



PAXEX / WHITEPAPER

The Future of Blockchain

An EVM-compatible ecosystem with a path toward post-quantum security.

TECHNICAL WHITEPAPER / VERSION 1.0

Architecture · quantum readiness · ecosystem · economics

10 OCTOBER 2026



00 / CONTENTS

Contents

A structured guide to Paxex, the network architecture, quantum readiness and the ecosystem.

01 / FOUNDATION

About Paxex	03
Vision and mission	04
Executive summary	05
System design principles	06

02 / NETWORK ARCHITECTURE

Trust and system boundaries	07
Transaction lifecycle	08
Consensus candidate	09
EVM execution compatibility	10
State and data availability	11
Network and node operation	12
Performance methodology	13
Fee accounting and economics	14

03 / QUANTUM READINESS

The quantum threat model	15
Cryptographic agility	16
Hybrid account authorization	17
Hybrid network transport	18
Post-quantum migration plan	19

04 / DEVELOPER EXPERIENCE

Developer interface and sample	20
Testnet as an evidence engine	21

05 / ECOSYSTEM & ECONOMICS

Ecosystem modules	22
PAXEX token economics	23
Presale and accounting controls	24
Staking and reward model	25

06 / SECURITY & DELIVERY

Governance and upgrades	26
Security assurance program	27
Milestone-based roadmap	28
Risk factors and limitations	29

07 / APPENDICES

Technical acceptance checklist	30
Glossary and implementation notes	31
References and document control	32



01 / FOUNDATION

About Paxex

A blockchain ecosystem designed to connect accessible products with an EVM-oriented network.

Paxex brings the user-facing presale application, future wallet and testnet experience into one coherent ecosystem. The product direction extends to a DEX, bridge, NFT and gaming experiences, and AI-assisted services. The network layer is intended to support smart contracts and interoperable developer tooling.

One ecosystem, distinct responsibilities

Layer	Role
User experience	Account access, purchase history and ecosystem navigation
Applications	Wallet, DeFi, digital assets and developer services
Network	Transaction execution, state, validators and finality
Assurance	Open specifications, measurement and security review

Each layer has a separate trust model. An intuitive interface does not replace independent chain verification, and EVM compatibility does not imply the security guarantees of another network. The architecture and operating evidence for the Paxex chain should be assessed on their own terms.

PROJECT FOCUS

Make blockchain functions understandable to new users while preserving the transparency and technical depth required by developers and network operators.



01 / FOUNDATION

Vision and mission

Build a usable blockchain platform whose security model can evolve as cryptography changes.

The vision

Paxex aims to connect everyday users, builders and applications through an approachable interface and an EVM-compatible execution environment. Wallets, DeFi, gaming and digital assets can share common network primitives without forcing users to navigate unrelated systems.

The mission

The mission is to deliver a chain that can be independently measured, operated and improved. Network speed, fees and finality should be described with reproducible workloads. Security claims should identify exact components and audit scope. Post-quantum migration should be staged through explicit account, transport and validator changes rather than asserted as a finished property.

Principle	What it means in practice
Accessibility	Clear transactions, visible permissions and recovery
Compatibility	Predictable EVM behavior and familiar tooling
Accountability	Published parameters, code and test evidence
Adaptability	Versioned cryptography and governance-controlled upgrades

LONG-TERM DIRECTION

The future of blockchain depends on useful products, verifiable operation and a security model that can adapt without losing user access.



01 / FOUNDATION

Executive summary

Paxex proposes a familiar EVM development surface coupled to deliberate cryptographic agility.

The product thesis is a single entry point for applications: users can enter through a presale dashboard, then interact with testnet tools and ecosystem services as those services become available. EVM compatibility is intended to reduce porting costs for contracts and tooling. A native Paxex chain, if launched, must publish its own consensus, validator, economic and bridge security specifications.

“Quantum ready” has a precise meaning here: the architecture reserves a migration path for post-quantum credentials, hybrid authorization, validator key rotation and secure transport. It does not mean current ECDSA accounts or deployed smart contracts are already quantum resistant. Standards referenced include NIST ML-KEM, ML-DSA and SLH-DSA [1–3].

