Concepts

Copy

🏛️Protocol architecture

How Ekubo is put together: one singleton Core contract, deferred settlement via the "till" pattern, and a thin periphery

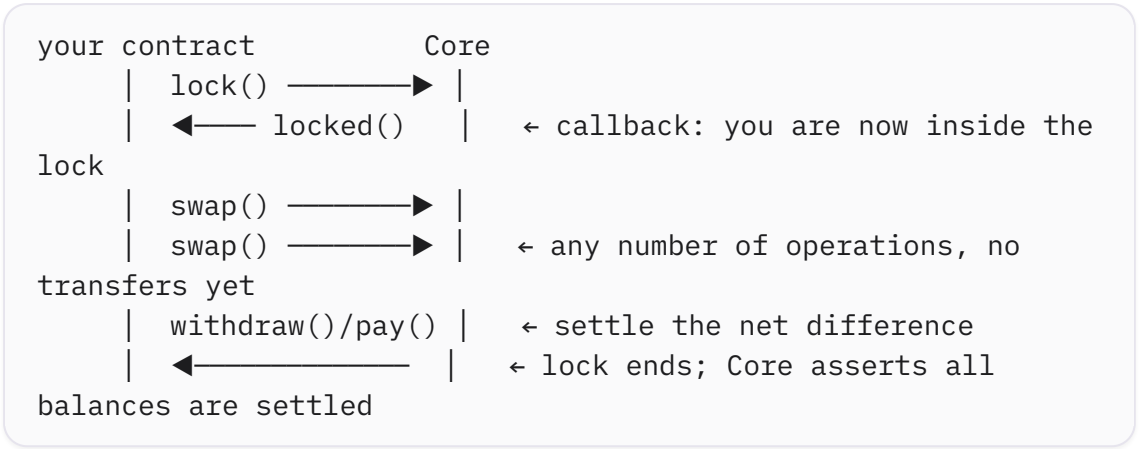
Ekubo is built as a **singleton**: a single Core contract holds every pool, every position, and all token balances. Rather than a separate contract per pool or per version, all liquidity lives in one place. This is what makes shared liquidity, cheap multi-pool routing, and a single integration surface possible — see the V3 whitepaper for the reasoning.

The "till" pattern

Every interaction that moves tokens starts with a call to `lock`. Core calls back into your contract, you perform any number of operations (swap, add or remove liquidity, collect fees), and only the **net** token amounts are settled at the end. A few operations that move no funds — notably pool initialization and extension registration — are callable directly, without a lock. Payments are deferred until you have finished — like a shop till that is reconciled once, rather than per item.

This is the "till" pattern, publicly introduced ↗ at EthCC[5] and described in more detail here ↗.

Archived copy · cryp2nova.com



Core tracks what you owe and are owed as **deltas** during the lock, and requires every delta to be zero before the lock can close. Until then, no tokens move.

> Even though the entrypoint for all methods is named `lock`, Ekubo supports reentrancy. Locks can be nested, so any contract you call from inside a lock can itself interact with Ekubo.

Flash accounting

Deferred settlement is what makes Ekubo cheap for anything more complex than a single swap. Trading across many pools, or opening several positions, requires only the minimum number of token transfers — the net difference — instead of one transfer per operation.

Two consequences worth knowing:

- **Saved balances.** Rather than withdrawing tokens at the end of a lock, you can leave them inside Ekubo for later use, avoiding token transfers entirely across repeated interactions.
- **Free flash loans.** Because balances are only checked when the lock closes, you can `withdraw` tokens and repay them within the same transaction at no cost.

Archived copy · cryp2nova.com

Pools

A pool is identified by its **pool key**: the two tokens (sorted, so `token0 < token1`), the fee, the pool type parameters, and the extension address. Pool state is deliberately compact — on EVM, the current price, tick, and liquidity pack into a single storage word (see Price representation) so that swaps touch as little storage as possible.

Ekubo V3 supports several pool types in the same Core contract: concentrated liquidity, stableswap (liquidity concentrated around a center price with an amplification factor), and full range.

Extensions

Extensions are separate contracts that Core calls at defined points in a pool's lifecycle — before and after initialization, swaps, position updates, and fee collection. They let developers add behavior (oracles, order types, custom fee logic) without reimplementing the AMM, and without fragmenting liquidity into a separate protocol.

Periphery

Core charges no protocol fee, and on EVM it is ownerless — there is no privileged account at all. (The Starknet Core is owner-upgradeable by governance, but likewise charges no protocol fee.) Everything user-facing lives in periphery contracts that hold locks on your behalf:

- **Positions** wraps liquidity positions as NFTs, and is where protocol fees are applied. It supports a fee on collected swap fees and, separately, one on withdrawn principal — the canonical deployment sets the latter to zero (see Providing liquidity)
- **Orders** manages TWAMM (DCA) orders
- **Routers** execute swap routes, including the gas-optimized Yul Router used in production on EVM

Archived copy · cryp2nova.com

- **Lens contracts** provide read-only helpers for prices, quotes, and pool state

This separation is deliberate: Core stays neutral and durable, while fee models and user experience live at the edges. See Contract addresses for what is deployed where.

Previous
Key concepts

Next
Extensions

Last updated 6 days ago

Was this helpful?



Archived copy · cryp2nova.com