

Technical Whitepaper

Deterministic execution, hybrid consensus and accountable AI services

Version 0.4.0 / 19 November 2026

Technical review draft

This edition describes inspected implementation and recorded experiments. Public testnet and mainnet acceptance remain incomplete. It is prepared for technical review; it is not an independent security audit, release authorization, or regulatory offering document.

Executive abstract

PQWERAI combines an account-based programmable ledger with a separately operated AI-service infrastructure. Its inspected Core derives from Core-Geth and integrates GhostRider CPU proof of work, registry-derived Smartnode checkpoint voting, EVM execution, a required WASM virtual machine, and optional predictive and speculative execution (PASE). AI inference runs outside block validation. On-chain contracts bind model and provider identities and support customer-funded service settlement; off-chain adapters supervise upstream inference runtimes and account for delivered output.

The retained canonical testnet has chain and network identifier 20777. Its issuance implementation allocates 67% to proof-of-work miners, 18% to participating Smartnodes, 1% to the founder destination, 9% to the Quantum Reserve and 5% to the IPFS reward pool under CR-0051, recorded active from block 34785, with exact integer rounding and no configured economic supply cap. These are shares of the same newly issued amount. Eligible IPFS operators claim from the shared pool after successful proofs; the allocation is not automatic income for every IPFS user. [E31] They are separate from AI-service revenue and from QAI, a simulated-euro-reference test token with post-quantum smart-account authorization and reserve-accounting contracts.

This paper is an evidence-based technical review of a development system. It separates inspected source, historical local measurements, dated deployment observations, design objectives and unresolved claims. It does not establish production security, independent operator diversity, sustainable network capacity, reserve solvency or regulatory compliance. The selected Core has unresolved WASM resource-cost findings. QAI's recorded threshold keys remain concentrated on one computer, and its reserves are simulated. Both public-testnet release acceptance and mainnet acceptance remain incomplete.

The principal contribution is the integrated architecture and its explicit boundaries: deterministic ledger execution, authenticated checkpoint evidence, ordered speculative execution, bounded AI authorization, customer-observed metering, and version-specific economic accounting. The accompanying source mappings and freshness checks make these descriptions reviewable as implementation evolves.

Earlier editions added end-to-end operational walkthroughs, a complete 74-feature status catalogue, descriptive reanalysis of historical performance, and four reproducible twelve-month QAI balance-sheet paths. Those paths examine capital requirements, losses and redemption liquidity under stated assumptions. They do not forecast market prices, assign probabilities or promise financial returns.

Edition 0.4.0 adds Chapter 23, which covers work between 14 and 19 November 2026: IPFS reward v2 with bonded, slashable proof of access; PATE adaptive trust execution, whose third phase runs only as shadow finality; the deployed AI market and same-model compute pools; native and multi-thread mining from the wallet; advisory internal agents; and live-difficulty throughput measurements on one host. [E32][E33][E34][E35] Chapter 25 sets out the evidence-gated path to mainnet and the resources it needs. That chapter is a plan, not an offer.

Reading this edition. Evidence links such as [E01] resolve to the source ledger in Appendix A. Numerical experiments are historical unless explicitly identified as executed for this edition. Every theoretical bound states its assumptions. No independent audit is claimed.

Source cutoff. Repository `1bf413ccef6c9bdfe658e44ebbae6e58a505fc50`; selected private

Core `f2eace6bc69fca48b290142b8232bd1ced6a690b`. Inspection and document preparation: 19 November 2026. This edition does not freshly query production services or sign transactions.

Contents

Executive abstract i

1 Introduction and evidence discipline 1

1.1 Scope and motivation . . . 1
1.2 Evidence hierarchy . . . 1
1.3 Reconciliation and limits of authority . . . 2

2 Purpose and design boundaries 3

2.1 Problem and existing approaches . . . 3
2.2 Three independent correctness questions . . . 3
2.3 Terminology . . . 3

3 System architecture 4

3.1 Execution, service and distribution planes . . . 4
3.2 State and persistence . . . 4
3.3 Authenticated network identity . . . 5
3.4 Dependencies and failure containment . . . 5

4 Hybrid consensus and finality 6

4.1 Roles and assumptions . . . 6
4.2 GhostRider seal and difficulty . . . 6
4.3 Epoch sets and checkpoint certificates . . . 7
4.4 Fork choice, partitions and recovery . . . 7
4.5 Smartnode lifecycle and penalties . . . 8

5 Networking and synchronization 9

5.1 Blockchain transport . . . 9
5.2 AI transport and NAT traversal . . . 9
5.3 Eclipse and partition resistance . . . 9

6 Transactions, state transitions and fees 10

6.1 Lifecycle . . . 10
6.2 Fee accounting . . . 10
6.3 Application safety . . . 10

7 EVM, WASM and developer tooling 11

7.1 Inherited EVM semantics . . . 11
7.2 Required WASM VM v1 . . . 11

7.3 Language and Studio boundary . . . 11

8 Predictive and parallel execution: PASE 12

8.1 Problem and prior work . . . 12
8.2 Prediction and prefetch . . . 12
8.3 General optimistic execution . . . 12
8.4 Correctness obligation and costs . . . 13

9 MeshLLM and accountable AI services 15

9.1 Runtime integration and resource sharing . . . 15
9.2 Provider discovery and routing . . . 15
9.3 Placement, cooperative pools and training . . . 15
9.4 Delivered-token accounting . . . 16
9.5 Models, storage and certification . . . 16

10 Colibri backend and orchestration 17

10.1 Implementation boundary . . . 17
10.2 Measured local behavior . . . 17
10.3 Tradeoffs and unresolved integration . . . 17

11 Native monetary and service economics 18

11.1 Asset roles and initial allocation . . . 18
11.2 Block rewards: accrual and actual payment timing . . . 18
11.2.1 Gross issuance is participation- and stake-dependent . . . 18
11.2.2 Exact split and per-block miner allocation . . . 19
11.2.3 Executed accounting examples, not a fixed subsidy . . . 20
11.3 Supply, burns and inflation . . . 20
11.4 Security budget and incentive limits . . . 21
11.5 Customer-funded service revenue . . . 21

12 IPFS sharing rewards: the five-percent issuance allocation 22

12.1 The missing allocation and its source . . . 22
12.2 Funding the pool without increasing supply . . . 22
12.3 Who can receive the allocation . . . 23
12.4 Proof, weighting, settlement and claim . . . 23
12.5 Evidence and limitations . . . 24

13 QAI, reserve accounting and post-quantum accounts 25

13.1 Purpose, identity and deployment status . . . 25
13.2 Authorization and cryptographic scope . . . 25
13.3 Separate reserve roles . . . 25
13.4 Valuation and the 110–130 percent band . . . 26

13.5 Issuance, redemption and burns . . . 27
13.6 Trust, custody and regulation . . . 27

14 QAI economics, conditional outlook and monetary risk 28
14.1 What can be forecast responsibly . . . 28
14.2 Why par-funded growth needs additional capital . . . 28
14.3 Loss absorption and insolvency sensitivity . . . 29
14.4 Twelve-month conditional paths . . . 30
14.5 Liquidity, redemption and market-price deviations . . . 31
14.6 How the 9 percent native treasury relates to money . . . 31
14.7 Costs and revenues without promised earnings . . . 31
14.8 What would permit a stronger outlook . . . 32

15 Security and threat model 33
15.1 Protected assets and adversaries . . . 33
15.2 Known selected-Core WASM findings . . . 34
15.3 Contract and service security . . . 34
15.4 Keys, cryptography and supply chain . . . 34
15.5 Incident response and assurance . . . 35

16 Governance, upgrades and authority 36
16.1 Observed governance . . . 36
16.2 Treasury and contract governance . . . 36
16.3 Upgrades and emergency procedures . . . 36
16.4 Proposed decentralization of authority . . . 36

17 Performance and experimental evaluation 37
17.1 What the evidence measures . . . 37
17.2 Gas-limited envelope . . . 37
17.3 Maximum-throughput series on a chain-20777 harness . . . 38
17.4 Reproducible evaluation protocol . . . 39
17.5 Available launchers and missing acceptance . . . 39

18 Feature statistics and evidence coverage 40
18.1 Statistical vocabulary and complete coverage . . . 40
18.2 Historical blockchain burst observations . . . 40
18.3 PASE execution statistics and their boundary . . . 41
18.4 Inference sample statistics . . . 42
18.5 QAI gas, reserve controls and economic statistics . . . 42
18.6 Missing statistics by feature family . . . 43

19 Operational walkthroughs: how the pieces interact **44**
19.1 A payment from wallet to finalized history . . . 44
19.2 A PASE conflict example . . . 44
19.3 An AI request from consent to payment . . . 45
19.4 A timeout after dispatch . . . 46
19.5 One model across providers versus independent replicas . . . 46
19.6 A QAI mint and redemption across the trust boundary . . . 46
19.7 From package discovery to safe local activation . . . 47

20 Participation and decentralization **48**
20.1 Node and operator participation . . . 48
20.2 Concentration and economic incentives . . . 48
20.3 Acceptance path . . . 48

21 Interoperability and distribution **49**
21.1 EVM interfaces and contract integration . . . 49
21.2 Content and software distribution . . . 49
21.3 Oracles and cross-system trust . . . 49

22 Applications and ecosystem scope **50**
22.1 Programmable ledger applications . . . 50
22.2 AI-powered applications . . . 50
22.3 Infrastructure and real-world integration . . . 50

23 Developments since edition 0.3.0 **51**
23.1 IPFS reward v2: bonded proof of access . . . 51
23.2 PATE: adaptive trust execution . . . 52
23.3 AI market, model pools and workflows . . . 52
23.4 Mining from the wallet . . . 52
23.5 Internal agents . . . 53
23.6 Operations and security posture . . . 53
23.7 Deployment lineage at the cutoff . . . 53

24 Acceptance roadmap and research **54**
24.1 Completed implementation slices . . . 54
24.2 Required work without invented dates . . . 54
24.3 Public testnet and mainnet prerequisites . . . 54
24.4 Research directions . . . 55

25 Path to mainnet and resourcing **56**
25.1 Where the network stands . . . 56
25.2 Milestones . . . 56

25.3 Network growth targets 56
25.4 Resource needs 56
25.5 Funding mechanisms under consideration 57

26 Conclusion **59**

A Implementation evidence ledger **60**
A.1 How to reproduce the evidence boundary 60

B Notation and reproducibility checklist **68**

C Complete feature status catalogue **69**

D Open technical and publication questions **74**

References **75**

List of Figures

3.1 Component relationships. Inference output never enters the block-validity predicate. Arrows show principal interactions, not a claim that every integration is accepted. 4

6.1 Transaction lifecycle. Inclusion and successful execution precede finality; reversion remains a valid execution outcome. 10

8.1 PASE execution boundary. Speculation changes how work is attempted; canonical order determines the committed result. 13

13.1 Distinct native treasury and QAI liability accounting. On testnet the reserve assets are simulated; the off-chain conversion boundary remains trusted. 26

14.1 Deterministic illustrative paths, not calibrated forecasts. All start from assumed simulated liabilities of 100,000 and valuation of 120,000 EUR-reference units. The unfunded-growth path remains at 120% because requested mints are rejected. A higher ratio does not imply sufficient immediate liquidity. 30

19.1 AI service lifecycle. On-chain funding and final credits remain separate from off-chain inference, observed delivery and semantic output quality. 45

List of Tables

1.1 Evidence vocabulary. These classes are not interchangeable. 1

3.1 Source-pinned genesis parameters, not a live network-health assertion. 5

10.1 Historical one-host Colibri measurements. These are not new measurements for this edition. 17

11.1 Pre-fork source-accounting vectors under explicitly chosen participation inputs; the later fork reduces PoW and adds the IPFS pool. Values exclude transaction tips and are not a fixed block reward, a live-network measurement or promised earnings. 20

12.1 Recorded post-fork settlement at height 34785; the same amounts were recorded at 34817 and 34849. These are dated counter deltas, not a fixed subsidy or promised future income. 23

13.1 Implemented QAI policy. Thresholds are controls on the simulated ledger, not price guarantees. 26

14.1 Illustrative loss sensitivity in EUR-reference units with fixed 100,000 liability. Values are assumptions and arithmetic, not observed money or portfolio forecasts. 29

14.2 Conditional scenario outputs, not forecasts of actual adoption, funding or redemption performance. Amounts are simulated EUR-reference units. 30

15.1 Attack surfaces, implemented controls and residual risks. 33

17.1 Measurements, limits and targets are kept separate. 37

17.2 One-host maximum-throughput observations. *Ceiling* is equation (17.1) evaluated at the run's mean block interval. 38

18.1 Status distribution across all registered features; not a success rate. 40

18.2 Historical test-builder burst rates, difficulty 2, two local nodes. *Separate coalesced observation. No row measures sustained or finalized TPS. 41

18.3 Historical Qwen2.5-Coder-0.5B Q4_K_M measurements on one host, not comparable model quality or capacity to the 30B-A3B Colibri model. 42

18.4 Historical local QAI operation costs and derived fee sensitivity. These are not live quotes, full lifecycle costs or euro prices. 43

23.1 Changes between editions 0.3.0 and 0.4.0 and their readiness level. 51

24.1 Evidence-gated roadmap. No delivery dates or automatic gate closures are implied. 54

25.1 Evidence-gated milestones from the current testnet to mainnet. 57

B.1 Principal notation and units. 68

C.1 Complete feature register: author status and remaining gaps, not runtime statistics. 69

Chapter 1

Introduction and evidence discipline

1.1 Scope and motivation

Programmable ledgers provide shared ordering and reproducible state transitions. AI services require heterogeneous, often nondeterministic execution, large model artifacts, resource scheduling and off-chain delivery. Combining these domains requires a settlement boundary that does not make consensus depend on a model's response or a provider's GPU availability. PQWERAI implements ledger and service components around this separation. It is an integration effort with inherited execution semantics, custom consensus extensions and application-specific contracts; this paper does not claim that every primitive is novel. [E01][E10]

Nakamoto consensus supplies the background for work-based chain selection [1]; checkpoint voting offers a different, assumption-dependent finality mechanism [2]. Ethereum's execution specification supplies the account, gas and receipt model [3]. These references explain concepts; their security theorems and benchmark results do not automatically apply to PQWERAI's modified client.

1.2 Evidence hierarchy

The audit began with workspace authority, actual Git state, source provenance, component ownership, the finding register and revision-specific records. The enclosing workspace contains active code, archives, toolchains and private operational material. The active repository is `current/`; archived work is context, not an active release. Sensitive stores and operational credentials are excluded from publication.

The baseline contains 65 registered components, 16 package identities and 74 feature entries. Their complete inventory is supplied in `evidence/component_inventory.md`, preserving the registry's stated limits. A file inventory is not an exhaustive behavioral audit of every dependency. Source inspection concentrated on consensus, execution, economic and service boundaries; selected private Core files were inspected at the registered revision and hashed separately. [E01]

Evidence class	Meaning in this paper
Inspected implementation	The cited source contains the described mechanism; activation is a separate question.
Local test evidence	A recorded test ran for a stated source and environment; it is not independent assurance.
Recorded deployment	A dated record identifies network activity; current remote state is not inferred.
Design or target	An intended property or acceptance criterion without sufficient execution evidence.
Unresolved	Sources conflict or the evidence is insufficient to state the property.

Table 1.1: Evidence vocabulary. These classes are not interchangeable.

1.3 Reconciliation and limits of authority

The strategic plan retains earlier 77240 and 79.2/19.8/1 text. ADR-0015 in the decision register and current invariants supersede that identity for the retained 20777 chain. The dated genesis directory also retains "may be regenerated" wording; it is not current reset authority. The original manifest calls the reserve contract unimplemented and PASE transfer-only, while later source and deployment evidence show subsequent development. These discrepancies are recorded explicitly rather than silently editing history. [E02][E09][E20]

This edition adds the focused CR-0051 economic supplement: repository `6f66e031b018b925c6427be03b9d8c0a73a5a245`, recorded deployed Core `12127fc1e41951f0d41db54e7d64799fe6ee851c`, and activation at block 34785. It supersedes earlier current-reward claims. Other chapters and the 74-feature catalogue retain their baseline scope; this correction is not a complete audit of later-branch functionality. [E31]

The retained `core/source/` snapshot is pinned to `bad4a06ea6c334e0b646ff1871b721704157dad9`; the earlier baseline release Core is `f2eace6bc69fca48b290142b8232bd1ced6a690b`. The former cannot prove the latter's reserve, WASM or post-quantum behavior. References to selected-Core paths in Appendix A mean that exact private source, not a relabelled snapshot. Publication rights and source-offer obligations remain unresolved under `LICENSE_DECISION.md`. [E01][E24]

Chapter 2

Purpose and design boundaries

2.1 Problem and existing approaches

Centralized inference endpoints simplify deployment but concentrate availability, pricing and model control. Uncoordinated resource markets can advertise capacity without proving delivery. Putting inference directly inside consensus would force validators to reproduce potentially nondeterministic outputs and to depend on model storage, accelerators and external services. PQWERAI instead uses the ledger for deterministic state, identity commitments and settlement, while providers supply computation outside consensus. [E10][E12]

The design aims to let application developers combine smart contracts, model discovery, signed quotes and bounded wallet authorization. It aims to let operators contribute CPU or GPU resources voluntarily. These are participation objectives, not evidence of a geographically diverse marketplace or profitable operator economics. Customer-observed delivery, provider identity and dispute handling remain essential even when payment is on chain.

2.2 Three independent correctness questions

Ledger correctness asks whether nodes agree on valid ordered state transitions. Service-accounting correctness asks whether an accepted charge stays within the quote, observed delivery and funded budget. Model-output quality asks whether an answer or training update is useful, accurate and safe. A valid block cannot establish the third property, and a provider signature alone cannot establish useful computation. [E07][E12][E13]

Separating these questions permits a provider outage without changing block validity and permits a conservative customer to reject a service without rejecting a valid chain. It also means the ledger does not remove arbitration, attestation or off-chain custody assumptions. Availability and quality require additional evidence.

2.3 Terminology

PQAI is the native 18-decimal asset used for gas, issuance, stake and service funding. **QAI**, whose test token symbol is **tQAI**, is a distinct six-decimal simulated-euro-reference asset. The **Quantum Reserve** receives native issuance and is distinct from the **QAI reserve vault**, which records simulated backing and margin. **Quantum Layer** denotes classical advisory software, not quantum hardware and not a consensus authority. [E16][E18][E19][E21]

Limitations. The inspected advisory layer can emit recommendations and restricted agent actions. It does not activate economic policy, own signing authority, or supply performance hints to the Go PASE executor merely because a Python reference implementation exists.

Chapter 3

System architecture

Evidence status: Implemented components with heterogeneous local and deployment evidence; full-system release acceptance remains open.

3.1 Execution, service and distribution planes

The Core node owns header and transaction validation, chain selection, EVM/WASM transitions, authenticated reward accounting and database persistence. Miners search for GhostRider seals. Smartnode signers submit checkpoint votes derived from executed state, and the registry determines eligible epoch sets. A signer is a role distinct from a mining process; four processes on one server are not four independent operators. Wallets construct transactions and bounded AI authorizations; explorers index observations without deciding consensus. [E01][E04][E05][E25]

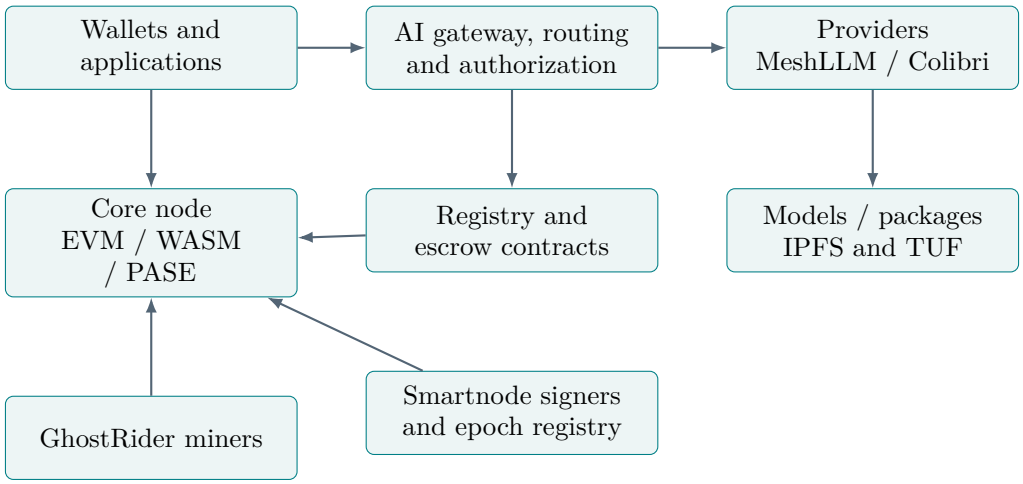


Figure 3.1: Component relationships. Inference output never enters the block-validity predicate. Arrows show principal interactions, not a claim that every integration is accepted.

The service plane includes provider/model/operator registries, inference gateways, runtime adapters, routing, usage receipts and settlement journals. MeshLLM and Colibri are upstream execution runtimes behind adapters, not validator modules. The distribution plane uses signed metadata, content-addressed artifacts and local TUF/CAR recovery tooling. Transporting a model or binary does not authorize its execution, and a CID does not establish continuing availability. [E10][E13][E15][E23]

3.2 State and persistence

The Core inherits account balances, nonces, code and storage from the Core-Geth state machinery. Block processing computes state and receipt commitments; the database retains canonical history and execution state subject to the client's storage configuration. Consensus extensions store reward counters, checkpoint state and validator lifecycle information in dedicated system state.

