

zapier.georgelambert.org · Markdown indexes

VERAE TIME × ZAPIER

TODO

site/TODO.md

Verae Zapier Platform — Implementation TODO

Workspace: `/Users/marchon/apps/zapier` (imported 2026-09-09). Packages live under `packages/`. Commercial billing is `packages/zappier` — do not reimplement it in middleware. Composition waves: see `docs/02-architecture/composition.md`.

Product goal: Hybrid **HTTP edge** + **NATS workers** middleware between Zapier servers and `https://api.veraetime.net`.

Architecture bottom line:

- Zapier never speaks NATS (HTTPS only).
- `verae-zapier` (CLI app) runs on Zapier infrastructure and calls middleware.
- `verae-zapier-middleware` exposes `/zapier/v1/*` (auth, billing, sync API).
- **NATS + JetStream** owns job watching, completion events, and webhook delivery.
- Workers call `api.veraetime.net` over HTTPS and POST REST Hooks to Zapier.

How to use this file

1. Work **top to bottom**. Phases are dependency-ordered.
2. Every phase ends with a **GATE** (tests must pass) before the next phase starts.
3. Mark items `[x]` only after the gate command succeeds and results are recorded under **Gate log**.
4. Do not skip gates. Parallel work is only allowed **within** a phase when items share no dependency.

Runtime debug

- Failure tracing is off by default.
- Enable at runtime: `DEBUG_VERA=1` or `DEBUG_VERA=auth,nats,jobs,webhooks,http,billing`.
- See `docs/developer/debugging.md`.

Primary docs

Doc	Purpose
README.md	Repo overview and quick start
docs/o2-architecture/overview.md	System diagram and data flows
docs/o2-architecture/nats-subjects.md	Subject topology and payloads
docs/modules/README.md	Per-module function reference
docs/plans/phase-gates.md	Gate commands and acceptance criteria
docs/api/middleware-openapi.yaml	Zapier-facing OpenAPI

Legend

Marker	Meaning
[]	Not started
[~]	In progress
[x]	Done and gate passed
BLOCKED	Waiting on listed dependency
GATE	Mandatory test validation — no next phase until green

Phase 0 — Repository baseline and documentation contract

Depends on: nothing

Unblocks: Phase 1+

Tasks



0.1 Create monorepo layout (`verae-zapier-middleware/` , `verae-zapier/` , `docs/`)



0.2 Write architecture overview and NATS subject map



0.3 Write developer module documentation skeleton for all planned modules



0.4 Document debugging facility contract (`DEBUG_VERA` , namespaces, redaction rules)



0.5 Write this dependency-ordered `TODO.md` with test gates



0.6 Add root `README.md` and package scripts for gated testing

GATE 0 — Documentation and structure

```
cd /Users/marchon/datacubes/verae-zapier-api
npm run gate:0
```

Pass criteria

- Required paths exist (see `scripts/gate-0-structure.mjs`)
- `TODO.md` , architecture docs, debugging doc, module docs present
- No empty required doc stubs

Gate log

Date	Result	Notes
2026-08-11	pass	structure + docs gate

Phase 1 — Debug runtime facility

Depends on: Phase 0 GATE

Unblocks: All code phases (debug used everywhere)

Tasks



1.1 Implement `src/debug/logger.js` — namespaced logger, levels, redact secrets



1.2 Implement `src/debug/trace.js` — request/job correlation IDs



1.3 Implement `src/debug/config.js` — parse `DEBUG_VERAE` / `DEBUG_VERAE_LEVEL` at runtime



1.4 Unit tests: enable/disable namespaces, redaction of tokens/passwords/api keys



1.5 Document every debug export in `docs/developer/modules/debug.md`

GATE 1 — Debug facility

```
npm run gate:1
# equivalent: node --test verae-zapier-middleware/test/unit/debug*.test.js
```

Pass criteria

- Logger silent when `DEBUG_VERAE` unset
- Enabling `DEBUG_VERAE=auth` only emits auth namespace
- Redaction masks `Bearer`, `zmw_`, `zmt_`, password fields
- Correlation ID stable across nested `trace.run` calls

