



pViTok
POLYGON NATIVE TOKEN

A reward token that cannot be quietly drained.

pViTok funds the reward programs of the VTS ecosystem. Ninety percent of its supply went at genesis into a vault that releases tokens on a linear drip, under a per-project lifetime budget and a rolling annual cap, owned by a two-day timelock behind a 2-of-3 multisig. The limits are contract code, not policy.

NETWORK	STANDARD	DECIMALS	TOTAL SUPPLY
Polygon	ERC-20	12	1B
GOVERNED	TIMELOCK		
900M	48h		

What pViTok is

pViTok is a Polygon-native ERC-20 token created to pay the reward programs of the VTS ecosystem — daily login rewards, NFT staking emissions, campaign payouts — without depending on a bridge and without depending on anyone's promise to behave.

The interesting part is not the token. An ERC-20 with a mint function is a weekend's work, and its supply is worth exactly as much as the discipline of whoever holds the key. The interesting part is the machinery around it: a treasury contract that enforces per-project budgets, linear vesting, and rolling annual caps as preconditions on every transfer, owned by a timelock that turns every privileged action into a public, two-day, multisig-signed proposal.

The design goal was narrow and testable: **make it impossible for any single party — including the operators — to accelerate emissions or sweep a committed budget**, and make every loosening of that constraint slow enough that holders can watch it coming.

A NOTE ON DECIMALS

pViTok has **12 decimals, not 18**. Every amount in the codebase is expressed in the token's own base units, so `parseEther` and `formatEther` are wrong here — and wrong silently, by a factor of one million. Integrators should read decimals from the contract rather than assuming the ERC-20 default.

Not the Ethereum ViTok

This must be stated before anything else, because the names are close enough to mislead.

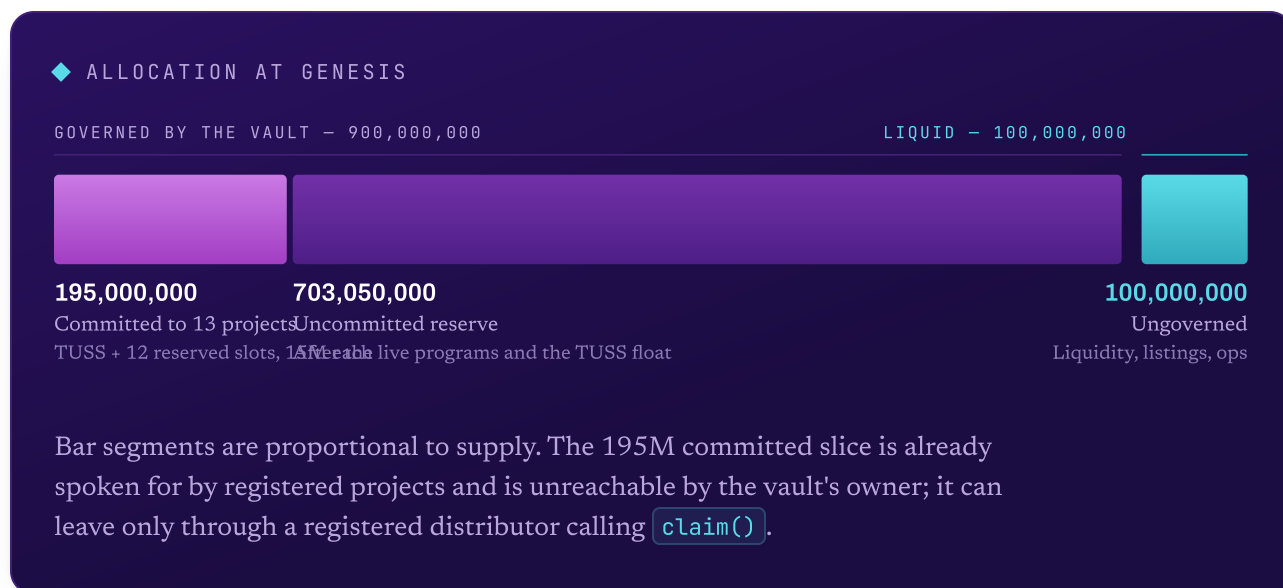
pViTok is not the same fungible asset as the ViTok token on Ethereum mainnet. They are unrelated on-chain. Holding one gives you no claim on the other, no redemption right, and no exposure to its price. There is no bridge, no peg, and no wrapper relationship between them.

