

ZXO Protocol

A trust-minimized concentrated-liquidity farm. Users mint a Uniswap V3–style position into the protocol, earn a capped stream of the pool’s own token, collect trading fees from that NFT, and always take the NFT back after a one-day lock. The owner cannot seize principal and cannot raise emission caps after a pool is created.

Product	Site	Solidity
Multichain V3 liquidity farm	Zxo.Farm	0.8.36 · shanghai · 200 runs
Status	License	Audit
Pre-mainnet source	Business Source 1.1 · change 2029-09-23	Source review 26 Sep 2026

CONTENTS

1. Overview
2. Problem
3. Solution
4. Architecture
5. Networks and DEX hosts
6. Pool lifecycle
7. User functions and flows
8. Owner capabilities
9. Formulas
10. Tokenomics
11. Technical design
12. Security
13. Gas and size
14. Interfaces and dependencies
15. Deployment
16. Governance
17. License
18. Roadmap
19. Appendix

1. Project overview

ZXO exists so a person can provide concentrated liquidity on a Uniswap V3-compatible pool and earn two things at once: the pool's trading fees (the ordinary V3 LP return) and a protocol emission paid in a freshly deployed `ZxoToken` that is itself one side of that pool.

The farm does not take custody of loose tokens as a vault balance the owner can rewrite. On stake, ZXO mints a fee-tier 100 position NFT to itself, books the position, and later returns that exact NFT. Rewards are minted at claim time against immutable per-pool ceilings. There is no pause switch on unstake.

Farm

Stake a V3 range around parity. Earn a capped APR in the pool's own token.

Harvest

Claim emissions and/or collect Uniswap trading fees while the pool tick is inside the user range.

Exit

After 24 hours the NFT goes back to you or to any address you name. Pending rewards are optional to forfeit.

2. Problem statement

Most on-chain “farms” fail one of four tests:

- **Custody.** The contract holds raw balances an admin can sweep.
- **Uncapped mint.** Reward token supply is a political knob with no hard ceiling after launch.
- **Frozen exits.** A pause, a health check, or an upgrade proxy can trap principal.
- **Mercenary liquidity.** Full-range or off-range LP earns emissions without actually making a market near the price people trade.

ZXO is built against those four. Principal is an NFT the owner cannot transfer. Caps are packed at `createPool` and have no raise path. Unstake ignores the health gate. Emissions are weighted toward in-range, two-sided liquidity via a composition bonus, and claims themselves require the pool tick to sit inside the user range (spot tick, not TWAP — accepted risk M-02).

3. Solution summary

Each pool is a pair of a new `ZxoToken` and an external pair token (USDC-class 6 decimals, BTC-class 8, or 18). The protocol seeds a large single-sided core LP just off 1:1 as a sell wall, books a narrower core-stake position that soaks part of the emission weight and never claims, then opens user staking on a ± 300 -tick band around parity at the 0.01% fee tier.

Reward rate is retargeted at most once per hour from TVL and a target APR, then clipped by two immutable caps: a bps cap (`maxNormalAPR`), doubled during the first year) and a 30-day token cap (`max30dRate`). Users who pair more of the external token than the ZXO token, above a 50% paired-ratio floor, receive extra reward *weight* (not extra minted principal) up to 20%, or 25% if they hold the chain’s holder token above threshold.

4. Architecture

Five contracts and one math library, compiled once per chain against a `ChainLib` of constants.

Zxo

Farm. Ownable2Step + ReentrancyGuard.
Holds user NFTs and both core NFTs. Only
minter of each pool token. Chain-id locked at
construction.

ZxoToken

OZ ERC-20 + burnable. Owner is permanently
the farm. `coreLP`, `coreStake`, `pairing`
are immutable. No supply cap in the token;
caps live in Zxo.

ZxoTreasury

Fee sink and default referral destination.
Owner sweeps to `destination`. Destination
change is 7-day delayed and can be locked
forever. Ownable is single-step.

