Lattica[‡]

Protocol v1.0

William Flanders[*‡] Stephen Flanders[†‡]

March 2026

Updated June 2026

Abstract

In this paper, we introduce a decentralized protocol: Lattica. Serving as the liquidity layer of informational finance, Lattica offers margin and credit against prediction-market exposure: position holders draw liquidity against ERC1155 outcome tokens they already hold, and traders open amplified exposure to a market outcome against posted margin. Instead of relying solely on liquidations, Lattica's calibrated risk engine, WARHORSE, protects capital providers by calculating and charging a state-conditional premium at origination.

[*]william@lattica.fi
[†]stephen@lattica.fi
[‡]https://lattica.fi

1 Introduction

1.1 Motivation

By allowing users to put their money where their mouth is, prediction markets produce price signals that aggregate dispersed beliefs into a common forecast (Hanson, 2003; Arrow et al., 2008). This informational function is already being incorporated into political science and polling theory (Clinton and Huang, 2025), economics[1], and media[2]. We expect their influence to deepen as these platforms expand into more specialized subjects and broader cultural domains.

Yet prediction markets today operate under a structural constraint that limits their potential. Market participation requires traders to lock capital into contingent claims with capped upside and no interim yield. For markets with resolution horizons spanning weeks, months, or years, this imposes significant opportunity costs. The result is thin books, wide spreads, and shallow depth, particularly in the longer-dated markets where forecasting value is arguably greatest.

The platforms themselves are clearly aware of this problem, and they have attempted a range of solutions: paying position holders and market makers "holding" and "liquidity" rewards from treasuries, contracting external market makers, and running internal market making strategies. These programs are uniformly capital intensive. Holder and liquidity rewards require treasury funds, external market makers require rebates, and internal strategies require the platform to absorb losses. More importantly, they treat the symptom rather than the cause. Locked capital remains inefficient, reward rates are set without rigorous connection to market quality, and prediction markets continue to suffer from illiquidity.

A protocol that lets traders borrow against locked positions would address the cause directly, and the case for building one extends beyond simple capital efficiency.

Forecast accuracy does not scale with raw volume; it scales with the presence and capital depth of informed traders[3]. When those traders' capital is trapped in existing positions, they cannot redeploy it to correct mispricings elsewhere. Freeing that capital to circulate across markets increases the effective reach of the informed flow that actually drives price discovery (Wolfers and Zitzewitz, 2004).

The benefits extend to retail participants as well. Della Vedova (2026) show that retail traders on Polymarket who correctly forecast outcomes still lose money because they arrive

[1] https://ir.theice.com/press/news-details/2025/ICE-Announces-Strategic-Investment-in-Polymarket/default.aspx.

[2] https://finance.yahoo.com/news/kalshi-becomes-cnn-prediction-market-200713149.html.

[3] https://fiftycentdollars.substack.com/p/minimum-viable-liquidity.

late to thin books and pay unfavorable prices. A protocol that recycles locked capital back into the ecosystem deepens those books at every price level, directly improving execution quality for the participants who need it most.

Finally, prediction markets today cannot support the derivative products, structured hedges, and secondary instruments that institutional participants expect from any serious asset class. Borrowable positions are a prerequisite for nearly all of these. Without a credit layer, prediction markets remain a spot-only venue, and the composability required for institutional adoption cannot develop. Building that layer is therefore not an incremental improvement to the existing ecosystem; it is infrastructure that makes an entirely new class of financial products possible.

1.2 Introducing Lattica

Lattica is a protocol designed to serve as that infrastructure. Its purpose is not merely to enable lending in isolation, but to provide the credit primitives on which a composable prediction market ecosystem can be built.

The core technical challenge is underwriting. Prediction market positions are not ordinary collateral. They are anchored by information rather than fundamentals, prices adjust discontinuously, and given a large enough movement, single-sided liquidity can suddenly evaporate as the market searches for a new equilibrium. Standard lending protocols that rely on liquidating underwater collateral in time cannot guarantee solvency under these conditions. Following our work in Flanders and Flanders (2026), where we design and calibrate a risk engine capable of solvent underwriting against this collateral class, this paper presents Lattica, the protocol built around that engine.

