



Uniswap V3 liquidity farming anchored by permanently locked, single-sided core liquidity.

UPDATED
6 October 2026

WEBSITE
Zxo.Farm

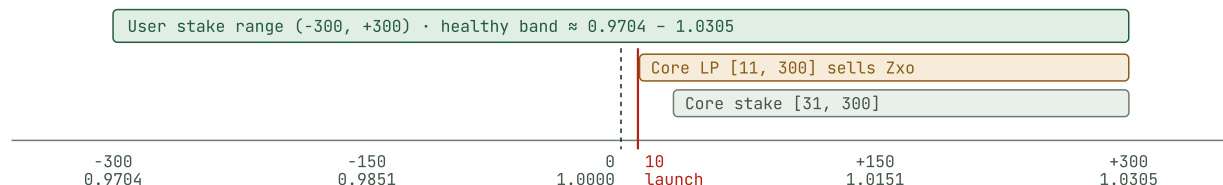
CONTRACTS
Solidity 0.8.36 · BUSL-1.1

SECURITY REVIEW
Revision 3.1 · 8.8 / 10

DEX MODEL
Uniswap V3 and forks

STATUS
Live on Arbitrum and
PulseChain

TICK MAP · 18/18 PAIR · Zxo AS TOKEN0 · PRICE = 1.0001^tick PAIR TOKENS PER Zxo



CONTENTS

- | | |
|-----------------------------|---------------------------------|
| 0. Abstract | 9. Tokenomics |
| 1. Project overview | 10. Governance and owner powers |
| 2. Problem statement | 11. Security |
| 3. Solution summary | 12. Gas efficiency |
| 4. Architecture | 13. Interfaces and dependencies |
| 5. Core functionality | 14. Deployment |
| 6. Staking and reward math | 15. Legal and compliance |
| 7. Technical design | 16. Roadmap |
| 8. User and owner functions | 17. Appendix |

Zxo in one page

Zxo is a liquidity farming protocol for Uniswap V3 and compatible DEXes. Each Zxo pool launches its own reward token, paired with an established asset such as a stablecoin or a major token, and designed to trade close to 1:1 with it. Users provide liquidity to that pair as a V3 position on a fixed, shared price range, stake the position NFT in the Zxo contract, and earn the pool's reward token while the price, and its 15-minute average, stay within the range.

Every pool is anchored by a **core LP**: a large, single-sided position of the reward token placed just above parity. As buyers push the price up, the core LP sells reward tokens and accumulates the pair asset. The contract has no function that can withdraw or reduce this position, so the core LP and everything it accumulates is locked permanently.

Rewards are minted on claim at a rate that the contract re-targets to a configured APR on staked value, capped by hard limits set at pool creation. A bonus rewards deposits that bring more of the pair asset than the reward token. Staked positions can always be withdrawn after a one-day lock, with no price condition and no owner approval.

FOR LIQUIDITY PROVIDERS

Stake a V3 position, earn trading fees plus minted rewards, withdraw at any time after 24 hours.

FOR THE ECOSYSTEM

Locked core liquidity that no one can pull, and a reward token priced against a real asset from its first block.

FOR VERIFIERS

No upgrades, no pause, no admin withdrawal path. Every rule in this paper is enforced by immutable code, and the live protocol contracts are exact-match verified on Sourcify.

01

Project overview

Zxo lets people farm, harvest, collect and earn on concentrated liquidity. One deployment of the Zxo contract on a chain can host up to 255 pools. Each pool consists of:

- a **reward token** (a ZxoToken) created by the pool, with the Zxo contract as its only minter;
- a **pair token** with 6, 8 or 18 decimals, such as USDC, WBTC or an 18-decimal asset;
- a V3 pool for the pair at the **0.01% fee tier**;
- a permanently locked **core LP** and **core-stake** position owned by the Zxo contract;
- an **emission controller** that targets an APR on staked value within hard caps.

The protocol consists of four long-lived contracts per chain: `Zxo` (the farm), `ZxoTreasury` (fee receiver), `ZxoV3Deployer` (bootstraps prices on other DEXes) and `ZxoOracle` (reads a 15-minute average price from V3 pools). A fifth, `ZxoVerify`, is a sanity check that is deployed only to run its checks during deployment. Each pool also deploys its own `ZxoToken`.

Zxo is live on two chains. It launched on Arbitrum One on 28 September 2026, on Uniswap V3, and on PulseChain on 3 October 2026, on LibertySwap, a PancakeSwap-style V3 DEX. The eleven contracts covered by the deployment reports are verified on Sourcify with an exact bytecode match; the other reward tokens are covered through the verified Zxo build. The addresses are in [section 14](#).

02

Problem statement

Liquidity that can leave overnight

New tokens usually launch with liquidity the team controls. Holders have to trust that it will stay, and many launches end with that liquidity withdrawn.

Farms that pay for idle capital

Classic MasterChef farms pay rewards on LP tokens whether or not that liquidity is useful. With concentrated liquidity, a position outside the active price earns no fees and does nothing for traders.

Unbounded emissions

Fixed emission schedules pay the same number of tokens whether the farm holds \$10,000 or \$10 million, which leads to runaway APRs early and dilution later.

Admin keys over user funds

Many farms can pause withdrawals, change reward rules or migrate funds. Users carry that key risk for as long as they stay.

