Quinvium

WHITEPAPER

Quinvium

A programmable proof-of-work network for GPU miners

Version	Published	Network	Web
1.0	14 November 2026	Mainnet · chain ID 52401	quinvium.com

Key facts

TICKER **QU**	CONSENSUS **Proof-of-Work, Ethash**
EXECUTION **Ethereum Virtual Machine**	BLOCK TIME **15 s target, LWMA difficulty**
BLOCK REWARD **4 QU, halving every 8,000,000 blocks**	SUPPLY CAP **64,000,000 QU, no premine**
FEES **Not burned, all to the miner**	MAINNET CHAIN ID **52401**
LAUNCH **14 November 2026, 00:00 CET (13 November 2026, 23:00 UTC)**	

MAINNET GENESIS HASH
`0x49c889ad9d4f44869cf7f97a6e079a50dbf13974263b1afedec3b774ebeeac39`

Contents

- Abstract
1 Motivation
2 Design goals and non-goals
3 Networks
4 Consensus
5 Execution
6 Monetary policy
7 Built-in contracts
8 Peer-to-peer network
9 Mining
10 Capacity and scaling
11 Software
12 Security
13 Decentralisation, upgrades and governance
14 Roadmap
15 Disclaimer
- References

Abstract

Quinvium (ticker QU) is an independent blockchain that combines proof-of-work consensus with the Ethereum Virtual Machine. Blocks are produced roughly every 15 seconds by miners running the Ethash algorithm on ordinary graphics cards. Smart contracts, wallets and developer tools built for Ethereum work without changes. The network launched without a premine, presale, team allocation or treasury: every QU is created by block rewards, which start at 4 QU, halve every 8,000,000 blocks and stop at a cap of 64 million QU. Transaction fees are not burned and go entirely to the miner of the block.

Quinvium is a decentralised network, not a company. No foundation, firm or person owns it, holds an admin key or can change its rules, freeze funds or reverse transactions. Anyone can run a node, mine, operate a public endpoint or build on it, and rule changes take effect only when node operators and miners choose to run them.

Twelve general-purpose contracts are part of the network from the start and have no owner: a notary, readable names, credentials, supply-chain records, escrow, polls, token and NFT factories, a multisig factory, wrapped QU, multicall and a factory for mining pools that pay out through a contract. A built-in proof-of-work check lets those pools run without trusting an operator.

This paper describes the rules as implemented in the reference node. Where this paper and the code differ, the code defines the network and the paper is wrong.

1. Motivation

Proof-of-work lets anyone with hardware take part in securing a network without asking permission and without first acquiring a large stake. When Ethereum moved to proof-of-stake in 2022, it left behind both a large community of GPU miners and a widely used programming environment. Quinvium brings the two back together in a new network with its own history:

- **Open participation.** Block production is open to anyone with a graphics card. ASICs for Ethash are allowed too; the algorithm is unchanged.
- **Familiar programming model.** The EVM, Solidity and the Ethereum JSON-RPC interface are the most widely known smart-contract stack. Developers have nothing new to learn.
- **Fair distribution.** No coins exist before the first block. Everyone, including the people who wrote the software, gets QU the same way: by mining or by trading with someone who mined.
- **Useful from block one.** Common applications such as timestamping, escrow, voting and token issuance are built into the genesis, so the network is usable without first deploying and trusting third-party contracts.

2. Design goals and non-goals

Goals

1. Simple, verifiable consensus: Ethash proof-of-work, heaviest-chain rule, a per-block difficulty adjustment.
2. Full compatibility with Ethereum tooling at the transaction, RPC and contract level.
3. A fixed, predictable issuance schedule written into the node.
4. Mining rewards that stay attractive as the block subsidy falls, by paying all fees to miners.
5. A small, readable implementation that can be audited and reimplemented.