Section 2 describes Lattica's architecture, including the first deployment of the calibrated risk engine, the Weighted Adaptive Risk Engine for Hyperjump-Optimized Rating & Stress Events (WARHORSE). Section 3 presents the protocol's smart contract design.

2 Protocol Architecture

Lattica's architecture is inspired by catastrophe reinsurance rather than traditional DeFi lending. In reinsurance, investors do not select which peril zones to underwrite; they buy equity in the book, and each risk is underwritten and priced individually rather than negotiated case by case. Lattica offers the analogous facility for prediction markets: margin and credit against a trader's market exposure, priced risk by risk at the moment of origination. Capital providers deposit USDC into a single pool. A trader draws against that pool either

by posting a prediction-market position, an ERC-1155 conditional outcome token, they already hold as collateral, or by posting margin against a position the protocol opens on their behalf; either way, WARHORSE prices the draw and an internal risk layer bounds it at the moment of origination.

The protocol's collateral standard, the ERC-1155 conditional token framework, is not specific to any single venue; it is the common representation used across permissionless on-chain prediction markets. Lattica's initial deployment underwrites positions on one such venue, but nothing in the Pool, Ledger, or WARHORSE design is venue-specific. Onboarding an additional permissionless prediction market platform is a matter of registering its markets and collateral tokens with the Configurator and extending WARHORSE's feature set to that venue's microstructure, not a protocol redesign.

All credit at Lattica, regardless of which product a trader is using, is drawn from one pool, priced by one engine, and bounded by one set of portfolio controls. That shared machinery is described first; Section 2.5 then covers the two products built on top of it.

2.1 Capital Pool

The pool is denominated in USDC. Capital providers deposit and receive share tokens representing pro-rata claims on pool capital. At the end of each epoch, net income (interest collected plus premiums minus realized losses plus any liquidation recovery) accrues to the pool, and share values update accordingly.

A single-pool design maximizes available liquidity and avoids the fragmentation that arises from category-specific or market-specific vaults, or from segmenting capital by product. Direct borrowing and leverage draw against the same balance and the same utilization cap; risk segmentation is handled entirely through internal pricing and the Configurator rather than through pool structure, so capital providers face one decision: how much USDC to deposit.

2.2 WARHORSE

WARHORSE (Weighted Adaptive Risk Engine for Hyperjump-Optimized Rating & Stress Events) is Lattica's pricing engine and the first production deployment of the risk engine developed in Flanders and Flanders (2026). Given an origination state vector containing the current price, epoch length, time to resolution, market category, and microstructure features, WARHORSE returns a risk premium quote for the exposure being underwritten. This premium covers the expected shortfall over the epoch and is separate from the interest rate, which compensates capital providers for the time value of their capital.

What that shortfall is measured against depends on what the trader has posted, so WARHORSE prices a direct loan and a leverage draw as distinct calibrated species rather than the same threshold with different labels. A direct loan is priced against a loan-to-value trigger: the request's implied LTV is matched to the nearest of a small set of certified LTV bands, and the premium reflects the drawdown needed to breach that band. A leverage draw carries no collateral to breach an LTV against, so it is priced as a mark-to-market species instead, one whose trigger is any drawdown past the position's maintenance buffer rather than a specific LTV level. The species is selected automatically from whether collateral is posted, not chosen by the trader, and both draw on the same underlying calibration in Flanders and Flanders (2026).

The full expected shortfall is baked into the upfront charge, and any proceeds recovered from an unwound position, collateral sale on a direct loan or position sale on a leverage draw, flow back to the pool as surplus. Risk engine parameters can be adjusted at the protocol level, allowing the platform to dial risk appetite up or down without retraining the underlying model. Quotes are generated off-chain with an expiry window to prevent execution against stale pricing.

2.3 Configurator

WARHORSE prices idiosyncratic risk on each draw, direct or leveraged. The Configurator bounds portfolio risk across all of them. These are separate systems by design: a position can be correctly priced in isolation and still be dangerous to the book if it concentrates exposure along a correlated dimension.