A branch reorganization must replay those effects consistently; off-chain journals must reconcile to the surviving branch. [E04][E07][E12]

The external settlement journal persists transaction identity and ambiguous submission state. Its role is recovery, not a second ledger. A timeout after dispatch is not evidence that execution failed. Restart handling must inspect the original transaction and canonical receipt before retrying. Local durability also depends on filesystem and device behavior; process-crash tests do not establish power-loss safety on every storage platform.

3.3 Authenticated network identity

Field	Retained 20777 profile
Chain / network identifier	20777 / 20777; hexadecimal chain ID `0x5129`
Native unit	PQAI; 10^{18} wei per PQAI
Genesis gas limit	15,000,000 gas
Genesis base fee	1,000,000,000 wei per gas
Epoch	32 blocks
Genesis features	WASM VM v1, pq-verify/v1, liveness/v1 (8-block duty window; 12-epoch inactivity setting)
Genesis identity	`0xe925fc110c0663ac319640f281d51caa7887b7171b2287fd28699b26f8cf1dc0`

Table 3.1: Source-pinned genesis parameters, not a live network-health assertion.

The profile commitment binds consensus configuration, including the helper identity and enabled extensions. Matching only a display name or chain ID is insufficient; installers, wallets and signers must verify the authenticated genesis and profile. Historical manifests can describe an obsolete package platform without describing the newer Windows-native adapters. [E02][E03]

3.4 Dependencies and failure containment

PoW, issuance, checkpoint rules, virtual machines and canonical PASE integration belong to one coordinated Core release boundary. They are not independently hot-swappable modules. Wallets, explorer services and AI adapters can have separate process lifecycles, but must maintain network and ABI compatibility. A provider/model registry read assumes an approved Core RPC; the inspected client is not a trustless light client. Public endpoint availability and geographically independent operation are separate acceptance requirements. [E01][E11][E24]

Chapter 4

Hybrid consensus and finality

4.1 Roles and assumptions

PQWERAI's inspected hybrid design uses GhostRider-sealed blocks for production and an FFG-style checkpoint overlay for finality. It is not an alternating PoW/PoS block lottery. The weight of a Smartnode determines checkpoint-vote influence and eligible reward participation, not the probability of finding a GhostRider nonce. The Core-Geth Ethash integration namespace remains in source even where the configured seal is GhostRider. [E03][E04]

PoW reasoning assumes the configured helper implements the intended hash, header checks are correct, and adversarial work and propagation do not dominate the relevant chain interval. Checkpoint reasoning additionally assumes an authenticated epoch set, sufficient honest participating weight, non-equivocating signers, correct ancestry checks and persistent safety locks. There is no proof in this edition that combines all of these assumptions into a complete dynamic-validator safety theorem.

4.2 GhostRider seal and difficulty

The helper receives an 80-byte domain-specific input containing the tag `G1GR`, parent hash, a commitment to the canonical RLP header with sealing fields zeroed, and the nonce. The remaining bytes follow the implementation's zero-initialized layout. The helper binary is hash-pinned; invalid output, timeout or a digest mismatch causes rejection. The verifier treats its displayed 32-byte digest as a big-endian integer H. For positive header difficulty D, the implemented acceptance predicate is

$$H \le \left\lfloor \frac{2^{256}}{D} \right\rfloor, \tag{4.1}$$

with a zero mix digest and the surrounding header/profile checks. The weak inequality is intentional: it reproduces the code rather than substituting another chain's target convention. [E03]

Let $d_0 = 64$ be the retained profile's base difficulty, D_p its parent difficulty and Δt the positive timestamp gap. With adaptive difficulty enabled and easing disabled, the selected rule is

$$D = \mathrm{clip}_{[d_0,16d_0]} \begin{cases} 2D_p, & \Delta t \le 2 \text{ seconds}, \\ \lfloor D_p/2 \rfloor, & \Delta t \ge 8 \text{ seconds}, \\ D_p, & \text{otherwise}. \end{cases} \tag{4.2}$$

Non-increasing timestamps are handled by surrounding validity rules; the adjustment function itself changes difficulty only on a positive gap. Without adaptive mode it returns d_0. A separately identified easing candidate changes long-gap halving behavior; it must not be attributed to 20777 merely because its flag exists. The bounded difficulty range is suitable for the retained experimental profile, not evidence of a production retarget design or an ASIC-resistance guarantee. [E03]

4.3 Epoch sets and checkpoint certificates

Checkpoints occur at heights divisible by 32. In registry mode, the validator set is read from executed state on the target branch and frozen for the target epoch. Let V_e be that set, w_i a member's effective weight, $W_e = \sum_{i \in V_e} w_i$, and Q the distinct valid voters in a certificate. The strict quorum predicate is

$$3 \sum_{i \in Q} w_i > 2W_e. \tag{4.3}$$

Votes bind version, chain ID, genesis, set root, epoch, stage and source/target identities. The selected implementation uses Ed25519 validator signatures. A valid signature from another chain, epoch set or checkpoint context does not contribute to this quorum. [E04][E18]

Algorithm 1 Selected-Core checkpoint transition, explanatory pseudocode

Require: Executed parent state S, source s, target t, certificate C

Check bounded encoding, epoch heights and source–target span
Derive expected context and epoch set from target branch state
Authenticate distinct votes and require equation (4.3)
Require fully executed s, t and $s \preceq t \preceq$ parent
Require s already justified; reject conflicting justification of t
Mark t justified and advance highest justified checkpoint if applicable
if $\text{epoch}(t) = \text{epoch}(s) + 1$ **then**
    Advance finalized checkpoint to s if higher than current finalized height
end if

The implementation bounds a link's ancestry span to 4,096 blocks and authenticates before walking potentially long history. Certificates are transported in transactions to a designated carrier; normal devp2p block propagation carries the resulting evidence. A submitted off-chain vote is not itself a finalized state transition. A target's justification finalizes its source only for a direct successor-epoch link; arbitrary distant justification does not finalize the preceding justified checkpoint. [E04]

4.4 Fork choice, partitions and recovery

The documented retained rule selects the greatest total-difficulty valid chain, retains the local head on equal total difficulty, and refuses reorganization below the BFT-finalized checkpoint. Equal-work partitions can therefore temporarily retain different heads; they require further valid work and compatible finalized ancestry to converge. An application must distinguish transaction inclusion, probabilistic work depth, justification and BFT finalization. Standard Ethereum `finalized` RPC-tag support must be verified separately from the custom G4 status interface. [E02][E04]

For a common fixed set, two strict quorums intersect in more than $W_e/3$: their weights sum to more than $4W_e/3$ within a universe of weight W_e. This is a set-intersection argument, not an independent proof of PQWERAI safety. Dynamic sets, long-range history, withdrawn stake, signer recovery, fork choice and inactivity-driven membership changes require additional analysis. A partition lacking a quorum should stall finality; elapsed wall time alone is not permission to discard a finalized lock. [E04][E05]

4.5 Smartnode lifecycle and penalties

The retained profile records a minimum stake of 1 PQAI, a maximum effective stake of 2,048 PQAI, up to 64 members in a frozen set and up to 256 registrations. Activation is delayed by two epochs; unbonding is four epochs. Withdrawals additionally depend on a BFT-finalized withdrawable epoch. Registration uses proof of possession; rotation, delegation, exit, unbonding and evidence operations have explicit state transitions. The profile states a three-epoch evidence window and a 25% slash fraction. These parameters describe this implementation, not a recommended production economic calibration. [E05]

Persistent signer journals reject double and surround votes and write safety state before releasing a signature. Restoring an obsolete journal can still create danger without an authenticated recovery anchor. The liveness profile tracks an eight-block duty window and can force-exit a validator unseen for twelve checkpoint epochs; inactivity exit does not itself slash stake. Membership changes intended to improve liveness must also preserve safety across partitions. The available tests and bounded recovery mechanisms do not eliminate that open assurance requirement.

Limitations. Historical local partition and restart tests demonstrate selected scenarios only. Independent implementation review, adversarial dynamic-set analysis, geographically distributed recovery and a production difficulty policy remain prerequisites for stronger claims.

Chapter 5

Networking and synchronization

5.1 Blockchain transport

The Core uses inherited Ethereum-family devp2p networking for block/transaction exchange and synchronization, with PQWERAI profile checks and peer-management additions. Discovery, persistent peer caches, signed seed information, retry policies and subnet diversity are availability mechanisms; none creates an independent operator by itself. The selected Core line includes a discv4 startup repair. An older package's discovery behavior must not be generalized to that selected revision. [E01][E06]

Node bootstrap requires chain ID, network ID, genesis, compatible profile and release identity. A peer with a familiar name or reachable IP is not trusted execution evidence. Synchronization must validate headers, work, transactions, state transitions and finality constraints rather than merely matching advertised height. A stale or inconsistent RPC can mislead service discovery even when the underlying protocol is correct. State download, historical execution and trusted checkpoint policies therefore belong in operator acceptance, not just a connectivity smoke test. [E02][E07][E11]

5.2 AI transport and NAT traversal

The separate Go AI sidecar uses libp2p provider advertisements and request streams, with protocol IDs `/pqwerai/ai/advert/1.0.0` and `/pqwerai/ai/http/1.0.0`. The code includes circuit-relay and hole-punch support and explicit policies for automatic P2P, opt-in direct ports and relay-only operation. Default application bounds in the inspected sidecar include 256 KiB request bodies and 8 MiB response bodies. These are transport limits, not model context limits. [E06]

NAT traversal depends on local routers, firewalls, upstream reachability and relay availability. A successful loopback or one-relay experiment cannot prove universal home-network connectivity. Resource-manager limits, request quotas and advertised bindings reduce exposure, but malicious peers can still consume bandwidth or attempt denial of service. Identity bindings must match the chain registry and exact model/runtime commitments. Publicly reachable control endpoints require their own authenticated configuration.

5.3 Eclipse and partition resistance

Independent seeds, outbound diversity and bounded backoff reduce dependence on one peer source. Residual risks include Sybil peers, common hosting, poisoned metadata and relay concentration. PoW or inference can continue while finality or safe settlement stalls. Recovery must preserve signer journals, verify finalized ancestry and package/profile identity, and reconcile ambiguous jobs before redispatch. These controls do not establish accepted multi-region recovery. [E04][E06][E12]

Chapter 6

Transactions, state transitions and fees

6.1 Lifecycle

A wallet constructs an Ethereum-style transaction with destination or creation data, nonce, value, gas limit and an appropriate fee envelope. The retained profile lists legacy EIP-155, EIP-2930 access-list and EIP-1559 dynamic-fee transactions. Native account authorization remains secp256k1 ECDSA. The node checks the envelope and admissibility before pool inclusion; a miner chooses an ordered block; peers validate and execute it; canonical receipts expose success, reversion, gas and logs. Checkpoint finalization is a later consensus event. [E07][E18][E25]

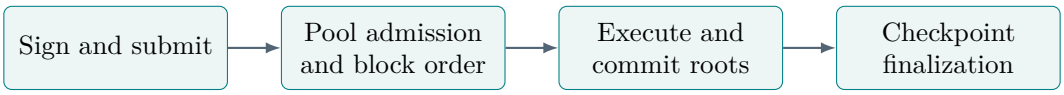


Figure 6.1: Transaction lifecycle. Inclusion and successful execution precede finality; reversion remains a valid execution outcome.

Execution is an ordered transition $S_{i+1} = T(S_i, \tau_i)$ for a valid transaction τ_i in state S_i. A contract-level revert restores the call's reversible effects while retaining protocol-defined nonce and fee consequences. An invalid transaction or block is a different condition from a valid receipt with failed execution. PASE must preserve this distinction, including cumulative gas and log indexing. [E07][E09]

6.2 Fee accounting

For an EIP-1559 transaction with maximum fee $f_{\max}$, priority cap $p_{\max}$ and applicable base fee b, the inherited effective price is $\min(f_{\max}, b + p_{\max})$, provided the transaction satisfies the fork's admissibility rules. For execution gas g, base-fee burning contributes bg and the priority component contributes $(f_{\text{effective}} - b)g$ to the miner. Actual refunds and intrinsic gas follow the configured VM rules; a simple multiplication by the submitted gas limit is not the settled charge. [E07][E16]

Base-fee burning is recorded separately from gross issuance. AI escrow transfers are payments from existing balances and do not mint additional PQAI. Similarly, locking validator collateral changes who can spend a balance, not total monetary issuance. Supply reporting must reconcile these categories and distinguish circulating balances from locked or operationally inaccessible balances.

6.3 Application safety

Wallets must check genesis and contract identity. Applications should verify receipt status and an appropriate finality anchor. Explorer indexing alone is insufficient. Nonce races, timeouts and reorganizations require explicit recovery; an API retry must not create a duplicate payment. [E02][E12][E25]

Chapter 7

EVM, WASM and developer tooling

7.1 Inherited EVM semantics

The Core retains the Core-Geth EVM account model, bytecode execution, contract creation, storage, gas metering and receipts. The genesis manifest describes London-family rules and explicitly configured later EIPs, including PUSH0, transient storage and MCOPY at block zero. This is a particular fork configuration, not a declaration of compatibility with every present or future Ethereum mainnet feature. The Shanghai Yellow Paper is a background specification, not an exact normative description of all PQWERAI extensions. [E07][3]

The implementation supports familiar JSON-RPC and wallet workflows, with first-party CLI, Windows wallet and developer documentation. Compiler output, opcode activation, chain-specific precompiles and RPC behavior still require conformance checks. A contract that compiles successfully can revert on deployment or rely on unavailable semantics. ABI and bytecode identity must be verified at the actual destination chain. [E25]

7.2 Required WASM VM v1

WASM is official required release scope, enabled from block zero in the retained 20777 profile. The inspected implementation uses a constrained, metered runtime and a shared ledger execution context, with cross-VM rollback and serial/PASE equivalence obligations. It is not a separate chain or an AI execution engine. Configuration commitments bind activation, and product classification does not authorize a silent fork. [E02][E08]

Resource isolation is an unresolved security issue. The 14 November author report identifies excessive locals/engine-stack consumption, cold-compile cache thrashing, host-copy allocation and failed-deployment compilation cost in the selected Core. A local candidate fixes selected cases, but locals metering requires an approved fork and does not automatically reinstrument previously stored modules. These findings prevent an unqualified claim that gas bounds every runtime resource. Chapter 15 records their scope without presenting attack instructions. [E08]

7.3 Language and Studio boundary

Studio contains multiple compiler paths and example contracts. Dated conformance evidence reports 95 cases passing against the selected Core after wallet-source hardening; that is finite language/ABI coverage, not certification of arbitrary applications. Compiler pinning, import confinement and checks against attachment substitution matter because a successful compiler can otherwise build different source from what the developer reviewed. Example ERC-20 contracts remain unaudited examples. [E25][E08]

Limitations. No production application audit, universal Ethereum equivalence, or absence of resource denial of service is established by the available compatibility tests.

Chapter 8

Predictive and parallel execution: PASE

Evidence status: Implemented opt-in executor, including general contract speculation; no verified sustained network-capacity gain.

8.1 Problem and prior work

Serial transaction order provides a simple execution oracle but can leave independent work unexploited. Access-list prefetching reduces read latency; optimistic execution speculates and validates dependencies before committing effects. Block-STM is an authoritative example of preset-order optimistic execution [4]. PQWERAI's comments use this comparison, but its scheduler, merge rules and measured performance must be assessed independently. [E09]

The retained early PASE report describes transfer-only execution. Current source additionally contains general contract speculation, explicit access recording and adaptive batches. This edition follows that code and the shipped-module identity rather than extending the old report's limitations indefinitely. `scripts/pase_pin.py` recomputes the Go module directory hash to compare the repository copy with the selected Core's recorded dependency. [E09]

8.2 Prediction and prefetch

The predictor normalizes bounded access hints and history; the prefetch component performs bounded state reads. Predictions are local optimization inputs, not authority over transaction validity. A missed access must lead to actual execution and dependency validation, not an incorrect result. Prediction budgets, cancellation, bounded workers and fallback prevent an untrusted hint from becoming an unbounded scheduler obligation. Neither a prediction score nor an advisory Python conflict graph changes the canonical order. [E09][E21]

8.3 General optimistic execution

General mode speculates on per-worker copies of committed state while recording account/storage reads, writes, resets and effects that cannot safely be merged. The initial default threshold is 64 transactions. The inspected general scheduler bounds batches between 16 and 1,024 transactions and uses a serial backoff of 128 transactions under severe contention. Eligibility, VM configuration and global limits can select serial execution instead. These constants are implementation policy, not consensus parameters. [E09]

For transaction i in one batch, let R_i denote its recorded reads and $W_{<i}$ the keys dirtied by earlier canonical transactions in that batch. A necessary merge condition is

$$R_i \cap W_{<i} = \varnothing, \tag{8.1}$$

with additional checks for account resets, opaque effects, execution errors and gas constraints. This expression summarizes only the conflict filter; it is not a complete correctness specification.

Fee balance deltas and account existence require care because some credits can commute while reads of those accounts cannot.

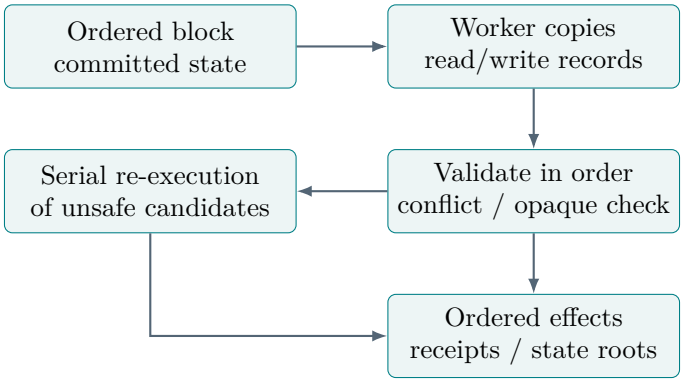


Figure 8.1: PASE execution boundary. Speculation changes how work is attempted; canonical order determines the committed result.

Algorithm 2 PASE general-mode batch, explanatory pseudocode

Require: Canonical transaction order, committed state, remaining block gas

Speculate a bounded batch on private worker state copies; record effects
$D \leftarrow \varnothing$ ▷ keys dirtied in canonical order
for transaction i in canonical batch order **do**
    if reads conflict with D, effect is opaque, or safe merge checks fail **then**
        Execute i serially on committed state; record actual writes
    else
        Merge verified final values and permitted balance deltas
    end if
    Update block gas, receipts, logs and D using committed effects
end for
Shrink or grow next batch using observed conflict fraction
Normalize surviving log indices to canonical block order

The adaptive policy shrinks and enters serial backoff when conflicts exceed half a batch, halves a batch above one-quarter conflict, and grows it when conflicts are below one-sixteenth. Worker time and wall-clock measurements affect scheduling statistics, not state commitments. A worker copy is discarded on fallback; canonical snapshots and VM semantics handle transaction-level reverts. The block processor remains responsible for rejecting a partial result after an execution error. [E09]

8.4 Correctness obligation and costs

For identical pre-state, ordered transactions and block context, the required equivalence is

$$\mathrm{Observe}(E_{\mathrm{PASE}}) = \mathrm{Observe}(E_{\mathrm{serial}}), \tag{8.2}$$

where observation includes state and receipt roots, status, cumulative gas, logs, balances, nonces, code and storage. Actual system transitions, fee burning and rewards must also match in integrated tests. This is an acceptance invariant, not a theorem proved for all EVM/WASM programs by this paper. Tests include contract conflicts, reverts and serial comparisons; historical process tests also cover reorg and restart scenarios. [E07][E09]

If n transactions have serial cost C_i, optimistic work includes their speculative execution, state copying, read/write recording, validation and any re-execution. Shared-account conflicts can make it slower than serial execution. A diagnostic model is

$$T_p \approx T_{\text{copy}} + T_{\text{record}} + T_{\text{validate}} + T_{\text{commit}} + \frac{\sum_i C_i}{p_{\text{effective}}} + T_{\text{redo}}, \tag{8.3}$$

an analytical decomposition, not a measured performance law. The effective parallelism is bounded by hardware, dependency chains and memory pressure. Serial fallback bounds unnecessary continuing speculation; it does not eliminate the cost already spent. No TPS multiplier follows from worker count.

Limitations. A benchmark must compare identical roots and workloads, report fallback rates, copy/memory overhead and tail latency, and include canonical commitment and persistence. Prefetch microbenchmarks, standalone EVM rates and RPC submission rates are not network throughput.

Chapter 9

MeshLLM and accountable AI services

9.1 Runtime integration and resource sharing

MeshLLM is an upstream inference runtime, supervised by PQWERAI adapters. The hardware interface distinguishes detected, verified, failed and unavailable backends; CPU, CUDA, ROCm and Vulkan paths depend on the pinned runtime and actual host support. A vendor identifier alone does not authorize scheduling. The inspected interface requires a real model load and inference probe for verification and refuses to fake unsupported layer-stage loading. [E10]

Participants authorize memory, compute and spending limits. The backend estimates weights, attention KV cache, non-layer tensors, buffers and headroom; an estimate is an admission aid, not proof that a model fits. Isolated child processes, cancellation, mining/chain guards and bounded job admission reduce resource contention. They do not constitute a strong multi-tenant isolation proof or confidential-computing guarantee. A hostile model or runtime dependency remains a supply-chain risk. [E10][E13]

9.2 Provider discovery and routing

The native registry client binds provider eligibility to chain/genesis, approved contract runtime hashes and a canonical finalized anchor. Current and finalized bindings must agree. Capabilities, endpoint, payout, receipt key, model and runtime commitments must match the registered records. The client trusts its approved Core RPC rather than verifying a light-client proof. [E11]

Customer selection first applies hard constraints for model identity, budget, resources, consent and freshness. It then ranks eligible candidates using estimated queueing, time to first token, completion time, price, customer-local reliability and operator concentration. Signed observations expire, while a durable high-water journal resists sequence rollback and equivocation across restart. Provider-reported speed remains a claim until measured by the customer. Reputation is local observation history, not an incorruptible global score.

