Orderly Permissionless Listing System

Technical Proposal & Reference Implementation

Prepared by: Arthur DEX (arthur-orderly) **Date:** February 8, 2026 **Version:** 1.0

Executive Summary

We've built a complete reference implementation for Orderly's permissionless listing system based on the published specifications. This includes production-ready smart contracts, a full frontend, and comprehensive test coverage. We propose this as either the official implementation or a starting point for Orderly's engineering team.

Problem Statement

- Orderly currently requires a manual listing process for new perpetual futures pairs
- This limits the speed of new market launches and creates a bottleneck
- Permissionless listing enables any qualified entity to list new perp markets, dramatically expanding Orderly's market coverage

Solution Architecture

Smart Contracts (Arbitrum)

1. **ListingRegistry** — UUPS upgradeable core registry. Manages listing lifecycle (Pending → Active → Suspended → Deactivated). Role-based access control (LISTER, ADMIN, ORACLE).

2. **ListingStake** — Staking mechanism requiring $50,000 USDC minimum with 6-month lock period. Implements three slashing tiers from the spec: 100% (rug pull), 50% (abandonment), variable 0-50% (manipulation).

3. **FeeVault** — Fee collection and distribution with configurable splits (default: 50% protocol, 30% lister, 20% insurance fund).
4. **SlashingOracle** — Decentralized governance for slashing decisions. 3-of-5 multisig with mandatory 48-hour timelock between proposal and execution.
5. **ListingLib** — Pure library implementing all parameter derivation from the spec, including leverage caps by market cap tier (<$30M→5x, $30-100M→10x, >$100M→20x), IMR/MMR calculation, funding rate bounds.

Frontend Application

- 4-step listing wizard matching the Configuration UX Flow spec
- Real-time dashboard with listing status, volume, and risk metrics
- Risk monitor with automated alerts for slashing threshold proximity
- Admin panel for multisig governance participants
- WalletConnect integration on Arbitrum

Key Design Decisions

- **UUPS Proxy Pattern:** Allows contract upgrades as the system evolves, critical for a V1 launch
- **Separation of Concerns:** Staking, registry, fees, and governance in separate contracts for security isolation
- **Conservative Defaults:** Minimum $50K stake, 6-month lock, 48hr timelock — all erring on the side of security
- **Spec Compliance:** All parameter derivations (leverage caps, IMR/MMR, fee bounds) follow the published specifications exactly

Parameter Derivation

Key formulas implemented in `ListingLib.sol`:

- **Leverage:** `min(tier_max, floor(1/base_imr))`
- **IMR tiers:** <$30M mcap → 20% (5x), $30-100M → 10% (10x), >$100M → 5% (20x)
- **MMR:** `IMR / 2` (with 6% exception for certain tiers)
- **Fee bounds:** Taker markup 0-5bps, Maker markup 0-2bps
- ~20 parameters auto-derived from just 5 inputs (token address, target leverage, fee preferences, market cap, desired spread)

Security Considerations

- All contracts follow Checks-Effects-Interactions pattern
- Reentrancy guards on all external calls
- Role-based access with granular permissions
- Timelock on all slashing actions prevents rushed/malicious slashes
- USDC approval pattern (approve → transferFrom) for stake deposits
- Comprehensive test suite: 20 tests covering edge cases, access control, math precision

Integration with Orderly Backend

How the on-chain layer connects to Orderly's existing infrastructure:

1. Lister stakes on-chain → `ListingCreated` event emitted
2. Orderly backend listens for events via indexer
3. Backend validates parameters and enables the trading pair on the orderbook
4. Ongoing: backend monitors MM activity, reports to SlashingOracle if abandonment detected
5. Fee markups applied at the matching engine level, collected on-chain via FeeVault

Test Results

- **20 unit tests**, all passing
- **Coverage:** listing lifecycle, slashing scenarios (all 3 tiers), fee distribution math, governance flow, access control, parameter validation
- **Gas estimates** for key operations included in test output

Deliverables

1. ✅ Smart contracts (Foundry project, Solidity 0.8.24)
2. ✅ Full test suite (20 tests)
3. ✅ Frontend application (deployed to Vercel)
4. ✅ Deployment scripts for Arbitrum
5. ✅ Documentation

Open Questions / Gaps in Current Spec

1. **Market Maker Requirement (Section 3.3)** — Spec marks this as TBD. Our implementation supports an optional MM account requirement but doesn't enforce specific MM behavior. Recommend: require lister to designate an MM account with minimum quote obligation.

2. **Broker-Specific Configurations (Section 6)** — Currently empty in spec. Each broker may want custom parameters. Our system supports per-listing parameter overrides.

3. **Oracle Data Source** — For market cap tier determination, need a reliable oracle (Chainlink, Pyth, or Orderly's own price feeds). Current implementation uses admin-set values.

4. **Cross-Chain Support** — Current implementation is Arbitrum-only. Multi-chain deployment would need cross-chain messaging (LayerZero, CCIP).

Proposed Next Steps

1. **Review:** Orderly engineering team reviews contracts and architecture
2. **Audit:** Professional smart contract audit (recommend Trail of Bits or OpenZeppelin)
3. **Testnet Deploy:** Deploy to Arbitrum Sepolia for integration testing
4. **Backend Integration:** Connect on-chain events to Orderly's matching engine
5. **Mainnet Launch:** Target March 2026 per Orderly's roadmap

Links

- **GitHub:** github.com/arthur-orderly/permissionless-listing
- **Frontend Demo:** orderly-permissionless-listing.vercel.app
- **Mockup (earlier version):** permissionless-listing-mockup.vercel.app
- **Arthur DEX:** arthurdex.com