The Ethereum-side ViTok has no native contract presence on Polygon – verified by querying for contract code at the same address on both Polygon and Base before any of this was built, and finding none. Bringing it across would have required either a hand-rolled lock-and-mint bridge – among the highest-risk categories of smart contract there is, and not a sensible thing to write for an unaudited project – or Polygon's official PoS bridge, which needs the original token's owner to register the mapping and real holders to move real supply across.

So pViTok was deployed as a separate, same-brand, Polygon-native token instead. It is deliberately named **pViTok** rather than ViTok: reusing the symbol of a token traded elsewhere would invite it to be mistaken for that asset in a wallet or on a block explorer, which is a problem whether or not anyone intended it. The 12-decimal choice matches the Ethereum-side token for brand and UX consistency only, and implies no on-chain link.

Supply

One billion pViTok were minted at deploy and split once, in the same transaction sequence that handed the token to its timelock. There has been no mint since, and any further mint is itself a timelock proposal.



The governed 900M

Nine hundred million pViTok went into the reserve vault at genesis. Of that, 195,000,000 was committed at deploy across thirteen projects — the TUSS collection plus twelve reserved placeholder slots at 15,000,000 each — and a further 1,500,000 was committed on 3 September 2026 to the two live distributors (section 06). The uncommitted remainder is 703,050,000 after the 450,000 TUSS milestone float above, and reaching it requires `registerProject`, `increaseBudget`, or `withdrawUnallocated` — each one a timelock proposal with a two-day delay.

The twelve placeholder slots exist so that a future collection does not need a governance fight to get an allocation; the allocation is already ring-fenced. They are registered *inert*: each carries a vesting start one hundred years in the future, so nothing accrues until

governance explicitly activates it. Activation is bounded too – a slot cannot be given a drip shorter than a year, and its start cannot be backdated more than thirty days, so activating a placeholder can never be dressed up as an instant unlock of its budget.

The liquid 100M

One hundred million pViTok went to the owner wallet, liquid and ungoverned. It is the company's operating reserve – held by Vivid Tokenz Studios for business expenses if and when the token becomes liquid. It is not a reward pool and no reward program draws from it. This slice is not subject to the vault's drip or caps, and this document makes no claim that it is. It is the portion a holder should treat as freely movable.

MILESTONE REWARDS ARE A FLOAT ON THE TUSS CONTRACT

TUSS milestone payouts (75 pViTok per accepted tier, so up to 225 per token) are paid from the TUSS contract's own pViTok balance, not by the vault at claim time – the vault pays only registered distributors, and the TUSS contract is not one. That float was drawn from the vault's uncommitted reserve by timelock proposal: 450,000 pViTok on 3 September 2026, enough for 6,000 accepted tiers, topped up the same way when it runs down.

The reserve vault

`RewardReserveVault` is where the governed supply lives. It never pushes tokens. A project's registered distributor contract calls `claim()`, and the vault pays the end user directly, but only after four independent limits all agree on the amount.

◆ FOUR LIMITS, EVERY CLAIM

LIMIT 1 • LOCKED

Committed budgets cannot be swept

The vault tracks `totalCommitted` – the sum of every project's unclaimed budget. Only genuine surplus above that figure is withdrawable by the owner. Registration reverts unless uncommitted reserve fully backs the new budget, so a registered budget is always covered.

