ERP Developer API Reference

REST API documentation for integrators building against the multi-tenant ERP. All paths are relative to your deployment origin.

Base URL: `https://erp.yallav.io/api`

Multi-tenancy

Every protected operation is scoped to a single tenant. The server rejects cross-tenant access at the service layer. You must never send another tenant's IDs expecting them to resolve.

Authentication

Two supported modes (see authentication.md (see authentication) for full detail):

For **ops scripts and dashboard workflows** (audits, remediation, migration), see TOOLS_MANUAL.md (see ../TOOLS_MANUAL).

Mode	Use case	How
Session (cookie)	User-facing apps, scripts after login	`POST /api/auth/login` → HTTP-only `session` cookie on subsequent requests
Service Bearer	Server-to-server automation	`Authorization: Bearer {ERP_SERVICE_SECRET}` plus required headers

Required headers (protected routes)

Header	Session auth	Service Bearer
`x-tenant-id`	Injected by middleware from session	**You must send**
`x-user-id`	Injected from session	Recommended for audited writes
`x-user-role`	Injected from session	Optional
`Content-Type`	`application/json` for JSON bodies	Same

`ERP_SERVICE_SECRET` is a **deployment-level** shared secret (not per-developer API keys). Contact your operator for provisioning.

Optional: `x-ai-agent: true`

When set, list endpoints cap `take` at **50** (see conventions.md (see conventions)). Intended for AI/automation callers; safe for integrators that want a hard upper bound.

Common response shapes

Success

- JSON body as documented per endpoint.
- Creates typically return **201** with the created entity or batch result.

Errors

Most routes return:

```
{ "error": "Human-readable message" }
```

Some settings/analytics routes use `{ "message": "..." }` instead.

Status	Typical cause
400	Validation failed (first Zod issue message)
401	Missing/invalid auth or missing `x-tenant-id`
404	Entity not found
409	Conflict (duplicate name, barcode, etc.)
500	Server error

Soft delete

`DELETE` on entities marks records as deleted (`isDeleted` / `deletedAt`); data is not physically removed.

Conventions

Shared patterns (pagination, bulk resolve/create, enums): **conventions.md (see conventions)**

API index

Foundation

Document	Description
authentication.md (see authentication)	Login, logout, session, password recovery
conventions.md (see conventions)	Pagination, bulk batches, enums, path aliases

Catalog

Document	Base path
catalog/inventory.md (see catalog/inventory)	`/api/inventory`
catalog/categories.md (see catalog/categories)	`/api/categories`
catalog/manufacturers.md (see catalog/manufacturers)	`/api/manufacturers`

Parties

Document	Base path
parties/customers.md (see parties/customers)	`/api/customers`
parties/suppliers.md (see parties/suppliers)	`/api/dealers`, `/api/suppliers` (alias)

Sales

Document	Base path
sales/invoices.md (see sales/invoices)	`/api/invoices`
sales/orders.md (see sales/orders)	`/api/orders`

Migration

Document	Description
migration-jobs.md (see migration-jobs)	Shamel migration worker API (JWT, batch, finalize)

Imports

Document	Description
imports/overview.md (see imports/overview)	Validate → import workflow
imports/products.md (see imports/products)	Product CSV / row import
imports/invoices.md (see imports/invoices)	Invoice import
imports/orders.md (see imports/orders)	Purchase order import

Expenses

Document	Base path
other-expenses.md (see other-expenses)	`/api/other-expenses`

Treasury

Document	Base path
cheques.md (see cheques)	`/api/cheques/analyze` (OCR autofill)

Fixed assets

Document	Base path
asset-groups.md (see asset-groups)	`/api/asset-groups`
assets.md (see assets)	`/api/assets`

Read-only reporting

Document	Base path

analytics.md (see analytics)	`/api/analytics`
treasury.md (see treasury)	`/api/accountant/treasury`
settings.md (see settings)	`/api/settings`, catalog profile

Recommended flows

Onboard products

Diagram omitted in PDF — see https://erp.yallav.io/docs/erp-api-reference.pdf

Create a sales invoice

Diagram omitted in PDF — see https://erp.yallav.io/docs/erp-api-reference.pdf

---

Out of scope (not documented here)

Admin, debug, migration, in-app chat, notifications, audit logs, webhooks, and internal user-phone lookup routes are excluded from this reference.