The Configurator holds a set of exposure budgets, from individual markets up to groups of related markets up to the pool as a whole, plus a concentration cap on exposure resolving within the same time window, and refuses any origination that would breach one, releasing the same headroom back as positions close.

The grouping is not fixed in advance. Related markets are clustered from offline simulations that stress the book against correlated moves, and both the cluster assignments and their budgets are re-estimated each epoch, so a budget tightens around markets that have begun to co-move before that correlation is realized as a loss. This keeps the portfolio controls honest about the risk that concerns them: exposure that is diversified by market label but concentrated in its underlying driver.

The Configurator also watches realized losses against what WARHORSE predicted, and can trip a circuit breaker that blocks new originations when actual losses run far enough ahead of the model. That breaker is deliberately asymmetric: an operational role can trip

it instantly, but only the timelocked admin can lift it, so the protocol is quick to defend and slow to stand back down.

2.4 Buffer

Buffer is Lattica's reserve: it is primed by a share of protocol revenue rather than by capital-provider deposits, and it backstops shortfalls on either product, direct or leveraged. It absorbs principal shortfalls that a position's own recovery and the loan's income leave uncovered, standing ahead of deposited pool capital in the loss waterfall. An admin-controlled treasury top-up channel can supplement it, and it reports healthy or unhealthy against a configurable target balance that the protocol's risk monitoring reads.

The loss-absorption waterfall on a liquidated position therefore runs, in order: recovered value from the position itself, the collateral sale on a direct loan or the trader's forfeited margin plus any execution-role recovery on a leverage position, offsets the loss first; the loan's premium and interest income absorbs the remainder next, which is precisely why that income is charged at origination and accrues to lenders in every state, including the good ones where realized losses land far below what WARHORSE predicted; Buffer then absorbs any shortfall that income does not; and only once Buffer is exhausted does deposited pool capital take a residual hit. Critically, Buffer does not rely on liquidation recovery to refill. In stress states, the protocol expects more expiry write-offs, worse marks, thinner books, and weaker recovery simultaneously; liquidation impairment is itself correlated with portfolio stress, so the conservative design prices assuming minimal recovery and treats all disposal proceeds as surplus on top of Buffer, not as a substitute for it.

2.5 Credit Products

The pool, WARHORSE, the Configurator, and Buffer are product-agnostic; they price and bound exposure however it is drawn. Two products draw against them today, distinguished entirely by what backs the exposure: a direct loan is backed by on-chain collateral the trader already holds, while a leverage position is backed by margin the trader posts up front against a position the protocol acquires on their behalf. Both are underwritten by the same engine at origination and settle into the same pool; nothing below is a separate protocol.

2.5.1 Direct Borrowing

Lattica operates on fixed-duration epochs. A trader posts a prediction-market position as collateral and selects a loan amount and an epoch length. The protocol returns a quote: a risk premium priced by WARHORSE and a fixed interest rate for the epoch. If accepted,

collateral transfers into the pool, the premium and interest are deducted, and net USDC disburses to the trader. The epoch length sets an outer bound, not a lock-up: the trader can repay, in full or in part, at any point during the epoch to recover some or all of their collateral early. Absent an early repayment, at epoch end the trader repays principal and recovers their collateral, walks away, or rolls the loan into a freshly quoted next epoch.

All loans are overcollateralized. If the value of the posted collateral falls below the maintenance threshold during the epoch, the protocol liquidates the position to protect pool capital; this is distinct from the walk-away case, which is the trader's own choice at epoch end when repayment is uneconomical. Premiums are priced assuming hostile recovery, so the protocol still attempts to recover value in either case, and treats speed as paramount when it does. The collateral is seized into a single backend wallet that is pre-authorised on the relevant venue and runs continuous automation; the moment the seizure transaction confirms, the backend sells the seized position directly against the venue's own order book and remits proceeds back to the pool as surplus. End-to-end resolution typically completes in seconds, so the protocol's outstanding loss exposure between seizure and remit is bounded by a single liquidation cycle rather than by the depth or speed of a public liquidation market. The trigger itself is permissionless: any caller can submit it, because the collateral destination is hardwired to the protocol's pre-set wallet at the contract level, so there is no caller who can siphon the collateral and no race over a seizure bonus, because none exists.