Gate log

Date	Result	Notes
2026-08-11	pass	11/11 unit tests

Phase 2 — Config, errors, and core HTTP shell

Depends on: Phase 1 GATE

Unblocks: Auth, clients, routes

Tasks



2.1 `src/config.js` — env load, plan limits, NATS flags, debug passthrough



2.2 `src/errors.js` — `AppError`, `asyncHandler`, `sendError` with debug dumps on 5xx



2.3 `src/app.js` / `src/index.js` — Express app, `/health`, `/zapier` mount



2.4 Unit tests for config defaults and health endpoint



2.5 Module docs: `config.md` (+ function-reference for errors/app)

GATE 2 — HTTP shell

```
npm run gate:2
```

Pass criteria

- `GET /health` returns `{ status: "ok" }`
- Unknown routes return structured JSON error
- Config reads `VERAE_API_BASE_URL`, `NATS_URL`, `NATS_ENABLED`, `DEBUG_VERAE`

Gate log

Date	Result	Notes
2026-08-11	pass	health + config unit tests; / zapier returns 501 until Phase 5–6 routes

Phase 3 — Persistence stores (tenants, usage, webhooks)

Depends on: Phase 2 GATE

Unblocks: Auth bridge, billing, webhook subscribe API

Tasks



3.1 `src/store/db.js` — load/persist store (file for MVP; interface ready for Redis/Postgres)



3.2 `src/store/tenants.js` — CRUD, API key lookup, `resolveLimits`



3.3 `src/store/usage.js` — monthly counters, overage



3.4 `src/store/webhooks.js` — subscribe/unsubscribe/list by tenant+event



3.5 Unit tests for limits, API key mapping, webhook isolation per tenant



3.6 Module docs for each store file

GATE 3 — Stores

```
npm run gate:3
```

Pass criteria

- Create tenant → API key resolves to same tenant
- Free plan limits applied; enterprise contract overrides
- Webhook for tenant A never returned for tenant B
- Persist + reload round-trip preserves data

Gate log

Date	Result	Notes
2026-08-11	pass	store unit tests

Phase 4 — Tokens and Verae HTTP client

Depends on: Phase 3 GATE

Unblocks: Auth service, timestamp/verify services

Tasks



4.1 `src/lib/tokens.js` — API keys (`zmw_`), session tokens (`zmt_`), HMAC sign/verify



4.2 `src/clients/veraeClient.js` — login, validate, timestamp, status, verify, waitForJob



4.3 Mock mode (`MOCK_VERAE=true`) for offline tests



4.4 Debug spans for outbound HTTP (url, method, status, duration; **no raw secrets**)



4.5 Unit tests: mock client lifecycle (create → pending → completed)



4.6 Module docs: `tokens.md`, `veraeClient.md`

GATE 4 — Client + tokens

```
npm run gate:4
```

Pass criteria

- Session token forge fails without secret
- Mock create returns `jobId`; wait reaches `completed`
- Debug log with `DEBUG_VERAЕ=http` shows request metadata without Authorization header values

Gate log

Date	Result	Notes
2026-08-11	pass	tokens + mock client

Phase 5 — Auth bridge and entitlements (HTTP, no NATS yet)

Depends on: Phase 4 GATE

Unblocks: Timestamp/verify routes, Zapier custom auth

Tasks



5.1 `src/services/authService.js` — login credentials, login API key, resolveAuthContext, validateSession



5.2 `src/services/entitlementService.js` — checkEntitlement, recordUsage



5.3 `src/middleware/authenticate.js` — Bearer / x-api-key



5.4 `src/middleware/rateLimit.js` — per-plan RPM



5.5 `src/routes/authRoutes.js` — `POST /login`, `GET /me` under `/zapier/v1/auth`



5.6 Integration tests: API key → me; quota 402; rate limit



5.7 Module docs for auth + entitlement + middleware

GATE 5 — Auth + billing gates

```
npm run gate:5
```

Pass criteria

- Seeded free tenant can call `/zapier/v1/auth/me`
- Exceeding free timestamp quota returns `402` + `QUOTA_EXCEEDED`
- Invalid key returns `401`
- Debug `DEBUG_VERA=auth,billing` traces tenantId/plan without password

Gate log

Date	Result	Notes
2026-08-11	pass	auth me + quota

Phase 6 — Sync HTTP API path (in-process poller fallback)

Depends on: Phase 5 GATE

Unblocks: NATS swap-in (Phase 7+), Zapier app wiring

Note: Implements full product behavior with `NATS_ENABLED=false` so Zapier can work before NATS.