Four design commitments

- **Compatibility:** preserve standard EVM bytecode, JSON-RPC behavior and predictable transaction semantics where feasible.
- **Measurability:** release workload profiles, hardware conditions, failure cases and raw test results alongside any speed or fee claims.
- **Defense in depth:** separate account, consensus, bridge, key management and operational trust assumptions.
- **Credible migration:** deploy post-quantum features behind staged gates, with rollback and independent review.

CORE PROPOSITION

Build useful blockchain applications now while retaining an auditable path to stronger authorization and network cryptography over time.



01 / FOUNDATION

System design principles

Paxex is framed as a stack of verifiable interfaces rather than a single “fast chain” claim.

- Minimize consensus-critical complexity. Introduce changes only through versioned specifications, test vectors and reproducible clients.
- Keep signing policies separate from execution when possible. Smart accounts can introduce alternate validators without changing every application [4].
- Prefer explicit trust boundaries: wallet, RPC, validator, bridge, oracle, custody and administrator keys must have different controls.
- Require objective release gates: compatible tooling, fault handling, audit remediation, recovery and measurable decentralization.

APPS + DAPP

Wallet · DEX · AI · NFT · Gaming

ACCOUNT + RPC

EVM calls · smart accounts · indexers

EXECUTION + STATE

Gas accounting · deterministic state root

CONSENSUS + P2P

Validator set · block propagation · finality

Architecture decision record

The final mainnet topology—independent chain, rollup or another model—must be chosen with an explicit security and data availability analysis. A sidechain has independent consensus; a rollup inherits different settlement assumptions [5]. This paper describes an independent EVM-oriented chain candidate, not an Ethereum-secured rollup.



02 / NETWORK ARCHITECTURE

Trust and system boundaries

A credible security model begins by identifying which actor may fail or behave maliciously.

Component	Assumption	Required control
User wallet	Keys may leak or be phished	Hardware support, simulation, recovery
RPC / indexer	Responses may be stale or censored	Multiple endpoints, proof checks
Validators	Subset may collude or go offline	Slashing policy, quorum, observability
Bridge	External chain state may be misreported	Caps, delay, independent verification
Presale operator	Manual payment decisions may be wrong	Dual approval, ledger, reconciliation

No wallet UI can make compromised credentials safe. No cryptographic signature prevents a user from signing a malicious call. Network security and product security must therefore be measured separately.

BOUNDARY RULE

Never reuse an administrator, bridge, validator and treasury key. Critical operations should have role separation, threshold control, time delay and a tested emergency procedure.



02 / NETWORK ARCHITECTURE

Transaction lifecycle

The execution pipeline must be deterministic even when accounts gain new signing algorithms.

Proposed flow

- The wallet constructs a chain-bound payload with nonce, expiry, gas ceiling and intended call.
- Admission verifies format, fee sufficiency, replay protection and the account's active signature policy.
- Consensus orders valid transactions. Every full node executes them against the same prior state.
- A receipt records success, gas consumed, logs and the resulting state commitment.

```
tx_hash = H(domain || chain_id || account || nonce || call || gas_limit)
assert now <= expiry and nonce == account.next_nonce
assert account.policy.verify(tx_hash, credential, signature)
state_next, receipt = EVM.execute(state_prev, call, gas_limit)
```

Pseudocode is a design illustration, not deployable consensus code. Hash domain separation, serialization, error semantics and signature verification costs require a normative specification and cross-client test vectors.

REPLAY PROTECTION

A migration must bind both classical and post-quantum signatures to the same chain ID, account nonce, payload and validity window.



02 / NETWORK ARCHITECTURE

Consensus candidate

A permissioned bootstrapping phase and eventual open validator participation have very different guarantees.

A BFT-style proof-of-stake engine is a candidate for deterministic finality. It is not selected or implemented by this document. The validator set, epoch changes, proposer selection, slashing evidence, downtime handling and censorship resistance must be specified before a mainnet decision.

Parameter	Candidate design question
Quorum	Does finalization require $> 2/3$ voting power?
Fault model	How many Byzantine validators can be tolerated?
Unbonding	What delay protects against equivocation and bridge exits?
Recovery	How are liveness failures resolved without unsafe shortcuts?