03

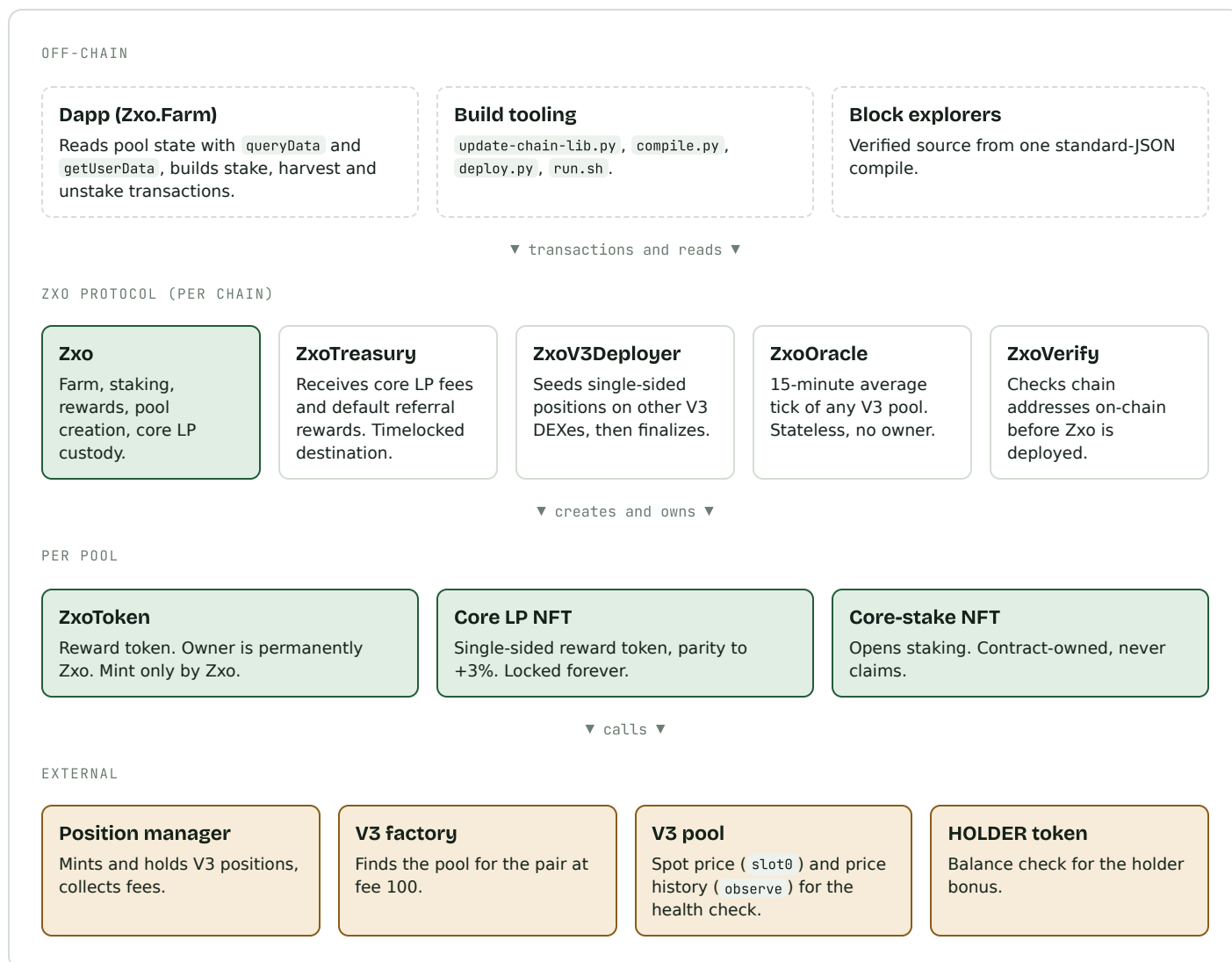
Solution summary

PROBLEM	ZXO DESIGN	WHERE IT IS ENFORCED
Removable launch liquidity	The core LP is minted to the Zxo contract, which has no function to transfer, burn or reduce it.	<code>Zxo.initializePool</code>
Rewards for idle liquidity	All user positions share one range around parity. Claims are only allowed while the price is inside that range.	<code>_getStakeTick</code> , <code>_requireFairRatio</code>
Price pushed for one transaction	The health gate also requires the 15-minute time-weighted average price to be inside the range, which a same-block swap cannot move.	<code>ZxoOracle.inRange</code> , <code>_poolRatio</code>
Deploying with wrong addresses	The Zxo constructor checks every configured address on-chain: code present, DEX wiring, fee tier, HOLDER token, and the treasury, deployer and oracle contracts.	<code>Zxo</code> constructor, <code>ZxoVerify</code>
Unbounded emissions	The reward rate re-targets to an APR on staked value, capped by a max APR and a max 30-day amount fixed at pool creation.	<code>_performAutoAdjust</code>
Admin control over funds	No upgrades, pause, migration or rescue functions. Unstake has no owner or price gate.	Entire codebase

PROBLEM	ZXO DESIGN	WHERE IT IS ENFORCED
Hidden launch allocations	Tokens used to seed other DEXes must be finalized (remainder burned, seeding closed) before staking can open.	<code>ZxoV3Deployer.finalizeToken</code>

84

Architecture



ChainLib. Each chain has a small library of constants: chain id, position manager, factory, HOLDER token and HOLDER threshold, and the treasury, deployer and oracle addresses. One library is compiled into the contracts for the target chain. Every constructor reverts if deployed on a different chain id, and the deploy script refuses to send a build to a chain that doesn't match.

Checked at deployment. The Zxo constructor reverts, with a readable reason, unless every configured address is a deployed contract, the position manager uses the configured factory, the 0.01% fee tier exists with tick spacing 1, the HOLDER token has 18 decimals and a threshold of at least one token, the treasury,

deployer and oracle identify themselves correctly, and the deployer is not already linked to another Zxo. These checks live in the constructor, so they cost nothing at runtime.

No proxies. Contracts are deployed directly and cannot be upgraded. The rules described here are the rules for the life of each deployment.

05

Core functionality

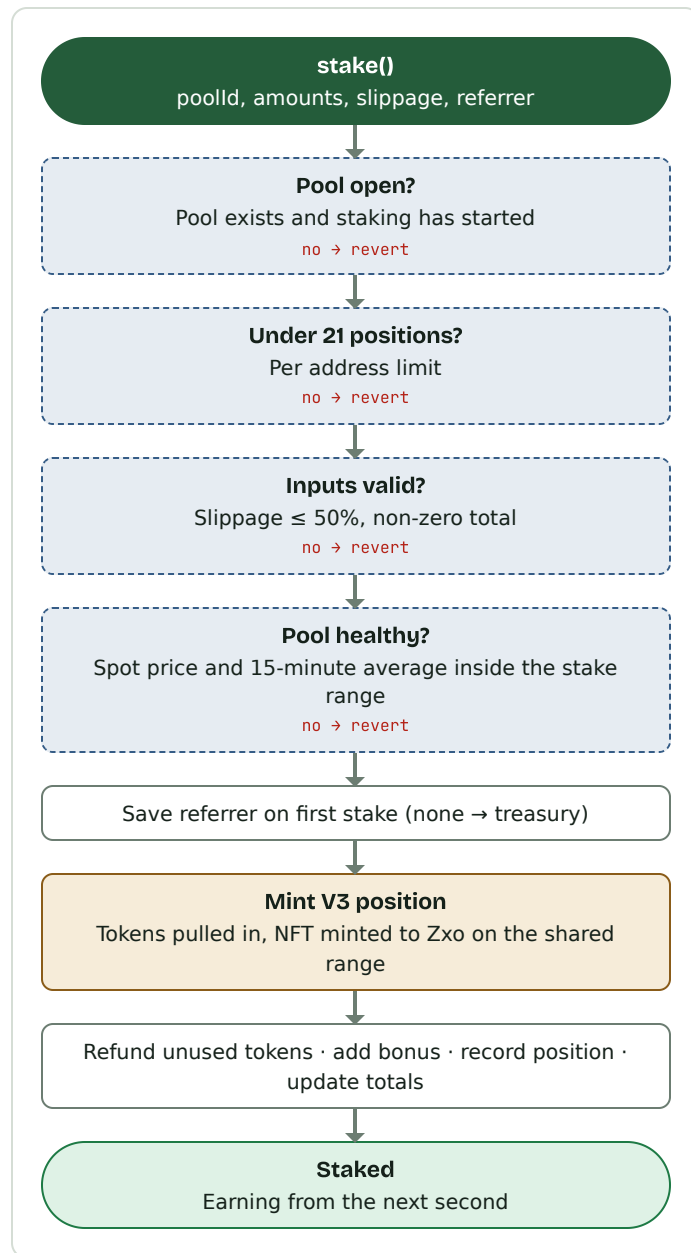
Start Check (reverts on failure) Contract step External DEX call Result

5.1 Pool launch

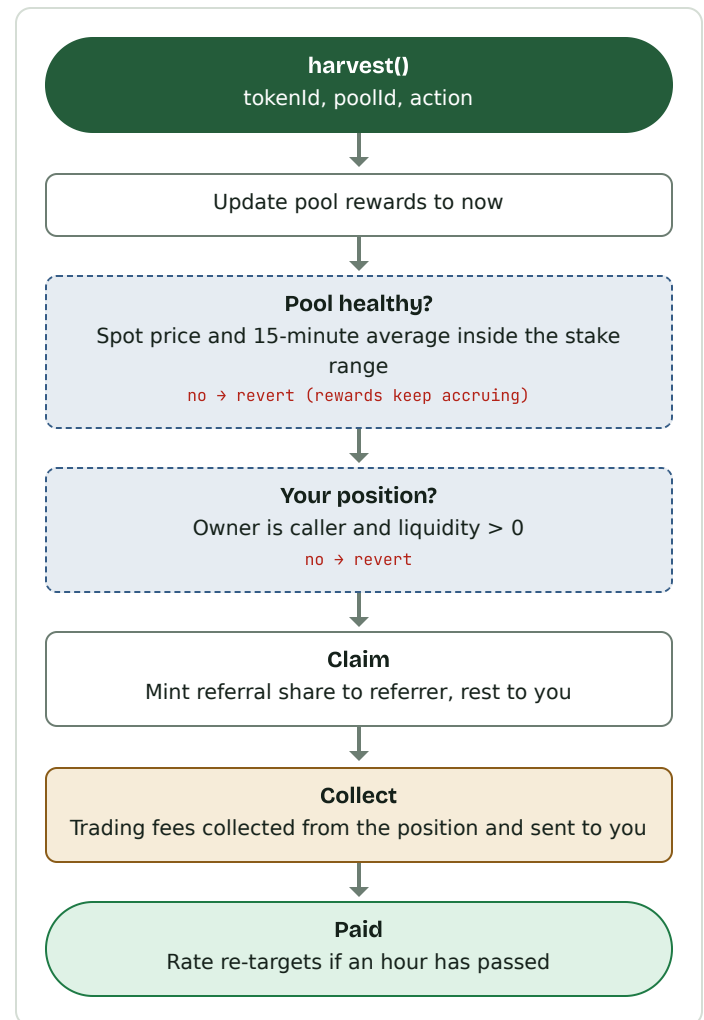
The owner launches each pool in six steps. The contracts enforce the order of the last two: the core stake cannot be minted until the reward token is finalized on the deployer, and staking cannot open until the core stake exists. Step 3 gives the new pool room to store 15 minutes of price history, which the health gate needs.

