy feed over TCP. See Protocols Overview.

Arrival sequence (`arrival_seq`): The number an order book assigns to an order each time the order is placed on a heap, strictly increasing within one book. It, not the client-supplied `ts_ns`, decides queue position at a price, so a back-dated timestamp cannot jump the queue. A new number is taken whenever an order goes to the back of its queue: when it first rests, when a stop is armed, when an iceberg shows a fresh slice, and when an amend raises the quantity or changes the price. The engine reports it as `priority_seq`. See The Order Book: Deep Dive.

Automated Corridor Expansion (ACE): The rule that governs how a symbol reopens after a halt. The reopening auction may only cross inside a corridor around the reference price; if the equilibrium price falls outside it, the call phase is extended and the corridor widened one rung of a ladder at a time, the last rung repeating until the price fits. Configured under `circuit_breaker_defaults.reopening` and tuned with `pm-config-gen --ace-*`.

Book checkpoint: The `system.book_checkpoint.<SYM>` message in which the engine publishes a symbol's whole book (resting orders in matching priority, armed stops in trigger order, trailing stops, and the last prices) at startup and at other defined moments, so that a replay can start from a known state rather than from the first message of the day.

Central pool: How `pm-eod` settles. Every trader settles with the exchange itself (acting as the one counterparty to everybody) rather than with other traders: a gain is *owed by the exchange* to the trader, a loss is *owed by the trader to the exchange*. The amounts of a day sum to exactly zero. See Closing the Day.

Close snapshot: The copy of every position — quantity, cumulative realized and unrealized P&L and the closing mark — that `pm-clearing` saves into `position_close_snapshot` when the session reaches `CLOSED` (or, if no `CLOSED` was ever announced, at the engine's graceful shutdown). Settlement is the difference between two consecutive close snapshots. View it with `pm-clearing-cli snapshots`.

Data directory (`EDUMATCHER_DATA_DIR`): The one directory in which an exchange instance keeps its deployed configuration, databases, logs and saved state. All processes of one exchange must use the same one. See Environment variables.

Episode (audit replay): In `pm-audit-replay`, a group of causally connected facts with one business meaning: one order, one trade, one quote or one command, together with its outcome. Episodes sit between the raw events and the narrated story, and are kept in the episode index (`audit_replay.db`). See Audit Replay.

Deployed configuration: The compiled configuration at `<data directory>/ref_data/engine_config.json` that every process reads. You edit `engine_config.yaml` and install it with `pm-config-deploy` (or `pm-setup`); editing the YAML alone changes nothing.

Engine (`pm-engine`): EduMatcher's matching engine, the single process that owns every order book and decides every trade.

Equity (settlement sense): In the settlement documentation, a position's *cumulative realized plus unrealized P&L*. Not to be confused with *equity* meaning shares; the general-terms entry below describes that sense.

`EXCHANGE` (ledger account): The account in `eod.db` that absorbs the rounding residual of a day (a cent or two at most), so the amounts of every day sum to exactly zero. A reserved name: no gateway may use it.

Health beacon: The small message every long-running `pm-*` process pushes to `pm-monitor` every two seconds (`monitor.beacon_interval_sec`). `pm-monitor` judges each process from its beacons and publishes `health.snapshot` and `health.event`. A process whose beacons stop is first SILENT, then DEAD. See Health Monitoring.

Lazy deletion: How an order book removes an order from the middle of a heap without searching for it: the entry is only marked invalid (`valid = False`), and whoever next reaches it at the top discards it (`_peek`). Because dead entries pile up deep in the book, `_compact_heaps` rebuilds the heaps once they hold more than twice the live entries plus 64. The consequence to remember is that the length of a heap is not the number of resting orders.

Gateway ID: The identity a participant presents when it connects, for example `TRADER01`. It is the participant's `id` from the configuration; messages and protocols call it `gateway_id`. See also *Gateway*.

Ledger (`eod.db`): The settlement database written by `pm-eod`: one row per participant per settled trading date (plus an `EXCHANGE` row that absorbs the rounding residual), plus a log of runs. Balances are never stored; they are the sum of the amounts plus the roster's opening balance.

Operator Dashboard (`pm-dash`): The browser view of `pm-monitor`'s health snapshot and history: every process's state, trading activity, participants, order books and events, live and over time. See Operator Dashboard.

Participant: Anyone configured to connect to the exchange — a student, a bot or the operator — with an ID and a role. Listed under `participants:` in the configuration. A participant is deliberately one combined gateway, session and user login. Real exchanges usually split this into several levels (member, then trader or user, with connectivity as a separate session layer, and clients or accounts for clearing); EduMatcher keeps a single level to stay simple for teaching.

pm-eod: The end-of-day command: settles a closed trading date, writes the statements and runs the housekeeping (audit archive, log compression, database backups). See Closing the Day.

Process profile: A named set of `pm-*` processes that `pm-opctl-cli` starts and stops together: `micro`, `mini`, `default`, `mm-demo` (the default set plus a market-maker bot) or `ai-swarm` (`mm-demo` with its market maker quoting around the market model's values, plus the model and twenty AI traders).

Recorder: A process that turns the engine's live events into durable records: `pm-stats` (prices and volumes), `pm-clearing` (positions and P&L) and `pm-audit` (the full event log). A recorder only captures what happens while it is running.

Roster: `ref_data/roster.yaml` — the optional list that connects gateway identifiers to a trader's name, email and opening balance for the statements. Never used to send anything.

Seed quote: A market maker's starting quote, defined per symbol under `market_maker_quotes` in the configuration and injected by the engine at startup so that a book is not empty. The `-nomm` example configurations have none.

Session phase: Where the trading day is: `PRE_OPEN`, `OPENING_AUCTION`, `CONTINUOUS`, `CLOSING_AUCTION` or `CLOSED`, plus the halt-related phases. Also called the *session state*.

Statement: The per-trader file `pm-eod` writes for each trading date: the headline (what the exchange owes the trader or the trader owes it), the result by symbol, the open positions at the close and the day's trades. See Your End-of-Day Statement.

TapeDeck (`pm-terminal`): The read-only browser market display. The other browser applications are the Trading GUI (`pm-trading-ui`), the Order Book Viewer (`pm-book`), the Log Operator Console (`pm-log-ui`), the Operator Dashboard (`pm-dash`) and the Configuration GUI (`config-gui`).

Trading date: The exchange-local calendar date that daily statistics and P&L summaries are grouped by, as opposed to the UTC date of an event's timestamp.

5.2 General terms

5.2.1 A

Adverse Selection: The risk a market maker faces when their quote is hit by a counterparty who has superior information about where prices are heading, causing the market maker to trade at a price that will quickly move against them. A market maker who is repeatedly adversely selected will lose money even while quoting at "correct" prices. MMP exists specifically to limit adverse selection damage.

Aggressive order: (also called a **taker**) — an incoming order that immediately trades against resting orders. It "sweeps" the book.

Aggressor / Taker: The participant whose incoming order triggered a match by crossing the spread to meet a resting order. The aggressor "takes" liquidity that was already available. Contrast with *Maker*, the participant whose resting order was already in the book.

Alternative Trading System (ATS): A trading venue that matches orders but is not a registered national securities exchange. Includes ECNs and dark pools. Subject to lighter regulation than exchanges in some jurisdictions.

Arbitrage: The simultaneous purchase and sale of the same (or equivalent) asset on different markets to profit from a price difference. Pure arbitrage is risk-free (buy cheap here, sell expensive there at the same moment); in practice, most arbitrage involves some timing or execution risk. Arbitrage is the mechanism by which prices on different venues are kept consistent, arbitrageurs quickly eliminate

price discrepancies, making markets more efficient. **Latency arbitrage** is a specific form exploiting the time it takes for price changes to propagate between venues.

Ask / Offer: The price at which a seller is willing to sell. The best (lowest) ask is the top of the sell side of the order book. The collection of all active asks forms the sell side of the book.

AON (All-Or-None combo): A combo type that checks, at entry, whether every leg could fill completely right now. If so the legs are matched; if not, none is matched on entry and all rest as interest. It is a guard at entry only, not a guarantee that the legs execute atomically later, so a combo still carries leg risk. See Known Limitations.

ATC (At-The-Close): An order TIF valid only during the closing auction. Expired automatically when the closing auction ends (transition to `CLOSED`).

ATO (At-The-Open): An order TIF valid only during the opening auction. Expired automatically when the opening auction ends (transition to `CONTINUOUS`).

Auction / Call Auction / Uncross / Fixing: A trading mechanism where orders are collected over a period and then matched simultaneously at a single equilibrium price. Contrasts with continuous matching where orders match one at a time as they arrive.

Auction Imbalance: The excess of buy orders over sell orders (or vice versa) at the indicative uncross price during an auction. Published by exchanges to encourage offsetting orders before uncrossing.

Audit Trail: The complete, immutable, time-ordered record of every event in the exchange system, every order, fill, cancellation, and rejection, with precise timestamps. Required for regulatory reporting and replay-based recovery.