ZxoV3Deployer

Owner seeds single-sided positions on
arbitrary V3 position managers and sends the
NFT to `0x...dEaD`. `finalizeToken` burns
leftovers and closes seeding.
`setZxoContract` is one-shot.

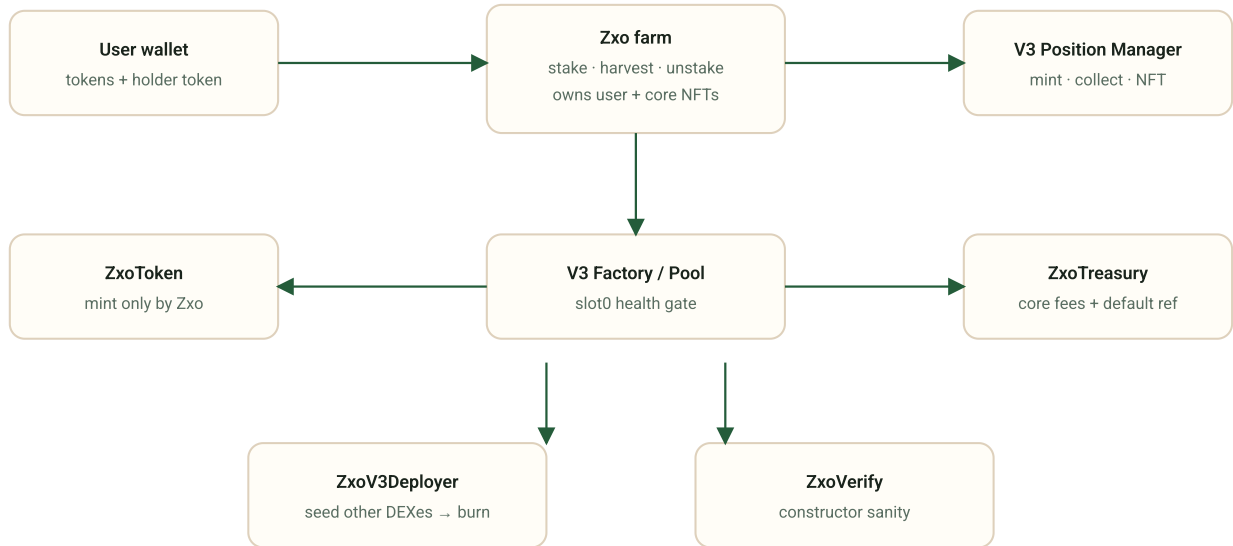
ZxoVerify

Constructor probe: factory owner, fee-tier 100
tick spacing, NPM.factory pointer, holder
supply, treasury marker = 1, optional deployer
marker = 2.

ZxoLib + ChainLib

Ticks, packing, bonus, caps, dust. Per-chain
constants: chain id, holder token, treasury,
deployer, factory, position manager.

System diagram



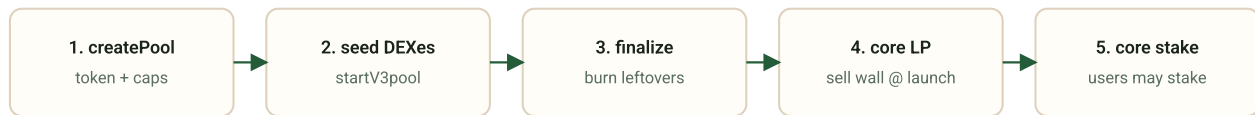
5. Networks and DEX hosts

One binary per chain. `update-chain-lib.py` selects the library, writes treasury / deployer / holder addresses, and rewrites the import. The constructor reverts if `block.chainid` does not match. PulseChain testnet and mainnet are Ethereum forks and retain the original Uniswap V3 factory and position-manager addresses; Pulse mainnet farming is configured against Liberty V3.

