zapier.georgelambert.org · Markdown indexes
VERAE TIME × ZAPIER

09 lookup tree nodes

site/user-docs/09-lookup-tree-nodes.md

9. Look up a hash on tree-node archives

Zapier search: **Find Hash (tree nodes + central chain)**.

Middleware: `GET /zapier/v1/hashes/{sha256}?includeAttached=true&includeTree=true`.

Sequence (you never configure this)

1. Central chain lookup (same as chapter 6).
2. If itemized, return that seal (and attached metadata if requested).
3. If miss, middleware broadcasts `verae.archive.query` with `kinds: ["tree", …]` to **every** tree-node NATS server.
4. A node whose bloom filter does not contain the hash **stays silent**.
5. A node that holds the leaf replies on `verae.archive.reply.<correlationId>` with `{ merkleRoot, proof, leafIndex, chainSealJobId }`.
6. Middleware checks the proof against the root, loads the **root's** chain seal, and returns both receipts.

Reading the result in a Zap

- `exists: true` and `itemizedOnMainChain: false` — this hash was in a bulk summary.
- `proofOk: true` — the leaf really is under `merkleRoot`.
- `receipts` includes `kind: seal` (of the root) and `kind: tree-leaf` (the proof).
- `archiveId` tells you which tree node answered (for support).

If every node is silent, the hash is unknown **or** the archive fleet is down. The simulator flags "all archives silent" when puts were known to exist; operators should treat that as an outage, not a miss.

Zap pattern

1. Search: Find Hash (tree nodes + central chain).

2. Filter / Paths:
   - Found + itemized → treat as a first-class seal.
   - Found + not itemized → store Merkle proof + root certificate.
   - Not found → optionally **Create Timestamp** to itemize it now.