5.2.2 B

Basis: In derivatives and futures markets, the price difference between two related instruments, most commonly between the spot/cash price and a futures price on the same underlying, or between two futures contracts with different expiry dates. In the implied order example in this document (the *Order Types* section of Part II), the spread price of −$2 between the January and February WTI contracts is the basis between those two contract months: January is $2 cheaper than February. The basis is not fixed; it changes continuously as supply and demand for the near and far months evolve independently. **Basis risk** is the risk that the expected relationship between two instruments diverges unexpectedly, a spread trader who expected the two legs to move together may find they do not, leaving an unintended net exposure. See *Implied Order*, *Calendar Spread*.

Bear Market: A sustained decline of 20% or more from a recent market peak. Originates from the image of a bear swiping its paws downward. Contrast with *Bull Market*.

Best Ask: The lowest resting sell price in the order book — the price at which a buyer can trade immediately.

Best Bid: The highest resting buy price in the order book — the price at which a seller can trade immediately.

Best Execution: The regulatory obligation for brokers and investment firms to take all reasonable steps to achieve the best possible result for their clients when executing orders. "Best" considers price, execution costs, speed, likelihood of execution, and market impact simultaneously, not just the quoted price in isolation. Mandated by MiFID II in the EU and the SEC's duty of best execution in the US. Best execution is the regulatory foundation that makes smart order routing (SOR) necessary: brokers must demonstrate their routing decisions genuinely serve clients' interests, not just the broker's economics.

Bid: The price at which a buyer is willing to buy. The best (highest) bid is the top of the buy side of the order book. The collection of all active bids forms the buy side of the book.

Bid-Ask Imbalance: A measure of depth asymmetry: bid_depth / (bid_depth + ask_depth), computed within a symmetric price window around the mid. Values near 1 indicate heavy buying interest; values near 0 indicate heavy selling interest. Used as a short-term price direction signal.

Blue Chip: A large, financially sound, well-established company, the most prestigious tier of the equity market. The term originates from poker, where blue chips represent the highest denomination.

Bond: A standardised debt instrument. The issuing company borrows money and promises to pay periodic interest (coupon payments) and return the face value (principal) at maturity. Bondholders are creditors, not owners, and are paid before equity holders in a bankruptcy. Bonds trade on secondary markets after issuance, allowing investors to exit before maturity. The financial term "bond" traces its roots back to the 14th century from the Old English word bindan and band, which eventually evolved into the standard English word "bind". The word was originally used to describe something that literally ties, fastens, or holds things together. Because a financial bond fundamentally operates as a formal, legally binding contract where an issuer promises to repay a loan with interest, the term stuck.

Book Depth: The total volume (quantity of shares) resting near the top of the order book. A deep book absorbs large orders without significant price impact. A thin (or shallow) book moves quickly on even moderate order flow.

Book Snapshot: A periodic complete view of the current order book state (all price levels and their quantities) published by the exchange for subscribers who need to get up to speed.

Borrow Rate: The annualised fee charged to a short seller for borrowing shares from a lender. Easy-to-borrow large-cap stocks typically have borrow rates near zero. Hard-to-borrow, heavily-shorted, or illiquid stocks can carry rates of 5–50% or more per annum, applied daily to the short position. Rising borrow rates signal increasing short interest and decreasing availability of lendable shares, and are themselves a market signal.

Bracket Order: A strategy combining an entry order with both a take-profit LIMIT and a stop-loss STOP, typically implemented as an OCO pair. See OCO.

Broker: An intermediary who executes orders on behalf of clients without trading for their own account. The word traces to Old French/Norse roots meaning someone who opens a cask and sells by the cup, a retail intermediary.

Bull Market: A sustained rise of 20% or more from a recent market trough. Originates from the image of a bull thrusting its horns upward. Contrast with *Bear Market*.

Buy-In: A remedy for failed settlement: if a seller cannot deliver securities, the buyer purchases them in the open market at the seller's expense.

5.2.3 C

Calendar Spread: A derivatives strategy involving the simultaneous purchase and sale of contracts on the same underlying instrument but with different expiry dates, for example, buy the March futures contract, sell the June futures contract. Used to profit from expected changes in the price difference (the "spread") between contract months. Requires cross-symbol coordination in the matching engine.

Call auction: Synonym for auction — "call" refers to the fact that all participants are "called" to submit orders before a single price is determined.

Capital Appreciation: An increase in the market price of a share (or other asset) above the price

paid. The primary way equity investors generate returns.

Capital Gain: The profit realised when an asset is sold for more than it was purchased for.

Cascade Cancel: When a COMBO order fails (any leg is cancelled or rejected), all remaining unfilled legs of the combo are automatically cancelled. Previously executed fills are not reversed.

CCP (Central Counterparty Clearing House): An entity that stands between buyer and seller in a trade, becoming the legal counterparty to both, eliminating direct counterparty credit risk.

Central Limit Order Book (CLOB): The canonical order-matching mechanism used by most regulated equity exchanges. Orders are queued by price and time; the best bid and best ask are continuously matched when they overlap.

Circuit Breaker: An automatic trading halt triggered when price movement exceeds a threshold within a time window, designed to prevent cascading automated trading crashes.

Clearing: The process of confirming and recording a trade, and (via a CCP) guaranteeing its settlement.

Closing price: The price of the last trade of the trading day in a symbol. In EduMatcher it is the *mark* used to value open positions at the close.

Co-location: Placing a participant's trading servers physically inside the same data centre as the exchange's matching engine, minimising network round-trip latency to microseconds.

Collar Band / Price Collar: A price filter that rejects orders whose submitted price deviates too far from a reference price. Two distinct variants: the *static price collar* (compares against the last close) and the *dynamic price collar* (compares against the last trade). See those entries for details.

Combo Order: A single instruction to execute orders in multiple instruments simultaneously. Each component is a leg.

Conflation (Market Data): Merging consecutive market data updates for the same price level into a single message under high-load conditions. Reduces message volume at the cost of losing intermediate states.

Continuous Trading: The normal trading mode where orders match as soon as they arrive and a compatible counterparty is present. Contrasts with auction mode.

Contract Multiplier: For futures and options, the dollar (or other currency) value of a one-unit price move per contract. The E-mini S&P 500 has a multiplier of $50, a 1-point move in the index is worth $50 per contract. Essential reference data: a wrong multiplier affects every P&L and margin calculation.

Corporate Action: An event affecting a company's share structure, stock splits, dividends, mergers, delistings. Requires adjustments to all open orders, positions, and reference data in exchange systems.

Counterparty Risk: The risk that the other party to a trade defaults before settlement is complete. Cleared via CCP novation, which replaces bilateral counterparty risk with exposure to the CCP.

Coupon: The periodic interest payment on a bond. Named after the physical coupon strips that bond holders historically detached and redeemed for payment.

Crossable: Two orders are crossable when they overlap in price: a buy at or above a resting sell's price, or a sell at or below a resting buy's price. Crossable orders produce a trade.

Crossing the Spread: Submitting an order at a price that matches an existing resting order, triggering an immediate fill.

Cum-Dividend: The period during which a share trades *with* entitlement to a declared dividend, the buyer receives the upcoming payment. A share trades cum-dividend from the dividend announcement up to (but not including) the ex-dividend date. After the ex-dividend date the share trades *ex-dividend* and the buyer is not entitled to the payment. The price typically drops by approximately the dividend amount on the ex-dividend date.

CUSIP: A 9-character identifier for US and Canadian securities used in clearing and settlement. Analogous to ISIN.

Custody: The safekeeping of securities on behalf of investors by a broker or custodian bank, with records maintained at a central securities depository.

Crossed book: An order book in which the best bid is at or above the best ask. It is normal while a call phase (auction, halt) collects orders without matching, and is resolved by the uncross at one price. In continuous trading a crossed book is a bug, and the order-book invariant I2 checks for it.

5.2.4 D

Dark Fibre: Unused optical fibre cable already laid in the ground but not yet carrying data traffic ("lit"). Trading firms and data providers lease dark fibre from infrastructure companies and run their own optical transceivers, creating private, dedicated network links between financial data centres. Dark fibre links are used between major exchange co-location sites (e.g., NYSE in Mahwah and NASDAQ in Carteret) to reduce cross-venue latency for arbitrage and hedging strategies.

Dark Pool: A private trading venue where orders are not displayed in a public order book. Participants submit orders and are matched against other participants' orders, typically at the midpoint of the NBBO, with the trade only reported publicly after it occurs. Used by large institutional investors to trade without revealing their intentions and causing adverse price movement. Contrast with *lit markets* (public exchanges) where resting orders are visible to all. See *Regulatory Surveillance*.

DAY: An order TIF valid only for the current trading session.

Delivery versus Payment (DvP): The settlement principle that the transfer of securities and cash happen simultaneously and conditionally, neither is released without the other. Eliminates principal risk in settlement.

Depth (Market Depth): The total quantity of resting orders available at each price level. Deep markets can absorb large orders with minimal price impact. Deep books have large quantities at each price, making it hard to move the price by trading. Shallow books are easier to move.

Depth-of-Book (Level 2): Market data showing multiple price levels and their resting quantities, rather than only the best bid and ask.

Designated Market Maker (DMM): A firm assigned by an exchange to maintain fair and orderly markets in specific securities, with both privileges and obligations. NYSE's term; Nasdaq calls them Market Makers.