Non-goals

- Quinvium does not try to replace Ethereum or carry its state or history.
- There is no company, foundation, treasury, on-chain governance or protocol fee.
- There is no staking, slashing or validator set at the consensus layer.
- The base layer is not designed to carry millions of payments per day by itself (see section 10).

3. Networks

	Mainnet	Testnet
Chain ID	52401	52402
Currency	QU	tQU (no value)
Target block time	15 s	15 s
Initial difficulty	2^27	2^22
Minimum difficulty	2^24	2^16
P2P port	30524	30525
JSON-RPC port	8545	8546
Stratum port	3333	3334
Public RPC	https://rpc.quinvium.com	https://rpc1.quinvium.com
Explorer	https://explorer.quinvium.com	https://testnet.quinvium.com
Seed	seed.quinvium.com:30524	seed.quinvium.com:30525

A development network (chain ID 52403) exists for local testing.

4. Consensus

4.1 Block format

A block header is an RLP list of exactly 15 fields: parent hash, miner, state root, transaction root, receipts root, logs bloom, difficulty, number, gas limit, gas used, timestamp, extra data (at most 32 bytes), base fee, mix hash and nonce. Numbers use the shortest encoding; a leading zero is rejected.

- `sealHash = keccak256(rlp(fields 0..12))`
- `hash = keccak256(rlp(fields 0..14))`

A block is `rlp([header, [transactions...]])`. There are no uncle blocks and no withdrawals.

4.2 Header validity

A header `h` with parent `p` is valid when all of the following hold:

1. `h.number = p.number + 1` and `h.timestamp > p.timestamp`.
2. `h.timestamp` is at most 60 seconds in the future of the local clock. A header from the future is rejected without penalty and may become valid later.
3. `h.gasUsed <= h.gasLimit`; the gas limit moves by less than 1/1024 of the parent's limit per block, never below 5,000,000 and never above the current maximum (15,000,000 at launch).
4. `h.baseFee` and `h.difficulty` equal the values computed from the parent (sections 6.3 and 4.4).
5. The Ethash proof is valid (section 4.3).

4.3 Proof-of-work

Quinvium uses Ethash exactly as Ethereum did, including the cache and dataset sizes and the 30,000-block epoch. A header is valid when

```
(mix, result) = hashimoto(sealHash, nonce, number / 30000)
mix == h.mixHash  and  result <= 2^256 / h.difficulty
```

The implementation is checked byte for byte against eight real Ethereum blocks in epochs 1 to 517, both in the JavaScript node and in the Rust GPU miner.

4.4 Difficulty adjustment

Difficulty is recalculated after every block with a linearly weighted moving average (LWMA) over the last 60 blocks and a target time `T = 15` seconds. Recent blocks weigh more than older ones, so the network reacts within minutes when hashrate joins or leaves.

```
nextDifficulty(p):
    if p.number < 6: return initial difficulty
    n = min(p.number, 60)
    for i in 1..n:  st_i = clamp(time(c_i) − time(c_{i−1}), 1, 40∗T)
    weighted = max(sum(st_i ∗ i), n^2 ∗ T / 20)
    d = sum(difficulty(c_i)) ∗ (n + 1) ∗ T / (2 ∗ weighted)
    return max(d, minimum difficulty)
```

Capping a single solve time at 40T limits the effect of false timestamps while letting the network recover in about half a day if 99.9 % of hashrate disappears. Simulations of 352 combinations of manipulated timestamps, even with 100 % of hashrate, did not push average issuance below 15.1 seconds per block.

The mainnet starts at difficulty 2^27, which corresponds to about 9 MH/s at a 15-second block time, so a handful of graphics cards can produce blocks from the first minute.

4.5 Chain selection and finality

The valid chain with the greatest total difficulty wins; on a tie, the node keeps the branch it saw first. A node rejects blocks more than 256 blocks below its head. A reorganisation deeper than 256 blocks (about an hour) is accepted only if the new branch carries at least twice the work of the current branch since the common ancestor, and only after every header of the new branch has been downloaded and its proof-of-work checked. Below a checkpoint published with a node release, history is never rewritten.