The documented default asks for two live operator identities and constrains recent concentration after enough observations. A local one-PC fixture relaxes those requirements explicitly. Directory identities do not prove independent owners, and a bootstrap directory can itself become a central trust point. Ranking estimates can use a different resource-cost model from final v3 output-token charging; only the signed quote and funded cap authorize settlement. [E11][E12]

9.3 Placement, cooperative pools and training

The pool planner considers single-node placement, contiguous layer pipelines and replica scale-out. It consumes measured free resources, compute time and link latency/bandwidth; it does not treat aggregate VRAM as one device. For a single sequential decoding request, its model sums

stage computation and hop transfer costs. In explanatory notation,

$$t_{\text{token}} = \sum_{j} t_{\text{compute},j} + \sum_{(j,k)} \left(\text{RTT}_{jk} + \frac{8a_{jk}}{10^6 B_{jk}} \right), \tag{9.1}$$

where RTT and times are seconds, a is activation bytes, and B is measured Mbit/s. This is the planner's estimate for one request, not a demonstrated distributed throughput bound. Replicas improve concurrent capacity differently from a pipeline. A slow or distant stage may worsen latency and be left on standby. [E14]

Collaborative inference and fine-tuning are established research directions [5]; their published results do not validate PQWERAI's independent-host pool behavior. The repository has training budgets, dataset permissions, task/coordinator and model-promotion components. Their existence does not establish accepted multi-host training, useful-update verification, poisoning resistance, or training compensation through inference-only escrow v3. Independent cooperative execution and contribution attribution remain acceptance work. [E14][E17]

9.4 Delivered-token accounting

The customer meter tracks monotonically sequenced chunks, ignores duplicates, rejects gaps and enforces a quoted output limit. Provider-signed cumulative acknowledgments bind job, model version, tokenizer digest, sequence, token counts and output digest. A final receipt must match delivered output. Prompt tokens are recorded for resource accounting; v3 charges accepted output tokens. For funded budget B, quoted output-token price p and accepted output count n,

$$C = \min(B, np). \tag{9.2}$$

Amounts use exact nonnegative integers. Sequence and token checks bound claims but do not prove answer quality or that the provider ran the stated model honestly. [E12]

No transaction is sent for each token. Settlement uses a job lifecycle: funding, proposal, acceptance or dispute, then settlement/refund according to contract rules and deadlines. A designated arbiter is a trust assumption. Durable reservation and reconciliation avoid treating ambiguous work as safe to redispatch; an "exactly once" objective must cover both off-chain attribution and the eventual canonical transaction, including reorganizations. [E12][E17]

9.5 Models, storage and certification

Manifests and registries bind artifact digests, formats, version and approval status. Candidate or experimental model use requires explicit policy. Content addressing verifies identity; signed certification expresses an approver's policy decision. Neither establishes benign behavior or factual answers. Automatic retrieval must preserve consent, size limits, signature checks and the exact approved model identity; a familiar model name is insufficient. [E13]

IPFS retrieval and release-fabric tools support content verification and recovery. Pinning and independently retained replicas remain necessary for availability [6]. Large models also require license and redistribution review. Prompts, embeddings and training data can reveal customer information to providers or logs; encryption in transit does not hide them from the executing host. Redaction, limited retention and explicit data permissions remain operational obligations.

Limitations. No universal useful-work proof, privacy guarantee, independent multi-host scaling result, or fully autonomous training economy is established. Service failure and fraud can still require dispute resolution outside cryptographic receipt checks.

Chapter 10

Colibri backend and orchestration

Evidence status: Concrete local backend with dated one-host CUDA evidence; distributed acceptance and payment integration remain open.

10.1 Implementation boundary

`ColibriBackend` implements the shared compute interface. It supervises `coli serve` on loopback with per-process authentication, verifies runtime/model hashes, supports JSON/tool calls and SSE, and confirms engine idleness after cancellation. This adapter does not establish a deployed orchestration network. [E15]

The report pins Colibri v2.1.0 and a Qwen3-Coder-30B-A3B int4 revision: 139 runtime files and 59 model files totaling 19,398,108,288 model bytes. Same-origin unsigned checksums establish consistency, not independent authenticity. GPU memory use was observed during loading. [E15]

10.2 Measured local behavior

Run-2 observation	Dated result, 13 November 2026
Environment	Windows 11; Ryzen 9 3900X, 24 threads; 15.9 GB RAM; RTX 4060 Ti 16 GB; driver 591.44; Python 3.12.4
Model/runtime configuration	Colibri v2.1.0 CUDA; Qwen3-Coder-30B-A3B int4; `--ram 10 --ctx 4096`
Load to healthy	38.8 seconds
Median of three 64-token probes	5.94 output tokens/second
96-token stream	9.39 seconds to first token; 17.9 seconds total
Cancellation	Engine idle after 0.59 seconds following client cancellation

Table 10.1: Historical one-host Colibri measurements. These are not new measurements for this edition.

The probes and stream have different timing intervals and cannot be combined into one capacity claim. No confidence interval, independent replication, AMD result or WAN measurement is established. Stand-in-server tests cover adapter authentication, health, cancellation and tamper refusal; they do not benchmark the runtime.

10.3 Tradeoffs and unresolved integration

Gateway selection, pool placement, settlement, IPFS distribution and signed model approval remain unaccepted by this report. Upstream cluster transport lacks authentication/encryption, and the adapter leaves it disabled. Distributed workers need authenticated transport and lifecycle controls. Local API authentication and engine-stop confirmation do not establish sandboxing or resistance to model-triggered resource exhaustion. [E15]

Chapter 11

Native monetary and service economics

11.1 Asset roles and initial allocation

PQAI pays transaction gas, collateralizes Smartnode participation, rewards block production and checkpoint participation, funds AI jobs, supplies the Quantum Reserve and funds the IPFS reward pool. It is not the QAI redemption liability. The retained genesis allocation sums, using integer balances rather than floating-point display fields, to

$$S_0 = 1{,}001{,}128 \text{ PQAI} = 1{,}001{,}128 \times 10^{18} \text{ wei}. \tag{11.1}$$

This includes the fixture's faucet/operational balances and four 32-PQAI validator balances. Stake locking is not additional genesis issuance. The manifest's operational allocations are testnet facts, not a selected mainnet premine or ownership distribution. Private wallet addresses are deliberately omitted. [E02][E16]

11.2 Block rewards: accrual and actual payment timing

A block's production, its miner's eventual issuance credit and the block that performs that credit are different events. The baseline 20777 Core has no fixed native subsidy paid anew on every block. Its issuance journal settles when the processed block has a positive parent height divisible by 32: for parent $h = 32e$, the settlement is executed in block $n = 32e + 1$. The payout covers recorded miner slots for the preceding 32 blocks. Ordinary blocks between settlements do not mint another copy of that epoch amount. Priority fees follow transaction execution separately; base fees are burned. [E16]

Before the IPFS fork, issuance used 72% PoW / 18% active-Smartnode / 1% founder / 9% Quantum Reserve. CR-0051 reallocates five percentage points from PoW to the IPFS pool from recorded activation block 34785: the post-fork split is 67/18/1/9/5. These are allocations of one gross settlement. Chapter 12 documents activation, eligibility and claims. [E31] In particular, the 9% credits the native-PQAI Quantum Reserve, not newly minted QAI tokens. A miner receives the PoW credit associated with its recorded block slots, while a participating validator receives its share of the epoch's eligible PoS allocation. The winning miner of a settlement block is not automatically the recipient of the entire epoch PoW amount. [E05][E16]

11.2.1 Gross issuance is participation- and stake-dependent

For a call to the actual reward function, let w_i be the supplied cohort member's effective stake in gwei and $W = \sum_i w_i$. These are effective cohort weights, not all deposited balances or the native coin's total supply. The code computes

$$q = \left\lfloor \frac{64 \cdot 10^9}{\lfloor \sqrt{W} \rfloor} \right\rfloor, \qquad b_i = \left\lfloor \frac{w_i}{10^9} \right\rfloor q. \tag{11.2}$$

Each b_i is a base reward amount in gwei. For each newly credited group of eligible voters U, the proposer journal increases by

$$\Delta J = \left\lfloor \frac{54 \sum_{i \in U} b_i}{448} \right\rfloor . \tag{11.3}$$

The floor is applied to the group's numerator at that crediting call, not to a continuous annual estimate. Repeated evidence does not repeatedly credit an already marked duty. In the finality profile, individually carried sub-quorum participation records can record liveness without earning a reward; eligible certificate-derived duty credit has its own checks and window. [E16][E05]

At settlement, let A be the sum of effective weights of members whose duty flags are already set, D those members, and J_e the stored proposer journal before current-block credit. Define the ordered weight sequence $(k_1, k_2, k_3) = (14, 26, 14)$. The source's gross amount is

$$G_e = 10^9 \left[J_e + \mathbf{1}_{e>1} \sum_{i \in D} \sum_{j=1}^{3} \left\lfloor \frac{b_i k_j \lfloor A/10^9 \rfloor}{64 \lfloor W/10^9 \rfloor} \right\rfloor \right] \text{ wei.} \tag{11.4}$$

The two separate occurrences of 14 are intentional. The condition $e > 1$, each integer division, the actual effective set and the journal state are part of the calculation. With liveness/v1, settlement uses the duties recorded before that block; evidence carried in the current block is credited afterward for the next settlement period. Thus four registered validators do not imply four credited duties in every epoch. [E16]

11.2.2 Exact split and per-block miner allocation

The generic helper applies

$$F_e = \lfloor G_e/100 \rfloor, \qquad R_e = \lfloor 9G_e/100 \rfloor, \tag{11.5}$$

$$N_e = G_e - F_e - R_e, \qquad P_e^{(0)} = \lfloor 80N_e/100 \rfloor, \tag{11.6}$$

$$V_e = N_e - P_e^{(0)}. \tag{11.7}$$

The actual gross calculation in equation (11.4) yields whole-gwei multiples. Consequently, before the fork the four helper shares are *exactly* 72/18/1/9, rather than approximate percentages perturbed by arbitrary one-wei inputs. The earlier edition's examples with gross amounts of 7, 101 and 10,000 wei were generic helper examples, not reachable epoch rewards, and are removed here. [E16]

After activation, the carve is $I_e = \min\{P_e^{(0)}, \lfloor G_e/20 \rfloor\}$ and actual PoW is $P_e = P_e^{(0)} - I_e$, giving the current five-way allocation. [E31]

The source credits 32 recorded miner slots from P_e, using a final-slot remainder in the generic algorithm. Current whole-gwei epoch amounts make P_e divisible by 32, so each recorded block slot receives $P_e/32$ wei. An operator mining multiple slots receives multiple such credits. The PoS amount is weighted among eligible duty-marked members; any integer residue goes to the last eligible member in the implementation's ordering. The founder and reserve shares come from the same G_e, not surcharges on transfers, fees or AI payments.

Allocation categories must also be distinguished from recipient accounts. In the retained genesis profile, all four validator payout fields point to the same destination as the founder field. Consequently, if those payout settings remain unchanged, that destination receives the eligible PoS allocation as well as the founder allocation: 18% plus 1% of a positive settlement. This does not change the separate 18% PoS and 1% founder counters or establish independent validator control. Later registry/payout changes must be checked in canonical state. The accounting vectors below deliberately use distinct test destinations to isolate category amounts; no operational addresses are reproduced. [E02][E16]

11.2.3 Executed accounting examples, not a fixed subsidy

The following vectors are explicitly pre-fork baseline examples; use Table 12.1 for recorded post-fork amounts. A documentation-only Go overlay calls the *actual* baseline-Core `g3Apply` function with four equal 32-PQAI effective weights and liveness window 8. Four eligible voter indices are supplied at block 34, then three at block 66. The accounting test observes zero new issuance at its bootstrap settlement block 33, the following distinct gross amounts at blocks 65 and 97, and no new issuance in the intervening ordinary blocks. It checks bucket conservation and total credited balances. The overlay does not change Core source, validate a certificate, seal a block, run a node or measure current network income. [E16]

Amount in PQAI	Four credited members	Three credited members
Gross epoch settlement	0.022079520000	0.012937215000
PoW total	0.015897254400	0.009314794800
PoS total	0.003974313600	0.002328698700
Founder total	0.000220795200	0.000129372150
Quantum Reserve total	0.001987156800	0.001164349350
PoW credit per recorded block slot	0.000496789200	0.0002910873375
PoS credit per eligible equal-weight member	0.000993578400	0.000776232900

Table 11.1: Pre-fork source-accounting vectors under explicitly chosen participation inputs; the later fork reduces PoW and adds the IPFS pool. Values exclude transaction tips and are not a fixed block reward, a live-network measurement or promised earnings.

The pre-fork four-member example gives an accounting average of $G_e/32 = 0.000689985$ PQAI gross per covered block, of which 0.0004967892 PQAI is that slot's miner allocation. These numbers are neither a new subsidy at each block nor a universal current rate: the three-member example already yields different values at unchanged registered stake and unchanged percentage split.

Separately, the dated dry-run report records cumulative issuance of 0.012937215 PQAI at observed height 1014, with the pre-fork four-way proportions. That is a cumulative observation since genesis, not evidence that every block or every epoch issued that amount. Current rewards must be established from actual consecutive canonical reward-counter deltas, effective cohort/duty state and miner slots, with fees accounted separately. [E16]

11.3 Supply, burns and inflation

For cumulative issued gross $\sum_e G_e$, cumulative fee burn B_f, registry slash burn B_s and native amounts designated irrecoverable by a consistent accounting convention B_o, economic net supply is

$$S = S_0 + \sum_e G_e - B_f - B_s - B_o. \tag{11.8}$$

Registry burn counters use gwei and must be converted to wei. Native transfers between ordinary accounts do not affect this equation. The Quantum Reserve's transfer to the zero address is labelled a burn by the contract; it does not delete that address's ledger balance. A report must state whether it subtracts such balances economically and must not subtract them twice. Circulating supply additionally depends on what locked, treasury or inaccessible balances the reporter excludes. [E16][E20]

The profile has no configured economic cap and no verified periodic halving. "Uncapped" is an economic policy, not a claim of infinite-width per-account arithmetic: accounts and counters

have explicit representational/resource limits. For an observed interval $[t_0, t_1]$, net inflation can be reported as $(S(t_1) - S(t_0))/S(t_0)$ with the same burn convention. Annualization requires assumptions about stake, participation, block intervals, fee demand and outages. No annual rate or price forecast is justified by the current evidence.

11.4 Security budget and incentive limits

A nominal issuance share does not determine attack cost. If p is an assumed currency price per PQAI and T_e the miners' aggregate tips in wei, their aggregate gross receipts would be $p(P_e + T_e)/10^{18}$ before operating costs. An individual miner receives its recorded slot allocation and its own tips, not the entire P_e. Validator revenue depends on eligible participation, effective stake and slash exposure. This dimensional scenario is not a measured security budget or an executable conversion quote. Thin markets, concentrated hash supply, common signer custody and correlated failures can undermine security. Founder/reserve concentration is an explicit custody risk. [E05][E16][E20]

11.5 Customer-funded service revenue

ServiceEscrow v3 has at most four configured non-provider payees, immutable basis-point shares whose sum is below 10,000, and a distribution commitment. For accepted charge C and configured share β_j,

$$c_j = \left\lfloor \frac{C\beta_j}{10{,}000} \right\rfloor, \qquad c_{\text{provider}} = C - \sum_j c_j. \tag{11.9}$$

The provider receives the exact remainder. The existing local default is provider-only; no universal live schedule is inferred. There is no founder or training-fund share introduced by v3. Historical v1 and v2 jobs retain their contract-specific terms; the proposed 85/5/5/5 policy is not activated by its appearance in design documents. [E12][E17]

Limitations. No evidence establishes long-term profitability, an optimal emission rate, economic attack immunity or sustainable real-money reserve funding. These require calibrated simulations, market/liquidity assumptions and independent review.

Chapter 12

IPFS sharing rewards: the five-percent issuance allocation

Evidence status: Implemented in the later CR-0051 source line, with recorded activation on 20777 at block 34785 on 16 November 2026. No new remote observation is claimed.

12.1 The missing allocation and its source

The five-percent IPFS reward is an allocation of *new native issuance*, taken from the PoW share. It is not the similarly named five-percent IPFS service-revenue proposal in the older 85/5/5/5 design. The later source implements `ipfs-reward/v1`; its recorded activation changes the block/epoch issuance split from 72/18/1/9 to **67% PoW / 18% PoS / 1% founder / 9% Quantum Reserve / 5% IPFS pool**. Gross issuance is unchanged. [E31]

Earlier editions audited repository `8ca81f7035cd95e6ea3a228fdebb38c44bdb8d1d` and Core `f2eace6bc69fca48b290142b8232bd1ced6a690b`, which predate this fork. This edition adds a focused supplement from repository `6f66e031b018b925c6427be03b9d8c0a73a5a245` and the recorded deployed Core `12127fc1e41951f0d41db54e7d64799fe6ee851c`. Older source-pinned measurements and the 74-feature catalogue retain their original historical scope; they are not silently relabelled as a complete audit of the later branch. The former four-way split is historical before activation, not the post-fork reward rule. [E31]

12.2 Funding the pool without increasing supply

Let G_e be the same duty-dependent gross epoch amount described in Chapter 11, and $P_e^{(0)}$ the helper's original PoW allocation. At a settlement processed in block n,

$$I_e = \begin{cases} 0, & n < 34785, \\ \min\{P_e^{(0)}, \lfloor 500 G_e / 10{,}000 \rfloor\}, & n \geq 34785, \end{cases} \tag{12.1}$$

$$P_e = P_e^{(0)} - I_e. \tag{12.2}$$

The minimum is a defensive arithmetic guard; the retained 72% pre-carve profile covers the 5% allocation. Founder, PoS and Quantum Reserve amounts are unchanged. For reachable whole-gwei gross amounts the five shares are exactly 67/18/1/9/5, and

$$P_e + V_e + F_e + R_e + I_e = G_e. \tag{12.3}$$

This does not charge transfers, gas fees or AI invoices, and does not issue QAI tokens. The 5% enters a reward-pool contract before individual operator claims. It is a shared budget, not 5% of issuance for every participant. [E31]

The fork retains chain 20777 and its genesis. Every executing node needs the same activation height and pool configuration; an incompatible older node can diverge at activation. Validator signers acting only as RPC clients do not themselves execute this rule. The dated rollout record

identifies the source and coordinated node configuration; an ordinary package update or IPFS content hash is not authority to change the schedule.

Post-fork allocation	Share	PQAI in the recorded settlement
PoW miners	67%	0.008667934050
Active Smartnodes	18%	0.002328698700
Founder	1%	0.000129372150
Quantum Reserve	9%	0.001164349350
IPFS reward pool	5%	0.000646860750
Gross	100%	0.012937215000

Table 12.1: Recorded post-fork settlement at height 34785; the same amounts were recorded at 34817 and 34849. These are dated counter deltas, not a fixed subsidy or promised future income.

The preceding recorded settlement at 34753 had 72% PoW and no IPFS allocation. At 34785, 34817 and 34849, all five shares sum to gross and the pool balance delta equals the IPFS credit. The source document's older "not activated" heading is superseded by this later execution record. This paper has reconciled those recorded integers; it has not freshly queried a public endpoint. [E31]

12.3 Who can receive the allocation

Eligible participants operate IPFS Kubo nodes, optionally associated with IPFS Cluster. The pool requires an active operator identity enrolled in OperatorDirectory, a registered payout and Kubo peer identifier, and declared content from the curator-managed manifest. Merely enabling IPFS, pinning arbitrary unrelated content or displaying a Cluster peer ID does not establish a claim. One reward node is registered per enrolled operator identity; enrollment is the system's Sybil boundary. [E31]

The curator adds content commitments and may retire content, but has no direct withdrawal path for reward funds. Content identity in the reward mechanism is the committed Merkle root and size; the CID is recorded as informational metadata. Holdings declared before round zero can count from round zero; later changes take effect from the next round. The contract caps declared holdings at 64 content entries per operator.

The recorded pool configuration starts at block 34785, uses rounds of 192 blocks and accepts proofs strictly after a round's first block through its 96-block proof window. These are block counts, not a fixed wall-clock SLA. Two challenges per round ask for 1,024-byte chunks, with a shorter final chunk where appropriate. Leaves hash a domain byte and chunk bytes; internal nodes hash a distinct domain byte and child hashes, and the tree is padded to a power of two. [E31]

12.4 Proof, weighting, settlement and claim

The challenge seed binds the round-start block hash, operator identity, round and challenge index. It selects a declared content item and chunk using deterministic hash/modulo operations. The operator supplies the challenged bytes and a Merkle path of the expected depth. The contract checks timing, identity, lengths and roots; a successful round proof records the weight as the total bytes of the operator's active declared holdings. Sampling two chunks is not verification of every byte or every subsequent retrieval request.

Let $w_{o,r}$ be operator o's recorded successful-round weight and W_r the total such weight. At ordered round settlement, B_r is the pool's then-unallocated native balance after subtracting already committed unpaid budgets. For $W_r > 0$, the operator's claim is

$$c_{o,r} = \left\lfloor \frac{B_r w_{o,r}}{W_r} \right\rfloor . \tag{12.4}$$

Previously claimed rounds cannot be paid again. Claims pay the registered payout destination; another caller may trigger a claim without changing that destination. Rounding dust remains reserved. A round without successful proofs allocates no new budget, and unallocated income rolls forward. Thus pool funding and operator payout can occur at different heights. External donations, if present, must be separated from consensus-issued credits when reconciling the pool. [E31]

For an assumed 1-PQAI round budget and successful weights of 370,000 and 300,000 bytes, the formula allocates approximately 0.552238805970149253 and 0.447761194029850746 PQAI, leaving one wei of rounding dust. These weights come from the rehearsal design; the 1-PQAI budget is an arbitrary arithmetic example, not a measured payout. With zero successful proofs neither operator earns a share merely for being registered.