Determinism (Exchange): The property that given the same initial state and the same ordered input messages, the matching engine always produces identical outputs. Essential for audit trail replay and disaster recovery.

Dilution: The reduction in existing shareholders' ownership percentage when new shares are issued. Selling 20% of a company to new investors reduces existing owners to 80%.

Dividend: A cash payment made by a company to its shareholders, distributed from retained profits. Not guaranteed, the board of directors decides whether, when, and how much to pay. Companies that pay regular dividends are typically mature, profitable businesses with stable cash flows. High-growth

companies often pay no dividend, choosing to reinvest profits in expansion. The key dates are: the *declaration date* (when the board announces the dividend), the *ex-dividend date* (buyers on or after this date do not receive the payment), the *record date* (the register of shareholders eligible to receive payment), and the *payment date* (when the cash is distributed).

Drop Copy: A real-time copy of all order lifecycle events for a specific participant, sent to a designated third party such as a clearing broker or risk system.

DTCC (Depository Trust and Clearing Corporation): The central clearing and settlement infrastructure for US equity markets. Its subsidiary DTC is the US central securities depository.

Dynamic Price Collar (Reference Price Filter): A pre-trade risk control that rejects any order whose submitted price deviates more than a configured percentage from the most recent traded price. Protects against algorithms that walk the price incrementally through a series of small trades. Contrast with *Static Price Collar*, which uses the last close as the reference. Together, the two collars provide overlapping protection against both sudden errors and gradual drift.

5.2.5 E

ECN (Electronic Communication Network): An electronic trading system that automatically matches buy and sell orders, typically with narrower spreads than exchanges but less regulatory oversight.

Equilibrium Price (Auction Price): The single price at which the maximum volume can trade in a call auction. See Auctions & Session Scheduling.

Equity: Ownership stake in a company. Equity holders (shareholders) have a residual claim on the company's assets and earnings after all creditors are paid.

ETF (Exchange-Traded Fund): A basket of securities (shares, bonds, commodities) that itself trades as a single share on an exchange.

Ex-Dividend Date: The date from which a stock trades without entitlement to a declared dividend. Buyers on or after this date do not receive the dividend.

Execution Algorithm (Algo): Software that breaks a large institutional order into smaller pieces and executes them over time to minimise market impact and achieve a target average price. Common types include VWAP (match the day's volume-weighted average price), TWAP (distribute uniformly over time), Implementation Shortfall (minimise the gap between decision price and execution price), and POV/In-Line (participate at a fixed percentage of market volume). Execution algorithms generate the majority of order flow on modern exchanges.

5.2.6 F

Face Value (Bond): The principal amount of a bond, the amount the issuer promises to repay at maturity, and the basis on which coupon payments are calculated.

Failover: The process of switching from a failed primary system to a secondary (backup) system.

Fat Finger: Industry slang for an accidental trading error caused by mistyping an order parameter (quantity, price, etc.).

FIFO (First-In, First-Out): Within a price level, orders are filled in the order they arrived. Earlier arrivals have priority.

Fill: An execution, a match between a buy and sell order resulting in a trade. Partial fills consume part of an order; a complete fill consumes all of it.

FOK (Fill-Or-Kill): An order that must be completely filled immediately or cancelled entirely.

FIX (Financial Information eXchange): The industry-standard protocol for electronic order submission, used by most exchanges and brokers globally.

Flat: Having no open position in a symbol (net quantity = 0). Neither long nor short.

Floor Broker: Historically, a broker who executed trades on the physical exchange floor. The term persists in regulations and informal speech.

FPGA (Field-Programmable Gate Array): Reconfigurable hardware chips that can implement market data parsing or order routing in dedicated logic circuits, achieving nanosecond-level latency.

Front Running: Acting on advance knowledge of a pending client order to trade for one's own account before executing the client's order, then profiting from the price movement the client's order causes. Illegal under market manipulation regulations and a serious breach of fiduciary duty. Front running is detected by surveillance systems that flag cases where a participant's own trades immediately precede large client executions in the same direction.

Futures Contract: A standardised agreement to buy or sell a specified quantity of an underlying asset (commodity, index, currency, interest rate) at a predetermined price on a specified future date. Obligations are legally binding. Futures are exchange-traded, margined, and marked-to-market daily. Unlike equity ownership, futures do not represent ownership of an underlying asset, they represent an obligation about a future transaction.

5.2.7 G

Gap (Market Data): A missing sequence number in a market data feed, indicating a lost message. Subscribers must request retransmission or resynchronise from the next full snapshot.

Gateway: The participant-facing interface to the exchange. Handles authentication, session management, and message translation.

Going Long: Owning an asset in expectation that its price will rise, so it can be sold later at a profit. You are "long" the asset. The term originates in physical commodity trading: a merchant who had purchased durable goods, grain, spices, metal, cloth, and was holding them in a warehouse was described as "long" in those goods. The word captured both *possession* (you own something tangible) and *duration* (durable goods last a long time and can be held through seasons or years). Being long in modern finance carries exactly the same meaning: you hold the asset and are exposed to its price over time. A long position profits when the price rises and loses when it falls.

Going Short / Short Selling: Selling an asset you do not currently own, typically by borrowing it from another party, with the intention of buying it back later at a lower price and returning it, pocketing the difference. The term originates in physical commodity trading: a merchant who had sold goods they did not yet possess, committing to deliver more than their warehouse held, was "short" of the goods. The same word family as "falling short," "short of supplies," or "shortage", deficient, insufficient, lacking. In the early forward markets of Amsterdam and London, a short seller would sell forward (promise future delivery), then purchase the goods in the market before delivery, hoping prices had fallen. Isaac Le Maire's 1609 campaign against VOC shares is the earliest recorded large-scale short selling operation. Today, a short position profits when the price falls (you buy back cheaper than you sold) and loses when it rises, in theory without limit, since a rising price has no ceiling.

GTC (Good-Till-Cancelled): A time-in-force value that keeps an order alive across trading sessions. GTC orders are serialised to storage (disk) at shutdown and reloaded next session. Hence, it survives engine restarts.

GTD (Good-Till-Date): An order Time-In-Force variant that keeps an order active until it fills, is explicitly cancelled, or reaches a specified expiry date, whichever comes first. On the expiry date the order is automatically cancelled. Provides the persistence of GTC without the open-ended risk of an order remaining active indefinitely. Requires the exchange scheduler to track each GTD order's expiry and issue the cancellation at the right time.

5.2.8 H

Heap: A data structure that gives O(1) access to the minimum (or maximum) element. Used in matching engines to efficiently maintain best price access.

High-Frequency Trading (HFT): Automated trading strategies that execute very large numbers of orders at extremely high speeds, typically holding positions for milliseconds or seconds rather than minutes or days. HFT firms invest heavily in low-latency infrastructure (co-location, microwave links, FPGAs) to gain speed advantages. Strategies include latency arbitrage, market making, and statistical arbitrage. Controversial because speed advantages can disadvantage slower participants; the subject of the IEX speed bump.

5.2.9 I

Iceberg Order / Hidden Order / Reserve Order: An order that shows only a small visible peak while hiding a large reserve. Each time the visible portion fills, the order replenishes from the reserve. See Order Types.

Imbalance: The difference between total buy and sell quantity at the equilibrium price. Expressed as a side (BUY or SELL) and a quantity.

Immediate-Or-Cancel (IOC): An order that fills as much as possible immediately and cancels any unfilled remainder. Never rests in the book.

Implementation Shortfall (IS) / Arrival Price Algorithm: An execution algorithm that minimises the difference between the *decision price* (the market price when the investment decision was made) and the actual average execution price. IS algos trade faster when the price moves against the position (urgency increases to avoid further shortfall) and slower when price moves favourably. More adaptive and sophisticated than VWAP or TWAP; preferred by quantitative funds whose models predict short-term price moves.

Implied Matching: A technique used in derivatives markets where the exchange combines spread orders and outright orders to synthesise matches that neither could achieve alone. For example: a spread order to buy March/sell June can be combined with a resting outright June sell to create an implied March buy. Implemented on CME Globex and Eurex for calendar spreads.

Implied Order: An order derived by the matching engine from a combination of existing spread and outright orders. Not submitted by any participant, computed from pre-existing orders and published in the relevant outright or spread book. When an implied order matches, all contributing orders execute simultaneously. If any contributing order is cancelled or filled by other means, the implied order disappears instantly.

Implied-In: A form of implied matching where an outright order in one month plus a spread order imply a new outright in the other month. Example: January sell at \$75 + Jan/Feb spread buy at −\$2 = implied February sell at \$77.

Implied-Out: A form of implied matching where two outright orders imply a spread. Example: January buy at \$75 + February sell at \$77 = implied Jan/Feb spread sell at −\$2.

Inactivation (Quote): Automatically cancelling or suspending the surviving leg of a two-sided

quote the moment the other leg is filled. The market maker must re-quote both sides afresh within their re-quoting window. Used in conservative quote refresh policies (Eurex-style). Protects the market maker from being left with a stale unhedged quote after their inventory position has changed.

Indicative Uncross Price: During an auction accumulation period, the price at which uncrossing *would* occur if matching happened at that moment. Recalculated after every order arrival and published as a signal to help participants decide whether to adjust their orders before the actual uncross.

