Lemon Markets v1: The Modular Protocol for Synthetic Asset Perpification

Lemon Markets Team

hello@lemonmarkets.finance

January 20, 2026

Abstract

Lemon Markets is a *modular*, decentralized protocol enabling the synthetic *"perpification"* of arbitrary value-bearing financial instruments. By architecturally decoupling market execution from protocol-wide liquidity, Lemon Markets facilitates the creation of perpetual exposure for any asset with a verifiable price feed. The protocol utilizes a tiered liquidity model comprising a protocol-owned Buffer Pool and a provider-funded Insurance Pool to achieve high capital efficiency while isolating systemic tail-risk. Every position is encapsulated as a unique, non-fungible token (NFT), enabling secondary market transferability and native composability within the DeFi ecosystem.

2. Disclaimer and Scope of Interpretation

2.1 Non-Financial Advice

This document constitutes a technical protocol specification and does not serve as financial, investment, legal, or regulatory advice. The Lemon Markets protocol is a decentralized software system.

2.2 Risk Acknowledgment

Participants (Traders and Liquidity Providers) acknowledge that:

- **Market Risk**: Synthetic assets are subject to extreme volatility and price gaps.
- **Smart Contract Risk**: Software vulnerabilities may exist; the protocol is under active development.
- **Oracle Risk**: The protocol is dependent on external data signatures; inaccuracies or liveness failures can lead to absolute loss of funds.
- **Liquidity Risk**: During extreme market-wide drawdown, the Insurance Pool may be depleted, resulting in capped payouts or insolvency.

3. Problem Statement and Motivation

3.1 Liquidity Fragmentation

Traditional Perp DEXs require deep, native liquidity for every asset pair. This fragments capital and limits the variety of tradable assets to high-cap cryptocurrencies.

Our team has decades of combined experience working in and with DeFI protocols, seeing the need for a defragmentalized-capital protocol for synthetic asset exchange.

3.2 Inventory Constraints

Borrow-based margin models are limited by physical asset supply. If a protocol has no underlying "token A" to lend, it cannot support a short position on token A.

3.3 The Lemon Paradigm: Modular Perpification

Lemon Markets introduces a "**Dumb Market, Smart Manager**" model. By separating the logic into specialized modules, it achieves:

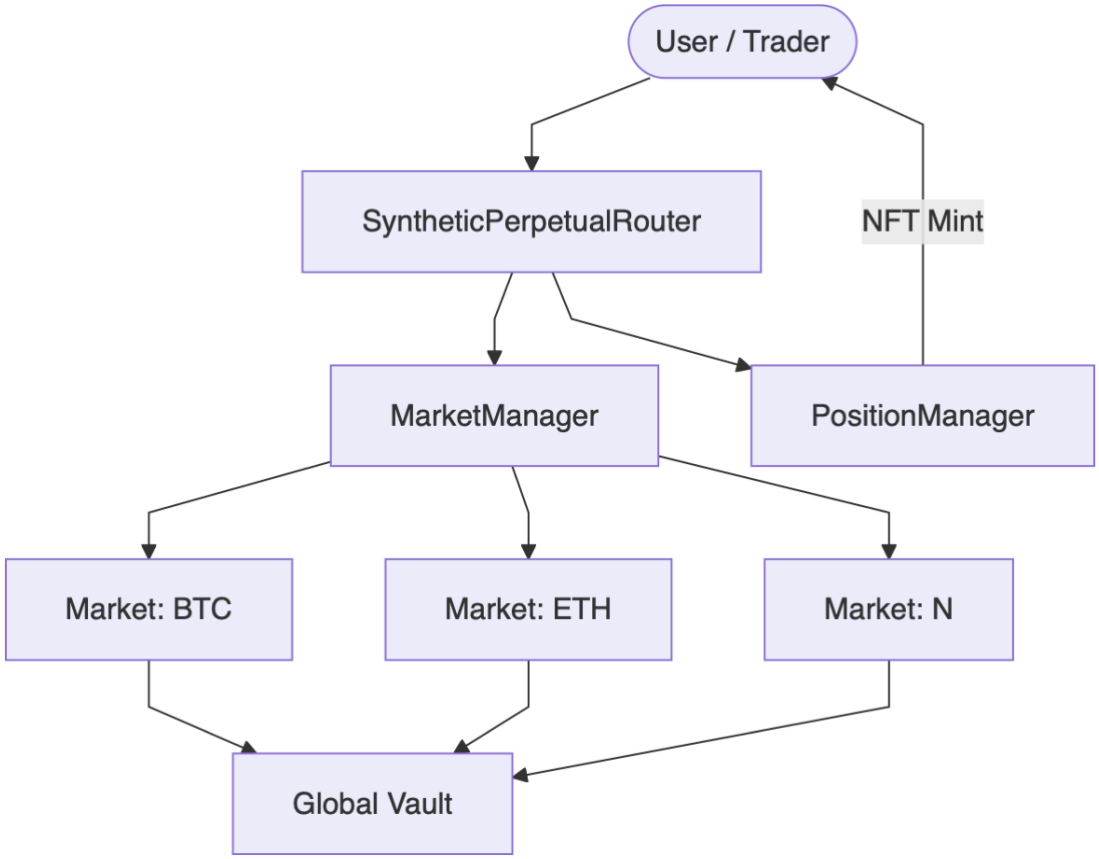
1. **Permissionless Growth**: New Market contracts can be deployed for any ticker without modifying the core liquidity logic.
2. **Pure Synthetics**: No borrowing, no inventory, no interest rates.
3. **Tiered Backstops**: Multi-layer protection for Liquidity Providers (LPs).

