CAPITAL - TECHNOLOGY - INTELLIGENCE

# WINMAR CHAIN

## A Sovereign Layer-1 Settlement Protocol

**TECHNICAL WHITEPAPER / VERSION 3.0 INSTITUTIONAL EDITION / 2026**

Prepared by Winmar Chain Team [ SYF ]

| NETWORK | CONSENSUS | IDENTITY |
| --- | --- | --- |
| Winmar Chain Mainnet | QBFT Proof of Authority | Chain ID 12142816 (0xb948e0) |

This institutional edition preserves the analytical depth of the original paper while aligning every network claim with the operating Winmar Chain mainnet. Live genesis configuration and released client software remain the executable source of truth.

# Abstract

Winmar Chain is a sovereign, EVM-compatible Layer-1 blockchain designed for programmable settlement, tokenized real-world assets, verifiable records, and institutional digital infrastructure. The live mainnet uses Hyperledger Besu with QBFT Proof of Authority consensus. Four authorized validators participate in block production and a separate public RPC node provides application access.

QBFT replaces the earlier draft consensus design. A proposed block is finalized after the protocol obtains a super-majority of validator commit messages. The current genesis configuration uses a three-second minimum block period, a 10-second initial request timeout, a 30,000-block epoch, a 30,000,000 gas limit, London and Berlin rules active from genesis, and zero base fee.

The native coin is Winmar Coin (WMC), with 18 decimals and 50,000,000 WMC allocated at genesis. This paper specifies network identity, consensus safety, validator governance, EVM execution, gas accounting, RPC access, token interfaces, interoperability assumptions, and operational security. Commercial asset backing, valuation, investment returns, and regulated-service claims are outside the protocol specification.

## Document status

Version 3.0 is the institutional edition aligned with the operating Winmar Chain mainnet. It replaces all legacy naming, identifiers, Clique assumptions, and retired endpoints while restoring the original paper's explanatory depth and extending the formal treatment of QBFT.

# 1. Purpose and engineering objectives

- Sovereign execution: network rules, validator admission, upgrades, and state are governed within the Winmar Chain protocol boundary.
- Deterministic finality: committed QBFT blocks do not depend on probabilistic mining depth under the stated validator assumptions.
- EVM compatibility: Solidity bytecode, Ethereum-style accounts, ABI encoding, JSON-RPC, wallets, and common tooling remain usable.
- Operational accountability: block production is limited to approved validator identities with explicit governance responsibility.
- Transparent verification: blocks, receipts, balances, bytecode, and event logs are publicly queryable through RPC and the explorer.
- Crypto agility: the roadmap evaluates post-quantum migration options without claiming that current secp256k1 signatures are quantum resistant.

# 2. Live network identity

| Parameter | Live mainnet value |
|---|---|
| Network name | Winmar Chain Mainnet |
| Network type | Sovereign EVM-compatible Layer 1 |
| Native coin | Winmar Coin (WMC), 18 decimals |
| Chain ID | 12142816 decimal / 0xb948e0 hexadecimal |
| Client | Hyperledger Besu |
| Consensus | QBFT Proof of Authority |
| Validator set | 4 authorized validators at launch |
| Block period | 3 seconds minimum |
| Request timeout | 10 seconds initial round timeout |
| Epoch length | 30,000 blocks |
| Block gas limit | 30,000,000 gas (0x1c9c380) |
| Fee mode | Zero base fee; minimum gas price configured as zero |
| RPC | https://rpc.winmarchain.org |
| Explorer | https://scan.winmarchain.org |
| Website | https://winmarchain.org |

The chain ID is a signing-domain identifier. Applications must verify eth_chainId before requesting signatures. It is not a security rating or a substitute for contract-level domain separation.

# 3. System architecture

## 3.1 Peer-to-peer layer

Validator and full nodes exchange signed transactions, proposals, prepares, commits, round-change messages, and finalized blocks over the network. Production topology disables open discovery and uses controlled peer relationships.

## 3.2 QBFT consensus layer

Authorized validators rotate proposal responsibility. Validators independently verify the proposed header and transaction execution, then participate in Byzantine fault-tolerant voting rounds.

## 3.3 EVM execution layer

Every full node deterministically applies the ordered transaction list to the parent world state. Execution consumes gas, may modify balances and storage, may emit logs, and either succeeds or reverts.