2.5.2 Leverage

Leverage lets a trader take amplified exposure to an outcome without ever posting the outcome token itself as collateral, because the position the leverage draws against does not exist on-chain until the protocol acquires it. A trader posts USDC into a margin account custodied by a permissioned execution role, either as an explicit deposit or pulled just-in-time from a linked wallet at open, then opens a position by choosing a market, a margin amount, and a leverage multiple. That multiple is bounded per market: each market carries an admin-set maximum loan-to-value, and the leverage available on it is $\min(10, \lfloor 1/(1-\text{LTV}) \rfloor)$, capped at 10x across the protocol regardless of how permissive an individual market's LTV is configured. This bound is applied by the execution role that sizes the position; the multiple never reaches the contracts, which see only the resulting principal and term (Section 3.2).

Opening sizes the position as margin times leverage; the margin funds the equity leg, and the remainder is drawn from the pool through a leverage-specific origination path, quoted by WARHORSE's mark-to-market species (Section 2.2) rather than the LTV species a direct loan would receive, priced into interest and a premium alongside the venue's own execution fee. The trader's margin and the pool's utilization cap are both checked and the position

provisionally reserved before either happens, so two opens against the same margin balance can never outrun what is actually available. The reserved notional is then routed to the market as a single fill against the venue's order book; a position is not durably recorded until that fill happens, and any borrowed amount left over after a partial fill is repaid back to the pool immediately.

As with a direct loan, the term is an outer bound, not a lock-up: a trader can close a position, in full or in part, at any point before its term ends, the same way an early repayment works on a direct loan. Positions carry either a fixed term, which simply closes at epoch end like a direct loan if the trader has not already closed it, or a floating term, which is rolled automatically at each epoch boundary, re-quoted and re-priced, unless the trader has capped the number of rolls or the premium they will re-accept, the roll would cross the market's resolution boundary, or margin can no longer cover the next term, in which case the position force-closes rather than rolling silently. Because there is no on-chain collateral to mark, a leverage position's health is tracked directly against its debt: it is closed the moment its mark-to-market value falls to a fixed buffer above the outstanding debt, independent of any on-chain liquidation trigger. Closing, whether voluntary or forced, sells the position back into the market and repays principal to the pool through the same settlement logic as a direct loan. A shortfall is absorbed first by the trader's own margin, locked up front and forfeited as realized loss, then by the loan's premium and interest income, and only beyond that by Buffer, so the trader's own capital is the first-loss layer ahead of the pool's shared reserve, not the other way around.

3 Smart Contracts

The contract stack is organised around a small set of functional modules rather than a flat list of independently meaningful contracts. A lending module handles the core supply/borrow/repay lifecycle. A leverage module extends that lifecycle to uncollateralized, executor-custodied positions. A liquidation module absorbs losses when a loan fails to repay. A risk and access-control module bounds portfolio exposure and governs who may touch protocol state. A thin periphery module adds permit-based routing and read aggregation without touching the core, built around the ERC-1155 conditional token standard used across permissionless on-chain prediction markets.