Safety and liveness

A standard BFT quorum argument assumes less than one-third Byzantine voting power under appropriate network conditions. That abstract threshold does not imply Paxex has met it. Validator concentration, network partitions, software monoculture and administrative keys can weaken the practical system.

RELEASE GATE

Publish a consensus specification and adversarial test report covering double signing, partitions, validator churn, clock skew and stalled finality.



02 / NETWORK ARCHITECTURE

EVM execution compatibility

Familiar tooling matters only if behavior is stable at the contract and RPC boundary.

The proposed execution layer targets Solidity/Vyper contracts, common EVM opcodes, event logs, gas metering and Ethereum-style JSON-RPC. Fork version, precompiles, chain ID, gas schedule, transaction envelope and tracing endpoints need a public compatibility matrix.

Surface	Acceptance test
Bytecode	Compiler fixtures and differential execution
RPC	Wallet, explorer and indexer conformance suite
Gas	Deterministic out-of-gas and refund behavior
Accounts	EOA and smart-account signing paths
Upgrades	Fork activation and replay-resistant versioning

EVM compatibility does not make the chain economically or security equivalent to Ethereum. Its validators, bridge, data availability and governance remain independent until proven otherwise [5].

DEVELOPER PROMISE

Applications should see documented incompatibilities early, with versioned migrations and a public deprecation window.



02 / NETWORK ARCHITECTURE

State and data availability

Independent verification depends on access to complete data and stable state commitments.

Every block should bind transactions, receipts and the post-state root. Archive and pruned nodes may store different history, but both need a documented path to validate new blocks. Snapshot format, retention policy and recovery procedure require testnet evidence.

Operational design targets

- Expose block headers and receipts through stable RPC endpoints; allow operators to run their own nodes.
- Measure state growth, synchronization time and disk requirements under realistic workloads.
- Publish the rules for handling reorganization, finality, replay and historical query retention.

```
state_root[h] = Commit(Apply(state_root[h-1], ordered_txs[h]))
block_hash[h] = H(parent_hash || tx_root || receipt_root || state_root[h])
```

The commitment function and serialization must be specified before deployment. The equation states a property, not a choice of Merkle structure or hash function.



02 / NETWORK ARCHITECTURE

Network and node operation

Nodes need predictable behavior under latency, overload and hostile peers.

A node should validate and rate-limit messages before expensive work, bound its mempool resources, reject duplicate or malformed transactions and retain enough information to diagnose equivocation or censorship. Peer discovery and inbound connection policies should be measurable and configurable.

Failure	Expected response
Peer flood	Admission limits and bounded queues
Network partition	No conflicting finalized blocks under the stated fault model
Stale RPC	Detect height lag and retry independent providers
Disk pressure	Fail safely; protect committed state
Validator outage	Document missed block and recovery behavior

OPERATOR OBSERVABILITY

Publish health metrics for block propagation, peer diversity, vote participation, finality latency, mempool rejection and node synchronization. Dashboards do not replace public reproducibility.

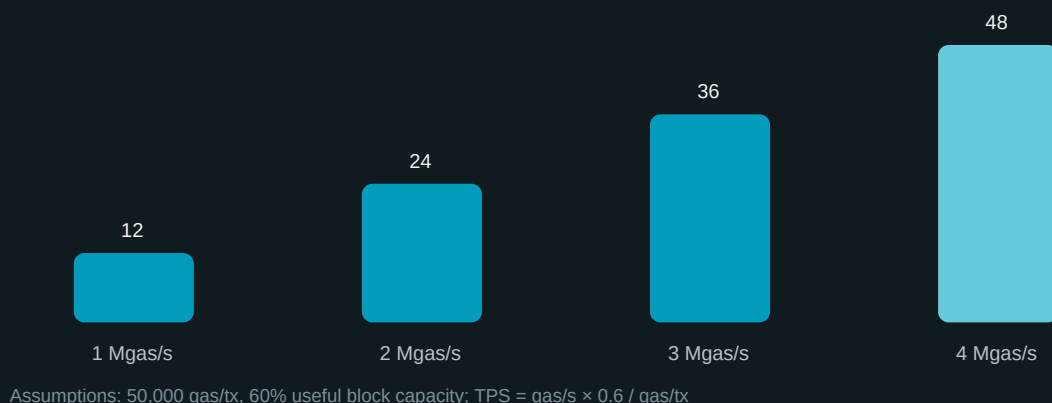


02 / NETWORK ARCHITECTURE

Performance methodology

Throughput and latency are workload-dependent; report distributions, not single best-case figures.