Tasks



6.1 `src/services/timestampService.js` — create, createAndWait, batch, getStatus



6.2 `src/services/verifyService.js` — verify, verifyBatch



6.3 `src/services/webhookService.js` — subscribe/unsubscribe/deliver (HTTP)



6.4 Routes: timestamp, verify, status, webhooks



6.5 In-process `src/workers/inProcessJobPoller.js` (legacy path, feature-flagged)



6.6 Integration tests with `MOCK_VERAE=true`, `NATS_ENABLED=false`



6.7 Module docs for services + routes

GATE 6 — Full HTTP middleware (no NATS)

```
npm run gate:6
```

Pass criteria

- `POST /zapier/v1/timestamp` → 202 + jobId
- `POST /zapier/v1/timestamp/wait` → completed status with certificate fields
- `POST /zapier/v1/verify` → valid true/false
- Webhook subscribe stores targetUrl; simulated complete invokes deliver (mock)
- All tests pass with NATS disabled

Gate log

Date	Result	Notes
2026-08-11	pass	full HTTP mock path

Phase 7 — NATS connection, streams, and publishers

Depends on: Phase 6 GATE

Unblocks: Workers, production multi-instance

Tasks



7.1 Add local NATS JetStream to `docker-compose.yml`



7.2 `src/nats/subjects.js` — subject constants



7.3 `src/nats/connection.js` — connect, reconnect, ensure streams/consumers



7.4 `src/nats/publishers.js` — enqueueWatch, publishJobEvent, enqueueWebhook



7.5 Unit/integration tests against test NATS (or mocked JetStream interface)



7.6 Module docs: `nats-connection.md`, `nats-subjects.md`, `nats-publishers.md`



7.7 Debug namespace `nats` for publish/ack/error tracing

GATE 7 — NATS infrastructure

```
npm run gate:7
# starts NATS if needed, ensures streams, runs nats tests
```

Pass criteria

- Streams `ZAPIER_JOBS`, `ZAPIER_EVENTS`, `ZAPIER_WEBHOOKS` exist
- Publish + pull consume one message end-to-end
- Connection failure surfaces structured error; debug logs connection lifecycle
- `NATS_ENABLED=false` still loads app without connecting

Gate log

Date	Result	Notes
2026-08-11	pass	nats-server JetStream

Phase 8 — NATS workers (job poller + webhook deliver)

Depends on: Phase 7 GATE

Unblocks: Hybrid production path

Tasks



8.1 `src/workers/jobPollerWorker.js` — durable consumer on `verae.zapier.jobs.watch`



8.2 `src/workers/webhookWorker.js` — durable consumer on `verae.zapier.webhooks.deliver`



8.3 Wire `timestampService` to publish watch jobs when `NATS_ENABLED=true`



8.4 Events consumer routes `jobs.events` → enqueue webhook deliveries for matching hooks



8.5 **Never** put raw Verae JWT in NATS when avoidable — use `tokenRef` or re-login



8.6 Integration tests: create → worker poll → event → webhook HTTP mock



8.7 Module docs for workers; failure tracing via `DEBUG_VERA=jobs,webhooks,nats`

GATE 8 — Async path with NATS

```
npm run gate:8
```

Pass criteria

- With NATS up and `NATS_ENABLED=true`, async create eventually completes via worker
- Webhook target receives POST with `timestamp.completed` payload
- Redelivery: failing webhook is retried (JetStream `max_deliver`)
- Two concurrent workers do not double-complete same job (queue group)
- In-process poller **not** started when NATS enabled