## 3.4 State and commitment layer

Each block commits to world state, transactions, and receipts. Hash-linked headers make tampering evident, while QBFT authorization determines which valid history becomes final.

## 3.5 Access and observability layer

A separate public RPC node serves wallets and applications. Blockscout indexes the chain independently. Metrics and node-health timers monitor the validator fleet without exposing validator signing interfaces publicly.

```
User/Application -> Wallet signature -> Public RPC -> Transaction pool -> QBFT proposal ->
Prepare/Commit quorum -> Final block -> EVM state + receipt + logs
```

# 4. QBFT consensus and algebra

## 4.1 Validator fault model

Let N be the active validator count and f the maximum Byzantine validators the network is designed to tolerate. Standard QBFT deployment sizing follows:

```
N >= 3f + 1 f_max = floor((N - 1) / 3)
```

For the launch set N = 4, f_max = floor(3/3) = 1. The network is designed to preserve safety with at most one Byzantine validator, assuming the protocol, cryptography, and network conditions satisfy the model. If more than one-third of validators stop participating, block production can stall.

## 4.2 Quorum

A block requires a super-majority of validator commits. Define the quorum:

```
Q(N) = ceil(2N / 3) For N = 4: Q = ceil(8/3) = 3 validator commits
```

Equivalent implementations may express the threshold as 2f + 1 for a set sized N = 3f + 1. With four validators and f = 1, both formulations require three commits.

## 4.3 Round progression

```
round = 0, 1, 2, ... proposer(round,height) = rotation(validators, height, round)
timeout(round) = T0 * 2^round (implementation policy; T0 = 10 seconds in genesis)
```

If a proposer is unavailable or the required messages do not arrive before the timeout, validators issue round-change messages and move to a later round. The increasing timeout allows the protocol to recover when network latency temporarily exceeds the previous round budget.

## 4.4 Finality condition

```
Final(B,h) iff ValidHeader(B) AND Execute(parentState, B.transactions).stateRoot = B.stateRoot
AND |CommitSignatures(B,h)| >= Q(N) AND every signer is in ValidatorSet(h)
```

A valid commit quorum finalizes a block at its height. Applications still need independent RPC checks, receipt-status validation, and operational controls; BFT finality does not protect against compromised application keys, malicious contracts, or a governance-authorized protocol change.

## 4.5 Safety and liveness separation

Safety means two honest nodes do not finalize conflicting blocks at the same height. Liveness means the system continues finalizing new blocks. QBFT is designed so that quorum intersection protects safety within the fault bound, while liveness requires at least Q(N) participating validators and sufficient network communication.

```
For any two quorums A and B with |A|, |B| >= Q: |A intersection B| >= 2Q - N For N=4 and Q=3:
|A intersection B| >= 2
```

Because the intersection contains at least two validators in the four-validator launch set, it must contain at least one honest validator when f <= 1. An honest validator does not validly commit conflicting proposals for the same sequence and round context.

# 5. Blocks, state, and execution algebra

## 5.1 World state

```
Let sigma map each 20-byte address a to an account: sigma[a] = (nonce, balance, storageRoot, codeHash) sigma_(i+1) = Upsilon(sigma_i, T_i) sigma_block = Upsilon(...Upsilon(sigma_parent,T_0)...,T_(n-1))
```

## 5.2 Transaction validity

```
ValidTx(T,sigma) iff validSignature(T) AND T.chainId = 12142816 AND T.nonce = sigma[sender].nonce AND T.gasLimit >= intrinsicGas(T) AND sigma[sender].balance >= T.value + maximumFeeExposure(T)
```

## 5.3 Block validity

```
ValidBlock(B,P) iff B.parentHash = Keccak256(Encode(P.header)) AND B.number = P.number + 1 AND AuthorizedQBFTProposal(B) AND CommitQuorum(B) >= Q(N) AND all transactions are valid AND sum(gasUsed_i) <= B.gasLimit AND computed stateRoot, transactionsRoot and receiptsRoot match B
```

## 5.4 Authenticated commitments

A block header commits to the post-execution state root, ordered transaction root, and receipt root. Except for a cryptographic collision, any altered committed leaf changes its path hashes and corresponding root.

# 6. Gas, fees, and throughput bounds

## 6.1 Intrinsic gas

```
G_intrinsic = G_tx + n_zero * G_datazero + n_nonzero * G_datanonzero + I_create * G_txcreate + n_access_address * G_accessaddress + n_access_key * G_accessstorage
```

Constants depend on the active EVM fork rules. Winmar Chain activates Berlin and London rules from genesis and operates with zero base fee; developers must still supply a valid gas envelope and should estimate gas before submission.