Illustrative capacity sensitivity — not a benchmark



Example calculation: if effective compute budget is 2,000,000 gas/s, the workload averages 50,000 gas per transaction, and only 60% of capacity is usable, then illustrative throughput is $(2,000,000 \times 0.60) / 50,000 = 24$ transactions/s. The chart deliberately uses a different scale label in Mgas/s: 1 Mgas/s yields 12 TPS under those inputs. Any actual benchmark must measure real execution, signatures, state growth and network overhead.

Benchmark contract

- State hardware, validator count, network geography, payload mix and software commit.
- Publish p50/p95/p99 inclusion and finality, failed transactions, CPU, bandwidth and disk growth.
- Repeat under sustained load and adversarial conditions; include confidence intervals and raw traces.

NO PERFORMANCE CLAIM

The values on this page are arithmetic examples only. They are not Paxex testnet or mainnet measurements.



02 / NETWORK ARCHITECTURE

Fee accounting and economics

Fees must cover resource costs without promising “zero fees” that a live network cannot sustain.

A candidate model separates computational gas from storage and bandwidth pressure. Governance may subsidize some user actions, but the subsidy must be funded and abuse-limited. Wallets should show the payer, maximum charge and actual receipt.

```
gross_fee = gas_used × effective_gas_price
net_user_fee = max(0, gross_fee - eligible_subsidy)
operator_cost = bandwidth + compute + storage + security_budget
```

Scenario	Gas used	Price / gas	Gross fee
Simple transfer (example)	21,000	0.2 gwei	0.0000042 PAXEX
Contract call (example)	150,000	0.2 gwei	0.0000300 PAXEX

Example amounts assume the native asset is the fee unit, 1 gwei = 10^{-9} unit, and omit storage rent, tip, subsidy and market dynamics. They are not announced fee parameters.

SUSTAINABILITY TEST

Publicly model validator revenue, storage growth and subsidy reserve before any claim of near-zero fees.



03 / QUANTUM READINESS

The quantum threat model

Long-lived keys and captured traffic create different migration priorities.

- **Store now, decrypt later**
Transport secrets

- **Long-lived public keys**
Account authorization

- **Validator signatures**
Consensus proofs

A sufficiently capable future quantum computer would threaten common public-key systems used for signing and key agreement. The time horizon is uncertain. The practical question is which Paxex assets must remain secure long enough that migration cannot be deferred [1–3].

Three separate surfaces

- Account signatures: exposed public keys and long-lived authorization policies create theft risk if a suitable attacker emerges.
- Consensus signatures: validator credentials and historical proof verification need coordinated migration.
- Transport confidentiality: information collected today may remain sensitive when current key exchange becomes vulnerable.

EXPLICIT LIMITATION

“Quantum ready” does not protect existing ECDSA wallets, unaudited bridges, weak recovery channels, compromised devices or contracts that bypass the new account policy.



03 / QUANTUM READINESS

Cryptographic agility

Versioned interfaces allow algorithms to change without redesigning every application.

Paxex should assign algorithm identifiers to credentials and require deterministic encodings, test vectors, key rotation and deprecation rules. ML-DSA [2] is a candidate signature family; SLH-DSA [3] provides a different standardized option. ML-KEM [1] applies to key establishment, not transaction signatures.

Use	Candidate	Caution
Transactions	ML-DSA via smart account	Signature size and gas cost
Fallback signing	SLH-DSA where appropriate	Larger signatures; verify costs
Transport	Hybrid classical + ML-KEM	Protocol and downgrade resistance
Validator votes	Phased PQ signature trial	Bandwidth, quorum, historical proofs

The chain cannot safely declare one algorithm permanent. A vulnerability, side-channel finding or implementation defect may require rapid revocation. Versioned authorization must include safe recovery and clear behavior for unsupported algorithms.

SECURITY PROPERTY

The authorization policy is part of consensus state. Its upgrade path must be at least as carefully protected as the keys it verifies.



03 / QUANTUM READINESS

Hybrid account authorization

A staged account policy can require both a conventional and a post-quantum signature.