STEP 1 createPool Deploys the ZxoToken, validates the pair's decimals, mints the core amounts to Zxo and the seeding amount to the deployer.	STEP 2 Core LP Creates the V3 pool at the launch price, checks the price, mints the single-sided core LP to Zxo.	STEP 3 Price history Raise the pool's observation slots once (<code>set-observations.py</code>). Anyone can do it; the size can only grow.
STEP 4 Seed DEXes Small single-sided positions on other V3 DEXes. NFTs go to 0x...dEaD.	STEP 5 Finalize Burns the deployer's remaining tokens. Seeding for this token is closed forever.	STEP 6 Core stake Requires finalization. Mints the core-stake position, which opens staking.

5.2 Stake

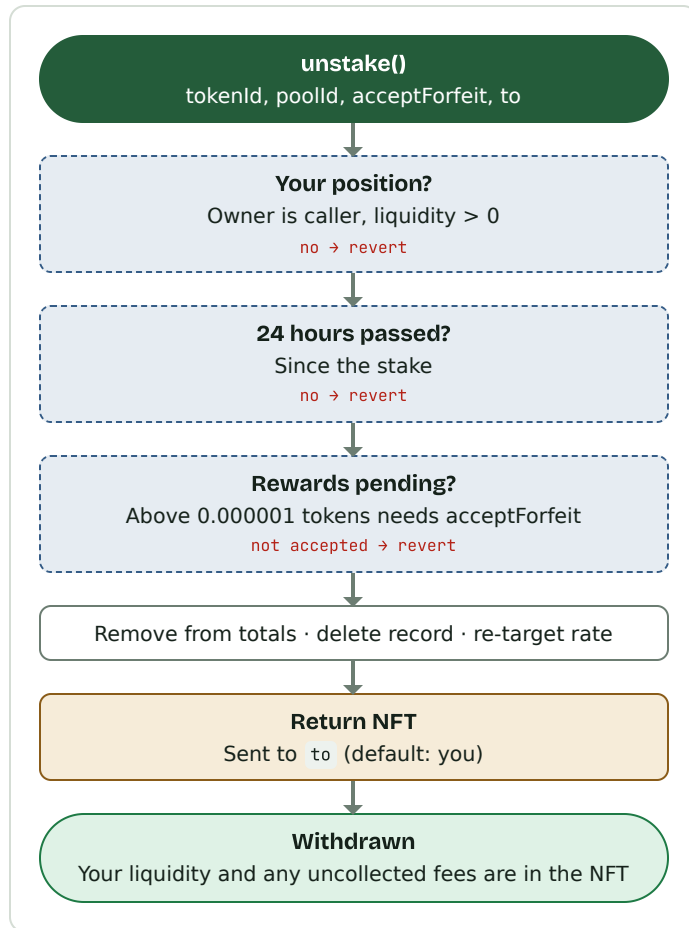


5.3 Harvest



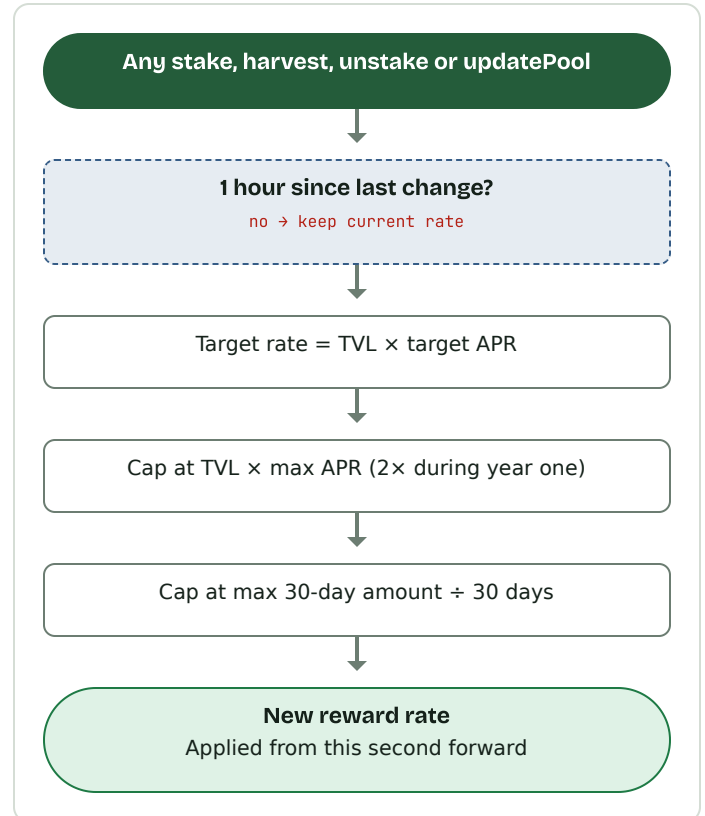
Action 0 claims rewards, 1 collects fees, 2 does both.

5.4 Unstake



No price check and no owner approval. Harvest first: pending rewards are forfeited on unstake.

5.5 Emission re-targeting



06

Staking and reward math

Every formula below is taken from the contract code. Amounts are integers in the token's smallest unit, divisions round down, and `PRECISION = 1018`. Examples use round numbers for readability.

6.1 Price, ticks and ranges

```
price(token1 per token0) = 1.0001 ^ tick  
sqrtPriceX96              = vprice × 296
```

A tick is a 0.01% price step. 300 ticks ≈ 3.05%.

For an 18/18 pair with Zxo as token0, tick 0 is exactly 1 pair token per Zxo (parity). For 6- and 8-decimal pair tokens, parity sits at tick ±276,324 and ±230,270 because of the decimal difference.

POSITION	RANGE (18/18, ZX0 = TOKEN0)	ZX0 PRICE BAND	PURPOSE
User stake	(-300, +300)	0.9704 - 1.0305	Every user position uses this exact range
Core LP	[11, 300]	1.0011 - 1.0305	Single-sided sell wall above parity
Core stake	[31, 300]	1.0031 - 1.0305	Opens staking, keeps totals non-zero
Launch price	tick 10	1.0010	One tick below the core LP

When Zxo is token1, all ranges mirror around parity (for example the core LP becomes [-300, -11]). The full table for every pair type is in the appendix.

6.2 Health gate

```
twapTick = ⌊ (tickCumulative(now) - tickCumulative(now - 900 s)) / 900 ⌋    (rounded toward -∞)
healthy  = lowerTick < currentTick < upperTick
          and lowerTick < twapTick < upperTick
```

`currentTick` is read live from the pool's `slot0`. `twapTick` is the 15-minute time-weighted average, read by `ZxoOracle` through the pool's `observe`. Both bounds are strict.

A swap inside one transaction can move the spot tick but not the 15-minute average, so pushing the price into range just to harvest does not work. If the pool does not yet have 15 minutes of history, no average is available and the pool counts as unhealthy.

Stake and harvest require a healthy pool. Unstake does not. Rewards accrue whether or not the pool is healthy, and become claimable again when the price returns to the range.

6.3 Deposit normalization

```
normalized = amount × 10^(18 - decimals)    (decimals ∈ {6, 8, 18})
totalAmount = normalized0 + normalized1
```

All TVL figures are 18-decimal amounts summed 1:1, which matches the pair's parity design. `totalAmount` is recorded as the position's contribution to pool TVL.

EXAMPLE

Deposit 5,000 USDC (6 decimals) and 4,990 Zxo: $5,000 \times 10^{12} + 4,990 \times 10^{18}$ in base units, recorded as 9,990 tokens of TVL.

6.4 Position liquidity

With v_a , v_b the range bounds and v_p the current price, for $v_a < v_p < v_b$:

$$\text{amount0} = L \times (v_b - v_p) / (v_p \times v_b)$$
$$\text{amount1} = L \times (v_p - v_a)$$
$$L = \min(\text{amount0} \times v_p v_b / (v_b - v_p), \text{amount1} / (v_p - v_a))$$

The position manager computes L from your deposit and returns any unused amount. Because every user position uses the same range, L is proportional to the position's value at any price, so weighting rewards by L is fair between users.

The mix of tokens you deposit is set by the current price, not by you. Above parity (Zxo token0) a deposit is mostly pair token; below parity it is mostly Zxo.

6.5 Stake bonus

```
pairedRatio = paired / (paired + local)           (normalized amounts)
bonusBps    = min( 2000, (pairedRatio - 0.5) * 59,538 )   only if paired > local
bonusL      = L * bonusBps / 10,000
HOLDER     : bonusL = bonusL * 1.25                 if HOLDER balance ≥ HOLDER_THRESHOLD
weight      = L + bonusL
```

`local` is the reward token in your deposit and `paired` is the pair token. The bonus rewards deposits that bring more of the pair asset than of the reward token, which happens when the price is above parity.