Gate log

Date	Result	Notes
2026-08-11	pass	workers watch→webhook

Phase 9 — `/timestamp/wait` via NATS events

Depends on: Phase 8 GATE

Unblocks: Efficient Zapier “Create and Wait” under multi-instance

Tasks



9.1 API waits on job event (subscription or inbox) after enqueue watch



9.2 Hard timeout returns `{ jobId, status: "pending" }` (document Zapier fallback)



9.3 Tests: fast complete; timeout path; cancellation on client disconnect (if feasible)



9.4 Docs update: wait semantics, Zapier multi-step fallback

GATE 9 — Wait path

```
npm run gate:9
```

Pass criteria

- Wait returns completed when worker finishes before timeout
- Wait returns pending + jobId on timeout without hanging forever
- Debug shows wait correlation id linking HTTP request → NATS event

Gate log

Date	Result	Notes
2026-09-09	pass	wait via NATS + timeout pending; NS1 tunnel 70.88.205.138

Phase 10 — Tenant signup / admin provision

Depends on: Phase 6 GATE (can parallel Phase 7–9 after 6)

Unblocks: Self-serve and enterprise keys for Zapier

Tasks



10.1 `src/services/tenantService.js` — `selfServeSignup`, `provisionTenant`



10.2 Routes: public signup, admin tenants (`X-Admin-Secret`)



10.3 Scripts: `seed-audiences.js`, `create-test-account.js`



10.4 Tests for free signup and enterprise contract validation



10.5 Module docs: `tenantService.md`, `tenantRoutes.md`

GATE 10 — Tenancy

```
npm run gate:10
```

Pass criteria

- Signup returns `apiKey` + free plan
- Enterprise without contract rejected
- Admin list does not leak passwords

Gate log

Date	Result	Notes
2026-08-11	pass	signup + admin

Phase 11 — Zapier CLI app (`verae-zapier`)

Depends on: Phase 6 GATE minimum; Phase 8 recommended for triggers

Unblocks: Private invites on Zapier

Tasks



11.1 `authentication.js` — custom API key → `/zapier/v1/auth/me`



11.2 `index.js` — `beforeRequest` API key; `afterResponse` map 402/403



11.3 Creates: `timestamp_and_wait`, `create_timestamp`, `verify`, `batch`



11.4 Search: `job_status`



11.5 Trigger: timestamp_completed REST Hook



11.6 Unit tests with mocked middleware (nock or local server)



11.7 `MIDDLEWARE_BASE_URL` env documentation



11.8 Developer docs for each Zapier operation module

GATE 11 — Zapier package

```
npm run gate:11
```

Pass criteria

- Auth test module returns tenant/plan label fields
- Each create/search perform builds correct middleware URL/body
- Trigger subscribe/unsubscribe payload shapes match middleware
- `zapier-platform validate` / local test suite green (when CLI available)

Gate log

Date	Result	Notes
2026-08-11	pass	zapier package unit tests

Phase 12 — End-to-end validation (local stack)

Depends on: Gates 8, 10, 11

Unblocks: Remote deploy

Tasks



12.1 `docker-compose.yml` : middleware + NATS (+ optional mock Verae)



12.2 Compose smoke script: health, auth, timestamp wait, webhook



12.3 Debug mode runbook: reproduce a failed job with `DEBUG_VERAЕ=*`



12.4 Record sample traces in `docs/developer/debugging.md`

GATE 12 — E2E local

```
npm run gate:12
```

Pass criteria

- Compose stack healthy
- One full async path and one wait path succeed
- One REST Hook delivery succeeds against mock receiver
- Gate script exits 0

Gate log

Date	Result	Notes
2026-09-09	pass	harness/scripts/smoke.sh health + signup + wait

Phase 13 — Production wiring to api.veraetime.net

Depends on: Phase 12 GATE

Unblocks: Real customer traffic

Tasks



13.1 Set `VERAE_API_BASE_URL=https://api.veraetime.net`, `MOCK_VERAE=false`



13.2 Provision dedicated Zapier service user on Verae (not shared admin long-term)



13.3 TLS, secrets management (`TOKEN_SECRET`, `ADMIN_SECRET`, NATS auth)



13.4 Public HTTPS URL for middleware (required for Zapier cloud → you)



13.5 Live smoke tests (non-destructive verify + one timestamp)



13.6 Runbook: incident tracing with debug flags (no secrets in logs)

GATE 13 — Production smoke

```
npm run gate:13
# requires env: PRODUCTION_SMOKE=1 and real credentials
```