## 6.2 Fee accounting

```
gasUsed = gasLimit - gasRemaining - eligibleRefund transactionFee = gasUsed * effectiveGasPrice
```

A zero-base-fee network can produce zero-fee transactions under current node policy. Governance may change fee parameters in a future coordinated release; applications must not hard-code a permanent zero fee assumption.

## 6.3 Capacity model

```
TPS_approx = (L_gas * u) / (g_avg * tau_block) L_gas = 30,000,000; tau_block approximately 3 seconds
```

For 21,000-gas native transfers and full gas utilization, the mathematical ceiling is floor(30,000,000 / 21,000) = 1,428 transfers per block, or approximately 476 transfers per second at three seconds per block. This is not a measured sustainable throughput claim. Contract complexity, propagation, database work, transaction mix, and validator hardware reduce actual capacity.

# 7. WMC native coin and token interfaces

Winmar Coin (WMC) is the protocol-native unit used for account balances, transaction value, and gas accounting. It has 18 decimals. The genesis state allocates a total of 50,000,000 WMC across two governance-controlled addresses.

WMC is not an ERC-20 contract. A WMC-20 token is a separate smart-contract asset that follows the ERC-20-compatible method and event interface on Winmar Chain. Token issuers define supply, minting, burning, pausing, upgrade, access-control, and compliance logic.

```
For a basic WMC-20 transfer in contract C: B'_C[s] = B_C[s] - x B'_C[r] = B_C[r] + x Sum_a
B'_C[a] = Sum_a B_C[a] (ordinary transfer; no mint/burn)
```

# 8. WINT settlement-token roadmap

Winmar Integrated Network Token (WINT) is a planned 1:1 USD-denominated settlement token. Its proposed uses include cross-border institutional, banking, and corporate settlement with a target of programmable T+0 processing. A target issuance capacity of up to 2.5 billion WINT and a proposed reserve reference to a Citi-issued bond are business and legal design statements, not properties enforced by QBFT consensus.

Before issuance, independent documentation must establish reserve ownership, custody, valuation, liquidity, legal enforceability, redemption rights, bankruptcy remoteness, sanctions and AML controls, attestations, and the precise relationship between outstanding WINT and eligible reserve value. Mention of Citi or any security does not imply endorsement, partnership, guarantee, or validation by that institution.

```
A simple collateralization disclosure metric may be defined as: CR(t) =
EligibleReserveValue_USD(t) / WINT_outstanding(t) A 1:1 fully backed target requires CR(t) >=
1 under the published valuation and eligibility policy.
```

The formula is an audit metric, not a proof of solvency. Oracle quality, settlement timing, market value, encumbrances, custody access, and redemption obligations remain off-chain risks.

# 9. Interoperability and bridges

EVM compatibility does not itself move assets between chains. Wrapped WMC and WINT require bridge or messaging infrastructure with separate contracts, operators or validator sets, custody rules, proofs, rate limits, and emergency controls.

```
For a lock-and-mint bridge under idealized accounting: locked_source(t) >=
wrapped_destination(t) netMint = verifiedDeposits - verifiedWithdrawals No bridge should claim
this invariant without independent monitoring and contract verification.
```

- Verify source and destination finality before minting or releasing assets.
- Separate bridge administration from validator key custody.
- Publish contract addresses, upgrade permissions, limits, audits, and incident procedures.
- Treat every wrapped representation as a distinct asset dependent on its redemption mechanism.

# 10. Network roles and current topology

| Role | Current function | Security boundary |
| --- | --- | --- |
| 4 validator nodes | Propose, prepare, commit, and finalize QBFT blocks | Signing keys and validator authorization |
| Public RPC node | HTTPS JSON-RPC for wallets and applications | Rate limits, allowed methods, availability |
| Block explorer | Independent indexing and public inspection | Indexer freshness and API integrity |
| Monitoring | Health timers, metrics, peer and block-height checks | Alert delivery and operator response |

The four-validator launch topology tolerates one Byzantine validator in the standard N = 3f + 1 model. Losing two validators prevents the three-commit quorum and stalls finalization. Expansion should prioritize organizational, cloud, regional, and administrative independence rather than merely increasing server count.