The bonus reaches its 20% cap (25% for HOLDER holders) at a 53.36% paired ratio, about 20 ticks above parity. At the launch tick the organic bonus is about 10–12%.

The bonus is virtual reward weight. It adds nothing to your principal and is removed from the totals when you unstake.

`HOLDER_THRESHOLD` and the HOLDER token are set per chain in its ChainLib and fixed at compile time. The threshold is at least one token. It is 3.69 billion tokens on both Arbitrum and PulseChain.

EXAMPLE

The price is 10 ticks above parity, so a deposit is 51.68% pair token. Excess = 0.0168, so $\text{bonusBps} = 0.0168 \times 59,538 = 1,000$ (10%). A position with $L = 1,000,000$ gets weight 1,100,000, or 1,125,000 with the HOLDER bonus.

BONUS CALCULATOR

Pair token share of deposit:

51.7%



Position liquidity L

1000000



Holds at least the chain's HOLDER threshold

1,012

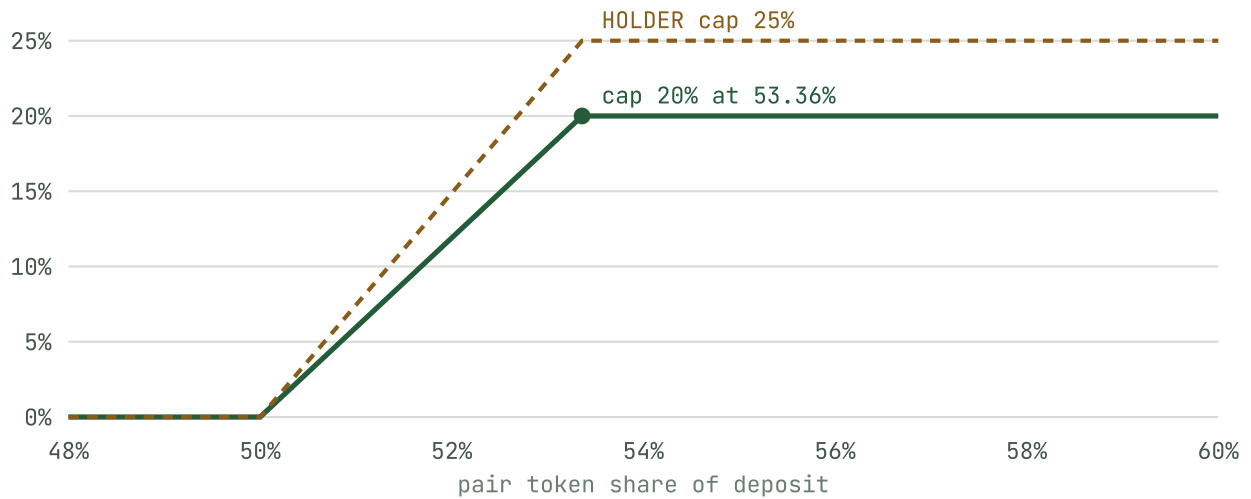
bonus (bps)

10.1%

bonus rate

1,101,200

reward weight



6.6 Reward accounting

On every pool update (Δt seconds since the last):

$\text{accRewardPerShare} += \Delta t \times \text{rewardRate} \times 10^{18} / \text{totalStaked}$

For a position:

$\text{pending} = \text{weight} \times \text{accRewardPerShare} / 10^{18} - \text{rewardDebt}$

$\text{rewardDebt} = \text{weight} \times \text{accRewardPerShare} / 10^{18}$ (reset on stake and claim)

totalStaked is the sum of all weights in the pool, including the core stake. Your share of emissions at any moment is $\text{weight} / \text{totalStaked}$.

Rewards are minted when claimed. Nothing is pre-funded, and rewards that are never claimed are never minted.

6.7 Emission rate

```
SECONDS_PER_YEAR = 31,536,000
targetRate = TVL × targetAprBps / (SECONDS_PER_YEAR × 10,000)
maxRate    = TVL × maxAprBps    / (SECONDS_PER_YEAR × 10,000)
monthlyCap = max30dRate / 2,592,000 (if set)
rewardRate = min(targetRate, maxRate, monthlyCap)

maxAprBps = maxNormalAPR × 2 during the first 365 days of the pool
           = maxNormalAPR afterwards (maxNormalAPR ≤ 32,000 = 320%)
displayed APR (bps) = rewardRate × SECONDS_PER_YEAR × 10,000 / TVL
```

The rate re-targets on the next stake, harvest, unstake or `updatePool` call after a 1-hour cooldown from the last change. The default target is 10% (1,000 bps).

EXAMPLE

TVL 2,000,000 tokens, target 10%: $\text{rewardRate} = 2,000,000 \times 1,000 / (31,536,000 \times 10,000) = 0.006342$ tokens per second, or 200,000 tokens per year.

6.8 Referral split

```
referral = pending × refRateBps / 10,000 (3% - 30%, default 3%)
you      = pending - referral
```

Your referrer is saved on your first stake and never changes. If you pass no referrer, the treasury is your referrer. If your referrer is your own address, you receive 100% and no split happens. The split comes out of your rewards; the total minted is the same either way.

6.9 Unstake threshold

```
allowed if pending ≤ 1012 base units (0.000001 tokens) or acceptForfeit = true
```

Pending rewards are not paid on unstake. Harvest first, then unstake. The threshold prevents accidental forfeits; `acceptForfeit` lets you exit immediately, for example while the pool is out of range and harvest is blocked.

REWARD ESTIMATOR (ILLUSTRATIVE)

Pool TVL (tokens)

2000000

Target APR (%)

10

Your stake (tokens)

10000

Your bonus (%)

10

Referral rate (%)

3

200,000

pool emissions / year

0.55%

your share of weight

1,066

your tokens / year after
referral

10.7%

your effective APR

Assumes weight is proportional to value and ignores the core stake's share, fee income and price changes. The contract's actual figures depend on every position in the pool.

07

Technical design

7.1 Storage

VARIABLE	TYPE	CONTENTS
<code>poolInfo</code>	<code>PoolInfo[]</code>	Per pool: <code>accRewardPerShare</code> , <code>totalStaked</code> , <code>totalTokens</code> , <code>token0</code> , <code>token1</code> , <code>rewardToken</code> , <code>rewardRate</code> (uint96), <code>lastRewardTime</code> , <code>lastAutoUpdate</code> (uint40), <code>targetAprBps</code> , <code>refRateBps</code> (uint16), <code>stakes</code> (uint32)
<code>poolConfig</code>	<code>uint8 → uint256</code>	Packed: <code>maxNormalAPR</code> (bits 240–255), <code>decimals</code> (232–239), <code>maxBonusEnds</code> (192–231), <code>max30dRate</code> (64–191), <code>corePositionId</code> (0–63)
<code>stakedPositions_</code>	<code>uint8 → uint64 → StakedPosition</code>	<code>rewardDebt</code> , <code>liquidity</code> (weight), <code>tokens</code> (TVL share), <code>owner</code> , <code>stakeTime</code> , <code>poolId</code>
<code>userStakedRefs_</code>	<code>address → PositionRef[]</code>	Up to 21 (<code>poolId</code> , <code>tokenId</code>) pairs per address
<code>userReferrer_</code>	<code>address → address</code>	Referrer, set once on first stake
<code>referralsPerUser_</code>	<code>address → uint32</code>	Count of addresses that chose this referrer
<code>totalGlobalStakes</code>	<code>uint32</code>	Active positions across all pools

7.2 Events

Zxo emits a single protocol event, `ZxoEvent`, through one shared internal function, to keep the contract under the 24,576-byte size limit. Indexers filter on `action`. A harvest with action 2 (ClaimAndCollect) logs a Claim and a CollectFees event.

```
event ZxoEvent(address indexed user, uint256 data0, uint256 data1, uint256 data2, Action_ action);
```

ACTION	CODE	USER	DATA0	DATA1	DATA2
Claim	0	staker	poolId	tokenId	rewards claimed (incl. referral share)
CollectFees	1	staker	poolId	tokenId	amount0 + amount1
Bonus	3	staker	poolId	tokenId	bonus weight added
Unstake	4	staker	poolId	tokenId	pending rewards forfeited
AutoAdjust	5	caller	poolId	new rewardRate	targetAprBps
Stake	6	staker	poolId	tokenId	weight (liquidity + bonus)
Referrer	7	referrer	0	0	0
Deployed	8	deployer	timestamp	chain id	0
PoolCreated	9	reward token	poolId	max30dRate	maxNormalAPR
CoreFeesCollected	10	treasury	poolId	amount0	amount1
ConfigUpdated	11	owner	poolId	dataType	value stored (target APR after the cap)
PoolInitialized	12	reward token	poolId	position tokenId	liquidity

7.3 Access control, errors and upgradeability

Access control

Zxo and ZxoV3Deployer use `OpenZeppelin Ownable2Step`. ZxoTreasury uses `Ownable`. ZxoToken ownership is fixed to Zxo and cannot be transferred or renounced.

