Nanite Token Whitepaper

Token Specification

Version 1.0

Abstract

Nanite is a game utility token built on Base using Uniswap V4 hooks. 100% of supply begins in the liquidity pool with no insider allocations. Community claims are accumulated through trading fees and are backed by ETH from the bonding curve. After claims are filled, the mechanism transitions to deflationary burns.

---

1. The Problem with Traditional Token Launches

In most token launches, tokens exist outside the liquidity pool that have no ETH backing.

Typical distribution:

- 50-70% in liquidity pool
- 15-25% team allocation
- 10-15% airdrop/community
- 5-10% advisors/marketing

The problem:

When team, airdrop, or advisor tokens are sold, they extract ETH that buyers deposited. These tokens were never purchased—they're claims against buyer liquidity with no backing.

```None
Pool: 100 ETH + 500M tokens (50% of supply)

Team sells 200M tokens

→ Team extracts ~40 ETH
→ Remaining buyers' tokens now backed by only 60 ETH
```

Early buyers lose value when insiders sell, even without malicious intent.

---

2. Nanite Distribution

Nanite addresses this with a different distribution model.

Distribution:

- 100% of tokens start in the liquidity pool
- 0% team allocation (team earns ETH fees)
- 0% pre-minted airdrops
- Claims are accumulated from trading, fully backed

The result:

Every token in circulation was either:

1. In the pool from genesis, or
2. Purchased through the bonding curve with ETH

No unbacked tokens exist outside the pool.

---

3. Token Mechanics

3.1 Supply

Parameter	Value
Total Supply	1,000,000,000 NANITE
Decimals	18
Initial Pool Allocation	100%

Team Pre-allocation	0%

3.2 Trading Fee Structure

Anti-Snipe Protection: Fee starts at 99% and drops 1% per minute until reaching 10% (89 minutes).

Time After Launch	**Fee**
Minute 0	99%
Minute 1	98%
Minute 10	89%
Minute 50	49%
Minute 89+	10%

Fee Distribution (of total fee collected):

Allocation	**Share**	**At 10% Fee**
Claims/Burn	50%	5% of trade
Reward Pool	20%	2% of trade
Team	20%	2% of trade
Treasury	10%	1% of trade

3.3 Transfer Restrictions

Nanite uses Uniswap V4 hooks to enforce transfer restrictions:

- Tokens can only be transferred through the official pool or by whitelisted distributors
- No secondary markets or OTC trades possible
- All trading goes through the bonding curve

This ensures the fee mechanism cannot be circumvented.

4. Three-Phase Tokenomics

Phase 1: Anti-Snipe Burns (First 89 Minutes)

During the anti-snipe period (fee > 10%), 50% of the fee is used to buy and burn tokens. **No tokens go to claims during this phase.**

```None
Trade occurs → Total fee taken (99% down to 10%)

     → 50% of fee buys and burn NANITE from pool.

     → Other 50% split: rewards (20%), team (20%), treasury
     (10%)
```

At launch, prices are low. Burning during this period prevents cheap tokens from flooding claims.

Phase 2: Claim Accumulation (Fee at 10%)

Once the fee reaches 10% (after 89 minutes), claims begin accumulating:

```None
Trade occurs → 10% total fee taken

     → 2.5% (25% of the fee) buys and burns NANITE from
     pool

     → 2.5% (25% of the fee) goes to the Claim contract

     → Other 5% split: rewards (2%), team (2%), treasury
     (1%)

     → Repeat until Claims Contract has 100M tokens (10% of
     total supply)
```

Phase 3: Deflationary Burns

Once the Claims Contract holds 100M tokens (10% of supply), the mechanism switches to full burning.

```None
Trade occurs → 10% total fee taken

    → 5% (50% of the fee) buys and burns NANITE from pool

    → Other 5% split: reward (2%), team (2%), treasury
    (1%)
```

The transitions are automatic and irreversible.

---

5. The Claims System

5.1 How Claims Work

Eligible users can claim tokens from the Claims Contract based on:

- Merkle proof of eligibility
- Pro-rata share of total allocation
- Unlock schedule (optional vesting)

5.2 Claim Eligibility

Claim eligibility is determined by snapshot criteria such as:

- Regular Punks NFT holders
- Early community members
- Game participants
- Other ecosystem contributors

5.3 Why Claims Don't Break the Model

Traditional airdrops are unbacked—free tokens that extract buyer ETH when sold.