Lemon Markets is built to ensure that any synthetic financial instrument (or asset) can be exchanged or traded as long as there are willing participants to that trade and the protocol achieves this through its organized and decentralized manager system.

4. Modular Architecture

The protocol is "modular" because it separates execution, risk management, and liquidity into independent, interoperable contracts. This allows for horizontal scaling where new assets are added by deploying new instances of a specific module rather than upgrading the entire system.

4.1 Component Decomposition



4.2 Functional Layers

- **The Orchestrator (SyntheticPerpetualRouter):** The single user entry point. It manages the handshakes between the MarketManager and PositionManager.
- **The Registry (MarketManager):** The "Brain" of the protocol. It maintains a registry of all active Market contracts and coordinates the PnL/Fee waterfall across the Vault.
- **The Market Instance (Market):** A lightweight contract holding state for a specific asset pair. It tracks $Margin$, $OpenInterest$, and localized liquidity.
- **The Execution Engine (PositionManager):** An ERC-721 contract that manages the position lifecycle and ownership.
- **The Liquidity Layer (Vault):** Manages the Insurance Pool (LP funds) and Buffer Pool (protocol reserves).

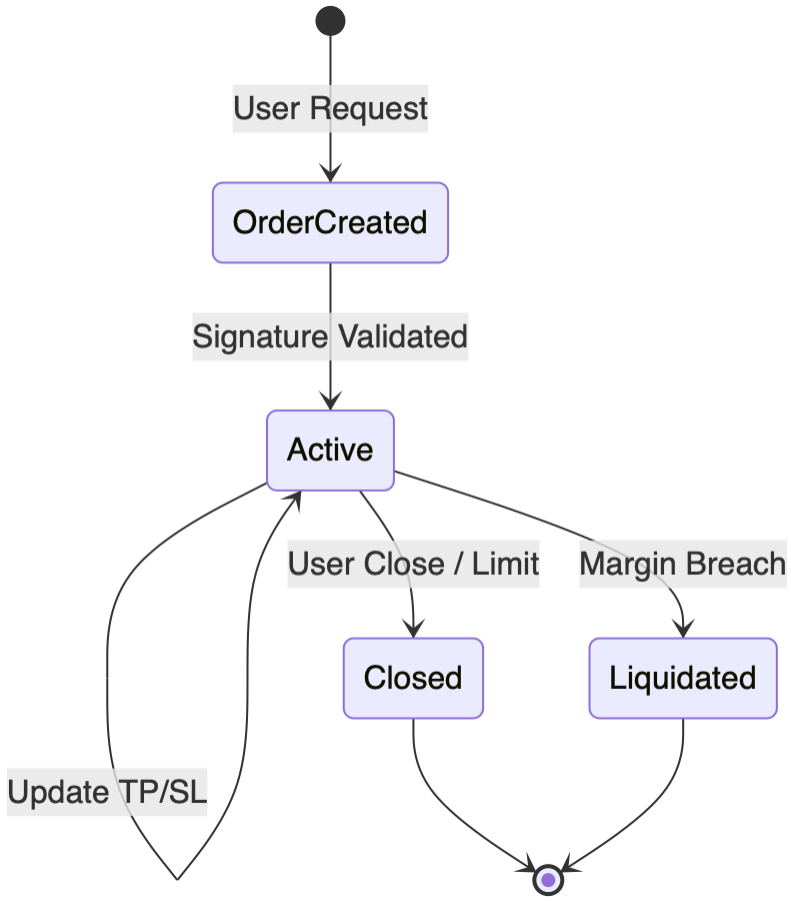
5. Market and Position Model

5.1 Position Representation

Positions on Lemon Markets are stored as non-fungible data structures containing the following immutable and mutable state:

	Description
Trader	The wallet owner of the position NFT.
Token Symbol	The market identifier (e.g., "BTC", "ETH").
Direction	Boolean flag indicating Long (True) or Short (False).
Margin	The current collateral backing the position (in collateralToken).
Leverage	The leverage multiplier applied to the margin.
Entry Price	The aggregate entry price of the position.
Liquidation Price	The calculated price at which the position is flagged for liquidation.
Take Profit / Stop Loss	Optional trigger prices for automated closing.
Open Timestamp	The block time when the position was created.