# 11. JSON-RPC and developer connection

Public RPC endpoint: https://rpc.winmarchain.org
Explorer: https://scan.winmarchain.org

```
await window.ethereum.request({ method: 'wallet_addEthereumChain', params: [{ chainId: '0xb948e0', chainName: 'Winmar Chain Mainnet', nativeCurrency: { name: 'Winmar Coin', symbol: 'WMC', decimals: 18 }, rpcUrls: ['https://rpc.winmarchain.org'], blockExplorerUrls: ['https://scan.winmarchain.org'] }] });
```

Critical systems should compare independent RPC providers, validate eth_chainId, verify receipt.status, require the expected contract address and event topics, and monitor the latest block height. Private keys must never be embedded in browser code, repositories, or public configuration.

# 12. Security, governance, and crypto agility

## 12.1 Principal risks

- Validator-key compromise, collusion, common administration, or correlated infrastructure failure.
- Consensus-client defects, unsafe upgrades, incompatible genesis configuration, or governance error.
- Compromised RPC, explorer, DNS, wallet, bridge, oracle, or software-supply-chain dependencies.
- Smart-contract vulnerabilities, privileged upgrade keys, incorrect token economics, or misleading off-chain data.
- Censorship and liveness failure when fewer than three of four validators participate.

## 12.2 Governance controls

- Multisignature approval for sensitive treasury and administrative actions.
- Documented validator admission, removal, rotation, and emergency procedures.
- Testnet rehearsal, version pinning, coordinated upgrades, rollback planning, and public release notes.
- Independent backups, monitoring, incident response, and periodic security assessment.

## 12.3 Post-quantum readiness

The current EVM account model uses secp256k1 signatures and Keccak-256 hashing. This version does not claim post-quantum security. The roadmap instead adopts crypto agility: inventory signature dependencies, follow standardized post-quantum algorithms, test hybrid authorization schemes, design address/account migration, and publish a reviewed activation plan before any consensus change.

# 13. Availability and storage models

```
If validator i has independent availability p_i, the probability that at least q of N validators are online is: P(X >= q) = Sum_(k=q)^N C(N,k) p^k (1-p)^(N-k) For current QBFT finalization: N=4 and q=3.
```

Independence is a strong assumption. Shared providers, regions, administrators, software versions, power sources, or networking create correlated failures. Real availability must be measured from production data.

```
A simplified full-node storage estimate is: S(t) approximately S0 + B(t) * (b_avg + r_state) B(t) approximately t / tau_block
```

Database amplification, snapshots, indexes, logs, Bonsai storage behavior, pruning, and archive history must be measured separately.

# 14. Limitations and disclosures

Winmar Chain is an authorized-validator network. Users rely on validator integrity, governance discipline, client correctness, infrastructure resilience, and application security. Deterministic QBFT finality does not guarantee truthfulness of off-chain data, smart-contract safety, bridge solvency, stablecoin redemption, wallet security, or regulatory compliance.

This paper is a technical and educational resource. It is not financial, investment, legal, accounting, or tax advice; it does not offer securities or promise asset value, return, liquidity, availability, regulatory status, or fitness for a particular purpose. Network parameters and software may change through published governance procedures.

Third-party names and software are cited for technical context only. Their inclusion does not imply endorsement, partnership, custody, audit, sponsorship, or guarantee.

# 15. Technical references

1. Hyperledger Besu Documentation, 'Configure QBFT consensus.' https://docs.besu-eth.org/private-networks/how-to/configure/consensus/qbft
2. ConsenSys, 'QBFT Formal Specification and Verification.' https://github.com/Consensys-Incorporated/qbft-formal-spec-and-verification
3. Hyperledger Besu Documentation, 'Proof of authority consensus.' https://docs.besu-eth.org/private-networks/concepts/poa
4. Gavin Wood, 'Ethereum: A Secure Decentralised Generalised Transaction Ledger.' https://ethereum.github.io/yellowpaper/paper.pdf
5. Ethereum Execution Specifications. https://github.com/ethereum/execution-specs
6. EIP-155, 'Simple replay attack protection.' https://eips.ethereum.org/EIPS/eip-155
7. EIP-695, 'Create eth_chainId method for JSON-RPC.' https://eips.ethereum.org/EIPS/eip-695
8. EIP-1559, 'Fee market change for ETH 1.0 chain.' https://eips.ethereum.org/EIPS/eip-1559
9. EIP-20, 'Token Standard.' https://eips.ethereum.org/EIPS/eip-20
10. Solidity Documentation, ABI specification and storage layout. https://docs.soliditylang.org
11. Ethereum JSON-RPC API. https://ethereum.org/developers/apis/json-rpc/
12. Lamport, Shostak, and Pease, 'The Byzantine Generals Problem,' ACM TOPLAS 4(3), 1982. https://doi.org/10.1145/357172.357176
13. Castro and Liskov, 'Practical Byzantine Fault Tolerance,' OSDI 1999. https://www.usenix.org/conference/osdi-99/presentation/practical-byzantine-fault-tolerance