Information Asymmetry: A situation where some market participants have access to information (faster data, better analytics) that others do not, giving them a trading advantage. Auctions reduce information asymmetry by letting all participants submit orders before any execution occurs.

Informed Flow: Order flow that originates from participants who have an informational advantage about near-term price direction, for example, participants who have seen relevant news before others. When a market maker is repeatedly hit by informed flow, they accumulate inventory at prices that are about to move against them. Informed flow is the root cause of adverse selection.

Initial Margin: Collateral deposited when opening a position to cover potential losses. Calculated by the CCP based on the instrument's historical volatility and the position size.

Insider Trading: Trading on material, non-public information, for example, buying shares before a publicly unannounced merger is announced. Illegal in all major jurisdictions. The exchange's audit trail is a key source of evidence in insider trading investigations, as regulators can reconstruct exactly who traded, when, and at what prices relative to when news became public.

Instrument / Symbol: The specific financial product being traded. Each instrument has its own order book.

Internaliser: A broker-dealer that matches client orders internally against its own inventory or other clients' orders, rather than routing to an exchange. A form of OTC secondary trading.

IOC (Immediate-Or-Cancel): A time-in-force (and order type) where an order sweeps available liquidity at the limit price and any unfilled remainder is immediately cancelled. Never rests on the book. See Order Types.

IPO (Initial Public Offering): The first sale of a company's shares to the general public, after which the shares are listed and traded on a stock exchange.

ISIN (International Securities Identification Number): A 12-character global identifier for a security, independent of the exchange it trades on. A single company has one ISIN regardless of where its shares are listed.

ITCH (NASDAQ TotalView-ITCH): NASDAQ's binary, message-based market data feed protocol for order book dissemination. ITCH publishes sequenced events such as add order, modify/replace, cancel/delete, executions, and auction/trade messages so subscribers can reconstruct the full limit order book deterministically from the stream. Official specification: NASDAQ TotalView-ITCH 5.0.

5.2.10 K

Kernel Bypass: Technology that allows a network application to communicate directly with network hardware, bypassing the operating system kernel and reducing latency from microseconds to sub-microsecond.

Kill Switch: An emergency control that immediately cancels all resting orders and quotes for a specific participant. Industry kill switches often also mark the gateway inactive until reconnection

and re-authentication; EduMatcher's does not — `risk.kill_switch/KILL` cancels exposure only and the gateway connection stays usable, so a disconnect-and-block effect has to come from the gateway's own `disconnect_behaviour` (see *Mass Cancel*) rather than from the kill switch itself. Triggered by the participant (emergency self-halt) or the exchange operator (`pm-admin`'s `KILL`). Mandatory under most exchange regulations including MiFID II.

5.2.11 L

Latency: The time delay between an action and its effect. In exchanges, latency is measured in microseconds or nanoseconds. Low latency is a competitive advantage.

Latency Arbitrage: Trading that profits from the brief period between when a price changes on one venue and when that change propagates to another venue.

Layering: A market manipulation practice: placing large orders to create a false impression of market depth, then cancelling them once the desired price movement occurs.

LCH: A major international clearing house, part of the London Stock Exchange Group. Clears interest rate swaps, equities, foreign exchange, and other products.

Ledger: A permanent record of money movements, like a bank statement. EduMatcher's settlement ledger is `eod.db`.

Leg: One component order within a combo order or spread strategy.

Leg Risk / Legging Risk: The risk that one leg of a combo order fills while another does not, leaving an unintended single-instrument position.

Level 1 Data: The best bid and ask prices and quantities only.

Level 2 Data (Market Depth): The full order book, all price levels and their quantities.

Limit Order: An order with a price constraint. A buy limit order at $150.30 will only execute at $150.30 or lower — the buyer is setting the maximum price they will pay. A sell limit order at $150.35 will only execute at $150.35 or higher. Limit orders that cannot immediately match rest in the book.

Limit Up-Limit Down (LULD): A US regulatory circuit breaker mechanism that pauses trading in an individual stock if its price moves outside a percentage band (the "price band") calculated from a reference price. The bands are tighter for liquid large-cap stocks and wider for smaller, more volatile ones. If the best offer falls below the lower band or the best bid rises above the upper band, trading pauses for 15 seconds; if the imbalance persists, a 5-minute trading halt begins. Implemented across all US equity exchanges.

Liquidity: The ease with which an asset can be bought or sold without significantly affecting its price. High liquidity = tight spread, large depth, easy trading.

Locate: The process of finding and reserving a source of borrowable shares before executing a short sale. Under US Regulation SHO, broker-dealers must have reasonable grounds to believe shares can be borrowed before accepting a short sale order. "Failing to locate" before shorting is called naked short selling, which is illegal in most jurisdictions.

Long Position: Owning a positive quantity of shares. You profit if the price rises. Expressed as a positive number in EduMatcher's P&L ledger.

Lot: The standard unit of quantity for an instrument.

5.2.12 M

Maintenance Margin: The minimum collateral balance that must be maintained in a margin account. Falling below this level triggers a margin call.

Maker: The participant whose resting order was already in the book when a fill occurred. Contrast with *Aggressor/Taker*.

Margin Call: A demand from a clearing house or broker for a participant to deposit additional collateral to restore their margin account to the required level.

Mark-to-Market: The daily revaluation of positions at current market prices, with cash payments made to reflect daily gains and losses. The mechanism by which variation margin is calculated.

Market capitalisation (market cap): The total market value of all a company's outstanding shares, calculated as share price multiplied by the total number of shares in existence. If Apple has approximately 15.4 billion shares outstanding and each trades at $190, Apple's market cap is roughly $2.9 trillion [1]. Market cap is the most widely used shorthand for a company's size. When rankings refer to "the world's largest exchange by listed market cap," they are summing the market caps of every company listed there.

Market data: The continuous stream of information about what is happening in the book: what prices orders are resting at, how much quantity, and what trades have happened. Display tools (viewer, board, ticker) consume market data.

Market Fragmentation: The distribution of trading volume for a single instrument across multiple competing venues, exchanges, dark pools, ATSs, and internalisers. Modern equity markets are highly fragmented; in the US, AAPL may trade simultaneously across a dozen registered exchanges plus numerous alternative venues. Fragmentation improves competition and reduces fees but complicates order routing and best-execution analysis. Regulatory frameworks like Regulation NMS (US) and MiFID II (EU) were introduced partly in response to fragmentation.

Market Impact / Slippage: The adverse price movement caused by a large order sweeping through multiple price levels.

Market Impact Estimation: The process of computing, before order submission, the average price you would pay (or receive) by sweeping through available book levels for a given order size. Derived from the cumulative depth profile. Used by execution algorithms to size order slices and choose between aggressive execution and passive accumulation.

Market Maker: A professional participant obligated to continuously provide both buy and sell prices, ensuring liquidity for other participants.

Market Maker Protection (MMP): An exchange-provided mechanism that automatically cancels all of a market maker's resting quotes when fills arrive at a rate exceeding a configured threshold within a rolling time window. Protects market makers from being rapidly picked off by participants with superior information (adverse selection). After MMP fires, the market maker enters a protection period before re-quoting. Parameters (fill count, time window) are configured per market maker per symbol.

Market Microstructure: The study of how trading mechanisms, order types, and market rules affect price formation and liquidity. EduMatcher is designed to demonstrate core microstructure concepts.

Market Order: An order that attempts immediate execution against available liquidity at the best obtainable prices. It prioritises execution certainty over price certainty, but execution is still subject to exchange state, available liquidity, price collars, and risk controls. A market order submitted

during a halt, against an empty book, or rejected by a pre-trade risk check will not fill.

Mass Cancel: A participant-initiated message type that cancels multiple resting orders simultaneously, all orders for a symbol, all orders of a given type, or all orders system-wide. The gateway connection remains active after a mass cancel; the participant can submit new orders immediately — in EduMatcher, this is the same behaviour as a *Kill Switch*, since neither action by itself disconnects the gateway. A gateway is only shut out by an explicit disconnect (`pm-admin`'s `KICK`, or a `disconnect_behaviour` of `CANCEL_ALL`/`CANCEL_QUOTES_ONLY` firing when the gateway drops on its own).

Matching Engine: The core software that manages order books and matches buy and sell orders to produce trades.

Maturity (Bond): The date on which a bond's issuer must repay the face value to the bondholder.

Microwave Link: A radio communications link using microwave frequencies. Between financial data centres, microwave links provide lower latency than fibre-optic cables because radio waves travel faster through air than light through glass.

Mid Price: The average of the best bid and best ask: mid $= \frac{\text{best bid}+\text{best ask}}{2}$. Often used as a reference price when the spread is non-zero.

Midpoint Peg Order: An order whose price continuously tracks the current mid price (the average of the best bid and best ask). Executes only when a counterparty arrives whose order is also willing to trade at the midpoint. The mid is a *better* price for both parties than the quoted bid or ask, the buyer pays less than the ask, the seller receives more than the bid; the spread is shared rather than fully paid. Provides price improvement at the cost of uncertain execution timing: there is no guarantee a counterparty will appear. Common in dark pools; also supported on IEX on the lit market. Never competes in the regular price-time priority queue, maintained as a separate midpoint matching layer.