Roadmap (not implemented)

Per-tenant API keys, OpenAPI generation from Zod, and removal of development `seed-tenant-001` fallbacks on some GET routes would be separate work. Today, use session auth or deployment `ERP_SERVICE_SECRET` as described above.

Authentication API

Auth routes live under `/api/auth/*`. Middleware **does not** require a session for these paths (see `src/middleware.ts`).

Protected routes elsewhere require either a valid **session cookie** or **service Bearer** token.

POST /api/auth/login

Create a session for a user.

Request body

Field	Type	Required	Constraints
`email`	string	Yes	Valid email
`password`	string	Yes	Min 6 characters
`rememberMe`	boolean	No	Extends session lifetime when true

Response 200

```
{
  "success": true,
  "redirect": "/dashboard"
}
```

`redirect` is `/admin` for `SUPER_ADMIN` users.

Sets an **HTTP-only** `session` cookie. Subsequent API calls from the same client send this cookie; middleware injects `x-tenant-id`, `x-user-id`, and `x-user-role`.

Errors

Status	Body

400	`{ "error": "<validation message>" }`
401	`{ "error": "<auth failure>" }`

Example

```
curl -s -c cookies.txt -X POST "https://erp.yallav.io/api/auth/login" \
  -H "Content-Type: application/json" \
  -d
'{"email":"user@example.com","password":"secret123","rememberMe":true}'
```

POST /api/auth/logout

Clear the session cookie.

Response 200

```
{ "success": true }
```

Example

```
curl -s -b cookies.txt -X POST "https://erp.yallav.io/api/auth/logout"
```

GET /api/auth/me

Return the current session (requires valid session cookie).

Response 200

```
{
  "session": {
    "userId": "uuid",
    "tenantId": "uuid",
    "role": "USER",
    "defaultLanguage": "en",
    "name": "Display Name"
  }
}
```

Response 401

```
{ "session": null }
```

Example

```
curl -s -b cookies.txt "https://erp.yallav.io/api/auth/me"
```

Use this after login to confirm `tenantId` before calling tenant-scoped APIs with cookie auth.

POST /api/auth/recover

Request a password reset email.

Request body

Field	Type	Required
`email`	string	Yes (valid email)

Response 200

```
{
  "success": true,
  "message": "Recovery link sent"
}
```

Email contains a link to `https://erp.yallav.io/reset-password?token=...`.

Errors

Status	Body
400	`{ "error": "<message>" }`

PUT /api/auth/recover

Complete password reset with token from email.

Request body

Field	Type	Required	Constraints
`token`	string	Yes	Non-empty
`newPassword`	string	Yes	Min 8 characters

Response 200

```
{
  "success": true,
  "message": "Password Reset Successfully"
}
```

Errors

Status	Body
400	`{ "error": "<message>" }` (invalid/expired token)

Example

```
curl -s -X PUT "https://erp.yallav.io/api/auth/recover" \
  -H "Content-Type: application/json" \
  -d '{"token":"reset-token-from-
email","newPassword":"newSecurePass1"}'
```

Calling protected APIs after login

Cookie session:

```
curl -s -b cookies.txt "https://erp.yallav.io/api/customers?take=10"
```

Service Bearer (no cookie; you supply tenant context):

```
curl -s "https://erp.yallav.io/api/customers?take=10" \
  -H "Authorization: Bearer YOUR_ERP_SERVICE_SECRET" \
  -H "x-tenant-id: YOUR_TENANT_UUID" \
  -H "x-user-id: YOUR_USER_UUID"
```

See README.md (see README) and conventions.md (see conventions).