# 16. Document integrity and change log

Official copies should publish a version, publication date, and SHA-256 file hash. Readers should obtain the document from an official Winmar Chain channel and compare the hash before relying on it.

| Version | Status | Material changes |
|---|---|---|
| 1.1 | Superseded | Legacy draft; identifiers and consensus model retired |
| 2.0 | Superseded | Condensed mainnet-aligned technical edition |
| 2.1 | Superseded | Mainnet-aligned complete edition |
| 3.0 | Current | Institutional edition: restored depth, expanded QBFT proofs, failure analysis, settlement model, and technical references |

# 17. Validator lifecycle and governance

## 17.1 Admission

- A candidate operator discloses its organization, infrastructure provider, region, network path, key-custody design, monitoring, backups, and incident contacts.
- Governance evaluates independence from existing validators, operational capability, jurisdictional exposure, and correlated-failure risk.
- The candidate synchronizes a non-validating node and demonstrates stable peers, reproducible deployment, health monitoring, backup restoration, and escalation readiness.
- Admission is complete only after the validator-set change is finalized on-chain and independently verified from multiple RPC observations.

## 17.2 Rotation and removal

Validator keys should be rotated after suspected exposure, operator change, custody-policy change, or scheduled lifecycle expiry. Emergency removal must preserve enough available validators to reach the post-change quorum.

```
N_after = N_before - r available_after >= Q(N_after) Q(N) = ceil(2N / 3)
```

## 17.3 Expansion policy

Adding machines under one administrator does not create equivalent Byzantine independence. Expansion should diversify operator, jurisdiction, cloud, region, power, network path, software release process, custody system, and human escalation path.

# 18. Key management and signing controls

| Key class | Purpose | Recommended boundary |
|---|---|---|
| Validator signing | QBFT identity | Dedicated validator host or controlled signer |
| Treasury signer | WMC and governed asset movement | Multisignature with separated operators |
| Contract administrator | Upgrade, pause, role control | Timelock and multisignature |
| Deployment key | Contract publication | Short-lived account with minimal balance |
| Monitoring credential | Read-only telemetry | No signing authority |

## 18.1 Separation of duties

No single credential should control consensus membership, treasury funds, bridge minting, and upgrade authority. A compromise domain is the set of actions available after one credential, operator account, or infrastructure account is lost.

```
Exposure(k) = Sum_j Impact(action_j) * I[key k authorizes action_j] Objective: minimize max_k Exposure(k)
```

## 18.2 Backup and recovery

- Encrypt backups, verify integrity, and test restoration without exposing private material.
- Maintain an offline recovery procedure with named custodians and documented authorization thresholds.
- Never transmit private keys through chat, email, repositories, tickets, logs, or monitoring output.
- Treat recovery tests as controlled security events with an auditable record.

# 19. Monitoring, service levels, and incidents

| Signal | Purpose | Example alert condition |
|---|---|---|
| RPC chain ID | Detect wrong-network routing | Value differs from 12142816 |
| Block progress | Detect consensus stall | No increase across several block periods |
| Peer count | Detect isolation | Below the operating baseline |
| Explorer lag | Detect indexer degradation | Indexed height materially behind RPC |
| Resources | Prevent exhaustion | Sustained disk or memory threshold |
| Service state | Detect process failure | Unexpected failed state |

```
lag_blocks(t) = max(0, h_rpc(t) - h_explorer(t)) lag_seconds(t) approximately lag_blocks(t) *
tau_block progress_rate = (h_rpc(t2) - h_rpc(t1)) / (t2 - t1)
```

Reachability alone is not health. Health requires correct identity, advancing height, sufficient peers, acceptable latency, fresh indexing, valid TLS, and successful service checks.

