Auriq development priorities

Proposed improvements and verification criteria based on Auriq Litepaper v0.3.

RANK	IMPROVEMENT	WHAT TO BUILD OR VERIFY
1	**Prove that payments keep confirming through both staking transitions.** Litepaper, pp. 2–4	Rehearse staking activation at block **129,600** and the eventual end of mining using an accelerated test schedule. Include very few eligible stakers, a temporary absence of eligible stakers, and their return. Once five mining blocks have arrived in a row, lowering difficulty can help an eligible staker produce the next block, but an eligible staker still has to come online. Publish the longest payment delay, the 95th and 99th percentile delays, and whether independent nodes agree on the resulting chain. Document what an operator should do during a prolonged halt.
2	**Limit and account for the 80 million AURIQ treasury before genesis.** Litepaper, pp. 3–4	Publish a first-year budget and consider keeping a modest operating balance available while releasing the reserve in installments. The original allocation is **10.4%** of gross issuance at the mining cutoff, but about **78.2%** of all coins created by staking activation, before burns. Test a treasury payment: two signers authorize it, the recipient's coins become eligible to stake after maturity, and change returned to the treasury remains restricted under an explicit rule. Publish how signers are replaced and how material spending will be reconciled.
3	**Give holders a safe way to stake and recover their coins.** Auriq staking rules	Auriq relies on people keeping staking wallets online. Design a restricted online staking key while keeping ordinary spending authority offline. Have an outside tester stake, verify that the staking key cannot transfer the principal or redirect its required return, retire the staking machine, restore the owner wallet on another device, and spend successfully. This exercise tests the custody experience a long-term holder would rely on.
4	**Make small-scale mining practical all the way through spending a payout.** Mining guide Litepaper, p. 7	Verify solo and pool mining on an ordinary CPU, a workstation CPU, and a SHA256D ASIC. Publish accepted work per kWh, rejected shares, pool fees, minimum payouts, and time to first spendable payout. Then test what it costs to spend **1, 10, and 100 accumulated payouts**. Each additional transaction input carries about 5,261 bytes of key and signature data, so paying a small miner too frequently can make the balance more expensive to move. Add an adjustable payout threshold and a wallet fee preview.
5	**Specify fee voting precisely enough that every wallet and node calculates the same fee.** Litepaper, p. 5	The starting floor is 100 atomic units per virtual byte. Its maximum change per staking vote is **100 ÷ 4,096 ≈ 0.024414** atomic units per virtual byte. Specify how the fractional rate is stored, how votes accumulate, and when the payable fee is rounded. Publish examples for increasing, decreasing, and alternating votes. Have a wallet and an independent calculator reproduce the node's results exactly. Show adjustment speed in both phases: at the target block pace, the staking-only phase has roughly twice as many voting opportunities per day as the hybrid phase.

Auriq development priorities

Proposed improvements and verification criteria based on Auriq Litepaper v0.3.

RANK	IMPROVEMENT	WHAT TO BUILD OR VERIFY
6	**Build one complete payment use case with an independent merchant.** BTCPay invoice workflow	Start with a business the community can reach, such as a hosting provider serving miners or node operators. Let a customer pay an invoice, see the fee before sending, distinguish payment detection from settlement, and receive a refund. Test late and partial payments and give the merchant an accounting export. Testnet can prove the checkout works. **The adoption milestone comes after launch, when an independent customer returns to pay a second real invoice.** This is the highest commercial priority and can start alongside the technical work.
7	**Make issuance, burns, and producer revenue independently auditable.** Auriq FAQ Litepaper, pp. 4–5	Correct the public wording on two points. The **768 million AURIQ** figure is gross creation before burns at the mining cutoff. Base fees above the burn ceiling go to the producer. Publish a per-block breakdown and an independent calculator. With supply near 768 million, a 4 AURIQ subsidy, and 16 AURIQ in base fees, about 11.301 AURIQ burns, the producer receives 8.699 AURIQ plus tips, and net supply falls 7.301 AURIQ. The node, explorer, and calculator should agree to the smallest unit, including at halvings and the mining cutoff.
8	**Measure who actually produces the blocks and casts the fee votes.** Block targets and fee voting	Publish two separate views. The first should show actual versus target block shares for CPU mining, ASIC mining, and staking. The second should show concentration within each group: observed pool shares, identifiable staking operators, and the share of fee votes attributable to the largest known participants. Keep unknown producers visible and state the limits of attribution. One operator can control several addresses. Testnet should establish that the measurements work. Mainnet data will show how broadly participation is distributed.
9	**Keep full historical signature verification available to an ordinary user.** Node modes and storage	Offer clearly labeled fast-sync and full-verification options. The default mode saves storage by skipping older signatures below its release checkpoint, which makes independent archives essential. Have an outside operator start from a clean machine **with the development team's archive offline**, obtain the historical data elsewhere, verify from genesis, and recover an old wallet. Record how long it takes, how much data moves, and which checks were performed. Publish the results alongside the storage estimates.
10	**Make releases, upgrades, and emergency decisions independently verifiable.** Auriq upgrade process Bitcoin Core release verification	Publish source tags, signed checksums, reproducible build instructions, and independent review findings tied to the release candidate. Rehearse a scheduled upgrade with an outside pool or node operator following only the public instructions. Include procedures for replacing software-signing keys, reviewing checkpoints, and deciding when deposits or payments can resume after an incident. Run one rehearsal with the primary developer unavailable. Record every step that needs additional documentation or a backup operator.