API Conventions

Shared behavior across ERP REST endpoints. Module docs link here instead of repeating bulk semantics.

Pagination (list GET)

Query param	Type	Default	Notes
`skip`	number	`0`	Offset
`take`	number	~`20`	Page size; see caps below
`search`	string	—	Module-specific text search
`sortBy`	string	—	Where supported
`sortOrder`	`asc` | `desc`	—	Where supported

List response shape (typical)

```
{
  "data": [ /* entities */ ],
  "total": 123
}
```

Exact field names match each service; module pages document module-specific filters.

`take` caps

Context	Max `take`
Most list routes (default)	Requested value (often uncapped above default)
`GET /api/inventory` (non-agent)	**100**
Routes with `x-ai-agent: true`	**50** (see below (see #x-ai-agent-list-cap))

---

x-ai-agent list cap

When request header `x-ai-agent: true`:

- `capListTakeForAiAgent` limits `take` to **50** (`AI_AGENT_MAX_LIST_TAKE` in `src/lib/ai-agent-request.ts`).

Affected list routes include:

Route
`GET /api/customers`
`GET /api/dealers`
`GET /api/invoices`
`GET /api/orders`
`GET /api/other-incomes`
`GET /api/other-expenses`
`GET /api/accountant/payment-vouchers`
`GET /api/accountant/receipt-vouchers`
`GET /api/accountant/treasury/transactions`
`GET /api/inventory` (with agent header)

`GET /api/settings` with `x-ai-agent: true` also triggers `ensureTenantBusinessProfile` before returning settings.

Bulk operations

Maximum batch size: 50 per request for all bulk endpoints unless noted otherwise.

Entity bulk resolve (customers, suppliers/dealers, categories, manufacturers)

POST body:

```
{
  "lines": [
    { "name": "Acme Corp", "nameVariations": ["Acme", "ACME Corp"] }
  ]
}
```

Field	Type	Required
`lines`	array	Yes, 1–50 items
`lines[].name`	string	No*
`lines[].nameVariations`	string[]	No

*At least one searchable term per line is required at the service layer.

Response:

```
{
  "lines": [
    {
      "lineIndex": 0,
      "status": "exact",
      "exists": true,
      "id": "uuid",
      "name": "Acme Corp",
      "customerId": "uuid",
      "customerName": "Acme Corp"
    }
  ],
  "summary": { "exact": 1, "similar": 0, "none": 0 }
}
```

Route handlers add entity-specific aliases (`customerId` / `customerName` , `dealerId` / `dealerName` , etc.) on top of shared `id` / `name` .

`status`	Meaning
`exact`	Single confident match
`similar`	Possible matches; check `suggestions`

`none`	No match

Matching uses normalized names and fuzzy similarity (`src/lib/entityResolution.ts`).

Entity bulk create (customers, dealers, categories, products)

POST body uses a plural entity key:

Module	Body key
Customers	`customers`
Dealers (suppliers)	`dealers`
Categories	`categories`
Inventory	`products`
Product aliases	`aliases`

Response (201):

```
{
  "results": [
    { "index": 0, "success": true, "customer": { "id": "uuid", "name":
"Acme" } },
    { "index": 1, "success": false, "error": "Customer already exists"
}
  ],
  "summary": { "succeeded": 1, "failed": 1 }
}
```

Each row is processed independently; partial success is normal.

Inventory bulk resolve (separate contract)

POST `/api/inventory/bulk-resolve`

Request `lines[]` item fields:

Field	Type	Description

`rawQuery`	string	Free-text query
`facets`	object	Catalog intent (categoryNames, typeTokens, modelTokens, etc.)
`nameVariations`	string[]	Alternate spellings
`productCode` / `code`	string	SKU / barcode
`productName`	string	Display name hint
`categoryId`	string	Scope search to category

Per-line `status`:

Status	Meaning
`exact_match`	Single product matched
`ambiguous`	Multiple `candidates`; disambiguate
`related_match`	Related product suggestion
`not_found`	No match

Response includes `lines`, `summary` (`exact`, `notFound`, `ambiguous`, `related`), and `meta` (`usedGinRetrieval`, `semanticUsed`, `catalogSize`, `matchReasons`, `catalogVersion`).