The supplied operator agent builds a manifest/root from its local Kubo API, declares only reproducible holdings, reads challenged byte ranges, submits proofs, and attempts settlement/claims. Its transactions require native gas. A quoted 5% pool allocation therefore does not imply positive net operator profit: funding volume, competitors' weights, successful proof timing, gas, storage, bandwidth and maintenance costs determine actual outcomes.

12.5 Evidence and limitations

The dated disposable six-node rehearsal first failed because pre-start holdings could not be declared. After the fix, it recorded 12/12 checks passing, eight post-fork epochs at the five-way split, honest operator proofs and claims, and a corrupted operator earning nothing. Its Kubo interfaces were stand-ins; the report does not establish independent real Kubo/Cluster operator availability. The deployed pool's activation record stated that no operator had yet enrolled, so that observation proves funding of the pool, not completed real-operator payouts. No later operator income is inferred. [E31]

For this correction, six focused tests on the exact recorded deployment Core pass: four fork/share/settlement tests and two real-EVM pool/Merkle tests. Six Python agent/Merkle tests pass in the separately pinned IPFS worktree. These are new local checks, distinct from the historical activation observations. No branch merge, contract deployment, node upgrade or live transaction was performed for the whitepaper.

Proof of producing sample bytes inside a window does not prove where or how long they were stored: an operator may retrieve them from another node in time. Cluster membership is recorded rather than cryptographically established. Directory enrollment and curator policy remain trust points; the round-start miner can influence the challenge seed. The scheme does not guarantee permanent retention, independent replicas, full-content availability or financial return. These limitations qualify the meaning of "IPFS sharing" without removing the implemented five-percent incentive.

Chapter 13

QAI, reserve accounting and post-quantum accounts

13.1 Purpose, identity and deployment status

QAI is implemented as a six-decimal test token, symbol tQAI, whose balances belong to quantum smart-account identifiers rather than ordinary ECDSA addresses. It is a valueless simulated-euro-reference system. It is neither native PQAI nor proof that a licensed euro-backed issuer exists. The token deliberately omits the ordinary ERC-20 approval/transferFrom authorization path; integrations must use its actual ABI. [E18][E19]

The 14 November deployment record reports the QAI reserve stack and Quantum Reserve deployed on chain 20777, with 19 phase-5 checks passing. That dated observation supersedes the earlier audit's "not deployed" statement. This edition has not re-queried remote bytecode or balances, and a recorded deployment does not establish full public-testnet acceptance. The sanitized deployment summary is sufficient here; account inventories and operational key material are not reproduced. [E20]

13.2 Authorization and cryptographic scope

QuantumAccountRegistry authenticates holder actions through ML-DSA-65 and recovery through SLH-DSA-SHAKE-192s. Signed contexts bind chain, genesis, registry/caller, account/key version, nonce and expiry. Relayers pay native gas; changing recipient, amount or an authorized fee invalidates the signed payload. Native verification is provided by pq-verify/v1 precompiles, with activation and gas schedule bound to the protocol profile. Constructor wiring to the correct verifier and registry remains a trust-critical deployment check. [E18]

ML-DSA and SLH-DSA are standardized in FIPS 204 and 205; ML-KEM in FIPS 203 is used for sealed backup functionality, not as a transaction-signature algorithm [7, 8, 9]. The repository reports algorithm-vector and negative-input tests. These do not constitute FIPS module validation, an independent audit or proof against all side channels. Base-chain ECDSA, validator Ed25519, RLPx transport and package-signing dependencies remain classical. QAI's restricted authorization path therefore does not make the whole PQWERAI system quantum-resistant. [E18]

13.3 Separate reserve roles

The native Quantum Reserve receives the 9% issuance share and permits bounded, role-authorized native outflows. Its contract hard ceiling is 5% of the day's captured base balance; the recorded deployment configures 1% per day. Governors and executors use separate threshold roles. Rule-based autonomy exists in source for margin top-ups, burns above a ceiling and scheduled subsidy, within a separate automatic sub-cap; the recorded deployment leaves those rules off. The

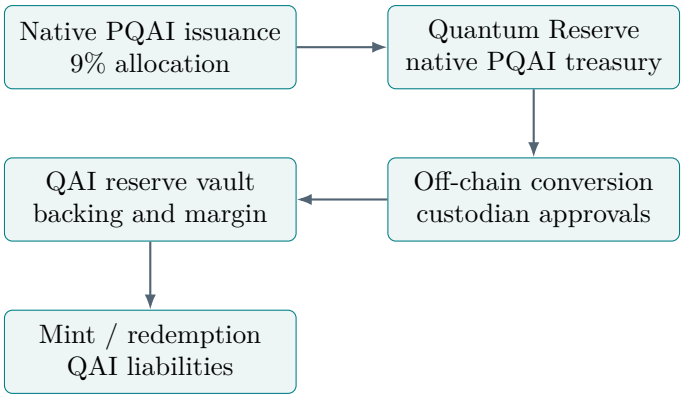


Figure 13.1: Distinct native treasury and QAI liability accounting. On testnet the reserve assets are simulated; the off-chain conversion boundary remains trusted.

supported automatic sub-cap is at most 0.25% per day. Contract presence is not evidence of activation. [E20]

The QAI reserve vault records off-chain simulated assets rather than holding bank money. It segregates customer backing from capital margin. A treasury nonce already spent on chain can be referenced once when custodians approve converted margin; that nonce proves a native outflow, not the price or completion of an off-chain currency conversion. Customer backing is debited on settled redemption and is not available for operating expenses or surplus release. [E19]

13.4 Valuation and the 110–130 percent band

For eligible asset a with customer amount C_a, margin M_a and haircut h_a in basis points, ledger valuation is

$$V_{\text{ledger}} = \sum_a \left\lfloor \frac{(C_a + M_a)(10{,}000 - h_a)}{10{,}000} \right\rfloor . \tag{13.1}$$

A fresh threshold attestation can impose a shortfall relative to ledger valuation; it cannot raise valuation above ledger value. Without a fresh attestation, usable valuation is zero. Let V be this conservatively adjusted value and L outstanding QAI redemption liabilities, both in six-decimal EUR-reference units. For $L > 0$, coverage is $r = V/L$. The display ratio is rounded down, but decisions use exact cross multiplication. With no liabilities, the code treats the band as surplus rather than performing division by zero. [E19]

Coverage state	Implemented decision
$r < 1.10$	Emergency: no mint; redemption paths remain available.
$1.10 \le r < 1.20$	Recovery: a mint may not worsen coverage.
$1.20 \le r \le 1.30$	Target: post-mint coverage must remain at least 1.20.
$r > 1.30$	Surplus: mint must retain target coverage; governor-approved margin release must leave at least 1.30.

Table 13.1: Implemented QAI policy. Thresholds are controls on the simulated ledger, not price guarantees.

The non-worsening recovery test for added reserves ΔV and liability ΔL is

$$(V + \Delta V)L \ge V(L + \Delta L). \tag{13.2}$$

Target admission instead tests $10{,}000(V + \Delta V) \ge 12{,}000(L + \Delta L)$. Checked arithmetic can reject excessive intermediate values; the equations do not imply support for unbounded monetary

inputs. The range 1.1–1.3 is thus already implemented in the reserve policy, while actual backing remains simulated. [E19]

13.5 Issuance, redemption and burns

Minting requires approved funding, threshold authorization, unused identifiers, per-approval/daily limits and coverage checks. Redemption starts with holder authorization and an escrow lock; threshold settlement burns the liability and debits backing. Rejection or an SLA reclaim returns tokens according to the lifecycle. The coverage band does not itself block redemption. A voluntary holder-authorized burn gives up the corresponding redemption right while backing remains segregated. Token accounting obeys totalSupply = totalMinted − totalBurned; pending redemption tokens remain part of supply until settlement. [E18][E19]

As an explicitly hypothetical stress scenario, $V = 120$, $L = 100$ gives 120% coverage. A 10% asset-value loss gives $V = 108$, so coverage becomes 108% and minting is blocked. A burn can reduce liabilities arithmetically but cannot restore a custodian's liquidity or create missing legal assets. A larger nominal ratio, restrictions on supply or a burn cannot guarantee appreciation or prevent depegging. There is no accepted automated market maker, deep secondary liquidity or price-defence mechanism established by these contracts.

13.6 Trust, custody and regulation

The deployment record says all eight signer keys and passwords remained on one PC. Consequently, the nominal 2-of-3 threshold did not establish independent custody. Attestors and custodians can misreport off-chain reserves; a hash of a report is not proof of an unencumbered asset. Correlated bank, custodian, conversion-agent and key failures, stale attestations, illiquid reserves and delayed redemption remain material risks. [E20]

If developed into an EU asset referencing one official currency, classification and obligations under MiCA require legal review. MiCA's e-money-token framework addresses authorized issuers and issuance/redemption at par, while safeguarding and related payment-service obligations require operational implementation [10, 11]. This observation is not a determination that QAI is compliant or a regulated offering whitepaper. The current simulated testnet cannot establish issuer authorization, real reserves or legally enforceable euro redemption.

Chapter 14

QAI economics, conditional outlook and monetary risk

Evidence status: New reproducible balance-sheet scenarios and algebraic sensitivities; no calibrated adoption, market-price or return forecast.

14.1 What can be forecast responsibly

QAI's implemented reference liability is one EUR-reference unit per token. The existing token and backing are simulated and valueless. A future real-money implementation would need an issuer, funding, independently verified reserves, liquidity, redemption operations and the applicable legal structure. No observed QAI market-price series, live reserve portfolio, audited cash-flow history or statistically defensible adoption model is established by the inspected evidence. A price target or probability of success would therefore be invented. [E18][E19][E20]

This chapter instead asks conditional questions: if a chosen amount of QAI were issued, how much eligible backing and loss-absorbing capital would be required? If reserves lost value, which controls would engage? If holders requested money back faster than assets could be converted to cash, what would remain payable? The answers are mathematical scenarios, not investment recommendations, money commitments or promises of returns. Currency amounts are mentioned because funding and liabilities matter, but all numerical inputs are explicitly assumed. [E28]

The contract's reference value, a market trading price and the money actually recoverable are different quantities. A holder does not own an automatic claim to the excess 20% target reserve margin. Demand growth can increase transaction activity and funding requirements without increasing the reference redemption amount. A stable-reference design should not be presented as a mechanism for guaranteed token-price appreciation.

14.2 Why par-funded growth needs additional capital

Let L be outstanding QAI liabilities, V fresh conservatively valued reserves, $t = 1.20$ the target, x newly minted liability, c eligible reserve value per newly funded unit and K added capital margin. The post-mint target requires

$$V + cx + K \geq t(L + x). \tag{14.1}$$

Consequently, a target-based planning requirement is

$$K_{\min} = \max\{0, t(L + x) - V - cx\}. \tag{14.2}$$

These are planning equations derived from the implemented target, not a new issuance rule. They assume fresh truthful valuation, correct authorizations and no other binding caps. Actual admission must still use integer arithmetic and all contract checks. [E19][E28]

At exactly 120% initial coverage with $c = 1$, each additional 1,000 QAI requires 1,000 of customer funding plus 200 of additional eligible margin to keep the target. Funding 1,000 at par alone

would dilute coverage. A zero-liability start is not exempt: issuing 1,000 against only 1,000 of reserves fails the target check. At a 2% haircut on the new funding asset, eligible funding is 980 and the target shortfall is 220. These amounts do not estimate actual issuance demand or promise that anyone will supply capital.

If $c < t$ and a pre-existing surplus is used instead, the target-based upper bound on new liability is

$$x \leq \max\left\{0, \frac{V - tL}{t - c}\right\}. \tag{14.3}$$

For assumed $V = 130{,}000$, $L = 100{,}000$ and par funding, headroom is 50,000 QAI, leaving 180,000 of reserves against 150,000 of liability. This does not mean 50,000 may actually be minted immediately: approvals, funding evidence, daily limits, freshness and other controls remain binding. Haircuts and micro-unit rounding can reduce admission at the boundary.

In the recovery band, the non-worsening test also matters. At 115% coverage, adding one unit of backing for one new liability worsens coverage and fails. A mint is not a source of free recapitalization; the reserve side must grow sufficiently relative to the new claim. If existing coverage is below 110%, the implemented check rejects minting even if that transaction proposes large additional funding. Restoring eligible reserve valuation is a separate operation. [E19][E28]

14.3 Loss absorption and insolvency sensitivity

Assume $L = 100{,}000$, $V = 120{,}000$, no intervening mints/redemptions and a fresh attestation reflecting a uniform value loss ℓ. Coverage becomes

$$r' = \frac{(1 - \ell)V}{L}. \tag{14.4}$$

Starting at 120%, a loss of more than $1 - 1.10/1.20 = 8.33\%$ of reserve value breaches the lower threshold. A loss above $1 - 1/1.20 = 16.67\%$ puts valued assets below nominal liabilities. These are balance-sheet sensitivities, not estimated probabilities of a loss, legal insolvency determinations or protections against fraud.

Assumed loss	Remaining value	Coverage	Capital to restore 120%
0%	120,000	120%	0
5%	114,000	114%	6,000
10%	108,000	108%	12,000
15%	102,000	102%	18,000
20%	96,000	96%	24,000
25%	90,000	90%	30,000

Table 14.1: Illustrative loss sensitivity in EUR-reference units with fixed 100,000 liability. Values are assumptions and arithmetic, not observed money or portfolio forecasts.

Counterparty default, FX exposure, custody restrictions, asset ineligibility and increased haircuts can all reduce usable valuation. An expired attestation sets usable value to zero in the controller; that is a conservative control state, not proof that the underlying bank assets have disappeared. Conversely, a fresh but false attestation does not create assets. A report should identify valuation date, legal ownership, encumbrances, settlement availability and concentration rather than publishing a ratio alone. [E19]

14.4 Twelve-month conditional paths

The accompanying standard-library model operates on exact six-decimal monetary integers. It transcribes only the economic controller's mint predicate and simple balance-sheet updates; it does not emulate the registry, approvals, limits, token contract, bank payments or full redemption lifecycle. All paths start from arbitrary $L_0 = 100{,}000$ and $V_0 = 120{,}000$. Each month is an illustrative step, not a launch-relative date. No likelihood ranking is assigned. [E28]

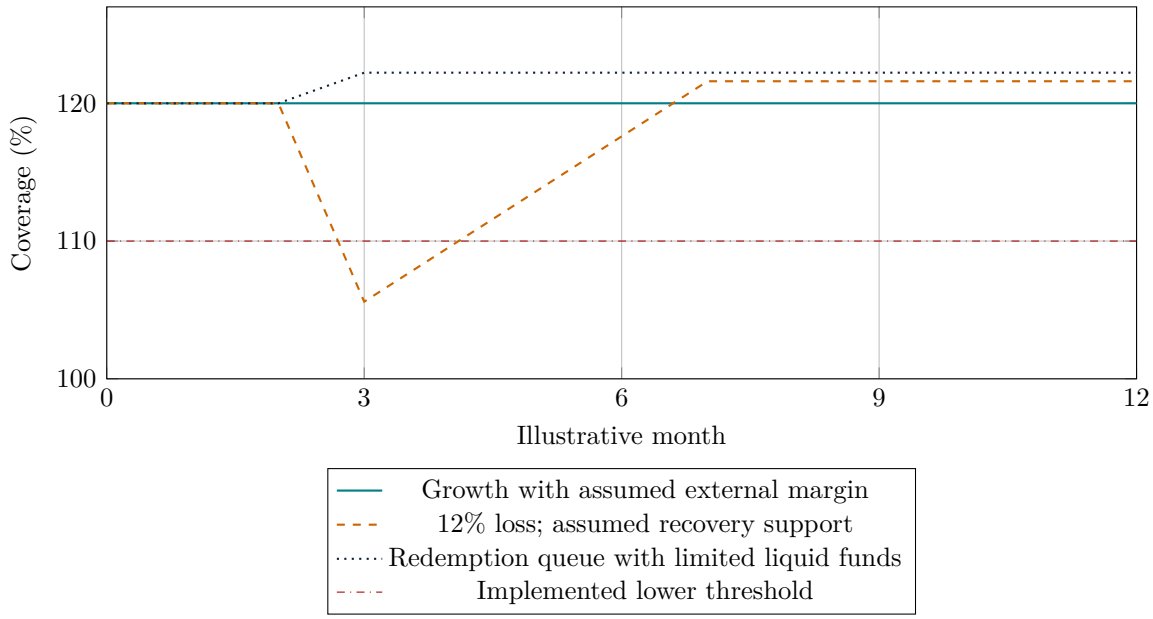


Figure 14.1: Deterministic illustrative paths, not calibrated forecasts. All start from assumed simulated liabilities of 100,000 and valuation of 120,000 EUR-reference units. The unfunded-growth path remains at 120% because requested mints are rejected. A higher ratio does not imply sufficient immediate liquidity.

Assumed path	Month-12 liability	Month-12 value	Additional result
Funded growth	220,000	264,000	120,000 new liability; 24,000 assumed external capital; coverage 120%.
Unfunded demand	100,000	120,000	120,000 requested new liability is rejected; no automatic issuance.
Loss and support	100,000	121,600	12% loss in month 3; 16,000 assumed external support over months 4–7.
Liquidity run	90,000	110,000	30,000 redemption request, 10,000 liquid payment capacity; 20,000 still pending.

Table 14.2: Conditional scenario outputs, not forecasts of actual adoption, funding or redemption performance. Amounts are simulated EUR-reference units.

The funded-growth path assumes monthly demand of 10,000 QAI and 2,000 of externally supplied margin before each mint. The unfunded path requests the same demand without margin; every mint is rejected rather than silently lowering the required ratio. Requested but rejected mints are unmet hypothetical demand, not circulating supply or a redemption claim.

The loss path applies a single 12% reserve-value reduction, taking coverage to 105.6%. Assumed support of 4,000 per month in months 4–7 raises coverage through 109.6%, 113.6%, 117.6% and 121.6%. Such contributions are scenario inputs, not automatic reserve revenue or a promised bailout. Their availability and legal eligibility would need evidence.

The liquidity path has 10,000 immediately available against 30,000 of requested redemptions. Only 10,000 is assumed payable. After that payment, $V = 110{,}000$ and $L = 90{,}000$, so coverage rises to about 122.22%, yet 20,000 remains pending and immediate liquid funds are exhausted. Pending tokens remain liabilities until settlement. This distinction is central: a healthy-looking coverage ratio cannot establish timely access to money. The simplified model treats payment and liability removal together for arithmetic; the actual on-chain settlement event and off-chain money transfer are separate trusted operations. [E19][E28]

14.5 Liquidity, redemption and market-price deviations

If immediately liquid eligible funds are A and redemption requests are D, the instantaneous funding shortfall is $\max(0, D-A)$. A mature service would need asset-maturity ladders, bank/payment access, custody-specific liquidity and stress-tested redemption plans. Contract caps and reserve haircuts are not substitutes for those operations. ECB analysis identifies parity deviations, runs and reserve fire-sales as risks of stablecoin arrangements [12]. This is general risk context, not a claim that QAI already holds such portfolios.

With a future real issuer, a nominal one-euro redemption claim can still trade below its reference if holders expect delay, fees, illiquidity or nonpayment. It can trade above reference if access to issuance or markets is constrained. There is no verified QAI market from which to estimate either magnitude. A voluntary burn reduces token liabilities while the holder gives up that claim; it does not entitle remaining holders to a guaranteed gain. A native PQAI treasury burn likewise does not mechanically create euro assets in the QAI vault. [E19][E20]

14.6 How the 9 percent native treasury relates to money

The treasury receives native units, not euros. For a reachable gross epoch issuance I in the selected Core, the reserve allocation is exactly $0.09I$; its settlement timing and whole-gwei quantization are explained in Chapter 11. Hypothetical convertible reserve value would additionally depend on the fraction authorized for conversion, native market price, executable market depth, transaction costs, custody acceptance and applicable caps. There is no verified future PQAI/EUR price or committed reserve-conversion pipeline in this edition. [E16][E19][E20]

A dimensional planning expression is $K \leq aRp(1-c)$, where R is native treasury units available, a the authorized convertible fraction, p an executable EUR-per-PQAI price and c proportional conversion costs. This is an upper-bound scenario requiring external inputs, not a treasury income forecast. If p or available liquidity collapses, an issuance percentage alone cannot maintain QAI coverage. A treasury outflow nonce proves that native units moved; custodians still attest to the actual off-chain conversion.

14.7 Costs and revenues without promised earnings

QAI actions consume native gas. At an assumed effective fee of 1 gwei, the historical 712,317-gas mint costs 0.000712317 PQAI; at 10 or 100 gwei, it costs 0.00712317 or 0.0712317 PQAI. These are fee sensitivities, not live quotes, euro prices or the total lifecycle cost. Deployment, attestations, holder signatures, redemption, relayers and off-chain operations add distinct costs. [E19][E26]

An AI provider's gross quoted revenue is also not profit. An illustrative 10,000 delivered-output-token job at 2 gwei per token charges 0.00002 PQAI before gas and operating costs, provided

funding, acceptance and settlement succeed. Euro proceeds depend on an unobserved exchange rate and executable conversion. For assumed idle power 50 W, active power 150 W, 720 hours per month, 25% active utilization and electricity at EUR 0.25/kWh, energy cost is EUR 13.50; adding an assumed EUR 20 of other costs gives EUR 33.50 before gas, taxes and any additional costs. These are arbitrary explanatory inputs, not measured provider expenses or an earning opportunity. [E12][E17][E28]

No token-holder interest, minimum profit, reserve yield, principal protection or price appreciation is promised. Economic feasibility must be demonstrated using actual accepted demand, billing, energy, hardware, staffing, custody, liquidity and compliance costs. Any future issuer's reserve income and any operator's service revenue would have to be reported separately from token-holder rights and protocol issuance.