Is Active	Status flag determining if the position contributes to Open Interest.

5.2 Position Lifecycle



6. Trading Mechanics and Economic Flows

6.1 Profit and Loss (PnL) Implementation

PnL is calculated linearly relative to price movement with great care in implementation is taken to ensure precision by performing multiplication before division.

$$PnL = Size\ x\ \Delta P.entry$$

$$Where\ Size = Margin\ x\ Leverage$$

6.2 The Fee Waterfall and Distribution

The protocol implements a tiered fee structure to ensure sustainable operations and LP compensation.

1. **Opening Fee (*fO*)**:
    - Rate: Defined by OPENING_FEE_RATE (20 bps).
    - Logic: $fO = Margin\ x\ Rate$.
    - Distribution: Deducted from initial margin and transferred to the TreasuryWallet.
2. **Referral Fee (*fR*)**:
    - Rate: 10% of *fO* (DEFAULT_REFERRAL_FEE_RATE = 1000 bps of Opening Fee).
    - Logic: If a valid referrer is provided during openPosition, a percentage of the Opening Fee is redirected to the referrer.
3. **Closing Fee (*fC*)**:
    - Rate: Defined by CLOSING_FEE_RATE (200 bps).
    - Logic: Applied only to profitable trades ($PnL > 0$).
4. **Duration Fee (*fD*)**:
    - Rate: Hourly rate (*bps*) per market.
    - Logic: $fD = Margin\ x\ Rate\ x\ TimeOpen$
    - Distribution: Collected during closePosition and used to replenish the Vault liquidity.

7. Risk Management and Exposure

7.1 Dynamic Liquidation

A position is eligible for liquidation when the maintenance margin is breached. The protocol uses an 80% threshold ($LIQUIDATION\_THRESHOLD = 8000$).

$$\text{P.}(liq,\ Long) = pE\ x\ (1 - 0.8/L)$$

$$\text{P.}(liq,\ Short) = pE\ x\ (1 + 0.8/L)$$

7.2 Exposure Throttling and Skew Management

Each Market instance tracks its own Long/Short skew. The MarketManager enforces global limits by checking the IVault.maxPositionDivider().

Accounting Invariants:

- totalLongMargin: Cumulative margin of all active long positions.
- totalShortMargin: Cumulative margin of all active short positions.
- realLiquidity: Total collateral tokens physically present in the Market contract.