This rule protects running nodes against long, cheap rewrites without blocking new nodes, which always follow the heaviest chain on their first sync.

5. Execution

Quinvium runs the EVM from the EthereumJS project with the Shanghai rule set plus EIP-1153 (transient storage), EIP-5656 (MCOPY), EIP-6780 (restricted SELFDESTRUCT), EIP-7823 and EIP-7883 (MODEXP limits and pricing). There are no blobs and no beacon roots. Transaction types 0, 1 and 2 are accepted; type 0 must carry EIP-155 replay protection.

Differences from Ethereum:

Item	Quinvium	Reason
Base fee	Paid to the miner, not burned	Keeps mining viable as the block reward falls
BLAKE2F	3 gas per round instead of 1	Otherwise the slowest operation relative to its price
PREVRANDAO (0x44)	Returns the parent's mix hash	There is no beacon chain; miners can influence it, so it must not be used as a randomness source

Transactions execute in block order on the parent state. After the last transaction the miner is credited with the block reward plus `gasUsed × baseFee`; priority fees are paid per transaction as in Ethereum.

The consensus-relevant dependencies have pinned versions, and the node carries fixed fingerprints of all genesis blocks and consensus outputs in its tests. Upgrading any of them is treated as a potential hard fork.

6. Monetary policy

6.1 Block reward

```
blockReward(n):
    n == 0:              0
    h = (n − 1) / 8,000,000
    h >= 64:             0
    otherwise:           4 QU >> h
```

Era	Blocks	Reward	Issued in era	Total at end	Approx. end
1	1 – 8M	4 QU	32M QU	32M QU	year 3.8
2	8M – 16M	2 QU	16M QU	48M QU	year 7.6
3	16M – 24M	1 QU	8M QU	56M QU	year 11.4
4	24M – 32M	0.5 QU	4M QU	60M QU	year 15.2
5	32M – 40M	0.25 QU	2M QU	62M QU	year 19.0
…	…	…	…	→ 64M QU	

The sum of all rewards is below 64,000,000 QU because of integer rounding. Era lengths assume the 15-second target; real calendar dates depend on the hashrate history.

6.2 Genesis and distribution

The genesis block contains only system contracts with zero balance. There is no premine, no presale, no allocation to developers or early supporters and no treasury. The first QU is created by block 1.

6.3 Fees

Quinvium uses the EIP-1559 fee mechanism with a target of half the gas limit:

```
nextBaseFee(p):
    target = p.gasLimit / 2
    above target: baseFee + max(baseFee * (gasUsed - target) / target / 8, 1)
    below target: baseFee - baseFee * (target - gasUsed) / target / 8
    never below 1 gwei
```

Unlike Ethereum, the base fee is paid to the miner. As block rewards halve, fees become a growing share of mining income, which is intended to keep the network secured by work rather than by subsidy alone.

6.4 Gas limit

The genesis gas limit is 10,000,000. Miners can vote it up or down by less than 1/1024 per block, up to 15,000,000; going higher requires a hard fork. The limit is based on measurement: the worst known block validates in about 6 seconds on an ordinary desktop processor, well inside the 15-second block time.

7. Built-in contracts

The genesis contains eleven system contracts and the `pools` rule change at block 1 adds a twelfth. None has an owner, an admin key or an upgrade path; they can only change by a hard fork.

Address	Contract	Purpose
0xae…01	Notary	Record the hash of a document with the block time
0xae…02	Names	Readable `.qu` names that point to addresses
0xae…03	Credentials	Issue and revoke credentials
0xae…04	Supply chain	Step-by-step history of a shipment
0xae…05	Escrow	Payment released when a condition is met, with an optional arbiter
0xae…06	Polls	On-chain polls and votes
0xae…07	Token factory	Create an ERC-20 token without writing code
0xae…08	Collection factory	Create an ERC-721 NFT collection
0xae…09	Multisig factory	Wallets that need several signatures
0xae…0a	Wrapped QU (WQU)	QU as an ERC-20 token
0xae…0b	Multicall	Several reads in one call
0xae…0c	Pool factory	Mining pools that pay out through a contract

Full addresses are `0xae0000000000000000000000000000000000000001` to `…000c`.