LIBRARY	CHAIN ID	PRIMARY V3 HOST
EthereumLib	1	Uniswap V3
OptimismLib	10	Uniswap V3
BNBChainLib	56	Pancake V3
UnichainLib	130	Uniswap V3
PolygonLib	137	Uniswap V3
PulseChainLib	369	Liberty V3
PulseChainTestLib	943	Uniswap V3 (fork copy)
RobinhoodLib	4663	Uniswap-style
BaseLib	8453	Uniswap V3
ArbitrumLib	42161	Uniswap V3
SepoliaTestLib	9991	Uniswap on virtual mainnet

Fee tier is always 100 (0.01%). Pair tokens may have 6, 8, or 18 decimals. ZXO token is always 18.

6. Pool lifecycle



1. `createPool` deploys the token. If `withoutPool` is false, it registers the pool, packs caps, mints `coreLP + coreStake` to Zxo and `v3StartAmt` to the deployer. If `withoutPool` is true, it only deploys a token — used to grind CREATE addresses until the token hex sorts below the pair so DEXScreener-class UIs render buys as buys. The live farm is a later call with `withoutPool = false` and `forceLower = true`.
2. Owner seeds satellite DEX pools via `startV3pool`. NFTs go to the burn address on a widened tick range.
3. `finalizeToken` sets `isFinalized` and burns any remainder. User staking cannot open until this is true.
4. `initializePool(..., isCore=true)` creates the canonical pool at the launch sqrt price if needed, requires slot0 to equal that price, and mints the single-sided core LP. Token id is packed into `poolConfig`.
5. `initializePool(..., isCore=false)` mints the narrower core-stake NFT, books it as owned by Zxo, increments `stakes` from 0 to 1. From that moment `stake()` is open.

Create and core-init are separate transactions (contract-size limit). A third party can initialize the factory pool at a wrong price and grief that pair; the team accepts that. A dead pair is abandoned and a new token/pool is opened.

7. User functions and flows

Everything a non-owner can call that changes state: `stake`, `harvest`, `unstake`, `updatePool`. Views: `queryData`, `getUserData`, `getTargetAprsByIds`, plus the public mappings.

7.1 Stake

```
stake(uint8 poolId, uint256 amount0Desired, uint256 amount1Desired,  
      uint256 slippageBps, address referrer)
```

- Pool must exist and already have `stakes > 0` (core stake is in).
- Caller may hold at most 21 positions across all pools.
- Slippage $\leq 5_000$ bps (50%). Combined notional, normalized to 18 decimals, $\geq 1e6$ wei-equivalent.
- Pool tick must sit strictly inside the user range (health gate).
- First referrer write wins. Zero becomes the treasury. Self or any other address is allowed and pays no treasury commission.
- Tokens are pulled, a V3 position is minted to Zxo on the user tick band, dust above the decimal-specific threshold is refunded, bonus weight is applied, the position is booked, auto-adjust may run.

7.2 Harvest

```
harvest(uint64 tokenId, uint8 poolId, Action_ action)
```

`Action_` on this path is only `Claim` (0), `CollectFees` (1), or `ClaimAndCollect` (2). Health gate applies. You must own the position and it must still have liquidity.

- **Claim** mints pending ZxoToken. If a referrer is set and is not you, they receive `refRateBps` of pending and you receive the rest. Total minted equals pending – the referral is a split, not extra inflation.
- **CollectFees** calls V3 `collect` on your NFT and forwards both tokens to you.

Rewards keep accruing while the pool is out of range. The gate only blocks when you can take them. Unstake does not require the gate.

7.3 Unstake

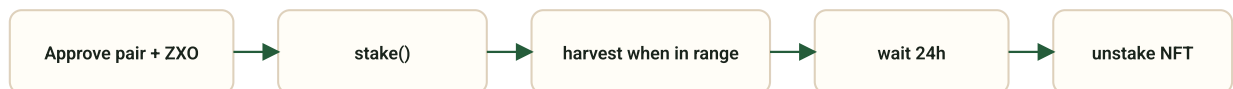
```
unstake(uint64 tokenId, uint8 poolId, bool acceptForfeit, address to)
```

- One-day lock from `stakeTime`.
- Pending is never paid here. Harvest first, or pass `acceptForfeit = true`. If `false`, pending must be $\leq 1e12$ (dust).
- `to = address(0)` means you. `to = Zxo` is rejected. A contract recipient must implement `onERC721Received`.
- Weight and TVL come off the pool, the storage row is deleted, the NFT is `safeTransferFrom` last.

