MarketSwarm — Model Maker (mmaker)

Instructions & Operating Charter

---

1. Purpose (Non-Negotiable)

Model Maker exists to transform market facts into interpretable structure.

It does **not**:

- ingest raw feeds
- execute trades
- predict outcomes
- issue signals
- optimize P&L

It **does**:

- consume verified market state
- construct domain models
- preserve ambiguity
- publish structure for downstream interpretation

> **mmaker answers "what is true right now?" — never "what should be done?"**

---

2. Inputs (Hard Constraints)

mmaker may only consume **authoritative state** published by upstream services.

Required Inputs

- massive:spot
- massive:chain

Optional Inputs (future-safe)

- volatility surface
- futures basis
- cross-asset proxies

Prohibited Inputs

- UI events
- user preferences
- execution feedback
- P&L
- signals from other models

> **No model may ingest another model's output unless explicitly designated as a terminal aggregator.**

---

3. Outputs (Strictly Declarative)

mmaker publishes **models**, not advice.

Canonical Outputs

- massive:model:gex
- massive:model:volume_profile
- massive:model:convexity
- massive:model:heatmap
- massive:model:market_mode
- massive:model:vix_regime
- massive:model:liquidity_intent

Each output must:

- be self-contained
- include timestamp + source
- declare its domain
- be reproducible from inputs
- avoid implicit normalization

---

4. Model Design Rules (Critical)

4.1 One Model → One Question

Every model must answer **exactly one question**.

Examples:

- GEX → *Where are dealers forced to hedge?*
- VP → *Where has the market accepted price?*
- Heatmap → *Where is convexity priced and repricing?*
- Market Mode → *Is structure absorbing or amplifying movement?*

If a model answers two questions, it must be split.

4.2 No Cross-Model Dependency Loops

Allowed:

```
spot + chain → model
```

Forbidden:

```
model → model → model → feedback
```

Terminal models (dashboards, AI narrators) may *interpret* multiple models — **mmaker may not**.

4.3 Models Must Be Interpretable Without UI

If a model:

- requires color to be understood
- relies on layout
- assumes user context

...it is **not a model**, it is a visualization artifact.

mmaker produces **semantic data**, not visuals.

---

5. The Tile Doctrine (Convexity Core)

5.1 Tile = Atomic Convexity Object

A tile represents:

- Strategy class
- Strike (Y)
- Width (X)
- DTE (page)
- Debit price
- Time derivative (optional)

Tiles must be:

- net debit

- long convexity
- strike-anchored
- real (market-priced)

No synthetic parity tricks.

No credit structures.

No normalization.

5.2 Placement Rules (Invariant)

- **Y-axis** → Strike
- **X-axis** → Width
- **DTE** → Discrete pages

Instrument rules:

- Calls & puts exist at all strikes
- Spreads:
    - Above spot → calls
    - Below spot → puts

Anchor rules:

- Single-leg → strike

- Vertical → long strike
- Butterfly → center strike

---

6. Time Discipline

mmaker operates on **structural cadence**, not tick cadence.

- Spot ingestion → fast
- Chain snapshots → 15–30s
- Models → event-triggered or scheduled
- No model recomputation without input change

Models must declare:

- update cadence
- staleness tolerance
- decay semantics

---

7. Silence Is a Feature

mmaker **must not publish** when:

- inputs are incomplete

- markets are illiquid
- data is stale
- assumptions are violated

> **Absence of a model is superior to a wrong model.**

Downstream consumers must be allowed to detect “no signal” cleanly.

---

8. Determinism & Reproducibility

Given:

- same spot
- same chain
- same parameters

mmaker must:

- produce identical outputs
- avoid randomness
- avoid hidden state

Historical replay must be possible.

---

9. Domain Separation

mmaker contains **no strategy logic**.

Strategy selection belongs to:

- playbooks
- coaches
- humans

mmaker exposes **structure**, not tactics.

---

10. Relationship to AI / Convexity (Vexy)

AI systems:

- may read mmaker outputs
- may narrate models
- may highlight tension
- may explain implications

AI systems **may not**:

- mutate models
- rank trades
- infer execution
- back-propagate decisions

mmaker is **authoritative**.

AI is **interpretive**.

---

11. Failure Modes to Guard Against

Explicitly test against:

- indicator creep
- implicit bias encoding
- “bullish/bearish” leakage
- normalization drift
- signal temptation
- over-smoothing

When in doubt:

> **Reduce output, not complexity.**

---

12. Success Criteria

A successful mmaker:

- Makes expert traders *slower*, not faster

- Makes regime shifts obvious in hindsight
- Feels boring when markets are boring
- Becomes vivid only under stress
- Never tells the user what to do

If users say:

> "It didn't give me a trade, but it showed me why the trade failed"

— it's working.

Final Note

If I were building **MarketSwarm's model maker**, I would treat it as:

> **A structural truth engine with zero ambition.**

Ambition belongs to humans.

mmaker's job is simply to **never lie**.

If you want, next we can:

- turn this into a literal README / charter
- define test cases for model integrity
- or specify what *should never be added*, even under pressure