Minor unit: The smallest division of a currency (the cent, the penny). Amounts are kept as whole numbers of minor units — 12.34 is stored as 1234 — so sums are exact. `pm-eod` uses two decimals.

MMP Threshold: The fill count and time window that together trigger Market Maker Protection. For example, "5 fills within 1 second." If this threshold is exceeded, the exchange automatically cancels all of the market maker's resting quotes. The threshold is negotiated per market maker, per symbol, and finding the right value is a key parameter of market making strategy.

Monotonic: Strictly non-decreasing. A monotonic sequence is one where each value is equal to or greater than all previous values. A strictly monotonic sequence is one where each value is strictly greater.

5.2.13 N

Naked Short Selling: Selling shares short without first locating and arranging to borrow them. The seller delivers nothing at settlement, creating a "fail to deliver." Illegal in most jurisdictions under short sale regulations (Regulation SHO in the US). Distinguished from covered short selling, where shares have been located and borrowed.

Nanosecond: One billionth of a second (10 seconds). Modern matching engines record timestamps at nanosecond precision.

NBBO (National Best Bid and Offer): The best available bid and ask prices aggregated across all US exchanges, as required by Regulation NMS. Brokers must offer clients execution at or better than the NBBO.

NAV (Net Asset Value): The total value of a fund's holdings divided by the number of shares

outstanding. For mutual funds, NAV is computed daily using the official **closing price** of each holding — one reason the closing price matters so much.

Netting: Combining many obligations into one. All of a trader's gains and losses in a day are combined into a single amount to pay or receive.

Novation: The legal process by which a CCP replaces the original buyer-seller relationship: the CCP becomes buyer to the seller and seller to the buyer, eliminating bilateral counterparty risk.

NTP (Network Time Protocol): A protocol for synchronising computer clocks over a network. Basic NTP achieves millisecond accuracy; higher-precision variants achieve sub-millisecond.

5.2.14 O

OCC (Options Clearing Corporation): The central clearing house for equity options in the US.

OCO (One-Cancels-Other): Two linked orders where filling either automatically cancels the other. See Order Types.

OHLCV: Open, High, Low, Close, Volume, the five standard summary statistics recorded for each trading session (or any time period). Open: the first traded price. High: the highest traded price. Low: the lowest traded price. Close: the last traded price (the official closing price in a closing auction). Volume: total shares traded. OHLCV data forms the basis of almost all price charts and historical market analysis.

Open Outcry: The historical method of trading on exchange floors, where traders shouted bids and asks and used hand signals in a sunken circular area called a pit. Largely replaced by electronic trading.

Options: Contracts that give the buyer the right, but not the obligation, to buy (a **call option**) or sell (a **put option**) an underlying asset at a specified price (the **strike price**) before or on a specified date (the **expiry date**). The seller of an option takes on the corresponding obligation. Options are priced using models (Black-Scholes being the most famous) that account for time to expiry, volatility, interest rates, and the distance between the current price and the strike. Cboe is the world's largest options exchange.

Order: An instruction to buy or sell a quantity of an instrument at a specified price or better.

Order Book: The central data structure of a matching engine, all resting buy and sell orders for a symbol, organised by price.

Order ID: A unique identifier assigned to each order upon submission. Must be unique across the entire system.

OTC (Over-the-Counter): Trading that occurs directly between two parties without going through an exchange. OTC trades are less transparent and subject to different regulations than exchange-traded transactions.

OUCH (NASDAQ OUCH): NASDAQ's high-performance binary order entry protocol used by member firms and low-latency trading systems to submit, modify, cancel, and manage orders directly at the matching engine. OUCH is session-based and message-oriented, with exchange acknowledgments and execution/cancel responses that let clients track the full order lifecycle deterministically. Official specification: NASDAQ OUCH 5.0.

5.2.15 P

P&L (Profit and Loss): The financial gain or loss on a trading position.

Passive Order: See *Resting Order*

Payment for Order Flow (PFOF): A practice in which retail brokers receive payment from wholesale market makers in exchange for routing client order flow to those market makers for internalisation rather than to lit exchanges. Market makers pay for retail flow because retail orders have lower adverse selection risk. Banned in the UK and EU under MiFID II. Controversial in the US, where it subsidises zero-commission brokerages but raises concerns about conflicts of interest and reduced price discovery. See *Best Execution*, *Internaliser*.

Pit: The sunken, octagonal or circular physical trading area on an exchange floor where open outcry trading happened. CME Group operated prominent futures pits in Chicago. Largely dormant today.

Portfolio: A collection of investments held by an investor, typically diversified across asset types (equities, bonds, etc.) to manage risk and return.

Position: A participant's current holding in an instrument, positive (long) if they own more than they have committed to deliver; negative (short) if they have committed to deliver more than they own; zero (flat) if both sides balance. Position tracking is the clearing system's primary responsibility.

Pre-Trade Risk Controls: Checks applied to orders before they reach the matching engine, quantity limits, notional limits, fat-finger filters, credit checks, and rate limiting. The gateway is typically the enforcement layer.

PRE_OPEN: Session phase before the opening auction. Orders are accepted and rest on the book, but no matching occurs.

Presence Obligation: A market maker's contractual requirement to have live two-sided quotes resting in the book for a minimum percentage of the trading session, commonly 85% or more. Prevents market makers from only quoting on easy days and disappearing on volatile ones when liquidity is most needed.

Price Discovery: The process by which the market determines the current fair price of an asset through the interaction of buyers and sellers. A well-functioning exchange enables efficient price discovery: prices reflect all currently available information and update rapidly as new information arrives. Price discovery is one of the three core promises every exchange makes. Poor price discovery, where the traded price diverges significantly from fair value for extended periods, is a sign of market dysfunction.

Price Level: All resting orders at the same price on the same side of the book. Their combined quantity is the depth at that level.

Price-Time Priority: The matching rule: better prices fill first; at the same price, earlier arrivals fill first.

Primary Market: Where new securities are issued and sold for the first time. The company (or government) receives the proceeds.

Primary Site: The active, live data centre running the production matching engine.

Private Company: A company whose shares are not publicly traded, held only by founders, employees, and private investors.

Pro-Rata Allocation: An alternative to price-time priority where fills at a price level are distributed proportionally among all resting orders at that level. Common in some futures markets.

Protection Period: The window of time after MMP fires during which a market maker has no active quotes and is expected to assess market conditions before deciding whether and at what prices to re-enter. The duration is typically a few seconds and is defined in the market maker agreement.

PTP (Precision Time Protocol, IEEE 1588): A network clock synchronisation protocol achieving sub-microsecond accuracy. Used in exchange infrastructure to ensure consistent timestamps across distributed systems.

Public Company: A company whose shares are listed on a stock exchange and available to any investor.

5.2.16 Q

Quote: A two-sided order submitted by a market maker containing both a bid (buy) and an ask (sell) simultaneously. Unlike a regular order, the two legs are linked, what happens to one (a fill, a cancellation) triggers a defined response on the other according to the quote refresh policy.

Quote Leg: One side (bid or ask) of a two-sided market maker quote. Filling one leg triggers the quote refresh policy for the other leg.

Quote Refresh Policy: The rule governing what happens to the surviving leg of a two-sided quote when the other leg is filled. Three main variants: (1) cancel both sides immediately, the most conservative, used by Eurex-style inactivation; (2) cancel only the filled side, leave the other active; (3) leave both active. The choice reflects the market maker's balance between speed of re-entry and risk of holding stale inventory.

Quote Stuffing: A market manipulation practice where a participant rapidly submits and cancels a very large number of orders to consume exchange bandwidth, slow down competitors' systems, and create artificial confusion in market data. A form of denial-of-service abuse in electronic markets.

5.2.17 R

Ratchet: The mechanism in a trailing stop that advances the stop price in the favourable direction but freezes it when the market reverses.

Rate Limiting / Throttling: A control that limits the number of orders a participant can submit per unit time. Prevents denial-of-service conditions from algorithmic misfires.

Re-quoting Obligation: A market maker's contractual requirement to post a fresh two-sided quote within a specified maximum delay after either leg of their previous quote is filled or after MMP fires. Failure to re-quote within this window counts against the presence obligation and may trigger penalties. The window is typically measured in milliseconds to low seconds.

Realized P&L: Profit or loss that has been "locked in" through a trade that reduced an open position. Once realized, it cannot be reversed by subsequent price moves.

Reference Data / Instrument Master: The configuration data describing a tradeable instrument, tick size, price scale, session schedule, contract multiplier, trading status, expiry date, position limits. Read by every exchange component; an error propagates system-wide. Treat it with the same rigour as code: version-controlled, tested, applied atomically.

Regulation NMS: A US Securities and Exchange Commission regulation requiring that equity orders be executed at the best available price across all exchanges (the NBBO). A primary driver of US market fragmentation.

Regulation SHO: A US Securities and Exchange Commission rule governing short sales in equity markets. Key requirements: broker-dealers must have a reasonable belief shares can be borrowed before accepting a short sale order (the "locate" requirement); persistent fail-to-deliver situations must be closed out; short sales must be marked as "short" in order records. Adopted in 2005 to address concerns about abusive naked short selling.