An opt-in smart-account design can require two independent signatures over the same domain-separated operation. It should include nonce handling, guardian rotation, upgrade delay and emergency freeze. ERC-4337 shows how account-specific validation can be introduced at the contract layer [4]; adoption on Paxex would still require compatibility and cost analysis.

```
function validate(op, policy) -> bool:
    digest = H("PAXEX:AUTH:v1" || chain_id || op.encode())
    require(verify_classical(policy.ecdsa_key, digest, op.sig_ec))
    require(verify_pq(policy.ml_dsa_key, digest, op.sig_pq))
    require(op.nonce == policy.next_nonce && op.deadline >= now)
    return true
```

This is non-production pseudocode. A real implementation must define byte-level encoding, algorithm parameter set, key registry, signature malleability behavior, validation gas limits and recovery semantics.

MIGRATION CAVEAT

A hybrid AND policy protects against a break in one signature family only while the second remains secure and its key has not been compromised. It increases cost and operational complexity.



03 / QUANTUM READINESS

Hybrid network transport

Key establishment and digital signatures solve different problems.

For peer-to-peer transport and sensitive API channels, a candidate hybrid handshake combines an established classical key agreement with ML-KEM [1], then derives traffic keys with a vetted KDF. This aims to keep confidentiality if one component weakens, provided the combiner and authentication are correctly designed.

```
secret_classical = ECDH(local_ephemeral, peer_ephemeral)
secret_pq        = ML_KEM.Decapsulate(local_sk, ciphertext)
traffic_key = HKDF(secret_classical || secret_pq,
                   transcript_hash || "PAXEX:CHANNEL:v1")
```

The diagrammatic pseudocode omits authentication, error handling and protocol details. Implementations should use a reviewed standard protocol profile; custom cryptographic combiners or ad hoc TLS modifications would create avoidable risk.

Property	Acceptance condition
Downgrade resistance	Transcript binds negotiated algorithms
Forward secrecy	Fresh ephemeral secrets; erasure verified
Interoperability	Known-answer tests across implementations
Operations	Certificate and rotation process documented

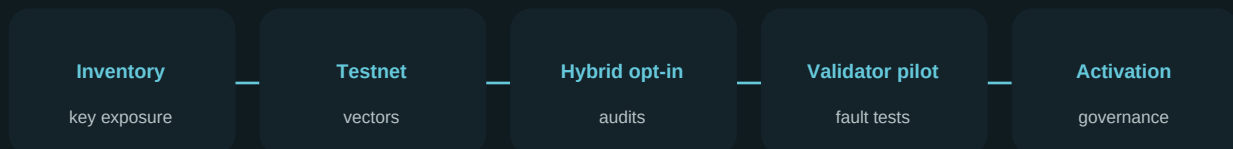


03 / QUANTUM READINESS

Post-quantum migration plan

Quantum readiness should be measured through completion gates rather than a slogan.

Migration gates and required evidence



No gate implies that a production quantum-resistant chain already exists.

Gate definitions

- Inventory: classify keys by value, exposure, lifetime and recovery path.
- Testnet: implement algorithm IDs, smart accounts, cost meters and cross-client vectors.
- Hybrid opt-in: ship audited authorization with user-visible backup and key rotation.
- Validator pilot: test message size, vote verification, partitions and key ceremony.
- Activation: publish governance decision, backward-compatibility rules, notice periods and rollback criteria.

Migration must include wallets that cannot upgrade, dormant accounts and long-lived contracts. A hard fork should not strand funds because a new signature format is unavailable to a user.



04 / DEVELOPER EXPERIENCE

Developer interface and sample

A coherent developer surface turns protocol concepts into testable integration work.

The target interface is standard EVM tooling plus documented Paxex extensions for account policy and network status. The example below is illustrative; addresses, ABIs and RPC methods are not production commitments.

```
// Illustrative TypeScript; not a deployed API
const network = { chainId: 0, rpcUrl: "<published-testnet-rpc>" };
const policy = { mode: "hybrid", algorithm: "ML-DSA", version: 1 };
const operation = await wallet.prepare({
  to: applicationAddress, data: callData, policy, deadline
});
const receipt = await wallet.submit(operation);
```

Required artifacts

- Versioned chain configuration and testnet faucet rules.
- Contract examples, deterministic ABI fixtures and JSON-RPC conformance results.
- Account policy SDK with test vectors, failure codes and recovery guide.
- Published source and reproducible binaries for node and wallet components.