14.8 What would permit a stronger outlook

A quantitative outlook needs historical accepted mint/redemption volumes, actual supply, fresh independently checked reserve composition, liquid assets by horizon, customer concentration, settlement delays, service revenues/costs and executable market liquidity. A model should disclose calibration windows, forecast errors, adverse scenarios and uncertainty without fitting probabilities to invented data. Until those records exist, the conditional paths in this chapter are more defensible than a nominal price target. Reproduction inputs and outputs are supplied in `evidence/scenario_assumptions.json` and `evidence/qai_scenarios.json`.

Chapter 15

Security and threat model

15.1 Protected assets and adversaries

The protected assets are canonical ledger state, finalized history, validator collateral, wallet authority, customer budgets, delivered AI work, model identity, reserve-accounting integrity and recoverable software distribution. Adversaries may control peers, miners, validators, providers, gateways, models, package mirrors or privileged roles; they may exploit dependency defects or compromise a shared operator host. The model includes partitions, crashes, stale backups and malicious data. It does not assume honest infrastructure merely because it uses signatures. [E03][E04][E12][E18][E23]

The trusted computing base includes the selected client, hash helper, cryptographic libraries, VM runtime, compiler/build inputs, OS and storage. Service clients additionally trust an approved Core RPC, registry bootstrap policy and relevant arbitration. QAI reserve users depend on verifier wiring, threshold custody, truthful custodians/attestors and off-chain redemption operations. These assumptions differ from the assumptions for block agreement.

Table 15.1: Attack surfaces, implemented controls and residual risks.

Surface	Controls with source evidence	Residual risk / evidence gap
Work majority / double spending	Work validation, cumulative difficulty, finalized-ancestor restrictions (E03–E04)	Concentrated hash supply, censorship and pre-finality reorganizations; no economic attack-cost proof.
Validator collusion / stale signer	Strict weighted quorum, domain checks, journal locks, slashing evidence (E04–E05)	Dynamic-set safety, long-range recovery and correlated custody require independent assurance.
Eclipse / network partition	Peer/discovery controls, separate AI transport, bounded messages (E06)	Common seed, subnet, relay or hosting failures; finality may stall.
State inconsistency / VM abuse	Serial oracle, root differentials, bounded execution paths (E07–E09)	Finite tests; selected-Core WASM resource-cost findings remain open.
Reentrancy / contract privilege	Role thresholds, explicit caller boundaries, guarded withdrawals and lifecycle checks (E17–E20)	Constructor miswiring, malicious verifiers, application bugs and governance compromise.
Provider fraud / wrong billing	Customer chunks, pinned tokenizer, signed receipts, funded cap, durable journal (E11–E12)	Quality/model-use proof incomplete; arbiter and approved RPC remain trusted.
Model poisoning / data leakage	Manifest hashes, approval policy, permissions and promotion controls (E13–E14)	Correct bytes may be malicious; providers can observe plaintext inputs.

Surface	Controls with source evidence	Residual risk / evidence gap
Reserve misstatement / depeg	Segregated ledger, haircuts, freshness, coverage and mint limits (E19)	Simulated assets; attestor/custodian fraud, liquidity and legal enforceability.
Key / dependency compromise	Keystore protection, threshold roles, package pins and TUF (E18–E23)	One-PC threshold custody; OS/compiler/runtime risk and classical signatures remain.

15.2 Known selected-Core WASM findings

The 14 November author investigation reports excessive locals/engine-stack resource use, host-copy allocation and cold/failed compilation cost in Core `f2eace6bc69fca48b290142b8232bd1ced6a690b`. It measured runtime-package heap growth and execution costs locally; it did not deliberately exhaust network validators. The report's network-outage consequence is an extrapolation from those measurements. A local candidate at `b6bc0ab2e` addresses selected memory/copy cases, while cold compilation and failed-deploy cost remain open. [E08]

The whitepaper does not treat the candidate as deployed. Locals instrumentation changes require an approved activation and a decision about already deployed code. A patch can preserve high-level language semantics while changing gas or admissibility, which is still consensus-sensitive. Publication of this draft does not close the existing findings or authorize a live attack, release or fork.

15.3 Contract and service security

Reentrancy guards, pull-credit withdrawals, role/domain separation, bounded amounts, nonce uniqueness and expiry reduce specific contract risks. They do not prove the absence of arithmetic, authorization or lifecycle bugs. QAI uses explicit contract-only mint/burn boundaries and PQ authorization for holders; constructor substitution and malicious verifier wiring remain critical. Escrow arbitration remains a trusted resolution mechanism rather than a useful-work proof. [E12][E17][E18][E19]

Provider signatures make a quote or usage claim attributable. They do not prove the model used, physical GPU capacity, semantic correctness or independent ownership. Customer metering can reject inflated delivery counts but may still depend on correct tokenization and output interpretation. Training requires independent evaluation, dataset consent, anti-poisoning controls and rollback; none is implied by a successful model download. [E11][E12][E13][E14]

15.4 Keys, cryptography and supply chain

Native transactions and validator certificates retain classical signatures. A sufficiently capable quantum adversary threatens those public-key systems even if QAI's holder operations use standardized post-quantum signatures. Hash algorithms, a product name containing "quantum", or a classical optimization algorithm do not change this boundary. PQ migration requires addressing consensus, transport, account recovery, package authentication and deployment trust together. [E18][E21]

Encrypted keystores reduce at-rest exposure, but an unlocked compromised host can defeat them. Filesystem overwrite is not reliable erasure on every SSD or snapshotting system. Signer backups must preserve anti-equivocation state; threshold members need independent devices and recovery procedures. The deployment's one-PC custody remains a material unresolved risk. [E05][E20]

TUF metadata, hash verification and content addressing address selected substitution, rollback and freeze threats [13]. Their effectiveness depends on trusted roots, freshness checks, correct clients and retained artifacts. Two matching local builds do not prove source safety; an SBOM lists dependencies without clearing their licenses or vulnerabilities. Runtime/compiler pins and notices must accompany distributable packages. [E23][E24]

15.5 Incident response and assurance

An incident process should preserve evidence, identify affected source/profile and contracts, isolate the failing service, protect signing journals, reconcile canonical state and communicate recovery instructions. Consensus changes require versioned activation; off-chain service disablement must not rewrite ledger rules. Reserve mint/outflow pause mechanisms are scoped controls and should not be described as a universal ability to reverse transactions. Release rollback is restricted when the database or consensus state has advanced incompatibly. [E05][E19][E20][E24]

Verified historical tests include signature/domain negatives, strict-quorum and duplicate-vote cases, reorg/restart execution comparisons, usage inflation/replay checks and hostile-verifier cases. Their exact scopes and source identities are linked in the evidence ledger. New checks for this edition validate documentation and selected local component behavior; they do not replace independent consensus, contract or infrastructure assurance.

Chapter 16

Governance, upgrades and authority

16.1 Observed governance

The inspected process is owner-led, with invariant, requirement, decision, gate, finding and release registers. It does not establish a token-voting DAO or community custody. Policy approval, implementation, release acceptance and deployment authority are separate. [E24]

The baseline decision register retains chain 20777 and its state, superseding older reset language. A protocol change needs an approved versioned future-height activation. The later CR-0051 record supplies such a coordinated activation for IPFS issuance at block 34785; it preserves genesis. Ordinary package updates, an IPNS pointer or a wallet UI change cannot legitimately change issuance, genesis or finality rules. Compatibility checks must bind network identity and supported activation schedule. [E02][E23][E24][E31]

16.2 Treasury and contract governance

Founder issuance has a genesis-pinned destination. Quantum Reserve governors/executors and QAI reserve roles have explicit thresholds, versions and emergency controls. These are contract authorities; daily caps do not prevent malicious policy within permitted limits. [E16][E19][E20]

The recorded automatic reserve rules are disabled. The classical advisory Quantum Layer cannot enable them or sign approvals. Owner-signed agent policy narrows capabilities; model submissions follow admission policy. Approval does not certify model behavior. [E13][E20][E21]

16.3 Upgrades and emergency procedures

Consensus modules must be released as a compatible Core bundle. Contract deployments retain their ABI, immutable policy and job terms unless an explicit migration is implemented; a new escrow version cannot retroactively change funded legacy jobs. Wallet, explorer and runtime upgrades need versioned interfaces and rollback plans, while rollback across consensus/state changes may be impossible without a coordinated procedure. [E01][E17][E23]

Release acceptance requires complete gates, exact-source tests, independent assurance, licensed reproducible packages and owner go/no-go. A particular deployment's permission does not waive these requirements. Mainnet identity and launch economics remain unselected. [E24]

16.4 Proposed decentralization of authority

Independent custody, release review, reserve reporting and operator acceptance remain requirements. Any future voting design needs an electorate, quorum, delegation, timelocks, emergency scope and capture analysis; no such parameters are invented here.

Chapter 17

Performance and experimental evaluation

17.1 What the evidence measures

No sustained network-capacity measurement is established for 20777. Edition 0.4.0 adds bounded one-host maximum-throughput series in section 17.3. Historical RC1 tests, PASE microbenchmarks and a one-host Colibri probe measure different systems and stages. They are retained as bounded observations rather than merged into a headline TPS figure. New document-build and unit-test durations do not measure chain performance. [E09][E15][E22]

Evidence	Reported observation	Interpretation
Historical RC1 scenario F	210 transactions including setup, 67 chain-seconds; 3.13 tx/s finalized	Paced workload on six local nodes; not capacity or a selected-20777 benchmark.
Historical RC1 block timing	Mean 9.14 s, median 4 s over 924 blocks	Contended one-PC, mostly one mining thread; not a production block-time guarantee.
Colibri run 2	Median 5.94 tok/s over three 64-token probes	One CUDA host/model/configuration; see table 10.1.
General PASE	Source and differential tests available	No accepted sustained TPS gain inferred.
10,000 TPS	Architecture target, unproven	A future objective, not measured or simulated achievement.

Table 17.1: Measurements, limits and targets are kept separate.

The RC1 environment was a Ryzen 9 3900X with 24 threads, WSL2 Ubuntu 24.04 and 7.7 GiB available RAM, shared with other local CPU workloads. The report identifies reproducible-build and historical client revisions separately; those binaries are not the selected 20777 Core. Its 80/20 reward observation belongs to that earlier profile. The reported 2+2 signer partition stalled finality and recovered after healing; it is not a dynamic-set WAN resilience proof. Resource soak was explicitly not run. [E22]

17.2 Gas-limited envelope

For block gas limit G_b, transaction gas g and mean block interval t_b, a basic homogeneous workload ceiling is

$$\mathrm{TPS} \leq \frac{\lfloor G_b/g \rfloor}{t_b}. \tag{17.1}$$

Assuming only 21,000-gas plain transfers, no additional gas consumers, full blocks and $G_b = 15{,}000{,}000$, at most 714 such transfers fit per block. At an assumed 8–10 seconds per block, the envelope is 71.4–89.25 tx/s. These are derived capacity bounds under the stated assumptions, not observations or finality rates. Contract calls and carrier transactions lower the count; independent CPU execution improvements cannot bypass a fixed gas budget. [E02][E22]

17.3 Maximum-throughput series on a chain-20777 harness

Between 15 and 19 November 2026 the committed ramp tool `benchmarks/tps/max_tps.py` offered pre-signed load in 60-second stages to a disposable six-node harness with 20777 genesis rules. The harness had three serial and three PASE nodes, four validator signers and one GhostRider miner, all on one 24-thread desktop. It measured inclusion from block timestamps and transaction counts. The first series used the harness default base difficulty 4 and produced blocks about every 2 seconds; the later series used the live base difficulty 64, which gave 7–10-second blocks close to the live chain. [E35]

Run (difficulty 64 unless noted)	Gas limit	Peak included	Ceiling	Note
Native transfers, difficulty 4 (15 Nov)	15M	391 TPS	—	2 s blocks; not the live block rate.
Native transfers (15 Nov)	15M	46 TPS	—	Saturated at 100 tx/s offered; blocks slowed under overload.
Native transfers (15 Nov)	30M	65 TPS	—	Saturated at 200 tx/s offered.
Native transfers (19 Nov)	15M	95 TPS	74.5 TPS	Peak in one stage with 6.3 s blocks; drained 100 tx/s.
Native transfers (19 Nov)	30M	169 TPS	192.4 TPS	Drained every stage up to 400 tx/s offered.
Native transfers (19 Nov, partial)	60M	141 TPS	—	Two stages only; the run stopped before a final report.
MultiSend batches of 100 (19 Nov)	15M	200 payments/s	189.4 /s	10,334 gas per payment; saturated at 600/s.

Table 17.2: One-host maximum-throughput observations. *Ceiling* is equation (17.1) evaluated at the run's mean block interval.

Three points follow. First, throughput tracked equation (17.1): blocks were full from modest offered loads, so the gas budget and the block interval, not the transaction pool or block import, set the ceiling. Followers stayed within about one second of the miner even with 30M-gas blocks. Second, batching roughly halved gas per payment and carried about four times as many payments per second as single transfers at the same gas limit, without a protocol change. Third, the two 15 November overload runs show that a transaction flood slowed block production on a shared host, from about 8 seconds to 23–37 seconds per block. That may not hold when the miner runs on its own machine, but it must be retested on separate hosts. The repeat runs on 19 November did not reach saturation. The live chain keeps the 15M gas limit by owner decision; without a fork, miners can move it by 1/1024 per block toward their configured ceiling. [E35]

Limitations. One machine, loopback networking, one miner and short stages. These runs do not measure WAN propagation, independent operators, hour-long load, state growth, finality delay under load, mixed contract workloads or any PASE or PATE speed-up. They are not a network capacity claim and do not approach the 10,000 TPS architecture target.

17.4 Reproducible evaluation protocol

A capacity evaluation should pin repository, Core, helper, PASE module, compiler/runtime, contract artifacts, genesis and node configuration; verify clean source and retain hashes. Record OS/kernel, CPU model, cores, RAM limits, GPU/driver, storage, process limits and concurrent load. Record each node's role, peer graph, network delay/loss, mining threads, PASE workers and effective validator weights. Use disposable chain state and keep real keys outside results. [E22]

Run independent transfers, shared senders, independent and contended contract storage, reverts, creations, token transfers, QAI operations and cross-VM calls. Compare serial and PASE with identical inputs, gas rules and hash-verified outputs. Include warm/cold state and model caches, saturation, overload/recovery, increasing conflicts and randomized prediction misses. Report the number of trials and all failed attempts; avoid choosing only the fastest run.

Measure submission-to-inclusion and submission-to-verified-finality separately. Specify clock sources, polling overhead and the numerator of every rate. Report latency quantiles and confidence intervals, execution/root/persistence costs, fallback counts, CPU time, resident memory, helper memory, storage growth and per-interface bandwidth. A model benchmark must identify tokenizer, prompt/context, output length, concurrency, queueing, time to first token and whether token rate includes prefill. Cancellation must verify engine stoppage, not just socket closure.

17.5 Available launchers and missing acceptance

`benchmarks/tps/run_private.py` supplies a source-pinned private-client launcher, while `benchmarks/tps/report.py` separates stage metrics. Its documented 128-transfer trials and alternating request modes are an RPC-ingestion experiment with head-convergence checks, not BFT finality. It requires private source and helper artifacts and is not a zero-dependency public reproduction kit. PASE's Go benchmarks isolate execution costs and deliberately exclude network, sealing and persistence. [E09][E22]

Independent-host sustained operation, bandwidth/resource soak, finality tail behavior, adverse routing, split inference and recovery under realistic failures remain missing. Chapter 14 adds reproducible deterministic economic scenarios, while Chapter 18 reanalyzes historical statistics; neither is a new runtime benchmark or calibrated market forecast. Publication of a benchmark harness alone is not a benchmark result.

Chapter 18

Feature statistics and evidence coverage

18.1 Statistical vocabulary and complete coverage

Every registered feature is included in the companion catalogue: 76 entries at the 0.4.0 cutoff (74 at the 0.3.0 baseline plus the two internal-agent features QL-007 and QL-008) covering the Core, networking, wallet/tooling, AI services, distribution and QAI. The catalogue records the registry's author status, named test hooks and acceptance gaps. Its denominator is feature entries, not deployed instances, independent operators or successful experiments. It therefore answers "what is documented and still missing?" without turning project bookkeeping into a security or availability statistic. [E29]

Registered class	Entries	Share	Meaning
design only	2	2.6%	Registry classification only.
implemented untested	7	9.2%	Registry classification only.
locally verified	35	46.1%	Registry classification only.
partial	32	42.1%	Registry classification only.

Table 18.1: Status distribution across all registered features; not a success rate.

Some recorded labels predate later implementation and deployment evidence. Appendix C explicitly reconciles known stale cases, while the CSV retains both original text and author annotations. The percentages above describe the register as recorded; they must not be interpreted as fresh acceptance percentages. [E02][E09][E20][E29]

For a runtime statistic, the denominator and observation window matter. Throughput is completed work divided by elapsed time at a specified stage. Availability needs service-success observations over a defined interval, excluding or explicitly including maintenance. A latency percentile needs the actual latency distribution and sample count. Reliability across independent operators needs evidence of independence. No global uptime, market-share, provider-profit or security probability is supplied where those observations are absent.

18.2 Historical blockchain burst observations

The historical capacity collection contains two-node local test-builder experiments at difficulty 2. The 512-transfer profile uses a 15-million gas limit, while the 2,048 and 8,192 profiles use 90 million and 350 million gas respectively. Header timestamps advance by 10 seconds without enforcing that cadence in wall time. More RPC streams change submission concurrency; they are not PASE workers. These settings differ materially from normal retained-chain mining and must accompany every quoted result. [E27]

For the 512-transfer profile, three trials per stream configuration give 1,536 canonical transactions per row. Rates of about 1,291–1,944 local end-to-end tx/s divide those transactions by the short measured execution/import interval. They do not imply that a 15-million-gas chain can sustain those rates at ten-second block intervals: the gas envelope at that cadence is 71.4 plain

Tx / trial	Gas limit (M)	Streams	Trials	Local tx/s
512	15	1	3	1,290.63
512	15	2	3	1,798.02
512	15	4	3	1,632.33
512	15	8	3	1,944.25
2,048	90	1	3	2,101.87
2,048	90	2	3	2,899.74
2,048	90	4	3	3,068.09
2,048	90	8	3	2,961.20
8,192	350	1	1	239.28
8,192	350	2	1	236.07
8,192	350	4	1	237.96
8,192	350	8	1	237.04
8,192*	350	4	3	237.96

Table 18.2: Historical test-builder burst rates, difficulty 2, two local nodes. *Separate coalesced observation. No row measures sustained or finalized TPS.

transfers/s. Both statements can be correct because the numerators and time boundaries describe different experiments. [E27]

At 2,048 transfers with the raised gas limit, four streams report about 3,068 local end-to-end tx/s, compared with about 2,102 for one stream. This is a within-fixture RPC-concurrency comparison, not a general scalability law. At 8,192 transfers the end-to-end observations fall to about 236–239 tx/s while accepted submission bursts remain several thousand tx/s. In the separate three-trial, four-stream coalesced observation, peer waiting accounts for about 95.1% of elapsed time. The source calls these local burst observations and leaves sustained/finalized TPS null; this edition preserves that classification. No 10,000-TPS success is established.

The descriptive extractor checks that each reported end-to-end rate equals canonical transactions divided by recorded elapsed time. It retains source hashes and exports only allowlisted metrics. A consistency check on a historical JSON file does not independently reproduce the original network run or verify all its correctness assertions. Current-source WAN, mixed-contract, sustained-load and finality measurements remain absent. [E27]

18.3 PASE execution statistics and their boundary

The older independent-transfer harness reports five-run medians. At 512 transactions, serial execution took 27.686 ms/block and versioned parallel execution 18.143 ms/block, corresponding to a fixture-specific ratio of about 1.53. At 1,024 transactions the medians were 55.497 and 31.877 ms/block, about 1.74. Allocated bytes fell from 18.52 to 13.30 MB/op and from 37.03 to 26.50 MB/op respectively, using decimal MB for those conversions. At 128 transactions the versioned entry point stayed on its serial-threshold path. [E22]

Those results do not contradict the absence of a current sustained PASE capacity claim. They describe a historical transfer-only in-memory harness with its own threshold, worker reuse and source pins; the current general executor has additional state-recording, conflict and VM behavior. The benchmark excludes RPC, block timing, persistence and BFT finality. A quoted 1.74 ratio must not be promoted to “74% faster blockchain” or applied to contended contracts, QAI verification or WASM compilation.

Useful current PASE statistics would include eligible-block fraction, speculative commits, serial re-executions, read/write-set sizes, copied-state bytes, worker occupancy, conflict/backoff rates and

time to commit roots. The code has statistics and differential hooks, but a unified current-source production distribution of those metrics is not established by the inspected reports. [E09]

18.4 Inference sample statistics

Colibri's two local runs each contain three 64-output-token probes with 20 runtime-reported prompt tokens. Recomputing 64/elapsed gives a median of approximately 5.06 tok/s in run 1 and 5.94 in run 2. Run 2's three rates span approximately 5.31–6.57 tok/s. The 17.4% difference between the two medians is an observed session-to-session difference; it is not an improvement attributed to a controlled software change. The same host, caches and ambient load can correlate these observations. [E26]

For equal-length probes, an arithmetic mean of per-probe rates and total tokens divided by total elapsed time answer different questions. The latter weights elapsed time and is the harmonic mean of the rates when all counts are equal. This edition exports the underlying counts, elapsed-time-derived rates, median, range, mean and sample standard deviation. With only three dependent local probes per session, it does not publish a confidence interval for population performance, an uptime estimate or an independent replication claim.