Reentrancy

Every state-changing external function is `nonReentrant`. State is written before external transfers, and unstake makes its NFT transfer the final step.

Errors and upgrades

Runtime failures use a bare `revert()` to save bytecode; the dapp checks conditions before sending. Deployment checks use readable messages. No proxy, no upgrade path.

User and owner functions

8.1 Functions anyone can call

FUNCTION	WHAT IT DOES	REQUIREMENTS
<code>stake(poolId, amount0Desired, amount1Desired, slippageBps, referrer)</code>	Mints a V3 position on the pool's stake range from your tokens and stakes it.	Approve Zxo for both tokens. Pool open and healthy. Fewer than 21 positions. Slippage $\leq$ 5,000 bps.
<code>harvest(tokenId, poolId, action)</code>	0 claims rewards, 1 collects trading fees, 2 does both.	You own the position. Pool healthy.
<code>unstake(tokenId, poolId, acceptForfeit, to)</code>	Returns the position NFT to <code>to</code> (address(0) = you).	You own the position. 24 hours since staking. Pending $\leq$ 0.000001 or forfeit accepted. <code>to</code> can receive NFTs.
<code>updatePool(poolId, updateAll)</code>	Accrues rewards and re-targets the rate for one or all pools.	None
<code>pool.increaseObservationCardinalityNext(n)</code>	A V3 pool function, not Zxo. Enlarges the pool's price history so the 15-minute average is always available. <code>set-observations.py</code> picks a size for the chain.	None. The size can only grow, so it cannot be used to break a pool.

8.2 Read functions

FUNCTION	RETURNS
<code>queryData(tokenId, poolId, 0, 0)</code>	Time of the last rate change
<code>queryData(tokenId, poolId, 1, 0)</code>	Pending rewards for a position
<code>queryData(0, poolId, 2, index)</code>	Pool config: 0 maxNormalAPR, 1 decimals, 2 bonus-year end, 3 max30dRate, 4 core LP id, 5 packed value. Other indexes revert.
<code>queryData(0, poolId, 3, 0)</code>	Current APR in bps
<code>queryData(0, poolId, 4, 0)</code>	<code>(healthy, sqrtPriceX96, tick)</code> ; healthy needs the spot tick and the 15-minute average in range. <code>(false, 0, 0)</code> when no price is available
<code>ZxoOracle.twapTick(pool) / inRange(pool, lower, upper)</code>	The 15-minute average tick (<code>ok = false</code> when history is missing), and whether it sits strictly inside a range
<code>queryData(tokenId, poolId, 5, 0)</code>	<code>(canUnstakeWithoutForfeit, pending, threshold)</code>
<code>getUserData(user, dataType)</code>	0: positions and ids · 1: ids and pool ids · 2: all

FUNCTION	RETURNS
<code>getTargetAprsByIds(poolIds)</code>	Target APR per pool
<code>poolInfo(i)</code> , <code>poolConfig(i)</code> , <code>stakedPositions_(pid, id)</code> , <code>userReferrer_(a)</code> , <code>totalGlobalStakes()</code>	Raw storage getters

8.3 Owner functions

CONTRACT	FUNCTION	EFFECT AND LIMITS
Zxo	<code>createPool(...)</code>	New token and pool. $\text{maxNormalAPR} \leq 320\%$. Pair decimals must match (6, 8 or 18). Max 255 pools.
Zxo	<code>initializePool(poolId, slippageBps, isCore)</code>	Once each: core LP (price must equal the launch price), then core stake (token must be finalized).
Zxo	<code>updatePoolConfig(poolId, 0, v)</code>	Target APR, clamped to the current max.
Zxo	<code>updatePoolConfig(poolId, 1, v)</code>	Referral rate between 3% and 30%.
Zxo	<code>updatePoolConfig(poolId, 2, v)</code>	Lower the reward rate only. Lasts until the next re-target. Any other type reverts.
Zxo	<code>collectFeesFromCoreLP(poolId)</code>	Sends core LP trading fees to the treasury. Liquidity is not touched.
ZxoV3Deployer	<code>startV3pool</code> , <code>finalizeToken</code> , <code>burnTokens</code> , <code>setZxoContract</code>	Seed external DEXes until finalized. ZXO_CONTRACT can be set once.
ZxoTreasury	<code>sweepToken</code> , <code>sweepAll</code> , <code>scheduleDestination</code> , <code>executeDestinationChange</code> , <code>lockDestinationForever</code>	Sweeps to the destination. Destination changes wait 7 days and can be locked permanently.

89

Tokenomics

Each pool has its own reward token, a standard ERC-20 with burn support. The name, symbol and core allocations are set by the owner when the pool is created. Only the Zxo contract can mint, and it only mints in three situations.

supply = coreLP + coreStake	minted to Zxo at creation, locked in positions
+ v3StartAmnt - burnedAtFinalize	seeding, remainder burned
+ Σ claimed rewards	minted on claim (user + referral shares)
- holder burns	

There is no fixed maximum supply. Emissions are bounded instead by the per-pool rate caps (max APR and max 30-day amount), which are fixed at creation.

Where value flows

- **Trading fees (0.01%)** on user positions go to the stakers who own them.
- **Core LP fees** go to the treasury.
- **Referral share** (3–30% of each claim) goes to the staker's referrer, or the treasury by default.
- **Pair tokens** bought by the core LP stay locked in it.
- Zxo charges **no fee on principal** and no deposit or withdrawal fee.

Utility and incentives

- The reward token trades against its pair asset near parity, backed on the upside by the core LP's sell wall.
- Deposits above parity get up to 20% extra reward weight.
- Holders of at least the chain's **HOLDER** threshold get 25% more bonus. The **HOLDER** token and threshold are set per chain; the token is only available on public DEXes.
- Referrers earn a share of the rewards of everyone who names them.

Parity is a design target, not a peg

The core LP only sells above parity. There is no redemption mechanism and no protocol buy support below parity. The price can fall below the stake range, and while it does, claims are paused.

Planned core LP sizes range from about \$5 million to \$15 million per pool, with larger pools possible. At those sizes, moving the price from the launch tick into the core-stake range requires absorbing about 7% of the core LP (roughly \$350,000 for a \$5 million core).

10

Governance and owner powers

Zxo has no token voting. Zxo, ZxoV3Deployer and ZxoTreasury each have a single owner address; ZxoOracle has no owner. On both live chains the owner of Zxo and ZxoV3Deployer was a single wallet, the account that deployed them, as of the deployment reports (events up to 1 October 2026 on Arbitrum and 5 October 2026 on PulseChain). The team plans to move ownership to a multisig after all mainnet deployments. Both contracts use two-step transfers, so a new owner must accept before control moves.

The owner can

- Create pools and their tokens, up to 255
- Initialize each pool's core LP and core stake once
- Set the target APR within the pool's cap, including 0
- Set the referral rate between 3% and 30%
- Temporarily lower a pool's reward rate
- Send core LP trading fees to the treasury
- Seed external DEXes for a new token until it is finalized
- Transfer or renounce ownership

The owner cannot

- Touch staked NFTs, deposits or unclaimed rewards
- Block, delay or pause unstaking
- Withdraw or reduce core LP or core-stake liquidity
- Mint reward tokens outside pool creation and emissions
- Burn anyone's tokens
- Raise a pool's max APR or 30-day cap after creation
- Change ranges, fee tier, DEX or reward token
- Upgrade the contracts or rescue any asset Zxo holds

If ownership is renounced, pools keep running with their last settings. No new pools can be created and core LP fees stay uncollected in the positions.

Treasury. The treasury owner can sweep tokens to a destination address. Changing that destination takes 7 days, and the destination can be locked permanently.

11

Security

SECURITY REVIEW

8.8_{/10}

Revision 3.1, AI-assisted. 6.6 before remediation, 8.5 after it, 8.8 as deployed on two chains.