COMPATIBILITY RULE

Never present invented chain IDs, contract addresses or endpoint URLs as final. Publish immutable release manifests when systems exist.



04 / DEVELOPER EXPERIENCE

Testnet as an evidence engine

The testnet should expose real behavior across wallets, contracts, validators and failures.

The proposed testnet journey includes wallet creation, faucet, claim, contract deployment, transactions, explorer and validator telemetry. Each feature should display its version and limitations. A public status page should distinguish planned, beta and verified services.

Track	Evidence to publish
Wallet + faucet	Success rate, abuse controls, transaction receipt
Contracts	Compiler matrix, deployment traces, known gaps
Validators	Count, concentration, uptime, finality
Bridge trial	Caps, challenge assumptions, incident drills
PQ pilot	Signature size, validation cost, recovery tests

Adversarial exercises

Run network partitions, malformed transaction floods, client version skew, key loss, bridge pause and governance delay drills. Record raw inputs and failures, not just headline averages.

DEFINITION OF READY

A testnet milestone is complete when a third party can reproduce the setup, run the test and compare results against a public acceptance threshold.



05 / ECOSYSTEM & ECONOMICS

Ecosystem modules

Products should be released as individually testable services with explicit dependencies.

Module	Purpose	First evidence
Wallet	Accounts, signing, recovery	Compatibility and recovery test
DEX	Asset routing and swaps	Contracts, liquidity and oracle review
Bridge	Cross-chain movement	Trust model, caps, monitoring
NFT + gaming	Digital assets and applications	Developer docs and live examples
AI services	Optional developer or UX tooling	Privacy, model and API disclosures

The ecosystem concept links the dapp, testnet and future chain. A product appearing in a roadmap is not evidence that it is live. Each launch should identify status, chain, source, audit scope, operator and exit path.

Experience objective

A new user should understand how to register, connect a wallet, obtain test tokens, deploy or use an application, inspect a transaction and recover access. UI convenience must not hide signature permissions or third-party bridge risk.

DEPENDENCY DISCIPLINE

Release wallet and explorer transparency before asking users to trust more complex bridge, staking or DeFi behavior.



05 / ECOSYSTEM & ECONOMICS

PAXEX token economics

The native asset connects network resources, security incentives and ecosystem participation.

Potential utility includes network fees, staking collateral, governance participation and access to ecosystem services where appropriate. Utility is an intended use, not a guaranteed demand source or a claim about token price.

Token utility framework

NETWORK

Fees and execution

SECURITY

Validator collateral

GOVERNANCE

Protocol decisions

Utility design does not imply a fixed price, yield or token demand.

Supply and circulation

A responsible token model distinguishes maximum supply, minted supply and circulating supply. Vesting, treasury release, validator emissions and burns must be accounted for in a single publicly verifiable schedule.

```
circulating_supply(t) = minted(t) - locked(t) - burned(t)
net_emission(t) = rewards(t) - burns(t)
```

Specific allocation percentages, sale tranches and vesting parameters should be read from the canonical token schedule and deployed contracts when published. No listing price or token return follows from the utility model.



05 / ECOSYSTEM & ECONOMICS

Presale and accounting controls

A dual-path purchase flow needs a single authoritative ledger and clear settlement rules.

The Paxex dashboard contemplates wallet-connected and manual payment purchases. Web3 receipts should be matched to chain ID, asset, destination, transaction hash and finality threshold. Manual payments require verified evidence, a review queue, duplicate detection and an immutable approval trail.

```
credited_tokens = floor(net_amount_usd / sale_price_usd × 10decimals)
refundable_balance = paid - settled - previously_refunded
assert credited_tokens <= tranche_remaining
```

Numerical example: \$125.00 net at an illustrative \$0.005 per token yields 25,000 tokens before any disclosed bonus or rounding rule. This is a worked example, not a Paxex price. If a bonus exists, publish whether it is additive, multiplicative and subject to vesting.