The counts are explicitly runtime-reported. They are not an independently audited tokenizer measurement, and probe elapsed time is not the same as a decode-only rate or a customer-metered settlement statistic. Time to first token, generation-only rate, total request latency, queueing and throughput under concurrency should be published separately. [E12][E26]

Historical MeshLLM profile	Probes	Load s	Median latency s	Tokens/s
CPU, context 1,024	5	17.41	1.711	54.00
GPU planning cap 1 GB	5	14.78	0.648	128.14
GPU cap 4 GB, context 4,096	5	14.44	0.663	127.49

Table 18.3: Historical Qwen2.5-Coder-0.5B Q4_K_M measurements on one host, not comparable model quality or capacity to the 30B-A3B Colibri model.

The nominal 1 GB GPU-cap profile used about 1,104 MiB of total GPU memory after jobs; the 4 GB profile used about 1,153 MiB. The first is a planning label, not proof that a physical 1 GB card can run the workload. Process peak RSS was about 635–638 MiB for those GPU profiles, excluding a complete accounting of all host/driver resources. The CPU measurement's GPU counter drift must not be interpreted as negative device memory. The source reports successful unload and released model VRAM, not a cross-platform memory-isolation proof. [E26]

18.5 QAI gas, reserve controls and economic statistics

These gas values belong to the dated reserve-accounting test at private Core `fe3ed8c42`, not a newly benchmarked current deployment. Input/signature shape and runtime rules can change the cost. A 15-million-gas block can fit at most 21 copies of a 712,317-gas operation by arithmetic alone, before any other gas use; admission, fees, roles, nonces and state dependencies still apply. This is not an achievable mint-throughput measurement. [E19][E26]

Reserve monitoring should publish outstanding and pending liabilities, independently valued eligible assets, customer/margin separation, attestation age, liquid coverage by horizon, mint/refusal counts, redemption latency/backlog, authorized treasury outflows and key/custodian concentration. The deployment report provides configuration and a dated verification result; it does not

Historical reserve-stack operation	Gas	PQAI at assumed 1 gwei
Mint	712,317	0.000712317
Attestation	451,866	0.000451866
Margin credit	394,128	0.000394128
Redemption settlement	429,199	0.000429199
Surplus release	386,181	0.000386181
Voluntary burn	221,596	0.000221596
Vault deployment	3,117,822	0.003117822

Table 18.4: Historical local QAI operation costs and derived fee sensitivity. These are not live quotes, full lifecycle costs or euro prices.

provide a current audited time series for these metrics. None is filled in with an invented current balance. [E20]

18.6 Missing statistics by feature family

For PoW and finality, independent hash/stake control, certificate-inclusion delay, reorganization depth and recovery latency remain important missing distributions. For P2P, peer diversity, reconnect success under different NATs, relay dependence and sustained bandwidth are needed. For EVM/WASM, gas-versus-CPU/memory tails and cold compilation require particular attention because known findings remain open. [E03][E04][E06][E08]

For AI routing/pools/training, publish accepted quote-to-completion latency, rejection causes, actual resource contribution, stage RTT/bandwidth, disputes/refunds and evaluated model-update quality. For wallets/explorers, publish compatibility coverage, synchronization lag, failed transaction handling and installation/recovery results by platform. For release distribution, publish independent failure-domain replicas, restore success and update rollback results. Absence of a comparable number is reported as missing evidence, not zero capability or 100% success. [E11][E14][E23][E25][E29]

Test counts also need context. The previous edition's local job-A run executed 574 Python tests with five skips, while focused suites overlap that broader collection. Adding those counts does not count distinct features or independent trials. No inference such as "99% secure" follows from passing unit tests. The verification record distinguishes retained implementation-test evidence from the new numerical/document checks for this edition.

Chapter 19

Operational walkthroughs: how the pieces interact

19.1 A payment from wallet to finalized history

Consider a native PQAI transfer. The wallet first selects the authenticated chain/genesis and derives the sender's nonce from a trusted view of state. It constructs and signs a transaction with amount, destination and gas/fee limits. The node validates its envelope before admitting it to a pool. Admission is not inclusion and does not yet change the recipient's canonical balance. The miner selects an order, executes transactions under the active VM rules and searches for a GhostRider seal. [E02][E03][E07]

A receiving full node verifies work, profile, ancestry and the ordered state transition. The resulting receipt distinguishes successful execution from a valid transaction whose contract call reverted. The state and receipt roots commit the execution result. A greater-work compatible branch may replace an unfinalized inclusion, so an application must follow the canonical receipt rather than treating the first explorer event as permanent. [E04][E07]

At checkpoint boundaries, Smartnodes derive the expected epoch set and sign a domain-bound source/target link using their persistent anti-equivocation journals. A certificate becomes effective through canonical block execution. Justification of the immediate successor checkpoint can finalize the source. A four-member equal-weight set needs three votes under the strict two-thirds rule; two votes are insufficient. This example counts equal weights, not arbitrary validator identities. No number of elapsed seconds substitutes for a valid quorum or permits a signer to discard its lock. [E04][E05]

If mean block time were ten seconds, 64–96 blocks would correspond arithmetically to roughly 10.7–16 minutes. That converts a historical reported depth to an illustrative time range; it is not a current finality SLA. Work variance, missed duties, certificate censorship, membership transitions and partitions can extend or stop progress. The standard RPC finalized tag and G4's own status must be checked for the selected client rather than assumed interchangeable.

19.2 A PASE conflict example

Suppose ordered transactions A and B both update the same contract counter. Both workers may initially read value 10 from copies of the batch pre-state and speculate writing 11. After A commits, its write dirties that storage key. B's read now conflicts and must be re-executed on the committed value 11, producing 12 under the serial semantics. Merging both stale writes as 11 would be incorrect; predicting that the transactions were independent cannot authorize that result. [E09]

With independent storage keys, both candidates may pass their full merge checks and commit in the preset order without re-execution. Nonces, balance reads, fee credits, account resets, logs and gas still matter: separate contract addresses alone do not prove independence. A reverted call is not automatically a scheduler error. The goal is the same observable result as serial execution, including the valid revert and its protocol-defined gas/nonce effects.

When conflicts become frequent, the adaptive scheduler reduces batch size or enters serial backoff. A busy shared contract can therefore remove the apparent parallelism even on a many-core machine. Predictor accuracy is useful only when it lowers total work without changing the acceptance invariant. Historical independent-transfer speedups do not establish the behavior of this counter example or of a mixed cross-VM block.

19.3 An AI request from consent to payment

The application resolves an exact model version, approved manifest and tokenizer identity. It obtains eligible providers from chain-bound records and applies customer resource, budget, consent and diversity rules. Signed availability/speed claims can help rank candidates, but the customer must not treat them as proof of physical resources. A local runtime probe verifies a backend on that host; it does not verify every remote provider. [E10][E11][E13]

The customer obtains a provider-signed quote binding job, model, token limit, price and service window, then funds the relevant escrow with existing native units. Wallet authorization and the off-chain budget journal reserve the allowed amount. A reservation prevents concurrent requests from each treating the same available budget as spendable. Neither a journal credit in a synthetic prototype nor an unsigned receipt can substitute for actual chain funding. [E12][E17][E30]

During delivery, the customer observes monotonically sequenced chunks and the output it received. Duplicate chunks do not create new accepted work; gaps, excess output or mismatching model/tokenizer identities are rejected. Provider acknowledgments authenticate usage claims and a final digest, while the customer bounds acceptance by delivered output and the quote. A plausible answer can still be wrong or produced by a substituted process: accounting integrity and model quality remain separate.

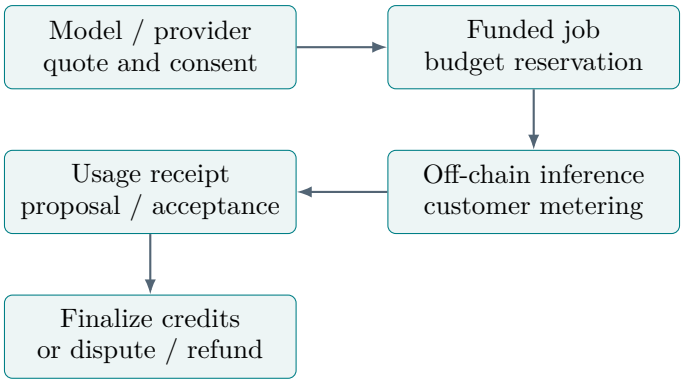


Figure 19.1: AI service lifecycle. On-chain funding and final credits remain separate from off-chain inference, observed delivery and semantic output quality.

In ServiceEscrow v3, a funded job can be proposed by the assigned provider with a bounded charge. The payer can accept a proposal, or an authorized party can open a timely dispute. After acceptance/challenge handling, finalization creates withdrawal credits for the provider/payees and refunds the uncharged budget to the payer's credit. The contract does not magically transfer money to every participant at the instant the last token arrives; withdrawals and canonical confirmation are additional operations. [E17][E30]

For an assumed 1,000-output-token quote at 2 gwei per token, full funding is 2,000 gwei. If 600 output tokens are accepted, the charge is 1,200 gwei and the unused 800 gwei becomes payer credit on settlement. With no configured non-provider payees, the serving provider's credit is 1,200 gwei. With configured payees, floor-rounded shares and the provider remainder apply.

These are explanatory unit amounts, not a live price, demand estimate or promised earning.

The arbiter can resolve a disputed charge only within its contract bounds. If the relevant funded/proposed/disputed job expires without a qualifying transition, the refund path depends on the specific state/deadline rules. A provider proposal cannot extend the payer's refund deadline indefinitely. Legacy escrow versions retain their own terms; v3's behavior must not be retroactively applied to them. [E17][E30]

19.4 A timeout after dispatch

An application sends a job, loses its connection and receives no final receipt. Several states remain possible: the provider never received it; work is running; output was produced but delivery failed; or a payment transaction was submitted but not observed. Blindly redispatching can duplicate compute or payment. The durable router/settlement journal retains identity and reservation, checks existing chain/provider evidence and retries only according to the implemented reconciliation rules. A timeout is uncertainty, not proof of non-execution. [E12]

If the chain reorganizes, the canonical funding, assignment or settlement anchor can change. The off-chain journal must reconcile against the surviving branch, while protecting against duplicate receipt/job use. Local idempotency and chain transaction uniqueness are useful mechanisms, but they do not prove universal distributed exactly-once behavior across every crash, filesystem failure and provider implementation. This is why restart/reorg acceptance and useful-work attribution remain explicit requirements.

19.5 One model across providers versus independent replicas

A replica serves a complete model on one eligible host; several replicas can serve different requests concurrently. A pipeline gives different contiguous layers of one model to different hosts. Pipeline decode latency depends on every participating stage and hop, so adding a slow host can make one request slower. Neither summed VRAM nor the number of advertisements is a performance result. The planner uses measured capacity and link inputs and can leave a candidate on standby. [E14]

For an explicitly assumed two-stage example, compute times of 60 ms and 40 ms plus a 20-ms activation hop imply 120 ms/token, or about 8.33 tokens/s in the planner's simplified sequential-request model. If the hop becomes 100 ms, the estimate falls to 5 tokens/s. Those are scenario calculations, not MeshLLM or Colibri observations. They omit many runtime, batching, queueing and failure costs and cannot prove a deployed pool's throughput.

Training adds further requirements: permission to use the dataset, an approved task/model lineage, verifiable contributions, quality evaluation and a promotion decision. The inference-only escrow does not pay arbitrary training work just because a training contract exists. Poisoned updates, colluding evaluators and non-useful computation remain risks requiring independent validation. [E13][E14][E17]

19.6 A QAI mint and redemption across the trust boundary

In a mint, customer funding evidence and custodian/approver signatures identify the beneficiary, asset and amount. The vault credits eligible backing only through its authorized path; the controller checks fresh conservative coverage and limits; the token's issuer boundary creates

the liability. Customer backing and capital margin stay distinct. The on-chain record cannot independently observe a bank deposit or prove legal segregation. [E18][E19]

For redemption, the holder signs a request that locks tokens in the redemption escrow. They remain outstanding liabilities while pending. Authorized settlement burns the corresponding tokens and debits recorded customer backing. The settlement event authorizes the off-chain payout under the documented trust model; it does not demonstrate that bank money has actually arrived. A timely rejection or an SLA reclaim returns the locked tokens according to the lifecycle. Returning tokens restores their use, not a completed money payment. [E19][E30]

A relayer pays native gas for these operations but cannot rewrite signed amount, recipient or fee fields. Recovery uses a different post-quantum key path and delay/cancellation rules. This narrows selected authorization risks while retaining dependence on correct verifier deployment, honest role custody, bank/custodian operations and the base chain. Multiple keys on one compromised PC do not become independent simply because the contract requires two approvals. [E18][E20]

19.7 From package discovery to safe local activation

The distribution client verifies metadata under its retained trust root, checks freshness/rollback constraints and content digests, and stages a compatible package. IPFS supplies bytes identified by a CID; an independently retained copy or CAR can provide another retrieval path. Hash equality answers “are these the expected bytes?” rather than “are they safe, licensed or forever available?” [E23]

Activation still needs compatibility with the retained chain, profile and allowed upgrade schedule. Local health checks and rollback can help an independent service recover, while a Core database or consensus transition can make an old binary unsafe to restart. A provider can stop serving without changing block validity; an updater cannot silently change issuance or regenerate genesis. Distribution, application automation, economic approval and consensus authority remain separate boundaries.

Chapter 20

Participation and decentralization

20.1 Node and operator participation

Full nodes need compatible Core software, authenticated genesis/profile, adequate storage and peer connectivity. Mining additionally requires the correct GhostRider implementation and CPU capacity. Smartnodes need locked stake, active duties, persistent signer journals and secure recovery. AI providers additionally need explicit participant authorization, a supported verified runtime, approved model identity and reachable service transport. These requirements are cumulative for operators combining roles. No universal minimum hardware specification has been independently validated. [E02][E03][E05][E10]

An independent operator controls infrastructure and keys separately from other operators. Multiple validator identifiers, payout addresses, processes or providers on one PC do not prove independence. The historical chain observation had zero peers on a single host, while the later QAI record still had threshold keys on one PC. These dated facts cannot establish current network topology; they do establish why node count and nominal multisig thresholds are insufficient evidence of decentralization. [E02][E20]

20.2 Concentration and economic incentives

Measure effective stake by beneficial operator, hash rate by independent mining control, AI jobs/revenue by provider operator, and artifact replicas by failure domain. Shared hosting, relays, package publishers, RPC endpoints and recovery devices can produce correlated failure despite many identities. The pool-selection policy's operator grouping is useful but depends on enrollment truthfulness. Incentive shares encourage participation only if real costs, service demand and access constraints allow independent operators to enter. [E11][E16]

For an explicitly analytical concentration indicator, let s_i be operator i's fraction of a measured resource and define $H = \sum_i s_i^2$. This is a reporting suggestion, not an implemented protocol rule or a verified network measurement. Operator identification uncertainty should accompany it; treating ten aliases as ten independent equal operators would understate concentration.

20.3 Acceptance path

The roadmap requires independent node/provider/storage operators, separate signer custody, retained recovery copies and verified onboarding without one development machine. Evidence should include distinct control/failure domains, source-bound packages, synchronization and finality traces, outage/rejoin tests and documented incident response. No accepted community governance or geographical distribution is inferred from an open participation objective. [E23][E24]

Chapter 21

Interoperability and distribution

21.1 EVM interfaces and contract integration

Ethereum-family transaction formats, JSON-RPC and EVM tooling provide an integration surface for wallets and contract applications. Compatibility remains conditional on supported forks, gas rules, precompiles and actual RPC methods. A QAI balance is keyed by smart-account identifier and does not expose ordinary ERC-20 authorization; it requires an adapter aware of that ABI. A contract address without network/genesis identity is insufficient integration metadata. [E07][E18][E25]

The repository contains developer documentation, explorer APIs and wallet tooling. They are useful application interfaces, not consensus authorities. External applications should verify canonical receipts and contract runtime identity and avoid equating an explorer's latest indexed head with finalized state. A cross-chain bridge, trustless external oracle or production liquidity venue is not established by these interfaces.

21.2 Content and software distribution

Model and package artifacts can be addressed by digest/CID and transported through IPFS. The implemented release fabric uses compatibility manifests, TUF metadata, verified retrieval/staging and local update rollback. CAR export/recovery and checkpoints support bounded restart recovery without defining another cryptographic trust root. The local fixture evidence demonstrates selected paths on one computer, not geographically independent retention or public package acceptance. [E13][E23]

The deployment target is IPFS-first distribution with independently retained replicas and offline recovery. IPFS content identity and pinning have distinct roles [6]; TUF authenticates authorized metadata under its trust model [13]. Neither guarantees future availability, clears redistribution rights or changes the network's consensus profile. Packages with incompatible genesis or activation schedules must be rejected even if their signatures are valid.

21.3 Oracles and cross-system trust

An AI answer, reserve attestation or webhook can feed an application contract only through an explicit trust mechanism. Provider signatures establish attribution; they do not turn external facts into consensus truth. An accepted oracle design would need named data sources, update/freshness rules, dispute behavior, replay/domain protection and failure semantics. Cross-system integrations and real-world oracles are therefore use cases or component-specific mechanisms rather than an accepted general interoperability protocol. [E12][E19][E25]

Chapter 22

Applications and ecosystem scope

22.1 Programmable ledger applications

The implemented EVM surface supports development of payments, contract-based registries and escrow applications on the retained testnet. Studio and wallet paths add multiple-language development and deployment workflows, subject to compiler and VM constraints. A deployed example or a passing conformance case is not an audited production application. WASM security findings also constrain application admission and release claims. [E07][E08][E25]

22.2 AI-powered applications

An application can combine model/provider lookup, a signed bounded quote, off-chain inference delivery and customer-funded escrow. A smart agent can operate within a wallet-authorized budget and capability policy. The gateway, authorization ledger, runtime adapter and settlement journal are concrete components; full independent-provider market acceptance and beneficial task outcomes remain separate. Automation must respect replay, budget, expiry, model consent and ambiguous-job recovery. [E10][E11][E12][E21]

Distributed inference can place a large model across authorized stages or serve parallel requests on replicas. The planner offers those modes, but multi-host throughput and safe contribution settlement require independent execution evidence. Training and model improvement have coordinator, budget and promotion components; they are not an active automatic reward economy funded by inference-only v3. [E14][E17]

22.3 Infrastructure and real-world integration

Content-addressed distribution, local recovery and explorer/RPC tools can support decentralized infrastructure applications. Reserve accounting illustrates how signed external attestations may constrain contract decisions, while also exposing the limits of trusting off-chain assets. An AI-assisted oracle, enterprise workflow, bridge or real-money stable-asset service would need its own trust, privacy, availability and legal design. This paper does not claim those future applications are available simply because compatible primitives exist. [E19][E23][E25]

Chapter 23

Developments since edition 0.3.0

Evidence status: Supplement prepared from repository `1bf413ccef6c9bdfe658e44ebbae6e58a505fc50` (19 November 2026) and the readiness ledger at that revision. Dated deployment records are cited as recorded; no fresh remote observation is claimed.

Edition 0.3.0 audited repository `8ca81f7035cd95e6ea3a228fdebb38c44bdb8d1d` (14 November 2026). In the five days that followed, roughly two hundred commits reached the main branch. This chapter summarises the changes that alter what the system does or how it is operated. Earlier chapters keep their baseline scope unless they point here. Status words follow the public readiness levels: *planned*, *implemented*, *locally tested*, *deployed on testnet*, *enabled on the running network*, *independently audited* and *mainnet ready*. No feature has reached the last two. [E38]

Area	Level at cutoff	Change since 0.3.0 and principal limit
IPFS reward v2	Deployed on testnet	Bonded, slashable proof of access; 5% share redirects at block 68,961. Unaudited. [E32]
PATE phases 1–2	Locally tested	Adaptive trust scheduling and opt-in recurring smart transactions; all flags off on 20777. [E33]
PATE phase 3	Enabled (shadow)	Provisional fast-finality signal beside real checkpoints; not consensus finality. [E33]
AI market	Deployed on testnet	ModelRegistry and AIJobMarket escrow verified on 20777; no independent provider yet. [E34]
Model pools	Deployed on testnet	Three federated directories, relay, power preview, heartbeats, reputation penalties. [E34]
Wallet node and mining	Enabled	Native Windows Core bundled; multi-thread GhostRider mining from wallet 0.8.13. [E37]
Internal agents	Deployed on testnet	Network Monitor (act mode, proposals only) and Security agent (dry run); no keys or funds. [E36]
Explorer	Enabled	Version 0.4.0 on PostgreSQL with a Model Pools page.
Measured throughput	Historical, one host	Live-difficulty series at 15M, 30M and partial 60M gas, plus batched payments. [E35]
Public testnet, mainnet	Planned	Unchanged: independent review, operators and an explicit launch decision are required. [E38]

Table 23.1: Changes between editions 0.3.0 and 0.4.0 and their readiness level.