FINDINGS

20 RESOLVED 9 ACCEPTED
0 OPEN

Both High and three of four Medium findings fixed. The core-stake weighting is accepted as a design trade-off, and the single-wallet owner is accepted until the multisig.

STILL OPEN

Main items: multisig ownership, a static-analysis run, invariant tests, a USDT-pair fork test before Ethereum, naming the seeding DEXes, and a published independent human audit. The reviews so far are AI-assisted.

11.1 User guarantees

GUARANTEE	WHY IT HOLDS
You can always withdraw after 24 hours	Unstake has no price check and no owner gate. Accounting totals are guarded so a drift can never block exits.
Only you control your position	Every action checks the stored owner. Zxo never approves anyone to move its NFTs.
Your NFT is only released by unstake	Unstake deletes the record before the transfer; no other function moves position NFTs.
Core liquidity cannot be pulled	No function transfers, burns or reduces core positions.

GUARANTEE	WHY IT HOLDS
A failed withdrawal changes nothing	If the NFT transfer fails, the whole unstake reverts and your position stays staked.

11.2 Threat model

THREAT	PROTECTION	STATUS
Reentrancy through tokens, NFTs or recipient contracts	<code>nonReentrant</code> on every mutating entry point; checks-effects-interactions; external call last in unstake	PROTECTED
Owner extracting launch tokens through a fake DEX	Finalization burns the remainder and closes seeding; staking cannot open before it. Users verify <code>isFinalized</code> and each seeding DEX.	MITIGATED
Pool pre-initialized at a wrong price	Core LP init reverts unless the pool price equals the launch price.	PROTECTED
DEX fork with a different <code>sLot0</code> layout	Price read decodes only the two fields every fork shares; failures report unhealthy. Confirmed live on LibertySwap, a PancakeSwap-style DEX.	PROTECTED
Flash swap to harvest while the price is out of range	The health gate also requires the 15-minute average tick in range, which one transaction cannot move	PROTECTED
Wrong or swapped addresses in a chain library	The <code>Zxo</code> constructor and <code>ZxoVerify</code> check every address on-chain; a bad build cannot deploy	PROTECTED
Non-standard tokens such as USDT	SafeERC20 transfers and <code>forceApprove</code> . Fee-on-transfer and rebasing pair tokens are not supported. A USDT pair has not yet been staked on-chain; a fork test is planned before the Ethereum deployment.	PROTECTED
Contract stakers that cannot receive NFTs	<code>to</code> parameter on unstake	PROTECTED
Flash swap to raise one stake's bonus	The bonus uses the deposit's token mix at the spot price. It is capped at 20% (25% with HOLDER), emissions stay capped and principal is not at risk	ACCEPTED
Price history not enlarged on a new pool	Run <code>set-observations.py</code> once per pool. Until then stake and harvest pause; unstake and funds are unaffected	OPERATIONAL
Core stake takes an unclaimed share of emissions	A core stake carries about 2.2 times the reward weight of a user stake of the same value. Stakers earn about 62% of the target APR when user stakes equal the core stake, about 90% at ten times and about 99% at one hundred times. While user stakes are far smaller than the core stake, they earn about half	ACCEPTED
Owner key compromise	Owner cannot reach user funds. The owner is a single wallet on both live chains until the planned multisig; transfers are two-step	OPERATIONAL

11.3 Accepted risks and design choices

- **Spot-based bonus.** The health gate uses both the spot price and a 15-minute average. The stake bonus still uses the deposit's token mix at the current price.
- **Core-stake dilution.** The contract-owned core stake counts toward TVL and reward weight but never claims; its trading fees cannot be collected. What that costs stakers depends on its size beside theirs. On PulseChain the core stakes were worth about \$330 to \$1,030 per pool at 5 October 2026 prices, so the gap closes quickly as a pool fills. On that date most PulseChain pools still held only test-size stakes, where stakers earn about half of the target APR.
- **Single-wallet owner.** Until the multisig, one key can create pools, change target APRs and referral rates, and collect core LP fees. It cannot touch staked positions, block unstaking or withdraw core liquidity.
- **Bonus curve.** The bonus reaches its cap about 20 ticks above parity, by design.
- **Deadlines.** Mint deadlines are set at execution time; price protection comes from the slippage minimums.
- **No rescue path.** Dust below the refund threshold and NFTs sent directly to Zxo cannot be recovered. This keeps the owner away from every asset Zxo holds.
- **Referrers.** Any address can be a referrer. Naming the Zxo contract loses that user's referral share; naming yourself skips the split.
- **Holder bonus.** The HOLDER token can be bought, used for a stake and sold again. This is intended.
- **Forfeiture.** Unstaking forfeits pending rewards. Rewards are paid only through harvest, while the pool is healthy.

11.4 Compiler

Contracts are compiled with solc 0.8.36 on the legacy pipeline (via-IR off), optimizer 200 runs, EVM version Shanghai. The three known bugs listed for 0.8.36 require via-IR, recursion, custom errors in `require`, or deleting an element of a memory `bytes` array. Zxo uses none of these.

11.5 What the reviews saw on mainnet

Confirmed from verified source and on-chain records

- All eleven contracts on Arbitrum and PulseChain are exact-match verified on Sourcify and are the audited code. The only differences are the license header, the chain library import and the chain settings.
- Six pools on Arbitrum and nine on PulseChain launched in the enforced order: every reward token was finalized before its core stake.
- A live reward claim was reproduced to the last wei from the formulas in section 6, on both chains.
- The HOLDER bonus applied at exactly 1.25 times on PulseChain.
- Stake, harvest and unstake ran on both chains. Nine unstakes succeeded after the one-day lock.

Read this with care

- The reviews are AI-assisted. They do not replace an independent human audit, which is still recommended.
- Most early activity was the team's own mainnet testing: all of it on Arbitrum up to 1 October 2026, and 94 of 100 transactions on PulseChain up to 5 October 2026.
- No contract state was read directly from a node. The owner and pool settings were inferred from events, and ZxoTreasury's owner and destination were not reviewed on-chain.
- No fuzzing, static analysis or fork tests were run by the reviews.

Sources: Zxo Protocol Security Audit, Revision 3.1; Zxo Arbitrum Deployment Statement, 2 October 2026; Zxo PulseChain Deployment Report, 5 October 2026. All three are published in full at zxo.farm/docs/audit.

12

Gas efficiency

CHOICE	SAVES
Bare <code>revert()</code> without messages	Bytecode and deployment gas
One shared event with an action code, emitted through one internal function	Bytecode; keeps Zxo under the 24,576-byte limit
Deployment checks in the constructor	Readable errors with no runtime bytecode cost
The oracle is read only when the spot check passes	An external call on unhealthy pools
Packed structs (uint96 rate, uint40 times, uint16 settings) and one packed config word per pool	Storage reads and writes
Rewards minted on claim	No reward funding transfers or balance bookkeeping
Chain addresses as compile-time constants	No storage reads for external addresses
Dust below thresholds kept instead of refunded	A token transfer on most stakes

CHOICE	SAVES
Rate re-targeting at most once per hour	Repeated storage writes
Bounded loops (21 positions per address, 255 pools)	Predictable worst-case gas
uint8 pool ids, unchecked counters where overflow is impossible	Arithmetic overhead

13

Interfaces and dependencies

DEPENDENCY	USED FOR
OpenZeppelin Contracts 5.x	ERC20, ERC20Burnable, Ownable, Ownable2Step, ReentrancyGuard, SafeERC20, IERC721, IERC721Receiver, IERC20Metadata
Uniswap V3 NonfungiblePositionManager (or fork)	<code>createAndInitializePoolIfNecessary</code> , <code>mint</code> , <code>collect</code> , NFT custody
Uniswap V3 Factory (or fork)	<code>getPool</code> for the pair at fee 100
V3 pool	<code>slot0</code> read by low-level call (price and tick only); <code>observe</code> read by ZxoOracle for the 15-minute average; <code>increaseObservationCardinalityNext</code> at launch
HOLDER token	<code>balanceOf</code> for the holder bonus
Standards	ERC-20, ERC-721 and the ERC-721 receiver hook

Project interfaces: `IZxo`, `IZxoToken`, `IZxoV3Deployer`, `IZxoOracle`, `Iv3PosManager`, `IUniswapV3`.

Libraries: `ZxoLib` (ticks, bonus, packing, validation) and one `ChainLib` per chain.

