

zapier.georgelambert.org · Markdown indexes

VERAE TIME x ZAPIER

Archive NATS, bloom, multi-receipt

<site/docs/02-architecture/archive-nats.md>

Archive NATS: jobs, multipart split, hash receipts, WORM bloom fan-out

Zapier never speaks NATS. HTTPS stops at zappier → middleware. Middleware owns jobs, splitting, chain lookup, and archive aggregation.

End-to-end

```
Zapier --HTTPS--> zappier (meter, x-api-key)
--HTTPS--> middleware /zapier/v1/timestamp[/wait]
  1. split multipart
    chain: SHA256 only (+ optional public-meta digest)
    archives: public JSON, encrypted private JSON, files
  2. publish jobs.watch → return jobId (202)
  3. waiters subscribe jobs.events
  4. lookup SHA256 on chain (mock or Verae)
    already sealed → original receipt, no new seal
    later attach records → extra receipts
  5. if includeAttached:
    publish verae.archive.query
    WORM nodes: bloom miss = silence
    bloom hit = reply
    aggregate until WAIT_ARCHIVE_MS
  6. jobs.events completed JSON → waiter / REST Hook
```

Blockchain stores **hash + time + block + certificate**. Public metadata, encrypted metadata, and file bytes live on **WORM archives**.

Splitter

```
splitRequest(body | multipart):
```

Field	Destination
data / sha256	Chain register or lookup
publicMetadata	Archive put (clear)
privateMetadata	Archive put (ciphertext)
files[]	Archive put; chain gets content hashes + ids
includeAttached	Whether wait path queries archives

Hash already registered

Return original `jobId` and original seal. Do not write a second chain timestamp.

If later attach jobs exist for that hash, `receipts` is an array: seal first, then attachment receipts in time order.

Subjects

Subject	Publisher	Subscriber
verae.zapier.jobs.watch	HTTP edge	job poller
verae.zapier.jobs.events	poller	waiter, webhook router
verae.archive.put	splitter	archive that owns the shard
verae.archive.query	aggregator	every archive (not a shared queue group)
verae.archive.reply.<correlationId>	archive on bloom hit	aggregator

Query payload: { correlationId, sha256, tenantId, kinds[] }.

`kinds` may include `tree` for Merkle leaf proofs (hashes sealed only as a bulk summary root).

Reply payload: { archiveId, sha256, records[] }.

Bloom miss → no reply. Aggregator timeout → complete with whatever arrived.

WORM archives

Each process holds append-only records + a bloom of SHA256 keys it stores. False positives OK; false negatives must be rare. Bloom is not an ACL — on hit, still check tenant/share.

Harness: three mock archives with overlapping hashes.

Completed job JSON (wait / hook)

```
{
  "jobId": "...",
  "status": "completed",
  "sha256": "...",
  "receipts": [
    { "kind": "seal", "timestamp": "...", "certificate": "...", "blockIndex": 42 },
    { "kind": "metadata-attach", "attachedAt": "...", "publicMetadata": {} }
  ],
  "files": [{ "id": "...", "sha256": "...", "archiveId": "archive-b" }],
  "archivesQueried": true,
  "archiveReplies": 2
}
```

Flag off or all blooms miss → `files` empty, extra receipts omitted.

Security

- Zapier never connects to NATS or archives.
- Private metadata only on authenticated archive replies.
- NS1 NATS stays loopback; use `scripts/nats-tunnel.sh`.