Bulk resolve idempotency (read-only)

All `POST .../bulk-resolve` routes are **read-only** and **idempotent**: they perform no writes and are safe to retry on `503` / timeout. Agent-service may retry these POSTs with exponential backoff.

Party bulk-resolve responses may include `meta.catalogTruncated: true` and `meta.resolveMode: "search"` for very large tenants (>50k entities); agents should log `catalog_truncated` and use search-scoped matching.

Full detail: catalog/inventory.md (see catalog/inventory).

Suppliers vs dealers (path alias)

The dashboard UI uses **suppliers**; the data model is **dealers**.

Preferred doc name	API paths (equivalent)
Supplier	`/api/dealers/*`
Supplier (alias)	`/api/suppliers/*` (re-exports dealer handlers)

Use either path; behavior is identical.

Date and money

- Dates in JSON are typically **ISO 8601 strings** (`YYYY-MM-DD` or full datetime).
- Money fields are **numbers** (decimal values, not formatted strings).
- Amounts are non-negative unless documented otherwise.

Development fallbacks (not for production)

Some **GET** routes fall back to `tenantId = seed-tenant-001` when `x-tenant-id` is missing or invalid:

- `GET /api/inventory`
- `GET /api/inventory/[id]`
- `GET /api/inventory/by-ids`
- `GET /api/inventory/semantic-search`
- `GET /api/invoices`
- `GET /api/orders`
- `GET /api/analytics/*`

Integrators must always send a real `x-tenant-id` (via session or Bearer). Do not rely on seed tenant in production.

Enums appendix

Payment method (invoices, payments)

Value	Description

`CASH`	Cash
`BANK_TRANSFER`	Bank transfer
`CHEQUE`	Cheque

Payment status (invoice/order filters)

Prisma `PaymentStatus` values used in query params (e.g. `paymentStatus` on `GET /api/invoices`):

Value	Typical meaning
`UNPAID`	No payment recorded
`PARTIALLY_PAID`	Partial payment
`PAID`	Fully paid
`OVERPAID`	Paid above total

Order status

Value
`PENDING`
`DELIVERED`

Product status (inventory update)

Value
`USABLE`
`DAMAGED`

Treasury voucher type (`GET .../treasury/transactions`)

Filter `type` with Prisma `TreasuryVoucherType` (e.g. cash in/out, transfer — see Prisma schema).

Treasury voucher source

Filter `source` with Prisma `TreasuryVoucherSource`.

Alias source (`POST .../aliases/bulk-create`)

Value
`user_correction`
`agent`
`user_confirm`
`transaction_confirm`

HTTP verbs

Verb	Usage
`GET`	Read / list
`POST`	Create, bulk, validate import
`PUT`	Update entity or settings
`DELETE`	Soft delete

Updates use **PUT**, not PATCH (except `PATCH /api/tenant/settings/catalog-profile`).

Financial year errors

When a tenant has **enabled the financial year module** (Settings → Accounting) and closed a period, creating or financially editing documents dated in that **closed** financial year returns **400** with a message indicating the financial year is closed. Tenants with the module off are never period-locked. Recording a payment **today** against an older invoice remains allowed (guard uses payment date). Notes-only invoice updates remain allowed.

Inventory (Products) API

Product catalog: CRUD, batch resolve/create, search helpers, and import. Base path: `/api/inventory`.

Inventory bulk resolve uses a **different contract** than party/category bulk — see bulk resolve section (see #post-apiinventorybulk-resolve).