Replay (Exchange): Replaying a recorded sequence of input messages through the matching engine to reconstruct past market states. Requires the engine to be deterministic.

Reserve Refresh Priority: The queue position rule for iceberg replenishment. Most exchanges place a newly replenished iceberg peak at the back of the queue at its price level, equivalent to a brand-new arrival, rather than preserving the original queue position.

Resting order: (also called a **passive order**) — an order that has been accepted by the exchange and is sitting in the book waiting for a counterparty. It does not execute immediately.

Retained Earnings: Profits kept within the business rather than distributed to shareholders. A source of self-funding for company growth.

Rolling Window (MMP): The time period over which fills are counted for Market Maker Protection purposes. Fills older than the window are discarded from the count; only fills within the current window matter. For example, a 1-second rolling window means: "if more than N fills have arrived in any sliding 1-second period, MMP fires."

5.2.18 S

Secondary Market: Where existing securities are traded between investors. The issuing company does not receive proceeds. Stock exchanges are secondary markets.

Secondary Site / Backup Site: A standby data centre that can take over exchange operation if the primary site fails.

Securities Lending / Stock Borrow: The temporary transfer of securities from a lender (typically a long-term investor or custodian) to a borrower (typically a short seller) in exchange for collateral and a lending fee (borrow rate). Short sellers must borrow shares before selling short. The lender retains economic ownership (receives manufactured dividends and can recall the shares) but transfers legal title temporarily. Recall risk, the lender demanding shares back at an inconvenient time, is a key risk for short sellers.

SEDOL: A 7-character UK security identifier used in settlement and clearing. Analogous to CUSIP.

Self-Match Prevention (SMP): A mechanism that detects when a participant would trade against their own orders and applies a cancellation policy.

Sequence Number: A monotonically increasing counter attached to events (in a drop copy or message feed) to enable detection of missed events and ordered replay.

Session State Machine: The formal model of an exchange's trading day phases (PRE_OPEN, OPENING_AUCTION, CONTINUOUS, CLOSING_AUCTION, CLOSED) and the permitted transitions between them. See A Full Trading Day.

Settlement: The final transfer of securities from seller to buyer and cash from buyer to seller. Legally completes the trade. In the US, equity settlement currently occurs on T+1.

Settlement Method (Derivatives): Whether a futures or options contract settles by cash payment or physical delivery of the underlying asset. Determines post-expiry processing required by the exchange and clearing house.

Shadow Replication / Shadow Mode: An architectural pattern for high-availability exchange systems. The secondary site receives and processes all input messages in parallel with the primary, maintaining an up-to-date copy of the order book, but does not publish outputs or accept participant connections. If the primary fails, the secondary's in-memory state is already current and it can begin accepting inputs and publishing outputs with minimal failover time. More common in production than true active-active matching.

Share / Stock: A single unit of equity ownership in a company. Each share represents a fractional claim on the company's assets, earnings, and votes.

Short Position: Selling shares you do not own (borrowed from a broker). You profit if the price falls. Expressed as a negative number in EduMatcher's P&L ledger.

Slippage: The difference between the expected fill price and the actual average fill price, caused by executing against multiple price levels in the book. Large orders in thin books experience more slippage.

Smart Order Router (SOR): Software that evaluates multiple trading venues and routes orders to achieve the best overall execution, balancing price, fees, available depth, and speed.

SmartNIC / Ultra-Low-Latency NIC: A specialised network interface card designed for trading and other latency-sensitive applications. Unlike commodity Ethernet cards, SmartNICs include an on-card kernel bypass networking stack (such as Solarflare's OpenOnload or Mellanox's RDMA), hardware timestamping of incoming packets to nanosecond precision, and CPU offload for networking tasks. Examples include the AMD Solarflare SFN series and NVIDIA Mellanox ConnectX series, widely deployed in exchange co-location environments.

Snapshot (Persistence): A complete dump of the matching engine's state (all resting orders, positions, key data) taken periodically to reduce recovery time by limiting the amount of write-ahead log that must be replayed.

Specialist: NYSE's historical term for the designated firm responsible for maintaining fair and orderly markets in a specific stock, including maintaining the physical order book. The modern equivalent is the Designated Market Maker (DMM).

Speed Bump: A deliberate artificial delay applied to all incoming orders to eliminate the advantage of being marginally faster than other participants. Pioneered by IEX.

Split-Brain: A failure mode in distributed systems where two nodes both believe themselves to be the active primary, leading to conflicting actions.

Spoofing: A market manipulation practice: placing orders with no intention of trading to move prices, then cancelling before execution. Illegal under market abuse regulations.

Spread: The difference between the best ask and the best bid. Market makers earn the spread; participants pay it as the cost of immediate trading. A tighter spread means more liquid, cheaper-to-trade market.

State Machine: A formal model consisting of a finite set of states and explicit rules about which transitions between states are allowed.

Static Price Collar (Fat-Finger Filter): A pre-trade risk control that rejects any order whose submitted price deviates more than a configured percentage from the last official close price. Designed to catch obvious fat-finger errors, an order at $15.00 on a stock trading at $150.00 is almost certainly a decimal error. Because the reference is the previous close, the static collar is stable throughout the trading day. Contrast with *Dynamic Price Collar*, which tracks the most recent trade instead.

Stop Order: A conditional order that is dormant until the market price reaches the stop price, then converts to another order type. Used to automatically exit a position if the price moves against you.

Stop Price: The trigger price on a stop or stop-limit order. The order remains dormant until the market's last trade price reaches or crosses the stop price, at which point it activates and enters the book (as a market or limit order, depending on type). A buy stop has a stop price above the current market; a sell stop has a stop price below it.

Stop-Limit Order: A stop order that converts to a limit order when triggered. Provides price protection but may not execute if the market gaps.

Stop-Loss: A sell stop order used to automatically exit a losing long position if the price falls to a specified level.

Surplus: The absolute difference between cumulative buy and sell quantity at the equilibrium price — the quantity that cannot be matched.

Sweeping: The process of an aggressive order filling against resting orders at successive price levels.

5.2.19 T

T+1 / T+2 Settlement: Settlement occurring 1 or 2 business days after the trade date. The US moved from T+3 to T+2 in 2017 and to T+1 in 2024; T+2 remains the standard in many other markets.

Tick: The minimum price movement for an instrument.

Tick Count / Tick-Based Price: A price represented as an integer number of minimum increments, rather than a decimal. Tick counts are exact; floating-point decimals are not.

Tick Size: The value of one tick, the minimum price increment for an instrument.

Tick-to-Trade Latency: The time from when a market data message leaves the exchange to when an order based on that message is received by the exchange. A key performance metric for market makers and HFT firms.

Ticker / Ticker Symbol: The abbreviated code identifying a tradeable instrument (e.g., AAPL for Apple Inc.). Named after the mechanical stock ticker machine that printed trade prices on paper tape using telegraph signals in the late 19th and early 20th centuries.

Ticker Tape: The narrow paper strip printed by the stock ticker machine, showing a continuous stream of trade symbols, prices, and volumes. The source of the term "tick" (minimum price movement) and related concepts.

Time-In-Force (TIF): The attribute specifying how long an order remains valid. EduMatcher's `TIF` enum has four members: `DAY`, `GTC`, `ATO`, and `ATC`. `IOC` and `FOK` are order *types* in EduMatcher (`TYPE=IOC`/`TYPE=FOK`), not `TIF` values, even though other venues sometimes model them as a time-in-force.

Top-of-Book (Level 1): Market data showing only the best bid price, best ask price, and their quantities. The minimum information needed to understand the current market.

Trade: An executed match between a buy and sell order. Also called a fill or execution. A trade record contains: the instrument, the price, the quantity, and which orders were involved. (also called a **fill** or **execution**)

Trailing Stop: A stop order whose trigger price automatically advances in the favourable direction as the market moves, protected by a fixed trail offset.

TWAP (Time-Weighted Average Price Algorithm): An execution algorithm that distributes a large order evenly across equal time intervals, regardless of trading volume. Simple and predictable, but less adaptive than VWAP. Used when a stock has no reliable volume profile or when the trader wants mechanically consistent participation. Performance is measured against the average of prices during the execution window, not the overall day's VWAP.

Two-Sided Quote: A market maker's simultaneous bid and ask for the same instrument. Both sides must be live at the same time to fulfil the two-sided quoting obligation. A one-sided quote

(only bid or only ask) is a contractual breach of market maker obligations.

5.2.20 U

UUID (Universally Unique Identifier): A 128-bit identifier designed to be unique without central coordination. UUID v4 is randomly generated; UUID v1 incorporates the current time and network address.

Uncross: The single atomic batch execution that occurs at the end of an auction phase. All crossable orders execute at the **equilibrium price** simultaneously.

Unrealized P&L: The theoretical profit or loss on an open position if it were closed at the current market price. Changes continuously as the market moves. Becomes realized P&L when the position is closed.

5.2.21 V

Variation Margin: Daily cash payments reflecting mark-to-market gains and losses on open positions. Prevents losses from accumulating to unmanageable levels before settlement.