Pass criteria

- Login + timestamp + status against live API succeed
- No secrets printed in CI logs
- Documented rollback: `NATS_ENABLED=false` still serves sync path

Gate log

Date	Result	Notes

Phase 14 — Private Zapier push and invite

Depends on: Phase 13 GATE

Unblocks: Beta users

Tasks

- ☐ 14.1 `zapier-platform register` / `link`
- ☐ 14.2 `zapier-platform push` private version
- ☐ 14.3 Invite internal users; validate real Zap (Drive/Sheets → Timestamp → Slack)
- ☐ 14.4 Collect failures; enable `DEBUG_VERA` only on middleware staging for repro
- ☐ 14.5 Update Progress / changelog

GATE 14 — Human acceptance

Pass criteria

- At least one real Zap runs end-to-end with production middleware
- Auth connection label shows tenant + plan
- Async trigger or wait path verified by human sign-off

Gate log

Date	Result	Notes

Phase 15 — Hardening (post-beta)

Depends on: Phase 14

Unblocks: Public listing consideration

Tasks

☐

15.1 Replace file store with Postgres/Redis

☐

15.2 NATS mTLS / account auth

☐

15.3 Webhook SSRF protections

☐

15.4 Usage stream consumer for billing export

☐

15.5 Optional: Verae core publishes completion to NATS (drop polling)

☐

15.6 Security review of tokenRef vs password-at-rest

GATE 15 — Hardening checklist

Pass criteria

- Written security review checklist completed
- Load test: N concurrent workers, no duplicate webhook storms
- Backup/restore drill for tenant store

Dependency graph (summary)

```
0 Docs/structure
└─ 1 Debug facility
   └─ 2 HTTP shell
      └─ 3 Stores
         └─ 4 Tokens + Verae client
            └─ 5 Auth + entitlements
               └─ 6 Sync HTTP API (+ in-process poller)
                  ├── 7 NATS infra → 8 Workers → 9 Wait-via-NATS
                  ├── 10 Tenancy (parallel after 6)
                  └─ 11 Zapier app (after 6; triggers prefer 8)
                     └─ 12 E2E local (needs 8,10,11)
                        └─ 13 Production Verae
                           └─ 14 Private Zapier push
                              └─ 15 Hardening
```

npm gate commands (root)

Script	Phase	What it runs
<code>npm run gate:0</code>	0	Structure + doc presence
<code>npm run gate:1</code>	1	Debug unit tests
<code>npm run gate:2</code>	2	Config/errors/app tests
<code>npm run gate:3</code>	3	Store tests
<code>npm run gate:4</code>	4	Tokens + client tests
<code>npm run gate:5</code>	5	Auth + entitlement tests
<code>npm run gate:6</code>	6	HTTP integration (NATS off)
<code>npm run gate:7</code>	7	NATS stream tests
<code>npm run gate:8</code>	8	Worker integration
<code>npm run gate:9</code>	9	Wait path tests
<code>npm run gate:10</code>	10	Tenant tests
<code>npm run gate:11</code>	11	Zapier package tests
<code>npm run gate:12</code>	12	Compose E2E
<code>npm run gate:13</code>	13	Live smoke (opt-in)
<code>npm run gate:all</code>	0–12	All automated gates in order; stop on first failure

Definition of done (platform MVP)

☐

All gates 0–12 green on clean machine

☐

`NATS_ENABLED=true` path used for job watch + webhooks

☐

`NATS_ENABLED=false` fallback still passes gate 6

☐

Developer docs cover every module's exported functions (params + returns)

☐

Debug can be enabled at runtime without redeploy (`DEBUG_VERA=...`)

☐

No secrets in logs when debug is on (redaction tests green)

☐

Private Zapier version pushed and one human E2E Zap succeeds (gate 14)

Working rules for implementers

1. **No phase advancement without gate.**
2. **Write tests first or with the code** for that phase's modules.
3. **Document every new export** in `docs/developer/modules/<name>.md` and JSDoc in source.
4. **Instrument with `debug`** at boundaries: auth resolve, outbound Verae, NATS publish/consume, webhook HTTP.
5. **Prefer failing a gate** over merging incomplete behavior.
6. Update the **Gate log** tables when a gate passes.