7.1 Proof-of-work precompile and contract pools

The `pools` rule set adds a precompiled contract at `0x0000000000000000000000000000000000000100`. Given the RLP of a block header, it recomputes Ethash and returns the miner, number, parent hash, difficulty, the amount of work and the header hash. It costs a fixed 100,000 gas and rejects headers from epochs older than the previous one.

Pools created by the pool factory use this check to verify miners' shares on-chain. The pool contract receives the block rewards and pays each miner according to verified work, so miners do not have to trust the pool operator with their earnings. Each verified share costs about 220,000 gas; the reference pool server reduces the number of shares it submits as blocks fill up so that pools do not crowd out payments.

8. Peer-to-peer network

- **Transport.** TCP with an X25519 key exchange, keys derived with HKDF-SHA256 and frames encrypted with AES-256-GCM.

- **Identity.** Every node has a persistent Ed25519 key. The handshake is signed over both ephemeral keys, so a relaying attacker cannot impersonate either side. Mainnet accepts only nodes with an identity.
- **Pinned seeds.** Seeds are written as `publickey@host:port`. A new node accepts the seed only if it proves that key, so a forged DNS answer cannot lead it to a different network.
- **Fork identification.** Nodes exchange an EIP-2124-style fork identifier and disconnect cleanly from software that follows incompatible rules.
- **Propagation.** New blocks and transactions are sent in full to the square root of the number of peers and announced by hash to the rest, so traffic grows with the square root of the peer count.
- **Limits.** Every message type has a rate and size limit; peers that send invalid data lose score and are disconnected. Outbound connections are spread across /16 address ranges.
- **Sync.** A node downloads headers first, checks their work and then downloads and executes the blocks. A peer's claim of work proves nothing until its headers have been verified.

9. Mining

- **Solo mining.** Any node can hand out work through JSON-RPC (`eth_getWork`) or Stratum. Public nodes switch the mining methods off.
- **Stratum.** The node includes a Stratum server compatible with common Ethash miners (EthereumStratum and ETHPROXY). One node has been tested with 10,000 connected devices.
- **quintium-miner.** The reference GPU miner is written in Rust on top of wgpu and runs on AMD and NVIDIA cards under macOS and Windows. The Quinvium desktop app starts the node and the miner with one switch.
- **Contract pools.** See section 7.1.

10. Capacity and scaling

Measured on a desktop computer with a quad-core processor:

Area	Measured
Network throughput	about 31.7 simple transfers per second at the 10M gas limit
Full block on one node	built in 0.35–0.5 s, verified and stored in 0.5–0.7 s
Transaction intake	664 transactions per second over RPC
RPC reads	9,500 per second for the block number, 2,900 per second for balances
Stratum	10,000 devices, new work to all of them in 261 ms
Proof-of-work check	13 ms per header
Data growth	about 8.5 kB per transaction

Like Bitcoin and Ethereum, the base layer alone cannot carry millions of daily payments. The paths forward are known and can be combined: raise the gas limit by a hard fork once multi-site tests confirm block propagation, move routine payments to second-layer systems, and serve reads from many public nodes. Before the network is busy, nodes need state pruning and snapshot sync; today every node keeps the full state history. Contract pools are limited to about three verified shares per second across the network; a design that verifies a random sample of shares committed in a Merkle tree would remove this limit and is on the roadmap.

