Athena Accounting

API Reference

Version 1.0.0-rc.1 · AthenaLab

Document date: 2026-08-07 · v1.0.0 (General Availability)

AI-first Malaysian double-entry accounting & e-Invoice

Athena Accounting — Public API Reference (v1)

Milestone 5E — Public API, Integrations & Developer Platform.

The public API exposes **existing** functionality safely. It defines **no accounting logic**: reads are organization-scoped queries and the single write reuses the internal customer service. The ledger, posting, journals, trial balance, forecasting, AI, MyInvois, bank reconciliation, tax, subscriptions, authentication and the security model are **not** modified or duplicated.

There are two planes:

Plane	Base	Auth	Audience
Public API	`/api/v1`	**API key** (`Authorization: Bearer ak_…` or `X-API-Key`)	third-party integrations
Developer portal	`/api/developer`	user **session** + RBAC	tenant admins managing the above
OAuth token	`/api/oauth/token`	client credentials	machine-to-machine (foundation)

Versioning

- The version is in the path: `/api/v1`.
- Every response carries `X-Athena-API-Version: v1`.
- `GET /api/v1` returns version metadata (`version`, `released`, `deprecated`, `sunset`, `resources`).
- When a version is later deprecated, responses add RFC-8594 `Deprecation: true` and `Sunset: <date>`

headers. v1 is current, so no sunset is set.

Authentication (API keys)

Create keys in the developer portal. A key is shown **once** at creation/rotation — only its **SHA-256 hash** is stored; the displayed `key_prefix` is masked (`ak_live_a1b2c3d4…`).

- **Live** keys begin `ak_live_`; **sandbox** keys begin `ak_test_`.
- **Sandbox keys are read-only** (they serve the sample tenant); writes return `403`.
- Present the key as `Authorization: Bearer ak_…` **or** `X-API-Key: ak_…`.
- Expired or revoked keys return `401`.

Scopes

`read`, `write`, `customers:read`, `customers:write`, `suppliers:read`, `accounts:read`, `invoices:read`, `reports:read`. `write` implies full read+write; a generic `read` satisfies any `*:read` requirement. A request lacking the required scope returns `403`.

Rate limiting

Per **API key**, sliding window, configurable per organization (default **120 req/min**) via the developer settings. It reuses the in-process limiter from Milestone 5A (no Redis). Exceeding it returns `429` with

`Retry-After: 60`.

Endpoints (v1)

Method	Path	Scope	Notes
GET	`/api/v1`	*(none)*	version metadata
GET	`/api/v1/customers`	`customers:read`	`?active=&search=`
POST	`/api/v1/customers`	`customers:write`	reuses the internal `parties` service
GET	`/api/v1/suppliers`	`suppliers:read`	
GET	`/api/v1/accounts`	`accounts:read`	chart of accounts
GET	`/api/v1/invoices`	`invoices:read`	`?status=&limit=`
GET	`/api/v1/reports/trial-balance`	`reports:read`	`?from_date=&to_date=` — reuses the reporting service

Examples

cURL

```
# List customers
curl -s https://your-host/api/v1/customers \
  -H "Authorization: Bearer ak_live_xxx"

# Create a customer (reuses the existing service; posts no journal)
curl -s -X POST https://your-host/api/v1/customers \
  -H "Authorization: Bearer ak_live_xxx" -H "Content-Type: application/json" \
  -d '{"customer_code":"CUST-100","customer_name":"Acme Sdn Bhd"}'

# Trial balance
curl -s "https://your-host/api/v1/reports/trial-balance?from_date=2026-01-01&to_date=2026-12-31" \
  -H "X-API-Key: ak_live_xxx"
```

Python

```
import requests

BASE = "https://your-host/api/v1"
HEADERS = {"Authorization": "Bearer ak_live_xxx"}

customers = requests.get(f"{BASE}/customers", headers=HEADERS).json()["data"]

requests.post(f"{BASE}/customers", headers=HEADERS,
              json={"customer_code": "CUST-100", "customer_name": "Acme Sdn Bhd"})
```

JavaScript (fetch)

```
const BASE = "https://your-host/api/v1";
const headers = { Authorization: "Bearer ak_live_xxx", "Content-Type": "application/json" };

const { data } = await (await fetch(`${BASE}/customers`, { headers })).json();

await fetch(`${BASE}/customers`, {
  method: "POST", headers,
  body: JSON.stringify({ customer_code: "CUST-100", customer_name: "Acme Sdn Bhd" }),
});
```

The live, always-current schema is the app's **OpenAPI** document at `/openapi.json` (Swagger UI at `/docs`), from which client SDKs can be generated.

Webhooks

Subscribe HTTPS endpoints to events in the developer portal. A **signing secret** (`whsec_…`) is shown **once** at creation.

Events

`invoice.created`, `invoice.approved`, `bill.created`, `payment.approved`, `journal.posted`, `myinvois.accepted`, `document.reviewed`, `forecast.generated`, `ai.suggestion.approved`, `subscription.changed`.

Events are **sourced from the existing audit trail** — no accounting module was modified to emit them. `POST /api/developer/webhooks/sync` maps newly-recorded audit actions to events and enqueues deliveries; `POST /api/developer/webhooks/process` drains the queue.

Delivery, signing & retry

Each delivery carries:

- `X-Athena-Event` — the event type,
- `X-Athena-Delivery` — the delivery id,
- `X-Athena-Signature` — `HMAC-SHA256(signing_secret, raw_body)` (hex).

Verify the signature before trusting a payload:

```
import hmac, hashlib
expected = hmac.new(secret.encode(), request.body, hashlib.sha256).hexdigest()
assert hmac.compare_digest(expected, request.headers["X-Athena-Signature"])
```

A non-2xx response (or a network error) is retried with **deterministic exponential backoff** (60s · 2■, capped at 24 h). After `webhook_max_retries` (default 5) a delivery is **dead-lettered** — never silently dropped — and can be **requeued** from the portal.

OAuth (client-credentials foundation)

Register a client in the portal (`client_id` = `oc_…`; the `client_secret` is shown once, hashed at rest). Exchange credentials for an opaque access token:

```
curl -s -X POST https://your-host/api/oauth/token -H "Content-Type: application/json" \
  -d '{"grant_type":"client_credentials","client_id":"oc_...","client_secret":"..."}'
```

This is a **foundation** only — it returns opaque token metadata; there is **no external identity provider** and full JWT/OAuth authorization-code flows are out of scope for 5E.

Developer portal (`/api/developer`, session-authenticated)

RBAC: `api.read` (view), `api.keys` (manage keys), `api.webhooks` (manage webhooks/queue), `api.manage` (OAuth clients + settings), `developer.portal` (catalog). Viewer = read-only; Finance Manager = manage; Super Admin = everything.

Key groups: `keys` (+ `/rotate`, `/revoke`), `oauth-clients` (+ `/revoke`), `webhooks` (+ `/test`, `/sync`, `/process`, `/deliveries`, `/deliveries/{id}/requeue`), `usage`, `logs`, `settings`, `catalog`.

Usage analytics & logs

`GET /api/developer/usage` aggregates request count, error rate, average latency, rate-limited count, sandbox count and top endpoints. `GET /api/developer/logs` returns recent request rows — **metadata only**: method, path, status, latency, sandbox flag and timestamp. **No request or response body and no secret is ever stored.**

Integration framework

A framework abstraction only — connectors are described but **no live third-party integration** (Stripe, Xero, QuickBooks, Shopify, WooCommerce, SAP, Oracle, …) is implemented in 5E.