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

Verae OpenAPI gaps

site/docs/01-product/api-gap-analysis.md

API gap analysis

Live spec snapshot: veraetime-openapi.yaml
Swagger UI: https://api.veraetime.net/docs/swagger/index.html

What Verae actually exposes

Auth: `POST /auth/login` → JWT. All API routes `Authorization: Bearer <jwt>`.

Operation	Path	Notes
createTimestamp	`POST /api/timestamp`	Body `{ data, hashAlg? }`, **202** `{ jobId }`
createBatchTimestamp	`POST /api/batch/timestamp`	`{ items: [...] }`
verifyTimestamp	`POST /api/verify`	`{ certificate }` → `{ valid, timestamp, blockIndex }`
verifyBatchTimestamp	`POST /api/batch/verify`	
getJobStatus	`GET /api/status/{jobId}`	pending / completed / failed
getJobVerification	`GET /api/verify/{jobId}`	
getBatchJobStatus	`POST /api/batch/status`	
Admin	`/admin/*`	dashboard, metrics, blockchain, queue, timestamps, hash
Users	`/auth/users`	admin CRUD

Gaps vs README a–m

- No customer API-key issuance (a) — **zappier owns this**.
- No billing, invoices, payment methods, subscriptions (b, j–m) — **zappier owns this**.
- No lookup-by-SHA256 or idempotent “return original timestamp” (c, d).
- No public/private metadata (e, h).
- No encrypted object storage or sharing (f, g).
- No certified retrieval receipt / extra seal (i).
- Timestamp input is `data` + `hashAlg`, not a first-class SHA256 key.

Adapter rule

The middleware may expose a **richer** Zapier-facing contract (hash index, metadata, receipts) implemented by `MOCK_VERAE` / mock stores.

The **live** client (`packages/verae-zapier-middleware/src/clients/veraeClient.js`) must only call paths in the snapshot OpenAPI. Unknown fields must not be sent to `https://api.veraetime.net` .