# Parser Refactor & Management Message Roadmap

# Parser Refactor & Management Message Roadmap

## Phase 0 — Prep (half‑day)

* \*\*Decide file paths & env key:\*\* `PARSER\_DICT\_YAML=runtime/data/parser\_dict.yaml`.
* \*\*Versioning:\*\* Add `dictionary\_version: "2025-08-15.1"` at top of YAML (will help debugging).
* \*\*Test corpus:\*\* Collect ~20 real messages (open + edits + management intents) for repeatable tests.

---

## Phase 1 — Pull dictionary into YAML (single source, no fallbacks)

\*\*Goal:\*\* Parser reads exclusively from YAML; code contains \*\*no embedded defaults\*\*.

1. \*\*Define schema (YAML)\*\*

dictionary\_version: "2025-08-15.1"

symbols:

aliases:

XAUUSD: ["XAU", "GOLD", "XAUUSD"]

keywords:

buy: ["buy", "long"]

sell: ["sell", "short"]

tp: ["tp", "take profit"]

sl: ["sl", "stop", "stop loss"]

tp\_open: ["tp open", "open tp"]

parsing:

entry\_patterns:

- '^@?\s\*(?P<entry>\d+(?:\.\d+)?)\s\*/\s\*(?P<worse>\d+(?:\.\d+)?)$'

- '^@\s\*(?P<entry>\d+(?:\.\d+)?)$'

tp\_line: '^(?:tp)\s+(?P<price>open|\d+(?:\.\d+)?)$'

sl\_line: '^(?:sl|stop(?:\s+loss)?)\s+(?P<price>\d+(?:\.\d+)?)$'

defaults:

default\_symbol: "XAUUSD"

default\_num\_legs: 5

require\_symbol: false

require\_price: true

2. \*\*Loader + validator\*\*

- Reads YAML, validates against a Pydantic schema, compiles regexes.

- No fallbacks: invalid/missing YAML → fail fast with error.

3. \*\*Parser refactor\*\*

- Replace in‑code constants/regex/aliases with reads from dict object.

- External behavior identical to baseline.

4. \*\*Tests\*\*

- Use ~20‑message corpus to compare JSON output to baseline.

5. \*\*Acceptance\*\*

- YAML present → parser runs.

- No hardcoded dictionaries remain.

- 100% parity vs baseline corpus.

---

## Phase 2 — Quoted message understanding

\*\*Goal:\*\* Resolve the “original OPEN” that a management message refers to.

1. \*\*Reference extraction module\*\*

- Resolve in order: `reply\_to\_msg\_id` → deep‑link → (later) text fingerprint.

- Output group key: `GK = f"OPEN\_{source\_msg\_id}"`.

2. \*\*Minimal DB indexing\*\*

- `signals(source\_msg\_id PK, chat\_id, msg\_ts, group\_key, text\_hash NULL)`.

- On each OPEN → insert row.

- ACK handler: `legs\_index(group\_key, leg\_tag, symbol, volume, broker\_ticket, status)`.

3. \*\*APIs\*\*

- `resolve\_group\_key(event) -> GK|None`

- `list\_open\_legs(GK) -> List[LegRef]`

4. \*\*Tests\*\*

- Reply‑based management resolves correct GK and enumerates live legs.

5. \*\*Acceptance\*\*

- Quoted/replied messages resolve GK.

- Can list open legs for GK.

---

## Phase 3 — Expand dictionary for more entry definitions

\*\*Goal:\*\* Let YAML define additional open signal variations.

1. \*\*Extend YAML\*\*

parsing:

entry\_patterns:

- '^@?\s\*(?P<entry>\d+(?:\.\d+)?)\s\*/\s\*(?P<worse>\d+(?:\.\d+)?)$'

- '^@\s\*(?P<entry>\d+(?:\.\d+)?)$'

- '^(?:entry|buy|sell)\s\*@\s\*(?P<entry>\d+(?:\.\d+)?)$'

2. \*\*Parser changes\*\*

- Iterate patterns in order; stop at first match.

3. \*\*Tests\*\*

- Existing corpus unchanged.

- Add new variants and verify.

4. \*\*Acceptance\*\*

- Old signals parse identically.

- New variants parse as intended.

---

## Phase 4 — Add trade management message types

\*\*Goal:\*\* YAML‑driven verbs for break‑even, risk‑free, move‑SL.

1. \*\*Extend YAML\*\*

management:

break\_even:

triggers: ["break even", "breakeven", "be"]

behavior: { type: "move\_sl\_to\_entry", cushion: 0.0 }

risk\_free:

triggers: ["risk free", "risk-free"]

behavior: { type: "move\_sl\_to\_entry", cushion: 0.10 }

move\_sl:

pattern: '^move (?:sl|stop) to (?P<price>\d+(?:\.\d+)?)$'

behavior: { type: "move\_sl\_to\_price" }

scope:

require\_reference: true

target: "group"

2. \*\*Builder\*\*

- `build\_management\_actions(GK, mgmt\_intent) -> List[MODIFY]`

- Resolve GK, load legs from index, apply behavior, emit per‑leg MODIFY.

3. \*\*Guards\*\*

- Require reference; no GK → ERROR.

- GK resolves but no open legs → ERROR.

4. \*\*Tests\*\*

- Each verb resolves and produces correct MODIFYs.

5. \*\*Acceptance\*\*

- Management verbs work without affecting OPEN flow.

- Clear errors for missing/ambiguous refs.

---

## Sequencing

* Phase 0–1: 0.5–1 day
* Phase 2: 0.5 day
* Phase 3: 0.5 day
* Phase 4: 0.5–1 day