## 19.1 Incident levels

- Informational: transient latency or one failed probe without user impact.
- Degraded: explorer lag, reduced redundancy, or repeated RPC errors while finalization continues.
- Critical: block stall, chain-ID mismatch, quorum loss, key compromise, conflicting finalized observations, or unauthorized validator-set change.
- Recovery: root cause documented and monitoring confirms sustained restoration.

# 20. Smart-contract production standard

## 20.1 Pre-deployment

- Pin compiler and dependency versions; preserve source, settings, constructor arguments, ABI, bytecode, and deployment metadata.
- Review upgradeability, pausing, minting, burning, access control, fee logic, oracle dependencies, and external calls.
- Run unit, integration, invariant, access-control, and failure-path tests away from mainnet.
- Define operational ownership and emergency authority before funding or user interaction.

## 20.2 Bytecode verification

```
runtime_expected = Compile(source, compiler, settings).runtime verified iff
Normalize(runtime_onchain) = Normalize(runtime_expected)
```

Verification demonstrates correspondence between published source and deployed bytecode. It does not prove security, economic soundness, legal compliance, reserve quality, or appropriate governance.

## 20.3 Release record

Every production contract record should include chain ID, address, transaction hash, block number, deployer, source commit, compiler, license, ABI, proxy implementation, administrator roles, multisignature, timelock, audits, and deprecation status.

# 21. Interoperability threat model

A bridge introduces at least two consensus domains plus a verification or custody domain. A wrapped asset is bounded by the weakest system involved in minting, custody, messaging, governance, or redemption.

```
Security_bridge <= min(Security_source, Security_destination, Security_verifier,
Security_custody, Security_admin)
```

## 21.1 Accounting invariants

```
wrapped_d(t) <= finalized_locked_source_d(t) - released_source_d(t) Sum_d wrapped_d(t) <=
eligible_locked_source_total(t)
```

The accounting set must exclude reverted, unfinalized, duplicated, frozen, or ineligible deposits.

## 21.2 Minimum controls

- Independent finality verification and replay protection for every message.
- Per-asset rate limits, emergency pause, and delayed high-value withdrawals.
- Separated upgrade, pause, mint, custody, and validator authorities.
- Continuous reconciliation of locked and wrapped supply with public incident procedures.
- Audited contracts and explicit disclosure of every external trust assumption.

# 22. WINT issuance control framework

WINT remains a planned product. The blockchain can enforce contract logic, but it cannot independently prove title to a reserve asset, legal enforceability, market value, liquidity, sanctions status, or redemption capability.

## 22.1 Required evidence before issuance

- Executed legal documentation establishing issuer, reserve owner, custodian, account control, and beneficiary rights.
- Independent confirmation of eligible reserve assets, valuation source, encumbrances, liquidity, and maturity profile.
- Published issuance, redemption, freeze, recovery, sanctions, AML, and dispute procedures.
- On-chain supply controls connected to independently reviewed operational approval and reconciliation.
- Periodic attestations and clear disclosure of exceptions, delays, shortfalls, and material counterparties.

## 22.2 Reconciliation algebra

```
CR(t) = EligibleReserveValue_USD(t) / WINT_outstanding(t) ReserveSurplus(t) = EligibleReserveValue_USD(t) - WINT_outstanding(t) IssuanceAllowed only if projected CR(t) >= 1 under the published policy
```

These are disclosure and control metrics, not mathematical proof of solvency or a guarantee of redemption.

# 23. Mainnet implementation checklist

| Domain | Minimum production verification |
|---|---|
| Identity | Chain ID 12142816, genesis hash, client version, validator set |
| Consensus | Four validators, quorum three, advancing height, round-change tested |
| RPC | TLS, correct chain ID, unsafe methods disabled, rate limits active |
| Explorer | Indexed height near RPC, transactions and receipts visible |
| Keys | Separated custody, backups tested, multisignature documented |
| Contracts | Source verified, roles published, tests and review complete |
| Assets | Supply, custody, redemption, reserve, and legal claims validated |
| Operations | Alerts, on-call, backups, patching, and post-incident review |

A checklist item is complete only when evidence exists, an accountable owner is named, and the evidence can be independently reproduced. Verbal confirmation or a single dashboard indicator is insufficient for high-impact controls.