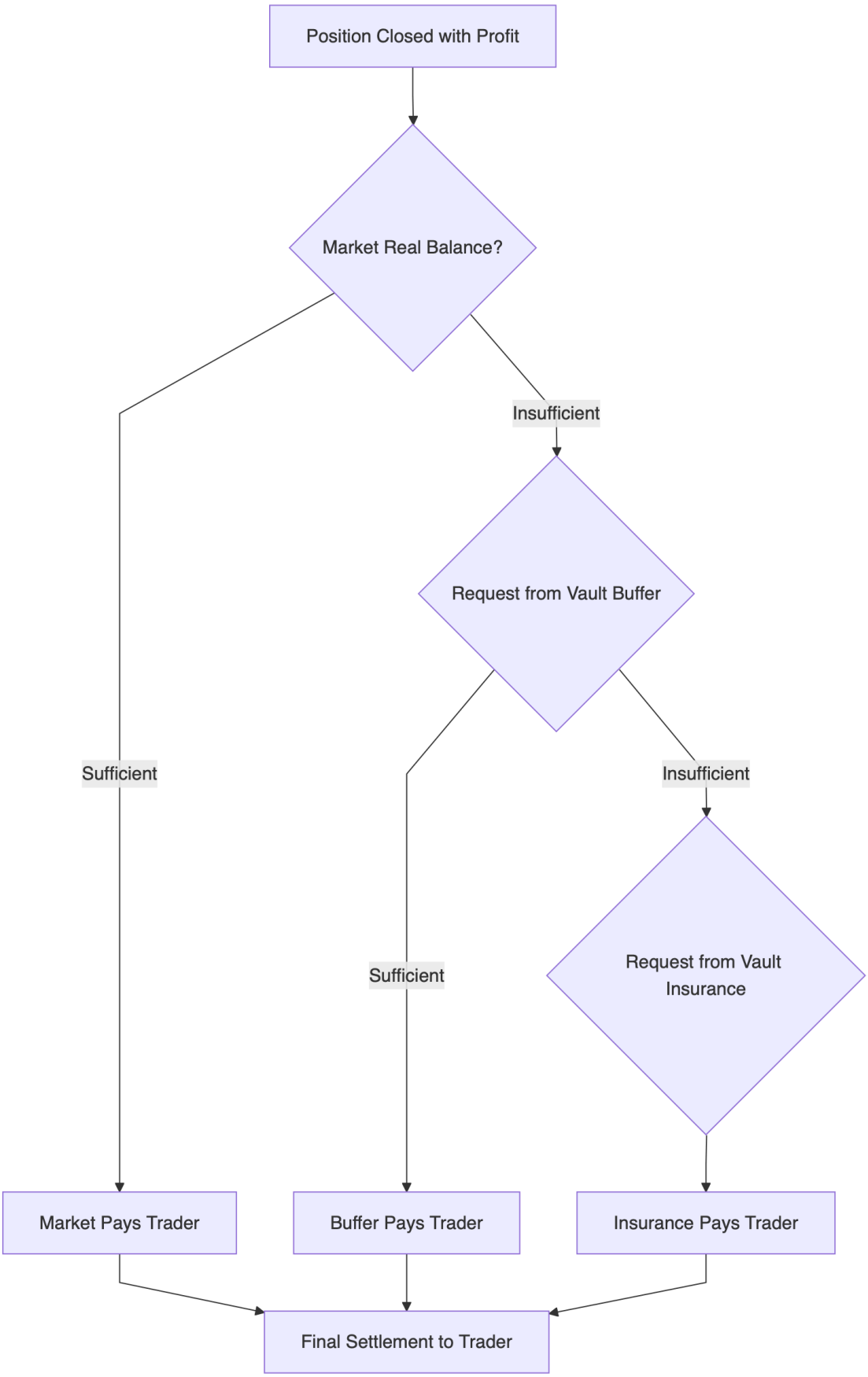
8. Liquidity Architecture and Solvency

8.1 Dual-Tier Payout Waterfall

Lemon Markets ensures trader payouts through a hierarchical liquidity structure implemented in MarketManager.removePositionMarginAndPayout.