Endpoint summary

Method	Path	Description
GET	`/api/inventory`	List products
POST	`/api/inventory`	Create one product
GET	`/api/inventory/{id}`	Get product
PUT	`/api/inventory/{id}`	Update product
DELETE	`/api/inventory/{id}`	Soft delete
POST	`/api/inventory/bulk-resolve`	Batch product match
POST	`/api/inventory/bulk-create`	Batch create products
POST	`/api/inventory/by-ids`	Fetch up to 50 by ID
GET	`/api/inventory/lookups`	Categories, manufacturers, dealers
GET	`/api/inventory/semantic-search`	Natural language search
GET	`/api/inventory/resolve-barcode`	Lookup by barcode
POST	`/api/inventory/aliases/bulk-create`	Save search aliases
POST	`/api/inventory/import/validate`	Validate import rows
POST	`/api/inventory/import`	Execute import

Import detail: imports/products.md (see ../imports/products).

GET /api/inventory

Headers

Session or Bearer + `x-tenant-id`. Dev fallback to `seed-tenant-001` if header missing — **do not use in production**.

Query parameters

Param	Type	Description
`search`	string	Text search
`skip`	number	Offset
`take`	number	Page size (max **100** without agent header; max **50** with `x-ai-agent: true`)
`categoryId`	string	Filter
`manufacturerId`	string	Filter
`dealerId`	string	Filter
`status`	string	Product status filter
`sortBy`	string	Sort field
`sortOrder`	`asc` | `desc`	Sort
`lowStock`	string	Low-stock filter when set
`lite`	`true`	Skip `summary` and `lowStockCount` side queries (faster). Auto-enabled when `x-ai-agent: true` and `search` is set.

Response 200

When `lite=true` (or agent search), response omits `summary` and `lowStockCount`.

```
{
  "data": [
    {
      "id": "inv-uuid",
      "productName": "Widget A",
      "productCode": "SKU-1",
      "categoryId": "cat-uuid",
      "purchasePrice": 10,
      "sellingPrice": 15,
      "quantity": 100
    }
  ],
  "total": 250
}
```


Example

```
curl -s -b cookies.txt "https://erp.yallav.io/api/inventory?
search=widget&take=20&lowStock=true"
```

POST /api/inventory

Create one product.

Request body

Field	Type	Required	Constraints
`productName`	string	Yes	Min 1
`categoryId`	string	Yes	Min 1
`purchasePrice`	number	Yes	≥ 0
`sellingPrice`	number	Yes	≥ 0
`quantity`	number	Yes	Int, ≥ 0
`productCode`	string	No	SKU

`manufacturerId`	string	No	Nullable
`dealerId`	string	No	Nullable
`shelfLocation`	string	No	Nullable
`purchaseDate`	string	No	ISO date
`expirationDate`	string	No	ISO date
`stockAlertThreshold`	number	No	Int, ≥ 0
`imageUrl`	string	No	Nullable
`barcodes`	string[]	No	Max 50 items

Response 201


```
{
  "product": { "id": "inv-uuid", "productName": "Widget A", "...":
"..." },
  "warnings": []
}
```


Errors

Status	Cause
409	Duplicate code/name, barcode conflict

GET /api/inventory/{id}

Response 200


```
{ "product": { "id": "...", "productName": "...", "...": "..." } }
```

Errors

Status	Cause
404	Product not found

PUT /api/inventory/{id}

Request body (all optional)

Field	Type	Notes
`productName`	string	Min 1 if sent
`productCode`	string	Min 1 if sent
`categoryId`	string	Nullable
`manufacturerId` , `dealerId`	string	Nullable
`purchasePrice` , `sellingPrice`	number	≥ 0
`quantity`	number	Int, ≥ 0
`fundingSource`	`CASH` | `BANK` | `CAPITAL`	Required when **increasing** quantity on stock **not** linked to a delivered order, after treasury setup is complete
`status`	`USABLE` | `DAMAGED`	
`shelfLocation` , dates, `imageUrl`		
`stockAlertThreshold`	number	Int, ≥ 0
`barcodes`	string[]	Max 50

Accounting behavior (PUT)