Minimum controls

- Idempotency keys, exact-decimal arithmetic, chain reorganization policy and treasury reconciliation.
- Two-person approval for manual credits and wallet-address changes; customer-visible transaction history.
- Separate pending, approved, allocated, claimed and refunded balances.

BUYER PROTECTION

Publish payment instructions, accepted networks, refund conditions and contract addresses in one canonical location. Never rely on a social post alone for payment destinations.



05 / ECOSYSTEM & ECONOMICS

Staking and reward model

Rewards should follow a disclosed budget and risk model, not a headline APY alone.

A proposed staking interface may present one-, three- and longer-term pools, active positions, earned rewards and withdrawal dates. The economic design needs an emission cap, source of rewards, penalties, early exit rule and validator relationship before launch.

```
simple_reward = principal × annual_rate × locked_days / 365
effective_apy = (1 + periodic_rate / periods_per_year)
               ^ periods_per_year - 1
```

Example: 10,000 tokens at a hypothetical 8% simple annual rate for 90 days yields about 197.26 tokens before fees and penalties. If rewards compound or rates float, the displayed formula changes. The example is not a Paxex advertised rate.

Risk	Disclosure needed
Smart-contract risk	Audit scope, upgrade keys, pause authority
Liquidity risk	Unlock timing and withdrawal queue
Emissions / dilution	Funding, maximum schedule, reporting
Validator risk	Slashing and delegation mechanics

NO GUARANTEED YIELD

Illustrative formulas explain accounting. They are not an offer, return forecast or claim that staking pools are deployed.



06 / SECURITY & DELIVERY

Governance and upgrades

A credible upgrade process defines who can change what and how users can object.

Protocol changes should move through public proposals, an implementation review, a timelock, a reproducible release and an activation window. Emergency powers may be necessary but must be narrow, logged and independently reviewed afterward.

Change class	Control
Routine parameter	Bounds, review period, published rationale
Consensus or crypto	Client release, test vectors, audit, hard-fork notice
Treasury transfer	Multisig, limit, public accounting
Emergency pause	Limited scope, expiry, postmortem

Governance design questions

- Who has proposal rights and who can veto unsafe changes?
- Can administrators alter token balances, bridge custody or account policy?
- What constitutes quorum and how is voting power concentrated?
- How are inactive users informed of a signature migration?

OPERATIONAL INTEGRITY

Publish current role holders and contract permissions. A governance token by itself does not decentralize an upgrade key.



06 / SECURITY & DELIVERY

Security assurance program

Security requires a continuous evidence trail from specification through incident response.

Independent review should cover consensus safety, EVM divergence, smart accounts, bridge logic, sale settlement, staking math and privileged operations. Each assessment needs scope, commit hashes, severity, remediation status and unresolved findings.

Stage	Evidence
Specification	Threat model and invariant list
Implementation	Unit, property and differential tests
Deployment	Reproducible builds and signed releases
Operations	Monitoring, incident drill and backups
Independent review	Public report with scope and fixes

Suggested invariants

- A finalized block cannot be replaced under the stated fault model.
- Total credited presale allocation never exceeds the published tranche cap.
- Bridge liabilities never exceed verified collateral under the declared model.
- A rotated credential cannot authorize operations after its revocation epoch.

AUDIT LANGUAGE

Do not display third-party audit badges or “verified” claims unless a current report covers the deployed contracts and the report can be inspected publicly.



06 / SECURITY & DELIVERY

Milestone-based roadmap

Release gates are more honest than dates without acceptance evidence.

01

Specification

02

Public testnet

03

Security review

04

Mainnet decision

Sequence is conditional. Dates require published acceptance criteria and release evidence.

Phase	Definition of done
01 / Specification	Protocol, economics, trust model and test vectors published
02 / Testnet	Public node, explorer, faucet, reproducible performance tests
03 / Review	Independent assessments and critical fixes complete
04 / Decision	Validator readiness, token disclosure, launch governance

Parallel application work may include wallet, DEX, AI, NFT, gaming and bridge modules. Their delivery order depends on dependencies and audit capacity. A mainnet launch should remain conditional on public evidence and operational readiness, not presale timing.

REPORTING CADENCE

Publish milestone changes with what shipped, what failed, blockers and next acceptance criteria. Avoid retroactive “complete” labels without artifacts.