11. Software

All software is written for the project:

- **Node** (JavaScript, Node.js 24): consensus, EVM, P2P, JSON-RPC with WebSocket subscriptions, Stratum, a pool server and a built-in Explorer.
- **Miner** (Rust): `quintium-miner` for graphics cards.
- **Desktop app** (macOS, Windows): wallet, full node, mining, Explorer and swap in one application.
- **Wallets:** the Quinvium wallet for Chrome and Wpayin for iOS and Android; any EVM wallet such as MetaMask also works.
- **Explorer:** blocks, transactions, addresses, tokens, network apps and documentation in eight languages, including right-to-left Arabic.
- **Command line:** the `quinvium` command for wallets, transfers and every network app.

12. Security

What has been done. The node passes the official Ethereum state tests for the supported rule set, a cross-check of its cryptography against reference implementations, attack, fuzzing and fork tests, and fixed fingerprints that detect any accidental consensus change. The reference software went through internal code reviews on 7–8, 10 and 13 November 2026, and the findings were fixed.

What has not been done. There has been no independent security audit yet. Until there is, Quinvium should be treated as experimental software.

Known risks.

- *Hashrate attacks.* A young proof-of-work network can be rewritten for a short time with rented hashrate. Recipients of large payments should wait for more confirmations; the 256-block rule limits how far back a rewrite can reach for running nodes.
- *Infrastructure concentration.* At launch the public RPC endpoint and the seed run on a single server. The chain itself lives on every node and anyone can run a public node or seed; more independent public nodes in different places are a priority.
- *Software distribution.* Desktop builds are not yet code-signed. Remote updates of the desktop app are switched off on mainnet until the update-signing key is kept off the development machine. Users should verify the published SHA-256 checksums.
- *Randomness.* `PREVRANDAO` can be influenced by miners and must not be used for lotteries or similar.
- *Bridge.* A bridge to other networks has been written but is not deployed. It will only run on mainnet with independent validators.

13. Decentralisation, upgrades and governance

Rule changes are scheduled as hard forks in the node: a fork names a block height and the rules that change from that height, for example new EVM features, the maximum gas limit or a new system contract. Nodes that do not upgrade disconnect cleanly through the fork identifier instead of silently following a different chain. There is no on-chain voting; a fork happens only if node operators and miners choose to run the software that contains it.

Quinvium is not a company. There is no foundation, no owner, no admin key and no entity that can change the rules, freeze funds or reverse transactions. The built-in contracts have no owner, and the software is published for anyone to run, verify and build on. Anyone can operate a node, a public RPC endpoint, a pool, an explorer or a wallet.

14. Roadmap

Intentions, not promises:

- **Done:** node, EVM, GPU mining and contract pools, Explorer, desktop app, Chrome wallet, Wpayin integration, command line, public testnet, mainnet launch on 14 November 2026.
- **Next:** an independent security audit; a second public node with another provider; the update-signing key off the development machine and remote updates switched back on; signed Windows and macOS builds and the wallet in the Chrome Web Store.
- **Later:** swap and a verified token list on mainnet under multisig governance; a bridge with independent validators; state pruning, snapshot sync and compact blocks; sampled-share verification for contract pools.

15. Disclaimer

This document describes software, not an investment. Quinvium makes no promise about the price or value of QU, and nothing here is financial, legal or tax advice. The software is experimental and provided as is, without warranty. Blockchain transactions cannot be reversed. Anyone using Quinvium is responsible for complying with the law in their country.

References

1. S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.
2. G. Wood, *Ethereum: A Secure Decentralised Generalised Transaction Ledger* (Yellow Paper).
3. Ethash specification, ethereum.org documentation archive.
4. EIP-155, EIP-1559, EIP-2124, EIP-1153, EIP-5656, EIP-6780, EIP-7823, EIP-7883, eips.ethereum.org.
5. Zawy, *LWMA difficulty algorithm*, 2018.
6. Quinvium consensus specification and source code (packages/node, docs/SPECIFIKACE.md).

Quinvium is open-source software under the MIT license. Source code, releases and the reference node: github.com/Lakylife/quinvium. Where this paper and the code differ, the code defines the network.