7.4 Public poke

`updatePool(poolId, updateAll)` accrues and, if the one-hour cooldown has passed, retargets `rewardRate`. Anyone may call it.

User journey



8. Owner capabilities

Owner is trusted for bootstrap and for policy inside the packed caps. Owner is not trusted with user principal. After every planned pool is live, ownership of Zxo may be renounced; users keep stake / harvest / unstake.

CONTRACT	FUNCTION	CAN	CANNOT
Zxo	<code>createPool</code>	Deploy a token, optionally open a pool, mint bootstrap, set immutable caps	Raise those caps later
Zxo	<code>initializePool1</code>	One-shot core LP, then one-shot core stake	Re-init, init after users have staked
Zxo	<code>updatePoolConfig</code> type 0	Set target APR, clipped to cap / 2× cap in year one	Exceed packed <code>maxNormalAPR</code>
Zxo	<code>updatePoolConfig</code> type 1	Set referral rate in [3%, 30%]	Go outside that band
Zxo	<code>updatePoolConfig</code> type 2	Strictly lower <code>rewardRate</code> , stamp cooldown	Raise the rate by this path
Zxo	<code>collectFeesFromCoreLP</code>	Collect the packed core-LP NFT fees to treasury	Collect user NFTs; collect core-stake fees
Zxo	—	—	Pause, rescue dust/NFTs, seize positions, mint except via createPool bootstrap and public claims
Deployer	<code>setZxoContract</code>	Once	Change it later
Deployer	<code>startV3pool</code>	Seed any V3 manager until finalize	Seed after <code>isFinalized</code>
Deployer	<code>finalizeToken</code> / <code>burnTokens</code>	Close seeding, burn ERC20Burnable balances	Un-finalize

CONTRACT	FUNCTION	CAN	CANNOT
Treasury	<code>sweepToken</code> / <code>sweepAll</code>	Send balances to current destination	Send to an arbitrary address in the same call
Treasury	destination change	Schedule (7 days), execute, lock forever	Change after lock; instant change

9. Formulas

Amounts of the non-18-decimal pair token are scaled to 18 decimals before any ratio or TVL math: $\times 10^{12}$ for 6 decimals, $\times 10^{10}$ for 8.

9.1 Accrual (MasterChef)

Each pool stores an accumulator and a per-position debt. Time is wall-clock seconds. R is `rewardRate` in token wei per second. S is `totalStaked` (V3 liquidity plus bonus weight). Precision $P = 10^{18}$.

$$\text{acc}_t = \text{acc}_{t_0} + \frac{(t - t_0) R P}{S} \quad (S > 0, R > 0)$$

$$\text{pending} = \frac{L \cdot \text{acc}}{P} - D$$

where L is the position's booked liquidity (base V3 liquidity plus any stake bonus) and D is `rewardDebt`, set to $L \cdot \text{acc} / P$ at stake and after every claim. If $S = 0$ the accumulator is frozen and `lastRewardTime` is stamped to now.

9.2 Auto-adjust of R

At most once per hour, using TVL $V = \text{totalTokens}$ (normalized notionals, not liquidity units) and target A in bps:

$$R_{\text{target}} = \frac{V \cdot A}{31\,536\,000 \cdot 10\,000}$$

clipped by two packed ceilings that cannot be raised:

$$A_{\text{max}} = \begin{cases} 2 \cdot \text{maxNormalAPR} & t \leq t_{\text{create}} + 365 \text{ d} \\ \text{maxNormalAPR} & \text{otherwise} \end{cases}$$

$$R \leq \min \left(\frac{V \cdot A_{\text{max}}}{31\,536\,000 \cdot 10\,000}, \frac{\text{max30dRate}}{2\,592\,000} \right)$$

`maxNormalAPR` itself cannot exceed 32 000 bps (320%). Default target at create is 1 000 bps (10%). Type-2 owner update may only lower R , not raise it; auto-adjust can raise R back up

to the clip.

