## 1. Tile: Definition (Non-Negotiable)

A **Tile** is the **atomic, irreducible convexity object** produced by mmaker.

It represents **one real, market-priced convex structure** at a specific location in convexity space.

A Tile answers **exactly one question**:

> *“What is the price and exposure of long convexity at this strike, width, and DTE — right now?”*

No inference. No ranking. No aggregation.

This is consistent with the Tile Doctrine in the charter .

---

2. Tile Invariants (Hard Rules)

Every Tile **must** satisfy all of the following:

Structural

- **Net debit**
- **Long convexity**
- **Strike-anchored**
- **Real market-priced**
- **Single expiry**
- **Single strategy class**

Prohibited

- ❌ Credit structures
- ❌ Synthetic parity (no call/put mirroring tricks)
- ❌ Normalization or scaling
- ❌ Embedded signals or labels

Violation of *any* invariant invalidates the Tile and must suppress publication.

---

3. Tile Coordinate System (Canonical)

The Tile exists in a **3D discrete convexity lattice**:

Dimension	**Meaning**	**Rule**
`Y`	`Strike`	`Absolute strike`
`X`	`Width`	`Distance between long legs`
`Z`	`DTE Page`	`One expiry only`

Placement rules are invariant per the charter :

- Above spot → **calls**
- Below spot → **puts**
- Single-leg → strike anchored
- Vertical → long strike anchored
- Butterfly → **center strike anchored**

---

4. Tile Strategy Classes (v1)

Tiles do **not** infer strategies — they **declare structure**.

Initial supported classes (explicit, finite):

```
single_call
single_put
vertical_call
vertical_put
butterfly_call
butterfly_put
```

Each class has a **fixed leg topology**. No dynamic leg counts.

---

5. Tile Schema (Authoritative)

This is the **minimum canonical schema**. Anything more belongs in a downstream model.

```json
{
  "tile_id": "SPX-20240118-4500-BFLY-50",
  "underlying": "SPX",
  "expiry": "2024-01-18",
  "dte": 0,

  "strategy": "butterfly_call",

  "strike_anchor": 4500,
  "width": 50,

  "legs": [
    { "type": "call", "strike": 4450, "side": "long" },
    { "type": "call", "strike": 4500, "side": "short", "qty": 2 },
    { "type": "call", "strike": 4550, "side": "long" }
  ],
  "debit": 1.35,
  "multiplier": 100,

  "greeks": {
    "delta": 0.02,
    "gamma": 0.18,
    "vega": 0.04,
    "theta": -0.22
  },

  "timestamp": "2024-01-18T14:32:05Z",
  "source": "massive:chain"
}
```

Notes

- greeks are **optional but allowed**
- No implied payoff curves
- No probability language
- No normalization across tiles

---

6. Tile Lifecycle (Deterministic)

Tiles are **ephemeral structural facts**, not persistent opinions.

Creation

Triggered only when:

- Chain snapshot changes
- Spot invalidates placement rules

- Expiry rolls

Mutation

❌ Forbidden

Tiles are **immutable**.

Expiry

A Tile **dies silently** when:

- Its expiry passes
- Required legs are no longer quoted
- Liquidity rules fail

Silence is success.

---

7. Tile Factory Responsibility

The Tile Factory (tile_factory.py) must:

1. Consume **only**:
    - massive:spot
    - massive:chain
2. Validate invariants **before construction**
3. Emit **zero tiles** if assumptions fail

4. Produce **reproducible output** given identical inputs

This aligns with determinism rules in the charter .

---

8. Relationship to Heat Map Model

Important separation:

- **Tile** → atomic convexity fact
- **Heat Map** → spatial aggregation of Tiles

Tiles **do not know**:

- Their color
- Their intensity
- Their relative importance

That belongs strictly to the **Convexity Heat Map model**, downstream.

---

9. Failure Modes to Explicitly Guard

At the Tile level, unit tests must catch:

- Debit flipping to credit
- Mis-anchored butterflies

- Width drift
- Strike mirroring above/below spot
- Missing legs
- Greek sign inversions
- Accidental normalization

If in doubt:

> **Drop the Tile.**