- **Order-linked variations** (created by a delivered purchase order): changing `quantity` or `purchasePrice` updates the linked `OrderItem`, order totals, and **accounts payable**. No treasury funding voucher.
- **Other stock**: quantity **increases** post a treasury funding adjustment (`fundingSource` required when Cash & Bank setup is complete). Quantity **decreases** may reverse funding when `fundingSource` is supplied.
- **HYBRID groups**: `purchasePrice` edits recalculate the base product weighted average; `sellingPrice` syncs across the whole product group.
- **Guards**: quantity cannot be set below the amount already sold on invoices for that variation row.

Response 200

```
{ "product": { "...": "..." }, "warnings": ["Order(s) updated: ORD-0001"] }
```

`warnings` may include order propagation or treasury funding messages.

DELETE /api/inventory/{id}

Requires `x-tenant-id`. Soft-deletes the product.

Response 200

```
{ "success": true }
```

POST /api/inventory/bulk-resolve

Match up to **50** product lines against the tenant catalog (fuzzy match, aliases, optional semantic/GIN retrieval).

HYBRID tenants: resolve and search return **base products only** (`isBaseProduct: true`). Purchase-batch variations are internal; `inventoryId` in the response is always the base product id. `quantity` is **aggregated sellable stock** (sum of variation quantities in the product group). Invoice creation accepts base ids and applies FIFO deduction across variations; order creation creates a new batch variation from the base id.

Request body

```
{
  "lines": [
    {
      "productCode": "MA-332",
      "nameVariations": ["Mazda 3"]
    },
    {
      "rawQuery": "brake pads toyota",
      "categoryId": "cat-uuid"
    }
  ]
}
```

Field	Type	Description
`lines`	array	Required, 1–50
`lines[].rawQuery`	string	Free-text query
`lines[].facets`	object	Optional catalog intent (`categoryNames` , `typeTokens` , `modelTokens` , `narrowTokens` , `broadTokens` , `queryYear` , `positionTokens`)
`lines[].nameVariations`	string[]	Alternate names
`lines[].productCode` / `code`	string	SKU — **exact** normalized match only
`lines[].productName`	string	Name hint
`lines[].categoryId`	string	Limit search scope

Code fragments in `rawQuery` : When the query contains a code-like token (e.g. `1679` , `23587-1679` , Arabic-Indic digits), the server runs **substring** matching on `productCode` before fuzzy/semantic fallback (`matchReason: product_code_substring`). Trailing

chat punctuation (`؟` , `?` , `.` , etc.) does not block extraction — e.g. `عنا بريك 1679؟` resolves the same as `بريك 1679` . For large catalogs, substring resolve may query the database when the in-memory GIN candidate set misses matches. When a code token is present but no product code contains it, the line returns `not_found` (no category-only fuzzy fallback). This mirrors dashboard list search with `search=` . Multiple matches return `ambiguous` with all candidates. Years (1900–2099) and small quantity numbers are not treated as codes.

Response 200


```
{
  "lines": [
    {
      "lineIndex": 0,
      "status": "exact_match",
      "exists": true,
      "inventoryId": "inv-uuid",
      "productName": "Mazda 3",
      "productCode": "MA-332",
      "categoryName": "Brakes",
      "purchasePrice": 50,
      "sellingPrice": 75,
      "quantity": 10
    },
    {
      "lineIndex": 1,
      "status": "ambiguous",
      "exists": false,
      "candidates": [
        {
          "id": "inv-2",
          "productName": "Brake Pad A",
          "productCode": "BP-1",
          "categoryName": "Brakes",
          "purchasePrice": 20,
          "sellingPrice": 30,
          "quantity": 5
        }
      ],
      "matchReason": "multiple_fuzzy_matches"
    }
  ],
  "summary": { "exact": 1, "notFound": 0, "ambiguous": 1, "related": 0
},
  "meta": {
    "usedGinRetrieval": false,
    "semanticUsed": false,
    "catalogSize": 1200,
    "matchReasons": ["code_exact"]
  }
}
```