Observed APR versus TVL, for dashboards:

$$\text{APR}_{\text{bps}} = \frac{R \cdot 31\,536\,000 \cdot 10\,000}{V} \quad (V \neq 0)$$

Users earn below headline A while the core stake holds a share of S and never claims (accepted M-03). Auto-adjust does not close that gap; it shrinks as user TVL grows.

9.3 Stake bonus (reward weight, not extra tokens deposited)

Let a be the normalized ZXO (local) side actually minted into the NFT and b the normalized pair side. $T = a + b$. If either side is zero, or $b \leq a$, bonus is 0. Otherwise:

$$\rho = \frac{b}{T}$$

Bonus starts only when the paired ratio clears 50%:

$$\text{bps} = \min\left(2000, \text{floor}\left(\frac{(\rho - 0.5) \cdot 59\,538}{10^{18}} \cdot 10^{18}\right)\right)$$

In code: `bonusBps = (pairedRatio - 5e17) * 59538 / 1e18`, then cap at 2 000. That constant reaches the 20% cap at a paired ratio of about 53.36% – roughly 20 ticks above parity. Then:

$$B = \frac{L_{v3} \cdot \text{bps}}{10\,000}$$

If the caller's holder-token balance $\geq 3\,690\,000\,000 \times 10^{18}$:

$$B \leftarrow B + \frac{B}{4}$$

so the max weight bonus is 25%. B is added to booked liquidity. TVL (`tokens`) stays T . The bonus changes how emissions are split, not how many tokens the user deposited.

Worked example. $L_{v3} = 1\,000\,000$, $\rho = 0.5336$, holder below threshold. Excess above 0.5 is 0.0336. $0.0336 \times 59\,538 \approx 2000$ bps. $B = 200\,000$. Booked liquidity = 1 200 000. TVL unchanged. With holder, $B = 250\,000$.

9.4 Referral split on claim

$$\text{ref} = \lfloor \text{pending} \cdot r / 10\,000 \rfloor \quad \text{user} = \text{pending} - \text{ref}$$

$r \in [300, 3000]$. If referrer is unset-as-self or equals the claimer, $\text{ref} = 0$ and the user receives all of pending. Treasury earns only when the stored referrer is the treasury (the default for `address(0)`).

9.5 Slippage on mint

$$\text{amountMin} = \text{amountDesired} \cdot (10\,000 - s) / 10\,000$$

Users: $s \leq 5000$. Owner init/seed: s is a frontend parameter; $s = 10000$ is a zero minimum; $s > 10000$ reverts.

9.6 Health gate

Pool is healthy iff the current tick is strictly inside the user stake band:

$$\tau_{\text{low}} < \tau_{\text{now}} < \tau_{\text{high}}$$

18-dec band is $[-300, 300]$. 6-dec and 8-dec bands are the documented parity tick ± 300 , flipped when ZX0 is token0. Spot tick, not TWAP.

10. Tokenomics

There is no single protocol token. Each pool has its own `ZxoToken`, minted by that pool's farm and burned by holders via ERC-20 burn. Utility is the farm reward and the inventory side of the V3 pair.

Bootstrap

`coreLP` and `coreStake` are chosen at create, required non-zero, minted once to Zxo. `v3StartAmnt` is minted to the deployer for satellite seeds and burned at finalize.

Emissions

Only minted on `harvest` claim. Bounded by $A_{\max}$ and `max30dRate`. Unclaimed pending on unstake is never minted.

Fees

User NFT trading fees belong to the user. Core-LP fees go to treasury. Core-stake fees are locked in the NFT (accepted).

Holder boost

A known 18-decimal token per chain. Balance $\geq 3.69\text{B}$ units adds 25% on top of the composition bonus. Decimals are not checked on-chain.

Address mining (`withoutPool` + `forceLower`) is how a pool gets ZXO as token0, so buy flow charts as buys on DEX screens.