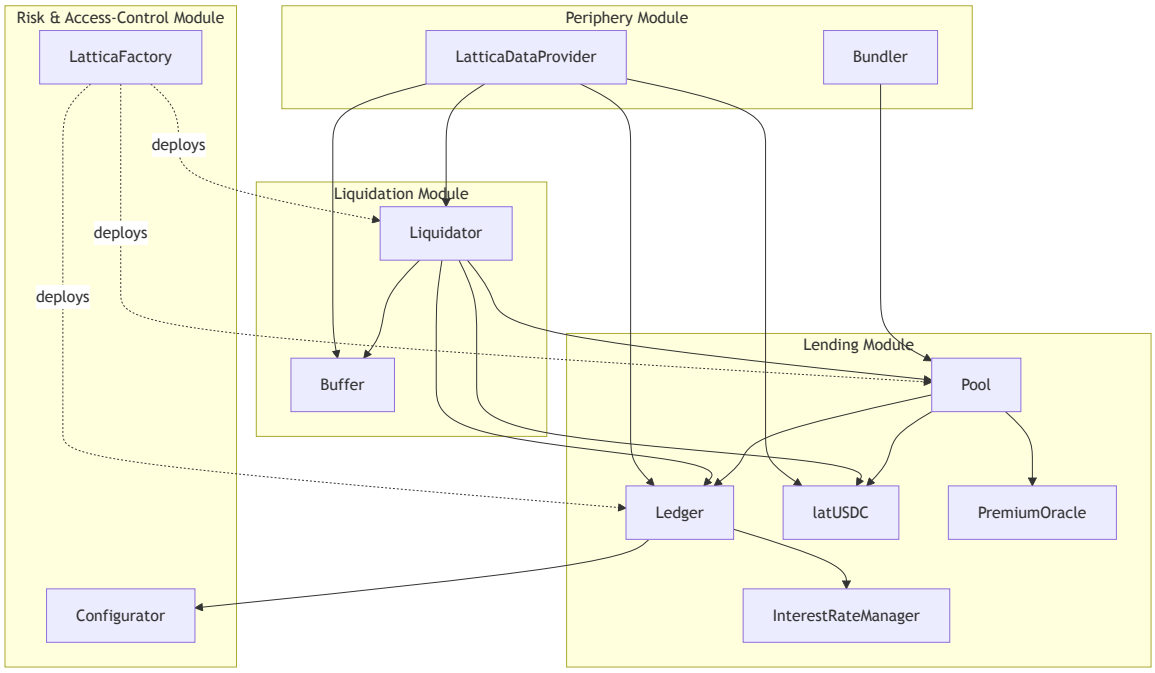


Figure 1: Lattica contract architecture. Arrows indicate call direction. `Pool` is the universal user entry point for supply, withdraw, borrow, repay, roll, and leverage; it is internally composed of state, lender, borrower, and leverage modules. `Liquidator` is the permissionless-trigger entry point for liquidation and expiration claims. The auth-list pattern described in §3.4 lets both `Pool` and `Liquidator` mutate state on `Ledger`, `latUSDC`, and `Buffer` without granting general write access.

3.1 Lending Module

The lending module is the core supply/borrow lifecycle, split across four contracts. `Pool` is the universal entry point for users: lenders call `supply`, `mint`, `withdraw`, and `redeem`; borrowers call `borrow`, `repay`, and `roll_loan`. Pool holds the protocol's CTF collateral; the underlying USDC sits in `latUSDC`. Internally, Pool is assembled from inline Vyper modules organised one domain per file, namely state, lender, and borrower logic, compiled into a single contract. Pool exposes one privileged CTF-movement function, `pool_send_collateral`, gated to the registered Liquidator address; outside a borrower reclaiming their own collateral on repayment, no caller can move CTF out of Pool.

`Ledger` holds all protocol state: loan records, idle USDC accounting, market parameters, premium-split parameters, the closing fee, the liquidation surplus split, and the running totals for borrowed amount and escrowed protocol revenue. Pool calls Ledger to originate loans, settle repayments, and settle rolls. The Ledger holds no tokens; it is pure state and pure logic. Origination, repayment, and roll-settlement mutations are gated strictly to Pool.

`latUSDC` holds all the protocol's USDC and serves as the lender deposit-receipt token, an ERC-20 with EIP-2612 permit support. Share issuance and burning are gated to Pool, and the USDC-out path used in normal operation is gated to the auth list described in §3.4; the only other exit is an admin-only sweep bounded to any balance in excess of tracked deposits and escrow. Share price is derived from that tracked accounting rather than the raw token balance, with a virtual-share offset that neutralises the first-depositor inflation attack. The vault is deliberately not ERC-4626: implementing the full ERC-4626 surface would expose a second user entry point for deposits and withdrawals, conflicting with Pool's role as the universal entry.

`InterestRateManager` is a pluggable interest-rate model. The current implementation is a kinked utilisation curve, admin-tunable per pool, that Ledger consults at origination to compute fixed interest for the epoch and at rate-snapshot time for share-price math.