# 24. QBFT message and certificate model

## 24.1 Consensus instance

A consensus instance is identified by chain domain d, block height h, and round r. Let V_h be the authorized validator set at height h and let H(B) be the Keccak-256 digest of the proposed block header. A signed vote is valid only when its recovered signer belongs to V_h and its domain, height, round, and digest match the active instance.

```
Instance = (d, h, r) Proposal = (Instance, H(B), proposerSignature) Prepare_i = Sign_i(d || h || r || H(B) || PREPARE) Commit_i = Sign_i(d || h || r || H(B) || COMMIT)
```

## 24.2 Commit certificate

```
Cert(B,h,r) = { Commit_i : i in V_h and Verify(Commit_i)=true } Final(B,h) only if |Cert(B,h,r)| >= Q(|V_h|)
```

Domain separation prevents a vote created for another chain, height, round, phase, or block digest from being accepted as support for the current proposal. Implementations must follow the exact Besu QBFT encoding and validation rules; the notation here is explanatory.

# 25. Safety proof sketch and failure boundaries

## 25.1 Quorum-intersection argument

```
Assume N = 3f + 1 and Q = 2f + 1. For any two commit quorums A and B: |A intersection B| >= |A| + |B| - N >= 2(2f + 1) - (3f + 1) = f + 1
```

At most f validators are Byzantine, so the intersection contains at least one honest validator. If honest validators obey the locking and voting rules, they cannot support two conflicting commit certificates in the same consensus context. Therefore two conflicting finalized blocks cannot both be produced while the fault bound and protocol assumptions hold.

## 25.2 What the proof does not cover

- More than f Byzantine or compromised validators.
- A client defect shared by enough validators.
- Incorrect or inconsistent validator-set transitions.
- Compromised signing keys used outside the intended protocol state.
- Application, bridge, oracle, custody, legal, or governance failures.

# 26. Liveness, timing, and availability algebra

Safety and liveness are distinct. Safety constrains conflicting finality; liveness requires timely communication among enough correct validators. If the initial request timeout is T0 and the implementation doubles the timeout after each failed round, the cumulative waiting budget through round r is:

```
T_round(r) = T0 * 2^r T_cumulative(r) = Sum_(j=0)^r T0 * 2^j = T0 * (2^(r+1) - 1)
```

This backoff does not guarantee progress during an indefinite partition. It allows the system to accommodate temporary delay after the network becomes sufficiently synchronous and at least Q(N) validators can exchange valid messages.

## 26.1 Four-validator availability model

```
Under the simplifying assumption that each validator is independently online with probability p, a four-validator network reaches the three-validator quorum with probability: P_live = C(4,3)p^3(1-p) + C(4,4)p^4 = 4p^3(1-p) + p^4
```

Independence is rarely exact. Shared cloud providers, regions, software versions, operators, DNS, network paths, and key-management systems create correlated failures. Operational planning must model common-cause events separately.

# 27. Institutional settlement control model

A blockchain finalizes state transitions; it does not by itself establish legal title, beneficial ownership, reserve adequacy, regulatory eligibility, or cash-account finality. Institutional settlement therefore requires an explicit correspondence between on-chain state and the governing off-chain records.

```
SettlementComplete(x,t) iff FinalizedOnChain(x,t) AND AuthorizedParticipants(x) AND
ComplianceChecks(x,t) AND AssetControlConfirmed(x,t) AND ReconciliationMatched(x,t)
```

## 27.1 Control objectives

- Bind each legal instrument and account to an authoritative identifier and governing documentation.
- Separate transaction initiation, approval, custody, issuance, and reconciliation roles.
- Require deterministic idempotency keys so an instruction cannot be executed twice.
- Record exceptions and reversals as new auditable state transitions rather than rewriting history.
- Define the point at which technical finality, accounting recognition, and legal settlement each occur.

# 28. Glossary and notation

| Term | Meaning |
| --- | --- |
| BFT | Agreement despite a bounded number of faulty or malicious participants |
| QBFT | Quorum Byzantine Fault Tolerance consensus used by Winmar Chain |
| PoA | Proof of Authority with an approved validator set |
| EVM | Ethereum Virtual Machine execution environment |
| Finality | A block is committed under the consensus model |
| Q(N) | Minimum validator support for the relevant certificate |
| sigma | World-state mapping from addresses to account records |
| WMC | Winmar Coin, native coin of Winmar Chain |
| WINT | Planned Winmar Integrated Network Token |
| RPC | Interface used by wallets and applications |
| T+0 | Settlement target on transaction date; not guaranteed legal finality |

## WINMAR CHAIN TEAM [ SYF ]

Capital - Technology - Intelligence