11. Technical design

State

LOCATION	WHAT IT HOLDS
<code>poolInfo[]</code>	Accumulator, TVL, liquidity weight, token addresses, reward token, R , timestamps, target APR, referral bps, stake count
<code>poolConfig[poolId]</code>	Packed word: maxNormalAPR (16) decimals (8) maxBonusEnds (40) max30dRate (128) corePositionId (64)
<code>stakedPositions_[pid]</code> <code>[tid]</code>	debt, liquidity, tokens, owner, stakeTime, poolId
<code>userStakedRefs_</code>	Up to 21 (poolId, tokenId) pairs per user
<code>userReferrer_</code> / <code>referralsPerUser_</code>	First-write referrer and counts

Events

One packed event: `ZxoEvent(user, data0, data1, data2, Action_)` with actions Claim, CollectFees, ClaimAndCollect, Bonus, Unstake, AutoAdjust, Stake, Referrer.

Access and reentrancy

Owner functions are `onlyOwner`. Every state-changing external on Zxo and the deployer is `nonReentrant`. Unstake performs the ERC-721 transfer last. ZxoToken disables `transferOwnership` and `renounceOwnership` with custom errors. Zxo itself can still renounce Ownable2Step — that ossifies admin, not users.

Upgradeability

None. No proxy. New behavior is a new pool or a new deploy. Caps cannot move up.

12. Security considerations

Pre-mainnet source review dated 26 September 2026. Classification: no principal-theft finding. A follow-up review will match creation transactions and deployed bytecode after launch.

Threat model — what is in scope

- Theft of another user's NFT or the tokens inside it
- Owner sweep of user assets
- Mint outside the packed caps
- Permanent freeze of unstake

Protections

- NFT custody + owner check + no rescue function
- Unstake independent of health and of owner
- SafeERC20 + `forceApprove` for USDT-class pair tokens
- Chain-id constructor lock + ZxoVerify before Zxo is deployed
- Finalize-before-stake on the deployer
- Core-LP slot0 must equal launch price
- User slippage cap, 21-position cap, 255-pool cap, uint128/uint96/uint16 downcast guards

Accepted risks (team + review)

ID	SUMMARY
M-01	Factory pool can be pre-initialized at a wrong price. Size limit blocks atomic create+init. Griefing is paid; pair can be abandoned.
M-02	Health gate and bonus read spot tick. A swap-in, act, swap-out costs two 0.01% fees. No TWAP. Principal untouched; caps still bind mint.
M-03	Core stake takes emission weight and never claims. Core-stake fees are locked. Users earn below headline APR until user TVL dominates.

ID	SUMMARY
L-01	Treasury commission only if referrer is left default. Self or third-party referral is intended.
L-03	Bonus ramp 59538 → 20% at ~53.36% paired ratio.
L-04	Deadline is <code>now + 20 minutes</code> at execution. Price protection is slippage.
L-05	No rescue for dust or stray NFTs. Owner has no path to assets in Zxo.
L-07	Treasury Ownable is single-step.
L-08	Deployer takes any position manager. Mitigated by finalize-before-stake; users can inspect seeds first.
L-09	No pause. That is the anti-freeze property.
L-10	Owner slippage uncapped; core-stake does not re-read slot0. Defended by the core-LP sell wall and the owner frontend.

Fee-on-transfer, rebasing, and fee-on-collect pair tokens are unsupported. Pair tokens should be canonical 6/8/18-decimal stables or majors. Holder token decimals are assumed 18.

13. Gas efficiency

- Plain `revert()` on failure (smaller bytecode, cheaper revert data).
- One packed `poolConfig` word instead of five hot slots.
- One event signature for all user and adjust actions.
- Unchecked increments where values are bounded by caps and by the permanent core stake.
- Hardcoded tick and sqrt-price constants; no on-chain $\log / \sqrt{}$.
- Dust below threshold is left in Zxo rather than refunded.
- Create+init were kept as two functions so Zxo stays under the 24 576-byte runtime limit.
- Optimizer 200 runs, via-IR off, EVM shanghai (PUSH0). Chains without Shanghai must compile to paris.