Settlement Execution Flow:

1. **Local Settlement**: The Market contract first attempts to pay the trader from its realLiquidity.
2. **Vault Rebalancing**: If realLiquidity < payout, the MarketManager triggers executeVaultRequest.
    - Priority 1 (Buffer): Request from Vault's buffer liquidity.
    - Priority 2 (Insurance): If Buffer is insufficient, request from Vaul's insurance liquidity.
3. **Loss Recycling**: When a trader takes a loss, the loss amount (up to the full margin) is sent to the Vault via executeVaultBufferDeposit or executeVaultInsuranceDeposit.


Position Closed with Profit
Market Real Balance?
Sufficient
Insufficient
Request from Vault Buffer
Sufficient
Insufficient
Request from Vault Insurance
Market Pays Trader
Buffer Pays Trader
Insurance Pays Trader
Final Settlement to Trader

8.2 The LP Share Model

Liquidity Providers (LPs) receive VaultToken (ERC-20) representing their proportional claim on the Insurance Pool.

- **Share Valuation**: The share price increases as fees (Duration Fees, Closing Fees) and Liquidation Surpluses are added to the pool.
- **Risk Profile**: LPs assume the role of the "Counterparty of Last Resort." In extreme market conditions where Buffer Pool is exhausted, LPs absorb winning trader PnL.

9. Smart Contract Inventory and Constants

9.1 Contract Responsibilities

- **SyntheticPerpetualRouter**: Manages and interacts with clients (native or not), and coordinates complex actions like modifying positions and closing limit orders.
- **MarketManager**: Stores the Market registry. It is the only contract authorized to call executeVaultRequest on the Vault.
- **PositionManager**: Owns the Position mapping. It performs EIP-712 style signature verification for oracle data. This is important for secure onchain verification of offchain oracle data.

9.2 Key Protocol Constants

Constant	Value	Description
MIN_TRADING_MARGIN	1 * 10^6	Minimum collateral required to open a position (e.g., 1 USDC).
MAX_TRADING_MARGIN	1,000,000 * 10^6	Maximum collateral per position.
OPENING_FEE_RATE	20 (0.2%)	Fee charged on opening a position.
KEEPER_FEE_RATE	10 (0.1%)	Fee paid to keepers for executing limit/liquidations.

10. Security and Threat Model

10.1 Trust Boundaries and Admin Authority

- **Oracle Integrity**: The protocol trusts a centralized `adminSigner`. All price data must be signed with a valid nonce to prevent replay attacks.
- **Emergency Controls**: In the case of an emergency the protocol has a designated owner to avert risks. The `Owner` can call `pause()` on the `Router` to halt all trading activity.
- **Upgradeability**: The system uses the UUPS (Universal Upgradeable Proxy Standard). Logic shifts are possible but restricted to the `ProxyAdmin`.

10.2 Attack Vectors and Mitigations

- Front-Running: Prevented by the requirement of a recent oracle signature in the same transaction as the position request.
- Price Manipulation: Prevented by referencing the `adminSigner` which aggregates off-chain data, rather than relying on onchain CLOB/AMM prices which are easier to manipulate.

11. Protocol Scope, Constraints, and Evolution

11.1 Current Scope (V1)

In its current iteration (V1), Lemon Markets is restricted to:

- Stablecoin Standard: Internal accounting is strictly in the `collateralToken` (e.g., USDC).
- Synthetic Only: No physical delivery or spot interaction.
- Centralized Oracles: Reliance on the `adminSigner` for all market prices.

11.2 Hard Constraints

1. **No Borrowing**: Positions are not funded by lenders. Leverage is a synthetic multiplier backed by protocol-wide liquidity.
2. **Fixed Settlement Ticks**: Settlement is only possible at prices signed by the authorized oracle.
3. **Maximum Leverage**: Hard-coded limit of 100x.

4. **Zero-Sum-Plus-Fees**: Every profit is compensated by either another trader's loss or the Vault, minus protocol fees.

11.3 Active Research Areas

- Asymmetric Funding Rates: Transitioning to dynamic funding rates $(Long - Short)\, x\, Rate$ to incentivize balance when one side dominates the market.
- Decentralized Oracle Networks (DONs): Moving towards a multi-signer threshold model or integrating Pyth/Chainlink.
- Cross-Chain Settlement: Implementing a bridge-aware PositionManager to allow users to move their position NFT between EVM-compatible chains.