LIMIT 2 • ANNUAL CAP

A hard ceiling per rolling year

No project may claim more than its `annualCap` in any rolling 365-day window. This is the backstop that holds even if a drip schedule is set aggressively – and in practice it is usually the binding limit, not the drip.

LIMIT 3 • LINEAR DRIP

Budgets vest over time, not on demand

Each project's lifetime budget vests linearly across its duration. A running project's schedule can only be tightened, never loosened – the vault rejects any reschedule that would vest more at this instant than the old schedule already had.

LIMIT 4 • LIFETIME BUDGET

An explicit ceiling, decided up front

Every project registers with a `lifetimeBudget` it can never exceed. The bound is enforced transitively: vested amount never exceeds the budget, and claims never exceed the vested amount.

Raising a cap is deliberately slow

Caps have to be raisable, or a project that outgrows its allowance is stuck with a migration. But an instant raise would reopen exactly the sweep this vault exists to prevent. So lowering a cap is immediate, while raising one is split into two calls with a thirty-day cooling-off period between them: `proposeCapIncrease`, wait, `executeCapIncrease`.

Both legs are owner-only, so in practice each is *also* a timelock proposal — the thirty days stack on top of the two-day timelock rather than replacing it. The pending value is public the entire time, so holders can see a loosening coming for a month before it lands. Re-proposing restarts the clock, so proposals cannot be stacked to shorten the wait, and a tightening step cancels any pending increase rather than letting a stale proposal silently undo it.

A cap can never be removed. `setAnnualCap(id, 0)` — which would mean "uncapped" — is rejected for any project that currently has a cap, because a one-call disable of the backstop is precisely the escalation the design is built to exclude.

Governance

Ownership of both pViTok and the vault sits with `VTSTimeLock`, a thin subclass of OpenZeppelin's audited `TimeLockController`. Every privileged action – minting, registering a project, raising a budget, sweeping surplus – becomes a two-step operation: a proposer schedules the call, and after the delay elapses, an executor runs it.

This is what removes the single-hot-key risk that a mintable token otherwise carries. No one address can inflate pViTok or drain the vault on a whim, and the delay gives holders a public window in which to see a pending change and react to it.

◆ ROLE SET — FIXED PERMANENTLY AT DEPLOY

ROLE	HELD BY	CAN DO
Proposer / Executor	Gnosis Safe, 2-of-3	Schedule and execute proposals after the delay
Guardian	Dedicated canceller key	Cancel a scheduled operation – and nothing else
Admin	renounced	No external admin. The timelock administers itself, so the role set can only change through its own proposal path – Safe proposes, 48 hours pass, Safe executes – never by a single key
Deployer	no role, no tokens	Nothing

`MIN_DELAY` is 172,800 seconds – 48 hours. The deploy script renounces the deployer's admin role as its final step, which is why no outside key can alter this table; only a proposal that itself waits 48 hours can. The Safe also holds the canceller role, as OpenZeppelin grants it to every proposer – the guardian is the one that holds *only* that role.

The multisig

The proposer/executor Safe is a 2-of-3 running Safe v1.4.1 on canonical singleton and fallback-handler contracts. Its configuration was audited against mainnet after setup: threshold 2, three owners, **zero modules**, and no transaction guard. The module check is the one that matters most – a Safe module can move funds bypassing the threshold entirely, which would void the 2-of-3 guarantee while leaving it looking intact.

Why a separate guardian

The guardian holds only the canceller role. It cannot propose and cannot execute; its single power is to veto a scheduled operation during the delay window. Splitting veto from proposal means that a compromise of the Safe does not automatically carry the ability to push a malicious call all the way through unopposed.

Reward programs

Each reward stream is a separate distributor contract registered as a vault project. No pViTok ever sits in a distributor — the vault pays the claimer directly, and the distributor's job is only to decide who is eligible and for how much. A distributor that misbehaves is bounded by its project's budget, drip and cap, not by the vault's whole balance.

LIVE **TUSS staking allocation** — 15,000,000 lifetime budget registered at deploy, 5-year drip, 1,500,000/yr cap.