Nanite claims are different:

1. Claim tokens were purchased from the pool
2. The ETH used to purchase them is in the pool
3. When claimers sell, they're selling backed tokens
4. The pool already has the ETH to support these sells

Claims are a redistribution of tokens already purchased from the pool.

5.4 Claim Selling Dynamics

When claimers sell:

- They pay the 10% trading fee
- Price impact works against large sells

---

6. Deflationary Mechanics

6.1 How Burns Increase Backing

When tokens are burned via buy-and-burn:

```
None
Before burn:

  Pool: 10 ETH + 1000 NANITE

  Backing: 0.01 ETH per token



Buy-and-burn with 1 ETH:

  1 ETH enters pool, buys ~91 NANITE

  91 NANITE sent to dead address



After burn:

  Pool: 11 ETH + 909 NANITE
```

```
Backing: 0.012 ETH per token (+20%)
```

Burns increase ETH backing per remaining token.

---

7. Technical Implementation

7.1 Smart Contracts

Contract	Purpose
Nanite.sol	ERC20 token with transfer restrictions
NaniteHook.sol	Uniswap V4 hook for fees and burns
NaniteClaims.sol	Merkle-based claim distribution with 10 periods

NaniteClaims Details

The claims contract manages 10 claim periods, each unlocked when 1% of supply (10M tokens) is deposited:

Period	Unlock Threshold	Cumulative
0	10M tokens	1% of supply
1	20M tokens	2% of supply
...	...	...
9	100M tokens	10% of supply

Each period has its own merkle root, allowing different snapshot criteria per period. Users claim by providing:

- Period number (0-9)
- Amount they're entitled to

- Merkle proof for that period

After 100M tokens deposited (10% of supply), claims complete and the hook switches to burning.

7.2 Uniswap V4 Hook Permissions

The hook implements:

- `beforeSwap` - Takes ETH fee on buys
- `afterSwap` - Takes ETH fee on sells, distributes fees
- `beforeInitialize` - Validates pool configuration
- `afterAddLiquidity` - Controls liquidity additions

7.3 Transfer Restriction Mechanism

Nanite uses transient storage to control transfers:

```
None
// Only hook can authorize transfers

function increaseTransferAllowance(uint256 amount) external {

    if (msg.sender != hookAddress) revert OnlyHook();

    // Set transient allowance

}



// Transfers check allowance

function _afterTokenTransfer(from, to, amount) internal {

    if (to == poolManager || from == poolManager) {

        // Verify transient allowance set by hook

        // Consume allowance

    } else {
        revert InvalidTransfer();
    }
}
```

7.4 Fee Distribution Flow

On Buy (ETH → NANITE):

```None
1. beforeSwap: Take fee from ETH input

2. Swap executes with remaining ETH

3. afterSwap: Distribute fee to claims/burn, reward, team, treasury
```

On Sell (NANITE → ETH):

```None
1. Swap executes normally

2. afterSwap: Take fee from ETH output, distribute
```

---

8. Game Integration

8.1 Reward Pool

The 2% reward pool allocation accumulates ETH for:

- Tournament prizes
- Gameplay rewards

8.2 In-Game Utility

NANITE serves as the primary currency for:

- **1v1 Fights**: Entry fees for player-vs-player matches
- **Faster Training**: Accelerate stat upgrades for your dog
- **Power Ups**: In-game boosts during fights
- **Resurrection**: Revive your dog after losing a fight or tournament

8.3 Future Burns

Game mechanics may include additional burn events:

- Fight entry fees partially burned
- Training acceleration costs burned
- Power up purchases burned
- Resurrection fees burned

These supplement the automatic 5% trading burns.

---

9. Summary

Nanite starts with 100% of supply in the pool. Community claims are accumulated through actual purchases, so every token in circulation has ETH backing.

The three-phase mechanism:

1. Anti-snipe burns (first 89 minutes)
2. Claim accumulation at half rate
3. Full deflationary burns

Team earns from trading fees. Claims are backed by ETH in the pool.