14. Interfaces and dependencies

- OpenZeppelin 5.x: IERC20, IERC20Metadata, SafeERC20, IERC721, IERC721Receiver, Ownable, Ownable2Step, ReentrancyGuard, ERC20, ERC20Burnable.
- Uniswap V3-compatible `INonfungiblePositionManager`: `mint`, `collect`, `createAndInitializePoolIfNecessary`.
- Factory: `getPool`, and for verify `owner` + `feeAmountTickSpacing(100)`.
- Pool `slot0()` via raw `staticcall`, first two words only, so extra fork return fields do not break the health gate.
- ZxoToken must expose OZ `burn(uint256)` for deployer finalize.

15. Deployment

Addresses are not canonical until the post-launch matching audit. Process per chain:

1. Run `update-chain-lib.py`, compile with solc 0.8.36.
2. Deploy `ZxoTreasury`.
3. Deploy `ZxoVerify(false)`. Require `isVerified()`.
4. Deploy `ZxoV3Deployer`. Write its address into the lib.
5. Recompile verify + Zxo. Deploy `ZxoVerify(true)`, then `Zxo`.
6. `setZxoContract` once.
7. Move treasury destination, wait 7 days, optionally lock. Move ownership to the production key. Record creation hashes for the second audit.

ZxoVerify will refuse a zero factory, NPM, holder, treasury, or (when asked) deployer, a factory without fee 100, or an NPM whose `factory()` pointer does not match.

16. Governance

No token vote, no Governor, no proxy admin. Policy knobs after launch are target APR (clipped), referral rate (banded), and lower-only reward rate. New pools require owner. When the protocol will not open more pools, Zxo ownership is renounced. Treasury destination is a separate, lockable sink. Production expectation: multisig or controlled key on Zxo and the deployer (2-step), careful single-step key on treasury.

17. License and compliance

The Licensed Work is under Business Source License 1.1, licensor Zxo, additional use grant none, change date 23 September 2029, after which it becomes GPL-7.0-or-later as specified in the header files. This whitepaper is a technical description, not an offer of securities and not legal advice. Pair-token listing and frontend geo-fencing are off-chain concerns.

18. Roadmap

1. Finish per-chain testnet farms (ChainLib + Verify + seed + finalize + core).
2. Mainnet deploy per chain with recorded creation hashes.
3. Post-launch audit matching source to bytecode and addresses.
4. Open user staking pool by pool once core walls are in.
5. Optional ownership renounce after the planned set of pools is live.

19. Appendix

Glossary

TERM	MEANING
Core LP	Protocol-owned single-sided V3 NFT just off parity. Fees collectable to treasury.
Core stake	Protocol-owned narrower NFT booked as a stake. Never claims. Fees locked.
Health / fair ratio	Current tick strictly inside the user band.
TVL / totalTokens	Normalized token notionals used to size R .
totalStaked	Liquidity weight used to split R , including bonus.
Holder token	External 18-dec token that adds +25% on the composition bonus.
Finalize	Irreversible close of deployer seeding for one ZxoToken.

Tick sheet (user band and core)

PAIR DECIMALS	PARITY TICK	USER BAND	CORE (ZXO LOWER)	LANDED TICK (ZXO LOWER)
18	0	[-300, 300]	[11, 300]	10
8	230270	parity $\pm$ 300	near parity, 20-tick tighten on core stake	see sqrt-price constants
6	276324	parity $\pm$ 300	near parity, 20-tick tighten on core stake	see sqrt-price constants

Source map

- `contracts/Zxo.sol`
- `contracts/libraries/ZxoLib.sol`
- `contracts/ZxoTreasury.sol`
- `contracts/ZxoV3Deployer.sol`
- `contracts/ZxoVerify.sol`

- [contracts/tokens/ZxoToken.sol](#)
- Companion audit: [Zxo-Security-Audit.html](#)

ZXO Protocol technical whitepaper · September 2026 · <https://Zxo.Farm> · source review, not a live-address certification