14

Deployment

14.1 Networks

NETWORK	CHAIN ID	DEX	POSITION MANAGER	STATUS
PulseChain	369	LibertySwap	0x19b1347900840e7a299f339868881750918fAD91	LIVE since 3 Oct 2026
Arbitrum	42161	Uniswap V3	0xC36442b4a4522E871399CD717aBDD847Ab11FE88	LIVE since 28 Sep 2026
Ethereum	1	Uniswap V3	0xC36442b4a4522E871399CD717aBDD847Ab11FE88	PLANNED

NETWORK	CHAIN ID	DEX	POSITION MANAGER	STATUS
Optimism	10	Uniswap V3	0xC36442b4a4522E871399CD717aBDD847Ab11FE88	PLANNED
Polygon	137	Uniswap V3	0xC36442b4a4522E871399CD717aBDD847Ab11FE88	PLANNED
Base	8453	Uniswap V3	0x03a520b32C04BF3bEEf7BEb72E919cf822Ed34f1	PLANNED
BNB Chain	56	PancakeSwap V3	0x46A15B0b27311cedF172AB29E4f4766fbE7F4364	PLANNED
Unichain	130	Uniswap V3	0x943e6e07a7E8E791dAFC44083e54041D743C46E9	PLANNED
Robinhood Chain	4663	Uniswap V3	0x73991a25C818Bf1f1128dEAaB1492D45638DE0D3	PLANNED
PulseChain Testnet	943	Test	configured for testing	TESTING

Only use Zxo contract addresses from section 14.2 or from the Contract Sources page at zxo.farm/docs/contracts.

14.2 Live deployments

These are the Zxo contracts on the two live chains, as recorded in the deployment reports. Each address links to its verified source on Sourcify, which reports an exact match of creation and runtime bytecode. The current list, including pools added after this paper, is kept at zxo.farm/docs/contracts.

PulseChain · chain id 369 · deployed 3 October 2026

CONTRACT	ADDRESS (LINKS TO SOURCIFY)	BLOCK	BYTECODE
Zxo Farm, staking, rewards, core LP	0x813984963d4e0CC7fF5d04E830767fe63D815B0A	27704394	EXACT MATCH
ZxoV3Deployer Seeds other DEXes, then finalizes	0x3D62C13A12F246A2CAC2d7fCe1A43D63ED253FBE	27704349	EXACT MATCH
ZxoOracle 15-minute average price reader	0xC8CaA5F5961d4679C79A18731C53b6023Ac8862B	27704319	EXACT MATCH
ZxoTreasury Fee and referral receiver	0x923cbe1C05e9c0014D6514a506A8E74B9254FF86	27704288	EXACT MATCH
ZxoVerify (first run) Configuration check before the deployer	0xa8D7A6A9291E60400F0CA64c5dfed64997Cd3817	27704333	EXACT MATCH
ZxoVerify (second run) Configuration check with the deployer	0x923028273B38b24fb98F846f0e4c809fAB58523f	27704375	EXACT MATCH

Arbitrum One · chain id 42161 · deployed 28 September 2026

CONTRACT	ADDRESS (LINKS TO SOURCIFY)	BLOCK	BYTECODE
Zxo Farm, staking, rewards, core LP	0x5356B3004D0394dEA51161d175265EC8F92c96b7	509563331	EXACT MATCH
ZxoV3Deployer Seeds other DEXes, then finalizes	0xb6CB455CB67d59DD269C865428acece75eA08594	509561805	EXACT MATCH
ZxoOracle 15-minute average price reader	0x98c9695bb012A42FdcB734324A78E63eE6c23D44	509547888	EXACT MATCH
ZxoTreasury Fee and referral receiver	0x0341A7a6CC031662526492e10796E551125AE2bc	509546642	EXACT MATCH

On Arbitrum no ZxoVerify address was provided to the review; the Zxo constructor repeats the same checks. Each chain's contracts were deployed by one externally owned account, which was also the owner of Zxo there as of the reports (inferred from events):
0x98C3eEc2b28D82b05edfA3D40F8A468a58BD1e24 on PulseChain and 0xFF9623ABe5D081c3c4E586330ab714FcB718fa53 on Arbitrum.

Reward tokens on PulseChain

POOL	REWARD TOKEN	PAIRS WITH	ADDRESS	SOURCE
0	ZXOWPLS	WPLS	0x4542fD562bBa19AE85e017d5b50Be7C4b16443DF	VIA THE ZXO BUILD
1	ZXOHEX	HEX	0x27381e9925B58133136297d7dd79dab1F2A1dD32	VIA THE ZXO BUILD
2	ZXOPLSX	PLSX	0x3647464C8e72CdB4Ea3Fc58002D2A1C9E0fe2daF	VIA THE ZXO BUILD
3	ZXOINC	INC	0x2dE787C0AB8DeCb4167d43ee00A0eE4d2e3f6f9E	VIA THE ZXO BUILD
4	ZXOPRVX	PRVX	0xbcE06Bb11FF7fEb628eE80589BC70206085A63Be	VIA THE ZXO BUILD
5	ZXOINT	INTARSIA	0x518bC3Ce004a6d6f074eD24cE0e6e730786aE99f	VIA THE ZXO BUILD
6	ZXOFARM	ZXO	0x8fA5C2a7CD2974b5b817136A986bf5475F310887	VIA THE ZXO BUILD
7	ZXOPCOCK	PCOCK	0x3BfefFA660696417b8068E19E8De1456211063d5	VIA THE ZXO BUILD
8	ZXOPDAI	DAI (pDAI)	0x204250574e7c7BE7fc6AEEdD38c365DA651695eaF	VIA THE ZXO BUILD

Reward tokens on Arbitrum One

POOL	REWARD TOKEN	PAIRS WITH	ADDRESS	SOURCE
0	ZXOWETH	WETH	0x304441b6BDADA52d41f4AA0A0b13dC4185B3979B	EXACT MATCH
1	ZXOPEPE	PEPE	0x113CA9e137ee2FD5bB89C979255010BBb3997D4f	VIA THE ZXO BUILD
2	ZXOARB	ARB	0x387BAfD20705c33Ff04A392C125BDeB39DFc6f98	VIA THE ZXO BUILD

POOL	REWARD TOKEN	PAIRS WITH	ADDRESS	SOURCE
3	ZXOUSDC	USDC	0x1BAbE35Eb60B5bdd4E4299EBdc6d511E29F0E906	VIA THE ZXO BUILD
4	ZXOLINK	LINK	0x3Ec1c1ED083156c2B33BF49550eCc9390e1b50eB	VIA THE ZXO BUILD
5	ZXOWBTC	WBTC	0x0cc4C9B093223E95d93FF10E6C6CdEae14579258	VIA THE ZXO BUILD

Reward tokens are ZxoToken instances created by `Zxo.createPool` from the token code inside the verified Zxo build. Only the Arbitrum pool 0 token was submitted to Sourcify on its own; the others link to the chain's block explorer. Pools as of 5 October 2026 on PulseChain and 1 October 2026 on Arbitrum.

Addresses compiled into each deployment

CHAIN	CONSTANT	WHAT IT IS	ADDRESS
PulseChain	<code>POS_MANAGER</code>	LibertySwap position manager	0x19b1347900840e7a299F339868881750918FAD91
PulseChain	<code>FACTORY</code>	LibertySwap V3 factory	0x796fcbDC956b85797EF21145Aa97599B7FB36a6
PulseChain	<code>HOLDER_TOKEN</code>	HOLDER token (DYNASTY)	0xdb3fF08284f0874860281b3A84a0D48ab25040d6
Arbitrum	<code>POS_MANAGER</code>	Uniswap V3 position manager	0xC36442b4a4522E871399CD717aBDD847Ab11FE88
Arbitrum	<code>FACTORY</code>	Uniswap V3 factory	0x1F98431c8aD98523631AE4a59f267346ea31F984
Arbitrum	<code>HOLDER_TOKEN</code>	HOLDER token	0x6F296078e80907E23014a619C83d98912E35E66A

These are third-party contracts, not Zxo contracts. `HOLDER_THRESHOLD` is 3,690,000,000 tokens on both chains.