23.1 IPFS reward v2: bonded proof of access

The first IPFS reward (Chapter 12) gates claims on OperatorDirectory enrollment and proof of pinning, which shows that bytes were produced in time but not where they are stored. Version 2 changes the incentive structure. An operator posts a PQAI bond (minimum one test PQAI), answers storage challenges in each 192-block round, and shares that round's five-percent budget with the other operators who answered correctly. A missed proof can be slashed from the bond. The issuance split is unchanged at 67/18/1/9/5. Only the recipient of the five-percent share changes, from the v1 pool to pool `0x16362198b0685B4a1d4C1f531B215C21568002b7` at block 68,961. Nodes need Core `f31a83ce5` or wallet 0.8.12 or later before that height, and an

older node diverges at activation. The remaining v1 backlog was moved into the v2 carry by a single-purpose migration key, which is to be revoked after a second pass. [E32]

Limitations. The redirect height had not been reached at the source cutoff. The contract and agent are unaudited test-network code, and no operator income is inferred from pool funding.

23.2 PATE: adaptive trust execution

PATE extends PASE (Chapter 8) in three phases. Phase 1 schedules transactions by learned conflict history: predictable conflicts go straight to the serial path instead of failing speculation. Phase 2 adds opt-in recurring smart transactions, authorised through a permissions contract with property, golden and security tests. Phase 3 lets validators sign an additional provisional finality signal next to the real 32-block checkpoint vote, so its latency can be measured without changing the fork choice. A signer-starvation defect found in the phase-3 rollout was fixed in Core `987d727fb`, which runs on both seed hosts. [E33]

Limitations. Phases 1 and 2 are off by default and not enabled on 20777. The RecurringPermissions contract is not deployed and needs an independent audit. Wallets, exchanges and the fork choice must continue to rely on checkpoint finality. Enabling faster finality would be a hard fork.

23.3 AI market, model pools and workflows

ModelRegistry and AIJobMarket were deployed and verified on 20777 on 15 November 2026. A client locks PQAI in escrow, a provider runs the model off-chain, and payment is released only for accepted work: 88% to the provider and 4% each to gateway verification, the CPU pool and the Smartnode pool. Same-model compute pools let computers running the same model register as one pool. Three federated directories (two seed hosts and a third pool host) and a members-only relay for machines behind NAT serve the pools. The directories record pool power, member heartbeats and reputation penalties; an on-chain bond is planned later. The first public pool, a two-billion-parameter model open to GPUs with at least 2 GiB of memory, was created on 18 November. The wallet also gained a local workflow canvas: templates for text, image, video and speech models, pre-flight checks, run history and pinned runtimes. These run on the user's machine and are not consensus features. [E34]

Limitations. At the cutoff no independent provider was registered, no paid job had run on 20777, and the first pool had no outside members. Pool members are not paid on chain. Splitting one model across several independent GPU computers has not been verified.

23.4 Mining from the wallet

Wallet 0.8.11 bundles a native Windows Core with its source archive and refuses private chains, so the same installer can run a full node on the official network without WSL. Wallet 0.8.13 adds multi-thread GhostRider mining. In a dated observation, one 24-thread desktop mined 42 consecutive blocks (53,031–53,072) on the live chain. [E37]

Limitations. Consecutive blocks from one computer demonstrate working software and, equally, the current concentration of hash power. They are not evidence of decentralisation. Pool mining is tested only on a disposable chain.

23.5 Internal agents

The Quantum Layer gained a runtime for small, local, advisory agents. The Network Monitor (QL-007) runs a 0.6-billion-parameter model on the CPU of seed1. Every two minutes it checks node health and decides to do nothing, write an alert or log an operation proposal for a human. Every decision is appended to a hash-chained log. The Security agent (QL-008) watches logs and configuration drift, tests the project's own public endpoints daily, and can propose capped, self-expiring firewall blocks; it runs in dry-run mode. [E36]

Limitations. Agents hold no keys and never sign transactions, move funds or change consensus. Restarts, resyncs and disk or clock operations remain human decisions. Two of six planned roles exist.

23.6 Operations and security posture

Administrative SSH on the two seed hosts and the web host now accepts keys only, and root passwords were rotated. HTTP security headers were corrected on all three hosts after a dated security test. The explorer moved to PostgreSQL with a table-by-table copy verification. The WASM resource-pricing findings described in Chapter 15 remain open. A candidate fix exists in private Core but needs a scheduled fork height and owner approval, so the earlier limitation stands.

23.7 Deployment lineage at the cutoff

The text above relies on several Core revisions, each named for what it supports: `f2eace6bc` is the baseline audited by earlier chapters; `12127fc1e` carries the CR-0051 IPFS fork; `987d727fb` adds the phase-3 signer fix; `f31a83ce5` adds the IPFS v2 redirect and runs on the seed hosts; `f5d9ce2c6` adds parallel mining and is bundled with wallet 0.8.13. This edition did not repeat the selected-Core source capture for the later revisions. Their behaviour is supported by dated records and focused tests, not by a new full-source audit.

Chapter 24

Acceptance roadmap and research

Chapter 25 turns the acceptance areas below into ordered milestones, network growth targets and resource needs. [E38]

24.1 Completed implementation slices

The repository provides a retained source snapshot, selected private Core lineage, registered components, local build/provenance tooling, EVM/WASM paths, GhostRider integration, checkpoint/registry mechanisms, PASE executors, AI runtime adapters and service/model contracts. Dated evidence includes local cross-VM differentials, wallet/compiler tests, one-host inference, reserve-accounting tests and a testnet QAI deployment. These are concrete slices, not completion of every gate. [E01][E08][E09][E15][E20]

24.2 Required work without invented dates

Acceptance area	Evidence required before stronger publication claims
Core safety	Resolve WASM resource findings; review activation and existing-code handling; freeze exact source and rerun adversarial/differential tests.
Consensus	Independent dynamic-validator/finality and recovery analysis; realistic partition, churn, censorship and stale-signer scenarios.
Packages	Complete license/source-offer review, signed reproducible role packages, compatibility and clean-client installation.
Distribution	Independent retained IPFS replicas, outage/offline recovery and fresh bootstrap without one developer host.
AI services	Independent providers, authenticated distributed runtimes, useful contribution verification and reorg/restart settlement.
QAI	Separate key custody, current bytecode/policy revalidation, independent contract/economic review; retain simulated status unless a real compliant structure is separately approved.
Performance	Sustained mixed workloads, resource soak, finality/latency distributions, WAN failure recovery and reproducible results.
Public acceptance	Complete applicable Gates 0–9, independent assurance, operating procedures and explicit owner go/no-go.

Table 24.1: Evidence-gated roadmap. No delivery dates or automatic gate closures are implied.

24.3 Public testnet and mainnet prerequisites

The current gate register remains the canonical acceptance queue, and the finding register retains granular unresolved requirements. Documentation checks cannot promote a gate. A running public endpoint or a permitted testnet contract deployment is not equivalent to a fully accepted public-testnet release. Mainnet follows accepted testnet operation, soak/incident/concentration

evidence, revalidated frozen source, supply reconciliation, custody and upgrade review, a selected mainnet identity and a separate owner decision. [E24]

24.4 Research directions

Further work should investigate composition of cumulative-work fork choice with dynamic checkpoint sets, economic effects of the bounded DAA, PASE correctness under adversarial state access and cross-VM effects, and resource pricing for cold compilation. Service research includes independent useful-work verification, reproducible model evaluation, privacy-aware routing and incentive compatibility under colluding providers. Reserve research requires liquidity and correlated-failure stress tests rather than a ratio-only argument. These are questions, not promised implementations.

The maintenance workflow accompanying this edition flags changed protocol sources and affected chapters. It never rewrites claims or publishes the PDF automatically. New measurements must retain source/configuration identity and failed attempts, and every edition must recheck stale plan, website and deployment statements against stronger evidence.

Chapter 25

Path to mainnet and resourcing

Evidence status: Planning chapter. It states the work and resources the project believes are needed. It is not an offer of tokens, securities or any other financial product, and nothing in it promises a mainnet date, a mainnet allocation or a return.

25.1 Where the network stands

Test network 20777 runs from two seed hosts and one pool-directory host operated by the founder. Four genesis validators produce real checkpoint votes and real finality, but they run on one host, so they are not independent. A single always-on VPS miner and occasional desktop miners seal blocks. Most features in this paper are implemented and many are enabled on that network; none is independently audited. Public testnet and mainnet are *planned* in the readiness ledger. [E38]

The binding constraint on reaching mainnet is no longer missing code. The gaps are independent people and machines: external security review, operators who do not report to the founder, hosts in more than one place, and enough sustained use to reveal failures that one developer cannot reproduce alone.

25.2 Milestones

The milestones below are ordered and evidence-gated. Each closes only when its exit evidence is recorded; no milestone has a promised date.

25.3 Network growth targets

Decentralisation is measured, not declared. The project will track and publish, for each milestone, the number of independently operated validator hosts and their share of stake, the number of distinct miners and the largest miner's share of blocks, the count and regional spread of seed and RPC hosts, the number of bonded IPFS operators, and the number of AI providers with accepted jobs. The minimum target for M2 is three independent validator hosts, because two hosts split evenly cannot keep finality through a partition. Mainnet needs a materially larger set, and the target should be fixed by the consensus review in M4, not chosen in advance for marketing.

25.4 Resource needs

The work above needs money and people that the project does not have today. The categories, in priority order:

Milestone	Work	Exit evidence
M1 Hardening	Fix and fork-schedule the WASM resource findings; complete the slashing and validator lifecycle; freeze a release candidate Core.	Adversarial and differential suites pass on the frozen source; fork activates on 20777 without divergence.
M2 Open testnet	Faucet, published operator guides, at least three independently operated validator hosts and more seed/RPC hosts in different regions.	Finality holds through a planned host outage; a fresh node bootstraps without any founder host.
M3 Incentivised testnet	Structured programme for miners, Smartnode operators, IPFS operators and AI providers; recorded incident drills.	Weeks of operation with published uptime, incident and concentration reports.
M4 Independent audits	Core consensus and WASM VM; contracts (Quantum Reserve, QAI, AI market, IPFS reward v2, pools); key custody and operations.	Audit reports published with every critical and high finding resolved or explicitly accepted.
M5 Genesis	Mainnet chain identity, supply reconciliation, founder and reserve key ceremony with separated custody, reproducible signed packages.	Public genesis manifest; independent operators confirm the same genesis hash.
M6 Launch decision	Go/no-go review against all gates.	Explicit, recorded owner decision.

Table 25.1: Evidence-gated milestones from the current testnet to mainnet.

1. **Security audits.** Independent review of consensus, the WASM VM and the deployed contracts is the largest single cost and the hardest prerequisite to skip.
2. **Infrastructure.** Seed, RPC, explorer and directory hosts in several regions, with monitoring and backups, for the open and incentivised testnet phases.
3. **Operator and contributor programmes.** Grants or reimbursements that let independent validators, IPFS operators and AI providers run test infrastructure, plus bug bounties.
4. **Engineering.** Hardening, release engineering and reproducible packaging beyond what one developer can sustain.
5. **Legal and compliance.** Entity, licensing of the Core-Geth-derived source, and review of any funding mechanism and of mainnet token design.
6. **Documentation and community.** Guides, translations and support for new operators.

A budget with amounts belongs in the funding campaign itself, where it can be updated as quotes arrive. Any campaign should publish that budget, report spending against it, and hold project funds behind multi-party approval rather than a single key.

25.5 Funding mechanisms under consideration

The project has not chosen a mechanism. The options differ in what a supporter receives and in the regulatory obligations they create.

Donations and sponsorship.
Supporters receive no token, equity or rights. This is the simplest option and fits public-good infrastructure, but it raises less.

Reward-based crowdfunding.
Supporters receive non-financial rewards such as recognition or hardware. Major platforms restrict or prohibit crypto-asset and financial rewards, so platform rules must be checked

before launch.

Grants.
Ecosystem, foundation or public research grants for specific deliverables such as audits or post-quantum research. Typically milestone-based and non-dilutive.

Token sale.
Selling a future mainnet token is regulated. In the European Union, an offer of crypto-assets to the public generally falls under the Markets in Crypto-Assets Regulation, which requires a legal-person offeror, a regulation-format crypto-asset white paper notified to the competent authority, rules for marketing communications and a withdrawal right for retail buyers. Other jurisdictions apply securities law. This technical whitepaper is not such a document.

Equity or convertible investment.
Investment in a company that develops the software, under the applicable investment and crowdfunding rules.

Whatever the mechanism, three rules apply. Test PQAI and test QAI have no monetary value and are not converted into mainnet value by this paper; any treatment of testnet participation at mainnet would be a separate, published decision. No campaign material may describe rewards as guaranteed, passive income or an investment return. Funds raised for audits and operators should be traceable to those uses.

Limitations. This chapter reflects the project's own planning at the cutoff. It is not legal or financial advice, and the regulatory summary is general; any campaign needs jurisdiction-specific legal review before launch.

Chapter 26

Conclusion

PQWERAI's inspected implementation joins a Core-Geth-derived programmable ledger, CPU work-based production, authenticated Smartnode checkpoints, optional ordered speculation and off-chain AI-service accounting. Its most consequential architectural boundary keeps inference and training outside block validation. The CR-0051 supplement corrects issuance to 67/18/1/9/5 from recorded block 34785, including the shared, proof-gated IPFS pool. QAI reserve contracts remain distinct from native issuance. Other implementation chapters retain their baseline revision scope. [E31]

Edition 0.4.0 records rapid progress on the running test network: a bonded IPFS reward, shadow fast finality, a deployed AI market, model pools, wallet mining and advisory agents. It also shows where the remaining work is. Most of that work is not code: independent audits, independent operators, more hosts and an explicit launch decision. Chapter 25 describes these needs. [E38]

The available evidence supports a technical review draft, not an unqualified production claim. Selected-Core WASM findings, incomplete independent consensus assurance, concentrated signer custody, simulated reserves, distribution rights and missing independent-host performance acceptance remain material. Preserving these limits makes the document useful for implementers and external reviewers. Its source ledger, unresolved-claim report and conservative freshness guard support revision as those gaps are resolved.

Appendix A

Implementation evidence ledger

A.1 How to reproduce the evidence boundary

Repository paths below are relative to the active checkout at `1bf413ccef6c9bdfe658e44ebbae6e58a505fc50`. Selected-Core paths refer to the separate private Core at `f2eace6bc69fca48b290142b8232bd1ced6a690b`. The historical retained snapshot must not be substituted for it. Full claim-to-source/test mappings, component inventory and normalized source hashes accompany this project under `evidence/`. The hashes detect changes; they do not independently validate implementation correctness or confer source-publication rights.

Each evidence identifier points to a bounded claim family. The narrative's equations are either transcriptions/derivations of the identified code or explicitly labelled analytical scenarios. Tests named in the matrix are existing evidence hooks; their execution status for this edition is recorded separately in `evidence/verification.json`. A referenced historical PASS is not represented as a newly executed PASS.

E01 Repository and selected Core are different source identities; both releases remain unaccepted.

Status: inspected source and registers. **Primary sources:** `README.md`; `core/SOURCE_REVISION`. **Limit:** Register status is not independent release assurance.

E02 Canonical chain 20777 retains its authenticated genesis; 77240 is historical.

Status: source and historical observation. **Primary sources:** `docs/project-control/01_INVARIANTS.md`; `docs/project-control/04_DECISION_LOG.md`. **Limit:** No fresh remote RPC observation in this session; dated manifest wording is partly superseded.

E03 GhostRider seal uses a pinned helper, bounded target comparison and clamped difficulty adjustment.

Status: selected core source inspected. **Primary sources:** `protocol/LOCAL_POW_PROFILE_V1.md`; `testnet/release/network/candidate-20777/GENESIS_MANIFEST.json`. **Limit:** Retained source supports historical lineage; exact selected Core files are pinned separately.

E04 G4 checkpoints use strict stake quorum and source-to-successor finalization.

Status: selected core source inspected. **Primary sources:** `protocol/LOCAL_FFG_EXPERIMENT_V1.md`; `protocol/G2_VOTE_ENCODING_CANDIDATE.md`. **Limit:** Quorum intersection is conditional on a common authenticated epoch set; not a dynamic-set safety proof.

E05 Registry implements staking lifecycle, evidence-based slashing and finalized withdrawal checks.

Status: selected core source inspected. **Primary sources:** `protocol/G4_VALIDATOR_LIFECYCLE_TESTNET.md`; `protocol/LIVENESS_CANDIDATE_V1.md`. **Limit:** External-operator lifecycle and recovery acceptance are incomplete.

E06 Core devp2p and AI libp2p are separate networking planes.

Status: implemented local evidence. **Primary sources:** `ai/p2p/node.go`; `ai/p2p/binding.go`. **Limit:** NAT reachability and operator diversity are not guaranteed.

E07 Core EVM transactions, gas and state remain canonical across execution modes.

Status: source and recorded tests. **Primary sources:** `testnet/release/network/candidate-20777/GENESIS_MANIFEST.json`; `core/source/core/state_processor.go`. **Limit:** No universal Ethereum fork or tooling compatibility assertion.

E08 WASM is required release scope, but selected Core has unresolved resource-cost defects.

Status: implemented with open security findings. **Primary sources:** `docs/architecture/decisions/ADR-0013-official-wasm-core-release.md`; `audit/evidence/2026-11-14-studio-adversarial/README.md`. **Limit:** Candidate b6bc0ab2e is not the selected or deployed Core; network OOM was not induced.

E09 PASE includes bounded prediction, prefetch and general optimistic execution with canonical-order fallback.

Status: implemented opt in. **Primary sources:** `execution/pase/engine/canonical_executor.go`; `execution/pase/engine/speculative_general.go`. **Limit:** Serial equivalence tests do not prove all programs safe or establish network TPS.

E10 MeshLLM is an upstream runtime supervised through PQWERAI adapters, separate from validation.

Status: implemented with scoped local evidence. **Primary sources:** `ai/runtime/mesh_client.py`; `ai/runtime/mesh_supervisor.py`. **Limit:** Hardware detection alone is not measured capability; remote stages need independent acceptance.

E11 Routing filters registry-bound providers; observations do not establish independent ownership.

Status: implemented customer policy. **Primary sources:** `ai/runtime/ROUTING_POLICY.md`; `ai/runtime/registry_client.py`. **Limit:** Trusted Core RPC and directory enrollment remain assumptions.

E12 Metering caps output-token charges and durable settlement tracks ambiguous work.

Status: implemented customer meter and journal. **Primary sources:** `ai/runtime/metering.py`; `ai/runtime/jobs.py`. **Limit:** Receipts authenticate claims; they do not prove inference quality or useful compute.

E13 Model identity, policy certification, storage retention and output quality are different properties.

Status: implemented controls partial acceptance. **Primary sources:** `ai/models/manifest.py`; `ai/models/trust.py`. **Limit:** No universal model-safety, confidentiality or persistence guarantee.

E14 Cooperative planner estimates pipelines and replicas; training contracts are not accepted distributed training.

Status: implemented planner and partial training. **Primary sources:** `ai/pool/planner.py`; `ai/pool/accounting.py`. **Limit:** Full independent multi-host useful-work and exactly-once payout acceptance remains open.

E15 Colibri backend passed a one-host CUDA measurement; cluster mode and settlement integration remain unaccepted.

Status: historically measured one host. **Primary sources:** `ai/hardware/colibri.py`; `audit/COLIBRI_BACKEND_2026-11-13.md`. **Limit:** No new GPU benchmark or distributed inference measurement in this session.

E16 Pre-fork baseline rewards settle by epoch with a 72/18/1/9 split; CR-0051 supersedes the current split from block 34785 (E31).

Status: selected core source and executed accounting vectors. **Primary sources:** `core/source/consensus/ethash/g3_rewards.go`; `audit/CHAIN_20777_CONSISTENCY_2026-11-13.md`. **Limit:** Overlay executes actual reward accounting only; it does not authenticate certificates, mine blocks, measure live rewards or authorize a changed monetary policy.

E17 ServiceEscrow v3 has immutable configured payees and provider remainder, separate from issuance.

Status: implemented contract policy. **Primary sources:** `contracts/ai/ServiceEscrow.sol`; `contracts/ai/legacy/ServiceEscrowV1.sol`. **Limit:** A proposed 85/5/5/5 schedule is not the deployed universal policy.

E18 QAI uses PQ smart-account authorization; base-chain signatures and transport remain classical.

Status: implemented with dated crypto tests. **Primary sources:** `qai/contracts/QuantumAccountRegistry.sol`; `qai/contracts/PQVerifiers.sol`. **Limit:** Algorithm conformance is not FIPS validation or whole-system post-quantum security.

E19 Reserve policy implements 110/120/130 thresholds, simulated backing, attestations and redemption lifecycle.

Status: implemented simulated reserves. **Primary sources:** `qai/contracts/reserve/QuantumEconomicController.sol`; `qai/contracts/reserve/QuantumReserveVault.sol`. **Limit:** Attestations cannot prove off-chain assets exist or are legally segregated.

E20 QAI/reserve stack deployment is recorded on 14 November; automatic rules disabled and keys concentrated.

Status: recorded testnet deployment. **Primary sources:** `qai/contracts/QuantumReserve.sol`; `audit/evidence/2026-11-14-qai-reserve-deploy-20777/README.md`. **Limit:** Not freshly reverified remotely; no real reserves; 2-of-3 keys on one PC do not establish custody independence.

E21 Quantum Layer is classical advisory code and cannot change consensus or move funds.

Status: implemented advisory shadow mode. **Primary sources:** `quantum/README.md`; `quantum/controller.py`. **Limit:** Python PASE hints are not consumed by the Go executor.

E22 Historical RC1 mixed workload achieved 3.13 tx/s under a paced six-local-node scenario.

Status: historically measured not current capacity. **Primary sources:** `testnet/release/PERFORMANCE_REPORT.md`; `benchmarks/tps/README.md`. **Limit:** Different source/profile; no extrapolation to 20777 capacity or WAN finality.

E23 Signed release-fabric retrieval, TUF and CAR recovery have local fixture evidence only.