Risk factors and limitations

The design has material technical, operational, financial and adoption risks.

- Protocol risk: consensus or execution defects can halt the chain or cause divergent state.
- Quantum risk: migration may be too slow, expensive or incomplete; standardized algorithms can still suffer implementation flaws.
- Bridge and oracle risk: external data, custody and governance assumptions can dominate protocol security.
- Economic risk: fees and incentives may not support validators; staking emissions may dilute holders.
- Sale risk: manual settlement, payment network errors and changing regulatory obligations require explicit controls.
- Liquidity risk: a token may never list or have a liquid market; no listing price or return is promised.
- Concentration risk: validator, token or administrator power may remain concentrated during early phases.

Privacy limitation

A public EVM chain exposes transaction and contract activity by default. “Private” product language must be narrowed to defined protections, such as transport confidentiality or optional application-level privacy, and verified against actual implementations.

NO ASSURANCE

This whitepaper describes technical direction. It is not investment advice, an audit report, a security guarantee, or evidence that every described feature has been deployed.



07 / APPENDICES

Technical acceptance checklist

A release candidate should pass a published checklist that an independent team can reproduce.

Area	Candidate criterion
Chain identity	Chain ID, genesis hash, consensus spec, clients
Compatibility	EVM and RPC differential test suite
Account migration	Hybrid vectors, gas bounds, recovery scenarios
Network	Sustained-load and partition results
Economics	Supply, allocations, contracts, vesting reconciliation
Safety	Audit reports, keys, incident exercise
Operations	Monitoring, restore drills, validator playbook

Measurement equations

```
TPS_window = successful_txs / window_seconds
finality_p95 = P95(time_finalized - time_submitted)
validator_HHI = sum((validator_power_i / total_power)^2)
net_supply = minted - burned
```

These metrics require precise definitions: whether reverted transactions count; whether latency begins at RPC acceptance or peer receipt; and whether validator power is attributed to legal operators or addresses.

PUBLICATION GATE

A metric without workload, environment, code version and raw evidence is marketing, not a reproducible benchmark.



07 / APPENDICES

Glossary and implementation notes

Terms used throughout this paper have narrow technical meanings.

Term	Meaning
EVM	Ethereum Virtual Machine execution semantics
BFT	Byzantine fault tolerant consensus family
Finality	A block accepted under a defined irreversible rule
ML-KEM	NIST key encapsulation standard FIPS 203
ML-DSA	NIST signature standard FIPS 204
SLH-DSA	NIST stateless hash-based signature FIPS 205
Hybrid signature	Two signature checks in one authorization policy
Smart account	Contract-controlled authorization logic

Scope of examples

Code is schematic and equations are dimensioned arithmetic illustrations. A production release should publish precise protocol parameters and a signed deployment manifest.

REVISION HISTORY

Version 1.0 (10 October 2026). Later revisions should enumerate changed assumptions and link to implementation evidence.



07 / APPENDICES

References and document control

Primary standards and protocol documentation consulted for the technical framing.

[1] NIST, FIPS 203, Module-Lattice-Based Key-Encapsulation Mechanism Standard (2024).

<https://csrc.nist.gov/pubs/fips/203/final>

[2] NIST, FIPS 204, Module-Lattice-Based Digital Signature Standard (2024). <https://csrc.nist.gov/pubs/fips/204/final>

[3] NIST, FIPS 205, Stateless Hash-Based Digital Signature Standard (2024). <https://csrc.nist.gov/pubs/fips/205/final>

[4] Ethereum Improvement Proposals, ERC-4337: Account Abstraction Using Alt Mempool.

<https://eips.ethereum.org/EIPS/eip-4337>

[5] Ethereum.org, Scaling documentation: rollups and independent sidechains.

<https://ethereum.org/developers/docs/scaling/>

Document control

Attribute	Value
Publisher	Paxex
Version / date	1.0 / 10 October 2026
Scope	Architecture, economics and migration framework
Artwork	Paxex logo supplied by project owner; diagrams original

Chain identity, source repository, contract addresses, token schedule and audit reports should be linked from the canonical project publication when available. Readers should verify such details at their source.

SOURCE INTEGRITY

The publisher should distribute a content hash and version history so readers can distinguish this edition from later updates.