14.3 Deployment order

STEP	ACTION
1	Deploy <code>ZxoTreasury</code> ; set <code>ZX0_TREASURY</code> in the chain library
2	Deploy <code>ZxoOracle</code> ; set <code>ZX0_ORACLE</code> in the chain library
3	Deploy <code>ZxoVerify(false, true)</code> to check the factory, position manager, HOLDER token and threshold, treasury and oracle on-chain
4	Deploy <code>ZxoV3Deployer</code> ; set <code>ZX0_V3_DEPLOYER</code> in the chain library
5	Deploy <code>ZxoVerify(true, true)</code> to confirm the deployer as well
6	Deploy <code>Zxo</code> (its constructor repeats the checks); call <code>ZxoV3Deployer.setZxoContract(Zxo)</code> once
7	Transfer ownership to the multisig. Not yet done on Arbitrum or PulseChain; the team plans it after all mainnet deployments
Per pool	After the core LP is initialized, run <code>set-observations.py</code> once so the 15-minute average is available

On PulseChain the on-chain order was exactly steps 1 to 6, with both ZxoVerify runs. On Arbitrum it was ZxoTreasury, ZxoOracle, ZxoV3Deployer, then Zxo.

14.4 Verification

The build compiles each contract once with `solc --standard-json`. The deployed bytecode and the explorer's verification input come from that single compile, and the input embeds every source file under its import name, so any explorer can reproduce the bytecode without local files. After deploying, the deploy script compares the on-chain code with the build (immutable values masked) and records the deployment.

Anyone can check a deployment by compiling the published `standard-json-input.json` with solc 0.8.36 and comparing the runtime bytecode with the chain. This has been confirmed on mainnet: all eleven contracts on Arbitrum and PulseChain are exact-match verified on Sourcify from that input, built with solc 0.8.36, optimizer 200 runs, EVM version Shanghai and via-IR off.

15

Legal and compliance

License. The contracts are released under the Business Source License 1.1. Copying, modification and non-production use are permitted. Production use requires a commercial license from the licensor. On the Change Date written in each deployment's published source (28 September 2029 for Arbitrum and 3 October 2029 for PulseChain, or the fourth anniversary of a version's first public release if earlier), the license converts to GNU GPL v3.0 or later.

No offer. This whitepaper describes software. It is not an offer or solicitation to buy or sell any token, security or financial product, and it is not investment, legal or tax advice.

Risk. Using Zxo involves smart contract risk, DEX and pair-token risk, price risk (including impermanent loss on concentrated positions), and network risk. Rewards and APR figures are variable and not guaranteed. Tokens can lose some or all of their value.

Jurisdiction. Users are responsible for complying with the laws of their jurisdiction, including rules on digital assets and taxes.

Software as is. To the extent permitted by law, the software is provided without warranties of any kind.

Roadmap

Milestones are planned and may change. Dates will be announced at Zxo.Farm.

Phase 1 • Protocol and review COMPLETE

- Core contracts, chain libraries and deployment verifier
- Security review and remediation with no open findings, now at Revision 3.1 (8.8/10)
- 15-minute average price gate (ZxoOracle) and on-chain deployment checks
- Reproducible build, verification and deployment tooling

Phase 2 • Testing IN PROGRESS

- Done: contracts deployed on PulseChain Testnet with test pools (not covered by a deployment report)
- Open: invariant tests, and a fork test of a USDT pair before the Ethereum deployment
- Open: static analysis (Slither or Aderyn) and a published independent human audit
- Dapp trust panel: finalization, seeding DEXes, burned NFTs

Phase 3 • Mainnet launch LIVE

- Arbitrum One live since 28 September 2026, on Uniswap V3
- PulseChain live since 3 October 2026, on LibertySwap
- All eleven deployed contracts exact-match verified on Sourcify
- Open: move ownership of Zxo and ZxoV3Deployer to a multisig

Phase 4 • Multichain

- Ethereum, Base, Optimism, Polygon, BNB Chain, Unichain and Robinhood Chain
- Set each chain's HOLDER token and threshold before it is deployed

Phase 5 • Growth

- Additional pools per chain, up to 255 per deployment
- Seeding new V3 DEXes for new pools as they launch

APPENDIX

Reference

A. Constants

CONSTANT	VALUE	MEANING
LP_FEE_TIER	100	0.01% V3 fee tier
DEFAULT_TARGET_APR	1,000	10% initial target

CONSTANT	VALUE	MEANING
MAX_NORMAL_APR	32,000	320% max per pool (×2 in year one)
BONUS_APR_ENDS	365 days	Doubled APR cap window
RATE_COOLDOWN	1 hour	Minimum time between rate changes
MIN_REF_RATE_BPS / MAX_REF_RATE_BPS	300 / 3,000	Referral share 3% – 30%
MAX_BONUS_BPS	2,000	20% bonus cap (25% with HOLDER)
BONUS_THRESHOLD	0.5×10^{18}	Bonus starts above 50% paired
HOLDER_THRESHOLD	per chain, $\geq 10^{18}$	HOLDER balance for +25% bonus (ChainLib)
ZxoOracle.WINDOW	900 s	15-minute average for the health gate
MAX_USER_STAKES	21	Positions per address
MAX_SLIP	5,000	50% max slippage
MIN_UNSTAKE	10^{12}	Pending above this needs forfeit
MIN_DUST_18 / _8 / _6	10^{12} / 10 / 1,000	Smallest refund sent back
Unstake lock	1 day	From stake time
PRECISION	10^{18}	Reward accumulator scale

B. Tick ranges for every pair type

PAIR	ZX0 IS	LAUNCH TICK	CORE LP	CORE STAKE	USER STAKE
18 / 18	token0	10	[11, 300]	[31, 300]	(-300, 300)
18 / 18	token1	-10	[-300, -11]	[-300, -31]	(-300, 300)
18 / 6	token0	-276,314	[-276,313, -276,024]	[-276,293, -276,024]	(-276,624, -276,024)
18 / 6	token1	276,314	[276,024, 276,313]	[276,024, 276,293]	(276,024, 276,624)
18 / 8	token0	-230,260	[-230,259, -229,970]	[-230,239, -229,970]	(-230,570, -229,970)
18 / 8	token1	230,260	[229,970, 230,259]	[229,970, 230,239]	(229,970, 230,570)

C. Code references

TOPIC	LOCATION
Staking	<code>Zxo.stake</code> , <code>ZxoLib._requireStakeValidations</code>
Bonus and TVL	<code>ZxoLib._calcBonusAndTotal</code>
Rewards	<code>Zxo._updatePool</code> , <code>Zxo._pendingRewards</code> , <code>Zxo._claimRewards</code>
Emissions	<code>Zxo._performAutoAdjust</code> , <code>Zxo.updatePoolConfig</code>
Health	<code>Zxo._readSlot0</code> , <code>Zxo._poolRatio</code> , <code>ZxoOracle.twapTick</code> , <code>ZxoOracle.inRange</code> , <code>ZxoLib._getStakeTick</code>
Deployment checks	<code>Zxo</code> constructor, <code>ZxoVerify</code>
Launch	<code>Zxo.createPool</code> , <code>Zxo.initializePool</code> , <code>ZxoLib._getCoreTicks</code> , <code>ZxoV3Deployer.startV3pool</code> , <code>ZxoV3Deployer.finalizeToken</code>
Config packing	<code>ZxoLib.packPoolConfig</code> and <code>unpack*</code>

D. Glossary

Tick	A 0.01% price step in a V3 pool. Price = 1.0001^{tick} .
Concentrated liquidity	Liquidity placed only within a chosen price range, earning fees only while the price is in that range.
Position NFT	The ERC-721 token a V3 position manager issues for each liquidity position.
Parity	One reward token equals one pair token in value.
Core LP	The protocol's permanently locked, single-sided reward-token position above parity.
Core stake	A contract-owned staked position that opens staking for a pool.
Healthy	The pool's current tick and its 15-minute average tick are both strictly inside the user stake range.
TWAP	Time-weighted average price. Zxo uses the average tick over the last 15 minutes.
Observation cardinality	How many price-history slots a V3 pool keeps. Raised once per pool so the 15-minute average is always available.
Weight	A position's liquidity plus any bonus. Rewards are shared by weight.
accRewardPerShare	Running total of rewards per unit of weight, scaled by 10^{18} .
Basis point (bps)	0.01%. 10,000 bps = 100%.
TVL	Sum of staked deposits in 18-decimal units, counted 1:1.
HOLDER token	A separate token, set per chain, whose holders at or above the chain's threshold get a larger stake bonus.
Seeding	Small single-sided positions on other V3 DEXes that set the initial price there.
Finalization	Burning the deployer's remaining launch tokens and closing seeding for that token.
Forfeit	Giving up pending rewards when unstaking without harvesting first.
ChainLib	The per-chain library of addresses compiled into the contracts.

This document describes the Zxo contracts as reviewed in Security Review Revision 3.1 and as deployed on Arbitrum One and PulseChain. The deployed code is the final authority; where this paper and the code differ, the code applies.