Status: implemented with local fixture evidence. **Primary sources:** `release_fabric/catalog.py`; `release_fabric/car.py`. **Limit:** No independent replica availability, cleared Core distribution rights or immutable future availability inferred.

E24 Existing governance and release gates are project-controlled, not an established token-voting DAO.

Status: observed process and open gates. **Primary sources:** `AGENTS.md`; `docs/project-control/07_PHASE_GATE_STATUS.md`. **Limit:** Readiness is expected to remain NOT READY; historical deployment permission is not blanket launch acceptance.

E25 Wallet, explorer and developer tooling exist; they are distinct from universal cross-chain interoperability.

Status: implemented tooling with scoped evidence. **Primary sources:** `docs/components/wallet/CMP-WALLET.md`; `docs/components/explorer/CMP-EXPLORER.md`. **Limit:** No accepted production bridge, trustless oracle or mainnet deployment inferred.

E26 Historical inference probes and QAI gas costs admit bounded descriptive reanalysis, not current service prices or independent replication.

Status: historical data reanalyzed. **Primary sources:** `audit/evidence/2026-11-13-colibri/verify-cuda-run1.json`; `audit/evidence/2026-11-13-colibri/verify-cuda-run2.json`. **Limit:** Counts are runtime reported; small dependent samples; different models/profiles cannot be pooled; gas is historical.

E27 Historical test-builder burst rates have reduced difficulty, altered gas profiles and no sustained/finalized TPS measurement.

Status: historical local bursts not current chain capacity. **Primary sources:** `audit/evidence/2026-11-04-capacity/512-streams.json`; `audit/evidence/2026-11-04-capacity/`

`2048-streams.json`. **Limit:** RPC streams are not PASE workers; timestamp cadence is not wall-clock enforced; no new runtime run is claimed.

E28 Four hypothetical QAI paths and capital/liquidity sensitivities are reproducible exact-integer analytical scenarios.

Status: new illustrative model not calibrated forecast. **Primary sources:** `qai/contracts/reserve/QuantumEconomicController.sol`; `qai/contracts/reserve/QuantumReserveVault.sol`. **Limit:** Arbitrary assumptions, no probabilities, actual bank payments, full contract emulator, observed demand, market-price estimate or promised capital/return.

E29 Every registered feature is catalogued with status, test-hook count and gaps; these administrative counts are not runtime or security statistics.

Status: complete register export not release acceptance. **Primary sources:** `docs/components/registry.json`; `docs/whitepaper/tools/analyze_features.py`. **Limit:** 76 entries do not establish 76 deployed or independently accepted capabilities; unavailable metrics remain explicit.

E30 End-to-end worked examples preserve distinct authorization, funding, execution, receipt, settlement and off-chain payout boundaries.

Status: source based explanatory walkthroughs. **Primary sources:** `contracts/ai/ServiceEscrow.sol`; `ai/runtime/metering.py`. **Limit:** Worked examples are not newly run network/AI/banking experiments; synthetic prototype credits are not actual money.

E31 CR-0051 allocates 5% of native issuance to the IPFS pool from recorded block 34785; post-fork shares are 67/18/1/9/5, with proof-gated weighted operator claims.

Status: pinned later source and reconciled dated activation plus new local tests. **Primary sources:** `docs/whitepaper/evidence/ipfs_reward_supplement.json`; `docs/whitepaper/tools/ipfs_reward_evidence.py`. **Limit:** Dated activation evidence is not a new remote observation. Rehearsal uses Kubo stand-ins; no real-operator payout or income is inferred from pool funding.

E32 IPFS reward v2 (CR-0065/CR-0070) replaces the open v1 claim path with bonded operators, 192-block storage-challenge rounds and slashable missed proofs; the 5% issuance share is redirected to pool 0x1636...02b7 from block 68961.

Status: deployed testnet contract with scheduled redirect. **Primary sources:** `protocol/IPFS_REWARD_V2.md`; `audit/evidence/2026-11-18-ipfs-reward-v2/README.md`. **Limit:** Redirect activation at block 68961 was scheduled, not observed, at the source cutoff. Minimum bond 1 test PQAI. Unaudited test-network code; no operator income is inferred.

E33 PATE phases 1-2 (adaptive trust scheduling, opt-in recurring smart transactions) are implemented and locally tested with all flags off on 20777; phase 3 runs only as a shadow fast-finality signal beside unchanged checkpoint finality.

Status: locally tested plus shadow deployment. **Primary sources:** `architecture/PATE_ADAPTIVE_TRUST_EXECUTION.md`; `architecture/PATE_PHASE2_SMART_TRANSACTIONS.md`. **Limit:** Shadow mode is not consensus finality. Activating faster finality is a hard fork requiring independent review. RecurringPermissions is not deployed.

E34 ModelRegistry and AIJobMarket escrow are deployed on 20777 (88/4/4/4 split of accepted jobs), and same-model compute pools run on three federated directories with heartbeats and reputation penalties.

Status: deployed testnet without independent providers. **Primary sources:** `audit/evidence/2026-11-15-ai-market-deploy-20777/README.md`; `audit/evidence/model-pools-2026-11-14/README.md`. **Limit:** No independent provider registered and no paid job run at the cutoff. The first public pool has no outside members. Pool members are not paid on chain; splitting one model across independent PCs is unverified.

E35 One-host six-node chain-20777 harness runs at live difficulty 64 measured about 46-95 TPS native transfers at 15M gas, 169 TPS at 30M gas and 200 batched payments/s at 15M gas, bounded by gas per block and block interval.

Status: historical one host measurement. **Primary sources:** `audit/evidence/2026-11-15-max-tps/README.md`; `audit/evidence/2026-11-19-max-tps/README.md`. **Limit:** Loopback, one miner, 60 s stages; the 60M series is partial. Not WAN, not independent operators, not a network capacity or 10,000 TPS claim.

E36 Internal agents run a small local model with no keys or funds: the QL-007 Network Monitor acts on seed1 by writing alerts and human proposals to a hash-chained log; the QL-008 Security agent runs in dry-run mode.

Status: deployed testnet advisory. **Primary sources:** `audit/evidence/2026-11-18-internal-agents-cr0072/README.md`; `quantum/network_agent.py`. **Limit:** Agents never sign transactions, move funds or change consensus; restarts and resyncs remain human proposals. 2 of 6 planned roles.

E37 Wallet 0.8.11+ bundles a native Windows Core (official network only), and wallet 0.8.13 mines GhostRider on every CPU thread; one 24-thread PC mined 42 consecutive live blocks (53031-53072).

Status: enabled network dated observation. **Primary sources:** `audit/evidence/2026-11-18-parallel-mining/README.md`. **Limit:** A run of consecutive blocks from one PC shows hash-power concentration, not decentralization. Pool mining is tested on a disposable chain only.

E38 The readiness ledger is the single source of public feature status; public testnet and mainnet remain planned, gated by independent review, a genesis ceremony and an explicit launch decision.

Status: ledger snapshot at cutoff. **Primary sources:** `docs/status/readiness.json`. **Limit:** A ledger snapshot is not an audit. Funding needs, budgets and milestones in chapter 27 are plans, not commitments or offers.

Appendix B

Notation and reproducibility checklist

Table B.1: Principal notation and units.

Symbol	Meaning
D, H	Header difficulty and GhostRider digest interpreted as an integer.
w_i, W_e	Effective validator weight and total epoch-set weight; use identical units within a quorum calculation.
G_e, F_e, R_e	Epoch gross issuance, founder share and native reserve share in wei.
P_e, V_e	Epoch PoW and active-Smartnode shares in wei; V_e is unrelated to QAI reserve valuation V.
S_0, S	Genesis and net native supply under an explicit burn convention.
B, p, n, C	Funded AI budget, output-token unit price, accepted output count and settled charge.
V, L, r	Conservative QAI reserve valuation, outstanding liability and coverage; valuation/liability use six-decimal EUR-reference units.
$R_i, W_{<i}$	Recorded reads for speculative transaction i and earlier committed writes in its batch.

For an updated edition: detect actual repository/Core revisions; preserve unrelated work; review decision precedence; compare source and deployment records; run freshness checks; reconcile amounts with integer arithmetic; rerun affected component tests; compile with Latexmk/Biber; inspect every page for overflow and diagrams; verify bibliography and links; scan exported material for sensitive information; record all failures, skips and remaining uncertainties. Author review remains distinct from independent review.

Appendix C

Complete feature status catalogue

The following register export accounts for all 76 features registered at the 0.4.0 cutoff; QL-007 and QL-008 are new since edition 0.3.0. Recorded status is not a runtime benchmark, independent audit or readiness decision. Some status/gap text is stale: the 77240 identity, transfer-only PASE, missing lifecycle and non-activation/non-deployment labels conflict with later source or evidence. The last column annotates these known cases without silently changing the registry or accepting a gate. The CSV preserves original text alongside reconciliation notes in full. Comparable measurements are discussed in Chapter 18; absent statistics are not invented. [E29]

Table C.1: Complete feature register: author status and remaining gaps, not runtime statistics.

ID	Feature	Recorded status	Gap / acceptance limit
CORE-001	Block construction and sealing work	locally verified	Independent operator acceptance
CORE-002	GhostRider G1 hashing	locally verified	Historical Windows gap; newer native-Windows local evidence exists. Independent helper / platform acceptance remains scoped (E01).
CORE-003	Difficulty adjustment	partial	Occasional 80-200 s gaps open
CORE-004	Peer discovery and propagation	partial	Seed outage and independent operator handshakes
CORE-005	Chain synchronization	locally verified	Cold-start repeatability (SIX-NODE-COLD-START-TIMEOUT-REPEATABILITY)
CORE-006	Transaction mempool	implemented untested	No PQWERAI-specific mempool tests recorded (upstream behavior)
CORE-007	EVM execution	locally verified	Independent assurance
CORE-008	WASM execution (official VM v1)	partial	Compiler / Studio / wallet / explorer / gas estimation, security assurance
CORE-009	PASE parallel scheduling	partial	Historical transfer / module gap superseded: general speculation exists and shipped-module h1 matches. Independent performance / correctness assurance remains open (E09).
CORE-010	Issuance and reward accounting	locally verified	Pre-fork baseline: 72 / 18 / 1 / 9. Later CR-0051 activation at 34785 supersedes this split with 67 / 18 / 1 / 9 / 5, including the IPFS pool (E31).
CORE-011	RPC service	locally verified	Public RPC policy deployment
CORE-012	Reorganization and crash recovery	locally verified	Power-loss and independent failure domains

ID	Feature	Recorded status	Gap / acceptance limit
CORE-013	Network identity (chain 77240, genesis pin)	locally verified	Historical 77240 label; retained canonical identity is 20777 under ADR-0015. Launch acceptance remains incomplete (E02).
CORE-014	Native Windows node build	locally verified	Windows 11 x64 local author acceptance; separate Windows 10 OS acceptance, ARM64 support, independent review and...
POS-001	Validator registration (genesis collateral)	partial	Registry lifecycle exists in selected source; independent lifecycle / slashing / finality acceptance is still incomplete (E05).
POS-002	Stake management	design only	Historical design-only label; selected source implements lifecycle / withdrawal checks. This annotation does not accept the full staking gate (E05).
POS-003	Epoch voting and vote transport	partial	Dedicated authenticated devp2p vote propagation
POS-004	BFT checkpoint finality	partial	Finality assurance remains open; retained genesis already enables liveness / v1 with 8 / 12 settings (E02 / E04 / E05).
POS-005	Signer persistence (anti-equivocation)	locally verified	Historical Linux-only label; later Windows signer persistence has scoped local evidence. Independent recovery / custody remains open (E01 / E05).
POS-006	Slashing	partial	Exactly-once attributable penalties not accepted
POS-007	Validator recovery	partial	Custody / backup rollback and weak-subjectivity recovery
AI-001	Model registration	locally verified	Signed approved stage fabric
AI-002	Provider registration	locally verified	Independent providers
AI-003	Hardware detection	locally verified	CPU / CUDA verified on one host only; ROCm, Vulkan, Metal, Intel unverified
AI-004	Provider discovery and routing	locally verified	Independent peers
AI-005	Model download (approved stages)	partial	chunk-verified, resumable, multi-source and layer-selective fetch verified on one offline Kubo; replication and ...
AI-006	Inference execution	partial	Three independent GPU providers
AI-007	Cooperative model splitting	partial	two-stage split of one request ran on one host and one GPU (two processes); no independent machines yet
AI-008	Dynamic pool membership	partial	signed join / leave / epoch / drain / replay control plane verified; live runtime topology change under load not tested

ID	Feature	Recorded status	Gap / acceptance limit
AI-009	Usage metering	partial	Independent eligible contribution
AI-010	Escrow settlement	locally verified	Pool-level exactly-once settlement
AI-011	Distributed training	partial	Accepted collaborative improvement deferred
AI-012	Model verification and promotion	locally verified	Extended finality run stopped by owner
AI-013	Isolated development proposals	locally verified	None recorded
WALLET-001	Keystore and signing	locally verified	Independent review
WALLET-002	Genesis / chain checks and RPC client	locally verified	Over-cap estimate intermittently does not revert
WALLET-003	AI escrow actions	locally verified	UI tests owner-PC only
WALLET-004	MetaMask compatibility	partial	Actual extension UI acceptance
WALLET-005	Distribution package	partial	Signed release; unified Core+wallet distribution
EXP-001	Reorg-safe indexing and reconciliation	locally verified	PostgreSQL acceptance run
EXP-002	JSON API v1	locally verified	Address transactions and token transfers are not paginated
EXP-003	HTML explorer views	implemented untested	No automated view tests
REL-001	Verified source and reproducible builds	locally verified	Cross-host reproducibility
REL-002	License and corresponding-source coverage	partial	go-jsonschema-walk grant; full source offer
REL-003	Signed catalog and trust root	locally verified	Production trust ceremony
REL-004	IPFS publication and CAR recovery	locally verified	Independent replicas
REL-005	Modular component update (stage, activate, health, rollback)	locally verified	Isolated local root only; no live deployment
OPS-001	Server bootstrap and Smartnode deployment	partial	No-GitHub bootstrap
OPS-002	Operator access security	partial	Key migration and rotation not executed
OPS-003	Monitoring and health	partial	Component-specific health checks exist only for the explorer update path
OPS-004	Hosted CI and verification governance	partial	Hosted jobs blocked by billing
DOC-001	Developer Hub and documentation accuracy	partial	Public endpoints; SDK examples
VER-001	Full-coin and unified integration acceptance	partial	Independent operators; sustained TPS
AI-014	Development bounties	partial	Wallet earnings view; independent contract audit
AI-015	AI P2P overlay and NAT traversal	partial	Real external relay, PCP, Windows home provider
AI-016	Streamed inference billing	partial	Phantom-token fix regression not executed
AI-017	Permissionless model publication	partial	publication to IPFS verified; registry calldata generated but not executed on a chain; wallet Publish UI open

ID	Feature	Recorded status	Gap / acceptance limit
AI-018	Scoped wallet authorization (one signature, bounded automation)	partial	library and gateway verified; wallet consent UI and session-key custody in the wallet open
AI-019	Pool contribution accounting	partial	signed receipt chain, exact allocation and exactly-once intents verified; no pooled on-chain settlement contract
AI-020	AI gateway (OpenAI-compatible)	locally verified	non-streaming chat + models only; streaming, embeddings, SDKs and MCP / A2A adapters open
AI-021	Wallet AI Hub	design only	not implemented
QAI-001	ML-DSA-65 authorized QAI transfer via untrusted relayer	locally verified	Multi-process node run; independent review
QAI-002	Quantum Smart Accounts: nonces, chain / genesis binding, expiry, rotation, SLH-DSA recovery	locally verified	A contested recovery (both keys in hostile hands) ends in a freeze, not a resolution; no guardian set
QAI-003	Multi-party post-quantum mint authority with reserve cap and limits	locally verified	Reserve attestations are signed claims, not proof of funds; testnet reserves simulated
QAI-004	Redemption lifecycle (request, settle / burn, reject, SLA reclaim)	locally verified	No off-chain payment rail; EMT legal and operational arrangements absent
QAI-005	PQ wallet key management (keystores, sealed backup, destruction)	locally verified	Historical wallet-UI gap needs revision-specific reconciliation; hardware-backed custody remains unverified (E18 / E25).
QAI-006	pq-verify / v1 native verification precompiles	partial	Historical NOT ACTIVATED label superseded by pq-verify / v1 at 20777 genesis. Independent cryptographic / system assurance remains open (E02 / E18).
QAI-007	QAI reserve accounting: 120% target, 110% emergency, 130% surplus; treasury-financed margin; independent attestation	locally verified	Deployment recorded 14 November (E20), superseding not-deployed text. Reserves remain simulated; custody, payments and independent review remain open.
QL-001	Quantum Controller (finalized-state reader, auditable decisions, safe fallback)	implemented untested	Unit tests only; acceptance scenario J (stop controller while real nodes finalize) NOT TESTED
QL-002	Economic Optimizer in shadow mode	implemented untested	Unit tests only; no real-node run yet
QL-003	Predictive compute routing over the deterministic baseline	implemented untested	Unit tests only; no real-node run yet
QL-004	Risk engine NORMAL / GUARDED / SAFE	implemented untested	Unit tests only; no real-node run yet
QL-005	PASE scheduling advisor (hints only)	partial	Hints not consumed by the Go PASE engine; equivalence proven only for the Python reference model

ID	Feature	Recorded status	Gap / acceptance limit
QL-006	Restricted agent orchestrator	implemented untested	Unit tests; dispatch exercised by the QL-007 Network Monitor against live 20777 (CR-0072)
QL-007	Internal agent runtime + Network Monitor agent (local small LLM chooser, deterministic guard)	locally verified	Network Monitor role only; real-model + live 20777 read-only run (audit / evidence / 2026-11-18-internal-agents-cr00...
QL-008	Security agent (log / drift watching, daily self-test of own endpoints, capped self-expiring nftables blocks)	locally verified	Parsers calibrated on seed1 logs and one live self-test from seed1 (audit / evidence / 2026-11-19-security-agent-cr0...

Appendix D

Open technical and publication questions

The companion `evidence/unresolved_claims.md` gives the detailed resolution queue. The highest-priority technical questions are selected-Core WASM resource pricing and activation, dynamic-set finality/recovery assurance, independent operator/custody separation, useful AI-work attribution and sustainable deployment evidence. The principal publication blockers are independent assurance, current remote identity/bytecode checks, redistribution/source-offer clearance and owner decisions on official publication status.

The whitepaper build is intentionally distinct from readiness. Successful compilation proves that the artifact can be produced; it cannot close a consensus finding, create reserve assets, approve a deployment or publish the paper. Any official edition must be reviewed against the then-current source and network, including superseding deployments and open security issues.

References

[1] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. URL: https://bitcoin.org/bitcoin.pdf (visited on 11/17/2026).

[2] Vitalik Buterin and Virgil Griffith. *Casper the Friendly Finality Gadget*. 2017. arXiv: 1710.09437. URL: https://arxiv.org/abs/1710.09437 (visited on 11/17/2026).

[3] Gavin Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger. Shanghai version efc5f9a, 4 February 2025*. Feb. 4, 2025. URL: https://ethereum.github.io/yellowpaper/paper.pdf (visited on 11/17/2026).

[4] Rati Gelashvili, Alexander Spiegelman, Zhuolun Xiang, George Danezis, Zekun Li, Dahlia Malkhi, Yu Xia, and Runtian Zhou. *Block-STM: Scaling Blockchain Execution by Turning Ordering Curse to a Performance Blessing*. 2022. DOI: 10.48550/arXiv.2203.06871. URL: https://arxiv.org/abs/2203.06871 (visited on 11/17/2026).

[5] *Petals: Collaborative Inference and Fine-tuning of Large Models*. 2022. arXiv: 2209.01188. URL: https://arxiv.org/abs/2209.01188 (visited on 11/17/2026).

[6] IPFS Documentation. *Persistence, Permanence, and Pinning*. URL: https://docs.ipfs.tech/concepts/persistence/ (visited on 11/17/2026).

[7] National Institute of Standards and Technology. *Module-Lattice-Based Digital Signature Standard*. FIPS 204. Aug. 13, 2024. DOI: 10.6028/NIST.FIPS.204. URL: https://csrc.nist.gov/pubs/fips/204/final (visited on 11/17/2026).

[8] National Institute of Standards and Technology. *Stateless Hash-Based Digital Signature Standard*. FIPS 205. Aug. 13, 2024. DOI: 10.6028/NIST.FIPS.205. URL: https://csrc.nist.gov/pubs/fips/205/final (visited on 11/17/2026).

[9] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. FIPS 203. Aug. 13, 2024. DOI: 10.6028/NIST.FIPS.203. URL: https://csrc.nist.gov/pubs/fips/203/final (visited on 11/17/2026).

[10] *Regulation (EU) 2023/1114 on Markets in Crypto-assets*. European Parliament and Council of the European Union, May 31, 2023. URL: https://eur-lex.europa.eu/eli/reg/2023/1114/oj/eng (visited on 11/17/2026).

[11] European Banking Authority. *Application of PSD2 Provisions to the Issuance and Redemption of Electronic Money Tokens. Single Rulebook Q&A 2025_7457*. URL: https://eba.europa.eu/single-rule-book-qa/qna/view/publicId/2025_7457 (visited on 11/17/2026).

[12] Piero Cipollone. *Stablecoins and Monetary Sovereignty*. European Central Bank. Oct. 18, 2025. URL: https://www.ecb.europa.eu/press/key/date/2025/html/ecb.sp251018~5280b1c98b.en.pdf (visited on 11/17/2026).

[13] The Update Framework. *The Update Framework Specification*. URL: https://theupdateframework.github.io/specification/latest/ (visited on 11/17/2026).