LIVE **12 reserved slots** — `RESERVED_01` ... `RESERVED_12`, 15,000,000 each, inert until governance activates them.

LIVE **Daily login rewards** — `DailyRewardV1`, deployed 3 September 2026; 1,000,000 lifetime budget, 5-year drip, 150,000/yr cap.

LIVE **Social follow campaign** — `SocialFollowRewardV1`, deployed 3 September 2026; 500,000 lifetime budget, 1-year drip, 150,000/yr cap.

Daily login

A holder of the TUSS collection may claim pViTok once per 24 hours — a base rate, plus a streak bonus of 5% per consecutive day up to seven days, so day one pays 0.020 and day seven pays 0.026. "Logging in" on-chain means sending the claim transaction, so the reward is trustless and needs no backend at all. A streak counter continues if you return within 26 hours, which is deliberately wider than the 24-hour claim interval so that claiming at hour 25 is not punished — but not so wide that skipping a day keeps the bonus.

The NFT gate is the anti-sybil mechanism. Logging in is free, so an ungated daily reward is farmed by scripted wallets — one person runs five hundred of them and emission is unbounded. Requiring a held NFT puts a real acquisition cost on every identity. The

bound is honest about its own limit: nothing stops one wallet claiming, transferring its NFT, and claiming again from a second wallet. The NFT is the scarce thing, not the wallet, so a collection of N tokens permits at most N claims per day however they are shuffled. That is the intended bound.

PARAMETER	VALUE	NOTE
Default rate	0.02 pViTok/day	Owner-settable
Hard rate ceiling	1 pViTok/day	Contract constant; a fat-fingered setter cannot exceed it
Claim interval	24 h	Per wallet
Streak window	26 h	Grace period before a streak resets
Streak bonus	+5%/day, max 7	Owner-settable up to 10%/day
Lifetime budget	1,000,000	Carved from uncommitted reserve at registration
Annual cap	150,000	The binding limit
Drip	5 years	Primed 30 days at registration

Social follow

Follow VTS on X, Facebook, Instagram, or TikTok and claim 5 pViTok, once per platform, per person – 20 pViTok for someone who follows on all four. Platforms are held as labels rather than an enum, so governance can add a fifth with a single call instead of a redeploy and a migration. The whole campaign is also capped at 5,000 pViTok per day, independently of the vault's annual cap.

A follow happens on the platform's servers, not on Polygon, so no contract can observe it. The check is off-chain and its *result* is carried on-chain: a follow is confirmed off-chain – today by a person checking the submitted handle against our own follower list, because no platform we pay for exposes a follow check to a commercial application (section 09) – and

the backend then signs an EIP-712 attestation naming the wallet, the platform, and the follower. The contract verifies that signature and pays. The signing key is trusted for exactly one claim – "this social account follows us" – and for nothing else. It cannot change the amount, re-pay a wallet, or reach anything else the vault governs. Because the wallet is inside the signed payload, a relayer can submit the claim and still cannot redirect the payment.

Two sybil axes are closed on-chain. A wallet is paid once per platform, forever, with no reset. And an attested follower is spent on first use, so one X account cannot be walked across a series of fresh wallets. Neither stops someone from registering five hundred real social accounts – that cost is the actual bound, which is why the amount is small and the annual cap is the hard ceiling on the whole campaign.

THE FOLLOWER IDENTIFIER IS A SALTED HASH

The attestation carries `keccak256(platform user id ++ secret server salt)`, never a raw id. It becomes a permanent public record, and platform user ids are small and enumerable – an unsalted hash would be a reversible, permanent map from wallet address to real social account. The salt is what prevents this from publishing an identity graph.

Deployed contracts

Token, vault and timelock were deployed to Polygon mainnet on 18 August 2026; the two reward distributors on 3 September 2026. All five contracts are verified on Polygonscan.