`PremiumOracle` is the on-chain attestation surface for WARHORSE quotes. It accepts EIP-712 signed premium quotes generated off-chain, verifies the signer against the registered pricer address, checks the deadline, and confirms the nonce has not been used. Quotes that have expired, replay an earlier nonce, or carry an invalid signature revert at origination. The borrow path calls this to validate the offered premium before the loan opens. The signing key is a cold operational key, rotatable by the protocol admin behind a timelock. Separately, that same borrow path verifies an oracle-signed collateral mark-price attestation under Pool's own EIP-712 domain, `LatticaPriceFeed`, rejecting any mark older than a fixed freshness bound; Pool's domain, PremiumOracle's, and the Liquidator's are all distinct, so a signature minted for one surface cannot be replayed against another.

3.2 Leverage Module

Leverage positions are originated through a dedicated set of Pool entry points, `open_leverage`, `close_leverage`, `close_leverage_partial`, and `roll_leverage`, gated to a permissioned execution role rather than open to any borrower directly. Section 2.5.2 describes the product; on the contract side, the on-chain surface is deliberately thin. Pool and Ledger see only a principal amount and a term, tracked against the same utilization and portfolio accounting as a direct loan; the leverage multiple itself, the margin account it draws against, position sizing, term rolling, and the mark-to-debt liquidation trigger are execution-role logic off-chain, not contract state. A leverage position carries no on-chain CTF collateral: Pool disburses principal to the execution role, which acquires and custodies the corresponding position off-chain. On close, the execution role returns principal; a shortfall relative to the recorded principal, arising from an off-chain liquidation, draws on `Buffer` up to its balance

and, beyond that, is realized as a pool loss, since there is no on-chain collateral to sell. A close therefore always completes and never depends on the Buffer being solvent, mirroring how a direct-loan seizure lets the pool absorb whatever recovery and Buffer do not. Leverage originations respect the same pause controls as ordinary borrowing; closes are never pause-gated, since principal must always be able to return to the pool.

3.3 Liquidation Module

`Liquidator` is the operator-facing entry point for the liquidation flow. It owns the mapping that tracks each liquidated loan's recovery status, along with running totals for expected and remitted proceeds. It exposes a health-factor path, `trigger_liquidation`, for use mid-epoch, an expiration path, `claim_expired`, for loans that pass their epoch end without repayment, and batch variants of each. Both trigger paths are permissionless: there is no role gate on either, and all seized CTF flows to a fixed `liquidation_wallet` address, set at deploy time and rotatable only by the protocol admin, regardless of who called the trigger. The seizure flow marks the loan liquidated in Ledger, routes escrowed protocol revenue on the loan to Buffer, calls Buffer to cover the principal up to its balance, and transfers the CTF collateral from Pool to the liquidation wallet via Pool's gated `pool_send_collateral`. The off-chain side then sells the seized position directly against the venue's own order book and remits proceeds back through `Liquidator.remit_proceeds`, a function role-gated to `REMITTER_ROLE` even though the triggers themselves are not, since faithful remit accounting determines share-price recovery for lenders. Recovered proceeds first make the pool whole on the principal the Buffer advanced; any surplus beyond that is split, a configured protocol cut to the treasury and the remainder retained by the pool. Liquidator carries its own EIP-712 domain, separate from PremiumOracle's and Pool's, so price attestations signed for one cannot be replayed against the others.

`Buffer` is the contract behind the reserve described in Section 2.4. It absorbs principal shortfalls when liquidation recovery, whether from collateral sale on a direct loan or execution-role repayment on a leverage position, does not cover the underwritten loss. It is primed by a share of protocol revenue plus an admin top-up channel for treasury contributions, and backstops shortfalls on either product alike. Recovered USDC is transferred directly to `latUSDC`, avoiding any intermediate custody. Buffer reports a health flag against a configurable target balance, surfaced through the read aggregator for off-chain risk monitoring; the Configurator's circuit breaker itself trips on realized-versus-predicted loss calibration rather than on this flag.