Venture Capital (VC): Private investment firms that provide funding to early-stage companies in exchange for equity stakes, before those companies are large enough to go public.

VIX (Volatility Index): A real-time index calculated by Cboe from the prices of S&P 500 options, representing the market's expectation of 30-day price volatility in the S&P 500. Commonly called the "fear gauge", when markets are uncertain or falling sharply, participants buy put options for protection, driving up option prices and therefore the VIX. Values above 30 typically indicate high market stress; values below 20 indicate relative calm. Derivatives on the VIX itself trade on Cboe.

VWAP (Volume-Weighted Average Price): The average price paid for shares, weighted by the quantity traded at each price level. EduMatcher uses VWAP as the cost basis for position tracking.

$$\text{VWAP} = \frac{\sum(\text{price}_i \times \text{qty}_i)}{\sum \text{qty}_i}$$

VWAP Algorithm: An execution algorithm that slices a large order and executes it throughout the day in proportion to expected trading volume, more during high-volume periods (open, close) and less during low-volume periods (mid-day). The goal is to achieve an average execution price close to or better than the day's volume-weighted average price. The most widely used institutional execution benchmark. Distinct from *VWAP (Volume-Weighted Average Price)*, which is a measurement; the VWAP algorithm is the execution strategy aimed at matching that measurement.

5.2.22 W

Wash Trading / Wash Trade: A transaction where the same participant is on both sides, generating artificial trading volume with no real change of ownership. Illegal in all major jurisdictions under market manipulation regulations (the EU's Market Abuse Regulation, the US SEC's Rule 10b-5, and others). Also called a wash sale. Sometimes performed accidentally, see *Self-Match Prevention (SMP)*, but when deliberate, it is a form of market abuse used to inflate trading volume figures or create false impressions of liquidity.

Write-Ahead Log (WAL): A persistent, append-only log of all input messages to the matching engine. Written before processing; used for recovery and replay. The source of truth from which all derived state (the in-memory order book) can be reconstructed.

5.2.23 Z

Zero-sum: A system in which one party's gain is exactly another's loss, so all results add up to zero. Trading on an exchange is zero-sum, and `pm-eod` refuses to settle a day that is not.

ZeroMQ (ZMQ): The high-performance messaging library used by EduMatcher for inter-process communication. Uses a broker-less PUSH/PULL and PUB/SUB topology. No external message queue server is required. See Architecture.

CHAPTER 6

Known Limitations & Bugs

Learning objectives

After reading this page you will understand:

- What the exchange **deliberately does not do** today: the boundaries of the model, collected in one table with a pointer to the chapter that explains each
- A real historical bug in how the engine reported fills for orders that sweep **multiple price levels** in a single execution
- Why the bug affected only the **average cost** (and therefore unrealized P&L) of the aggressing order, never the **quantity** of any position
- A worked numeric example that makes the size of the (former) error concrete, and what the engine reports today for the same scenario
- How the fix actually works, as a short case study in fixing a reporting bug on a hot code path without changing the number of messages a gateway receives

Prerequisite: Read Positions and P&L first — this page builds directly on the VWAP average-cost formula described there.

6.1 Current limitations and design boundaries

EduMatcher is a teaching exchange. Some things a production venue does are left out on purpose, and a few are simply not built yet. This table collects them in one place so a student, or an operator planning an exercise, does not have to find them by running into them. Each row names the chapter that explains the detail.

Area	Boundary	What to do instead / why
Combo orders	A combo is a parent with independent child orders on separate books, so it has **leg risk**: if one leg fills and the other never does, the combo ends `FAILED`, the other leg is cancelled, and the fill that happened stays a real, committed trade. There is no atomic combo book	An `AON` combo only guards the *entry*: if every leg could fill completely right now the legs are matched, otherwise none is matched on entry and all rest as interest. It is not an atomic guarantee for later fills, so size legs with the risk in mind. See Guided Tour

Area	Boundary	What to do instead / why
Implied / synthetic orders	No spread book and no implied quotes. Combo handling is parent/child tracking over the outright books	See Implied Orders, which describes the mechanism as a concept only
Amending	Only resting `LIMIT` and `ICEBERG` orders can be amended. A stop, trailing stop or any other type answers `Cannot amend <TYPE> orders`	Cancel and re-enter. An amend that raises the quantity or changes the price loses time priority by design. See The Order Book: Deep Dive
Stops before the first trade	A stop is compared with `last_trade_price` only. In a session with no trade yet every stop is armed, however near its trigger is to the market	See the deep dive, *Stop Orders*
Instruments	One order book per symbol, priced on an integer tick grid of 0–8 decimals. No contract multipliers (a design proposal only), no per-price tick table	Model a different size with `outstanding_shares` and `tick_decimals`
Positions and P&L	P&L is gross (no fees or commissions), tracked per gateway ID with no netting across gateways of one participant, and marked intraday. Settlement dates (T+1/T+2) are not tracked	See Positions and P&L
Engine concurrency	The engine is one single-threaded event loop and the single writer for a data directory: no hot standby and no failover. One data directory is one exchange	A deliberate choice: it removes locks and makes matching deterministic. Restarting restores resting orders and combos from the persisted state. See Persistence
Lossy event bus	The `PUB/SUB` feeds drop messages for a subscriber that falls behind its high-water mark, silently. The engine's inbound `PULL` queue can also fill	Every published message carries a per-topic sequence number in its envelope, so a consumer can detect a gap; the drop-copy feed adds a `from_seq` replay. See Architecture Overview and Order Flow End to End
Drop-copy replay	The replay buffer holds the last `drop_copy_buffer_size` events (default `10000`). A consumer that was away longer receives only the subset still buffered	Fill the gap from the audit log. See Drop Copy

Area	Boundary	What to do instead / why
Security	The internal bus has no authentication or encryption and binds to `127.0.0.1` by default. The `pm-index` command socket accepts any non-empty `gateway_id` and uses it only to route acks. BALF has no protocol-level encryption	Keep the bus on a private interface; put external gateways behind TLS or a tunnel. See Environment variables
BALF 1.0.0	No OCO, no multi-leg combos, no market-data broadcast, no sequence recovery (reconnect on a gap), symbols of at most 8 ASCII characters	Use ALF for OCO and combos, `pm-md-gwy` for market data. See BALF Gateway
Indices	At most 5 indices, cap-weighted	See the `indices` block in Configuration Schema
Market-maker bot	The `passive` strategy does not skew for inventory, and cannot be combined with `inventory_skew`. Its notion of "covered" is judged from aggregated price levels	See The Market-Maker Bot

The rest of this chapter is a worked case study of one limitation that *was* fixed.

6.2 What this page covers

EduMatcher is a teaching exchange, and part of teaching is being honest about the places where the implementation used to take a shortcut — and what it took to fix it properly. This chapter is a case study of one such bug: when a single aggressive order traded against several resting orders at *different* prices in one shot (a "sweep"), the engine used to collapse those fills into **one** fill notification that carried the **last** trade price rather than the true volume-weighted average.

The bug was subtle: nobody lost or gained shares, the public trade record was always correct, and a single-price fill was always reported perfectly. The error only ever showed up in the **average cost** that a trader's gateway derived from the fill notification, and only for an order that walked through more than one price level.

!!! success "Status: fixed" This was a genuine bug in early versions of EduMatcher. It was fixed in the matching engine (internally tagged **H6**, alongside the related duplicate-fill guard **H5**) before release 0.16.1. This page keeps the worked example and the diagrams as a case study — the mechanics below are exactly what shipped.

6.3 Background: how a fill used to be reported

When an order arrives, the engine's `_handle_new_order` handler hands it to the order book's `process()` method. Matching happens inside `_apply_fill`, which is called **once per resting order that the aggressor trades against**. Each call:

1. Creates a `Trade` record at that resting order's price.
2. Decrements the aggressor's `remaining_qty`.
3. Appends the **aggressor `Order` object** to the `events` list.

The crucial detail is in step 3: the engine appends the **same Python object** each time. After the sweep finishes, that one object only holds its **final** state — the total quantity filled and a `remaining_qty` of zero. The per-level breakdown (how much filled at each price) is not visible on the aggressor object afterwards; it only ever existed in the separate list of `Trade` records.

After matching, the publish loop walks `events` and emits an `order.fill` message for each fill. Because the aggressor object was appended N times (once per price level it consumed), a guard (a `seen` set in `_publish_dispatch`, tagged H5) ensures the engine still sends **exactly one** `order.fill` message for the aggressor instead of N duplicates — that part of the design was correct from the start and is unchanged today. What *that* single message reported was the bug:

```
# The bug: every fill message used the price of the LAST trade in the
# sweep, no matter how many price levels the order actually walked through.
fill_price = from_ticks(book.last_trade_price, order.symbol)
```

`book.last_trade_price` is the price of the **last** fill in the sweep — the deepest, least favourable level the order reached. That single price was then reported for the **entire** filled quantity, at every site in the engine that published a fill (the main order path, seeded quotes, combo legs, OCO legs, and auction fills all shared the same mistake).

The resting (passive) side was always correct

Every resting order the aggressor hits is a **distinct** object and is filled exactly once at its own price, so each passive participant has always received a correct per-price `order.fill`. The inaccuracy was limited to the **one** aggregated message sent to the **aggressing** order's gateway.