◆ POLYGON MAINNET

CONTRACT	ADDRESS
pViTok	0xBBbAF3D60ea0e3ab89A65bbb5d522E29057475f7
RewardReserveVault	0x17BEc4BeCBB9B8de8351d65aA09A5c366F929C7d
VTSTimelock	0x01804BbAB42859915c856AD28aD5Fc0DA47A35b7
DailyRewardV1	0x22016CBCb368764481b484610118A132A66bCe3f
SocialFollowRewardV1	0x1Dec6642fdEcbe69cC183E419338e0d11Eeb800f

The distributors are owned by the operations Safe rather than the timelock, deliberately, so `pause()` and the rate knobs are a Safe transaction rather than a two-day proposal. Verifying the timelock required naming the contract explicitly – it is a thin subclass, so its bytecode also matches OpenZeppelin's `TimeLockController` and the explorer refuses to guess between them.

Security posture

The vault's guarantees are only worth the paths that enforce them, so those paths carry the test suite's weight: that the owner cannot sweep a committed budget, that a drip cannot be loosened for a running project, that a cap cannot be removed or raised without the cooling-off period, and that a stream fails loudly rather than paying nothing when it is out of budget. The suite currently runs **112 passing tests** across the token, vault, governance, and both distributors.

The verifier key

The social-follow backend signs unattended on a server, which makes its key a hot key. If it leaks, the holder can mint attestations for wallets they control and drain that stream up to its rolling annual cap. **That cap is the blast radius**, and it is sized for the leak rather than for expected uptake.

The response is `pause()` first, then rotate to a fresh key. Both are owner-only, and that contract's owner is the operations multisig rather than the timelock that owns the vault — deliberately, so stopping the bleed is a Safe transaction rather than a two-day proposal. Attestations carry deadlines, so a leaked batch of otherwise-legitimate ones expires on its own.

What the timelock does not protect

The liquid 100,000,000 is ungoverned by construction. It is not behind the timelock, not subject to a drip, and not capped. Anyone assessing counterparty risk should treat that slice as movable at the owner's discretion — that is what it is for, and no part of this document should be read as a lock-up commitment over it.

What is not done

A whitepaper that lists only what works is a brochure. This is the current state of the things that are not finished, as of 12 September 2026. Two items that stood here previously – consumer wiring, and the reward distributors being undeployed – were completed on 3 September 2026; the reward programs are live and the addresses are in section 07.

NOT AUDITED

No third-party audit has been performed. The contracts are unaudited. The test suite is not a substitute for one, and the fact that the vault's limits are enforced in code does not mean the code is free of defects.

PENDING

The TUSS staking distributor does not exist. The TUSS project holds its 15,000,000 budget and has been vesting since 19 August 2026, but no contract has been written that can claim from it, so its distributor is unset. Its annual cap window is passing unused – the budget is not lost, but the window does not carry over.

MANUAL

Follow rewards are verified by a person, not an API. No platform the campaign pays for exposes a “does this account follow us” check to a commercial application, so a submitted handle is confirmed by hand against our own follower list before it is paid. That is a deliberate trade rather than a placeholder: it is honest about what is actually being checked, and it does not scale past modest volume.

THIS DOCUMENT IS NOT FINANCIAL ADVICE

pViTok is a utility token for paying rewards inside the VTS ecosystem. It carries no claim on revenue, no equity, no governance vote over anything but its own reward parameters, and no promise of value. Nothing here is an offer to sell, a solicitation, or investment advice. The supply and governance figures describe how the contracts are configured today; verify them on-chain rather than trusting this page.

pViTok — Polygon Native Token

Every figure in this document is drawn from the deployed contracts and the deployment record: supply and allocation from the governance deploy script, vault limits from `RewardReserveVault.sol`, reward parameters from the distributor contracts and their deploy scripts, and role assignments from the mainnet deployment audit. Addresses are verifiable on Polygonscan.

pViTok is not affiliated with, backed by, or redeemable for the ViTok token on Ethereum mainnet.