3.4 Risk and Access-Control Module

`Configurator` is the contract behind the portfolio-risk system described in Section 2.3: it enforces portfolio-level risk constraints and houses the circuit breaker. Ledger checks and updates it at every origination and settlement, against exposure budgets that a risk-manager role sets rather than the contract deriving on its own. It also holds a realised-versus-predicted loss ratio, updated at settlement, that can trip the circuit breaker automatically; the breaker can also be tripped instantly by a dedicated operational role, while clearing it is admin-only.

Three contracts, namely Ledger, latUSDC, and Buffer, gate their state-changing functions through a small auth-list mapping rather than a single hardcoded caller. `LatticaFactory` sets both Pool and Liquidator as authorised callers at deploy time. Future periphery contracts that need to mutate state on these contracts can be added by granting them auth post-deploy, without redeploying the core. Functions that should never be reachable from the auth list, namely origination on Ledger and share mint and burn on latUSDC, retain a hardcoded check against the Pool address instead.

`LatticaFactory` deploys the complete stack from EIP-5202 blueprints in a single transaction: Pool first, then Ledger and the supporting contracts, then Liquidator, wiring the auth list and role grants as it goes. After construction, the Factory renounces its own admin role on every deployed contract, leaving a configured timelock as the sole admin authority. This is the mechanism that removes any standing operator key from the deployed stack; the timelock, not the deployer, holds upgrade and parameter authority from that point forward.

3.5 Periphery Module

`Bundler` is a stateless one-transaction router for permit-driven workflows. Lenders and borrowers who want to act without a separate `approve` call submit a signed EIP-2612 permit alongside their request; Bundler consumes the permit and routes the action through Pool in the same transaction. It holds no state, no funds, and no privileges; its security model is that of a transient router.

`LatticaDataProvider` is a read aggregator holding no protocol state. It exposes consolidated views over Ledger, latUSDC, Buffer, Configurator, Liquidator, and Pool for off-chain consumption, including pool snapshots, protocol economics, lender balances, redeemable amounts, liquidation-price previews, and outstanding liquidation proceeds, so that user interfaces, indexers, and risk monitors read from one contract rather than querying each piece of the stack individually.

4 Future Work

The design presented here is a first production deployment, not a fixed endpoint. We are working towards several extensions to the architecture already in place: internal matching of offsetting trader flows, direct or leveraged, or of offsetting positions on opposite sides of the same market, before they touch an external venue; improved oracle infrastructure, including redundant price sourcing and tighter attestation freshness bounds; and, longer term, extending the protocol beyond its initial venue to any permissionless prediction market platform that exposes tokenized outcome positions, since the underwriting engine and portfolio controls described in Section 2 are already venue-agnostic by construction. None of these are commitments on a fixed timeline; they are the directions the current architecture was built to accommodate.

References

Arrow, Kenneth J., Robert Forsythe, Michael Gorham, Robert Hahn, Robin Hanson, John O. Ledyard, Saul Levmore, Robert Litan, Paul Milgrom, Forrest D. Nelson, George R. Neumann, Marco Ottaviani, Thomas C. Schelling, Robert J. Shiller, Vernon L. Smith, Erik Snowberg, Cass R. Sunstein, Paul C. Tetlock, Philip E. Tetlock, Hal R. Varian, Justin Wolfers, and Eric Zitzewitz, "The Promise of Prediction Markets," *Science*, May 2008, *320* (5878), 877–878.

Clinton, Joshua D and TzuFeng Huang, "Prediction Markets? The Accuracy and Efficiency of $2.4 Billion in the 2024 Presidential Election," December 2025.

Flanders, William and Stephen Flanders, "Unlocking Liquidity on Prediction Markets," February 2026.

Hanson, Robin, "Combinatorial Information Market Design," *Information Systems Frontiers*, January 2003, *5* (1), 107–119.

Vedova, Joshua Della, "Who Profits from Prediction Markets? Execution, Not Information," February 2026.

Wolfers, Justin and Eric Zitzewitz, "Prediction Markets," *Journal of Economic Perspectives*, June 2004, *18* (2), 107–126.