6.4 How the gateway turns a fill into a position

A trading gateway tracks positions with VWAP average-cost accounting (see Positions and P&L). On every `order.fill` it calls `_update_position(symbol, side, fill_qty, fill_price)`, which folds the fill into the running average:

$$\text{avg_cost}_{\text{new}} = \frac{\text{avg_cost}_{\text{old}} \times |q_{\text{old}}| + \text{fill_price} \times \text{fill_qty}}{|q_{\text{new}}|}$$

The gateway trusts `fill_price` as the price for the whole `fill_qty`. Before the fix, when the engine sent one aggregated fill at the last sweep price, the gateway booked the **entire** quantity at that single, least-favourable price — overstating cost basis for a buy sweep and understating proceeds for a sell sweep.

6.5 Worked example

Suppose the ask side of the book holds three resting sell orders:

Resting order	Price	Quantity
A	$10.00	100
B	$10.01	100
C	$10.02	100

A trader sends an aggressive **BUY 300 (MARKET)** order. It sweeps all three levels and three trades print:

Trade	Price	Quantity
1	$10.00	100
2	$10.01	100
3	$10.02	100

The **true** volume-weighted average price the buyer paid is:

$$\text{VWAP} = \frac{100 \times 10.00 + 100 \times 10.01 + 100 \times 10.02}{300} = \frac{3003}{300} = \mathbf{10.01}$$

Before the fix, the engine emitted a single aggregated fill using the last trade's price:

```
order.fill  →  fill_qty = 300   fill_price = 10.02 ✗   remaining_qty = 0
```

giving a reported average cost of $10.02 — a $0.01/share, $3.00 total, overstatement of the true $10.01 cost basis.

Today, the engine still sends exactly **one** `order.fill` message for the sweep (the message count did not change — see How it was fixed below), but the price on that message is the order's own VWAP across every level it touched:

```
order.fill  →  fill_qty = 300   fill_price = 10.01 ✓   remaining_qty = 0
```

which folds into the gateway's running average as exactly $10.01 — matching the true cost basis.

The three sellers (A, B, C) each receive, as they always did, a correct fill at their own price, and the `trade.executed` market-data / drop-copy feed carries all three trades at the right prices.

6.5.1 Direction of the (former) error

The last level reached in a sweep is always the **worst** price for the aggressor, so the old bug had a predictable sign:

Aggressor side	Last level is…	Effect on the aggressor's books (before the fix)
BUY	Highest price	Cost basis **overstated** → unrealized P&L understated
SELL	Lowest price	Proceeds **understated** → unrealized P&L understated

In both directions the old, buggy report was **pessimistic**: it made a sweep look like a slightly worse

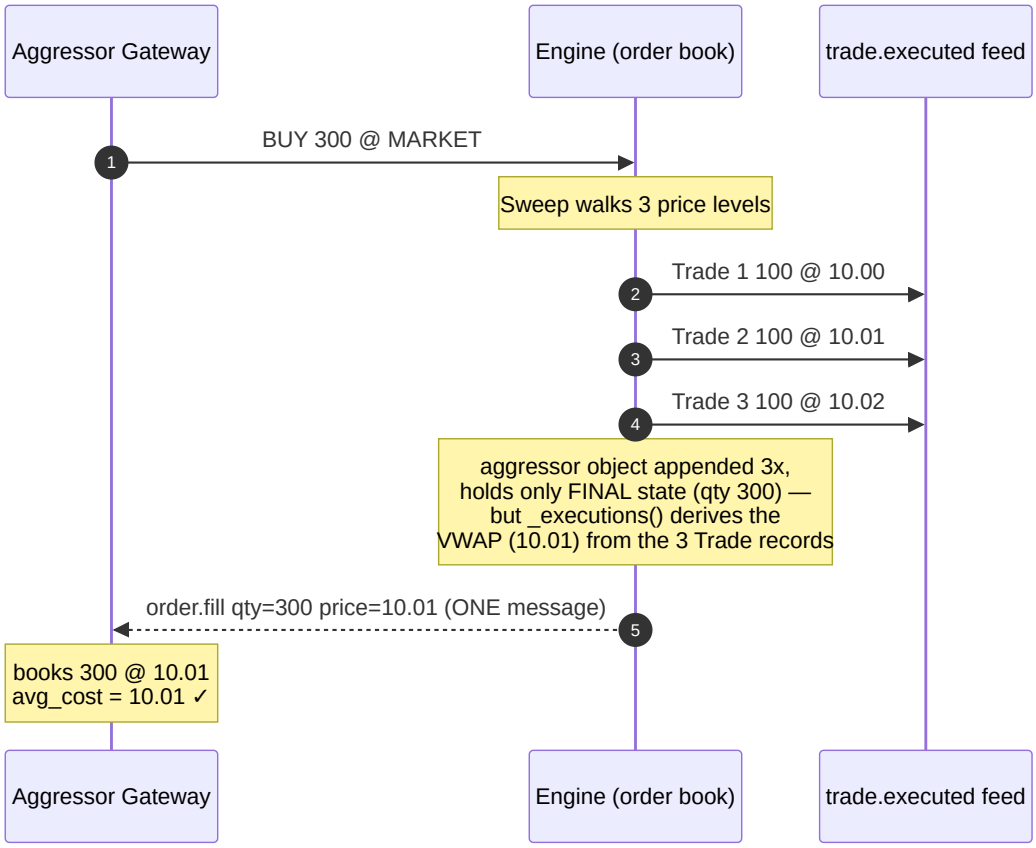
trade than it actually was. The magnitude equalled the spread between the sweep's average price and its last price, multiplied by the swept quantity.

Position quantity was never wrong

The dedup guard (the `seen` set in `_publish_dispatch`, H5) always guaranteed the quantity was counted **once**, so the net position size was always exact, even before the price fix. Only the **price** attached to that quantity was ever approximate.

6.6 Visualising the fix

The diagram below contrasts the **three real trades** generated inside the book with the **single fill** delivered to the aggressor's gateway, as it works today.



The same idea as plain ASCII:

```
Book side (asks)                Trades printed            Fill sent to buyer
----------------                --------------            ------------------
100 @ 10.00  ◄── swept ──►  Trade 1: 100 @ 10.00
100 @ 10.01  ◄── swept ──►  Trade 2: 100 @ 10.01   ===►  order.fill
100 @ 10.02  ◄── swept ──►  Trade 3: 100 @ 10.02           qty   = 300
                                                           price = 10.01  ✓
                            true VWAP = 10.01              (per-order VWAP)

Correct quantity (300) ✓     Correct trade prints ✓     Correct average price ✓
```

Before the fix, the last line of the gateway note read `price = 10.02 ✗ (last level only)` — the trade prints and the per-level book walk were identical; only the number handed to the gateway was wrong.

6.7 How it was fixed

The aggregation lives in the engine's hottest code path: `_publish_dispatch`, which every path that matches (new orders, quotes, combo children, OCO legs, amends, auctions) calls with the result of one `process()`. A fix that is correct but still cheap on that path mattered. The engine does **not** split the aggregated message into one-per-level (which would double or triple outbound message volume for every sweep); instead it keeps sending exactly one `order.fill` per order per `process()` call, and computes that message's price from the `Trade` records the book already produced:

```
@staticmethod
def _executions(trades: list[Trade]) -> dict[str, _Execution]:
    """Each order's execution in THIS dispatch, from *trades*, in one pass."""
    out: dict[str, _Execution] = {}
    for t in trades:
        agg = t.aggressor_side
        for oid, side in ((t.buy_order_id, "BUY"), (t.sell_order_id, "SELL")):
            ex = out.get(oid)
            if ex is None:
                ex = out[oid] = _Execution("TAKER" if side == agg else "MAKER")
            ex.qty += t.quantity
            ex.notional += t.price * t.quantity   # integer ticks
            ex.trade_ids.append(t.id)
    return out
```

The fill then reports `notional / (qty * 10 ** tick_decimals)`: the notional is summed in integer ticks and divided once, so the VWAP is correctly rounded. An order gets an `order.fill` only if it has an entry here, so there is no fallback price to reach for.

This is cheaper than it looks: `trades` for a single dispatch is small (one entry per price level swept, rarely more than a handful), so building the per-order map costs one pass over data the book already produced — no additional allocation on the matching path itself, and **no change** to how many messages a gateway receives per order. The dedup guard (H5) that ensures exactly one `order.fill` per order is unchanged; only the *price* it carries is now correct.

Two properties of the message are worth stating precisely, because the aggregation could otherwise be misread:

- `fill_qty` is the quantity matched **in this dispatch**, not a running total. A resting order that is hit again later gets a second `order.fill` carrying only the new quantity; its running total is `quantity - remaining_qty`, and the engine keeps the running notional on the order itself.
- The fill carries the `trade_ids`, in match order, of the trades it aggregates, so a consumer can join a private fill to the public tape and recompute the VWAP itself.

Because no test asserted the specific (wrong) last-trade-price value, fixing this required no changes to the existing test suite — only new tests asserting that a multi-level sweep's fill price equals the true VWAP.