`status`	Action
`exact_match`	Use `inventoryId` and catalog prices
`ambiguous`	Pick from `candidates` or refine query
`related_match`	Review suggestion / `expansion`
`not_found`	Create via bulk-create or import

Example

```
curl -s -X POST "https://erp.yallav.io/api/inventory/bulk-resolve" \
  -H "Authorization: Bearer $ERP_SERVICE_SECRET" \
  -H "x-tenant-id: $TENANT_ID" \
  -H "Content-Type: application/json" \
  -d '{"lines":[{"productCode":"SKU-1"}]}'
```

POST /api/inventory/bulk-create

Create up to **50** products in one request.

Request body

```
{
  "products": [
    {
      "productName": "New Widget",
      "categoryId": "cat-uuid",
      "purchasePrice": 10,
      "sellingPrice": 15,
      "quantity": 0,
      "productCode": "NW-1",
      "manufacturerId": null,
      "dealerId": null
    }
  ]
}
```

Field	Type	Required
`products`	array	Yes, 1–50
`products[].productName`	string	Yes
`products[].categoryId`	string	Yes
`products[].purchasePrice`	number	Yes, ≥ 0
`products[].sellingPrice`	number	Yes, ≥ 0
`products[].quantity`	number	Yes, int ≥ 0
`products[].productCode`	string	No
`manufacturerId`, `dealerId`	string	No, nullable
`shelfLocation`, dates, `stockAlertThreshold`, `imageUrl`		No

Response 201

```
{
  "results": [
    { "index": 0, "success": true, "product": { "id": "uuid",
"productName": "New Widget" } }
  ],
  "summary": { "succeeded": 1, "failed": 0 }
}
```

Successful batches may trigger background tenant business profile refresh.

POST /api/inventory/by-ids

Fetch multiple products by ID (max **50**).

Request

```
{ "ids": ["inv-uuid-1", "inv-uuid-2"] }
```

Response 200

```
{ "products": [ { "id": "...", "productName": "..." } ] }
```

GET /api/inventory/lookups

Returns id/name lists for dropdowns and import resolution.

Response 200

```
{
  "categories": [{ "id": "...", "name": "Electronics" }],
  "manufacturers": [{ "id": "...", "name": "Acme Mfg" }],
  "dealers": [{ "id": "...", "name": "Main Supplier" }]
}
```

Requires `x-tenant-id`.

GET /api/inventory/parse-query

Deterministic query parsing (intent facets, name variations, ontology-expanded queries). Used by agent-service as the single source of truth for query understanding.

Query parameters

Param	Type	Required
`q`	string	Yes

Response 200


```
{
  "query": "امامي ريو",
  "intent": { "categoryNames": [], "typeTokens": [], "modelTokens":
["ريو"], "yearToken": null, "positionTokens": ["امامي"] },
  "nameVariations": ["امامي ريو"],
  "expandedQueries": ["امامي ريو", "front rio"]
}
```

POST /api/inventory/search-feedback

Record search feedback (selection, confirmed sale, not_found, admin correction) for active learning.

Body


```
{
  "feedback": [
    { "query": "امامي ريو", "inventoryId": "uuid", "feedbackType":
"selection", "source": "agent" }
  ]
}
```


Max 50 items per request.

GET /api/inventory/search-metrics

Tenant search quality metrics (exact/not-found/ambiguous rates, semantic/reranker usage).

Query parameters

Param	Type	Default
`days`	number	7 (max 30)

Environment flags (search overhaul)

Variable	Default	Purpose
`HYBRID_SEARCH_ENABLED`	`false`	Enable RRF hybrid + optional rerank in bulk-resolve
`SEARCH_RERANKER_ENABLED`	`true` when `JINA_API_KEY` set	Jina rerank over top-K
`PRODUCT_ENRICHMENT_QUEUE_ENABLED`	`false`	BullMQ async enrichment worker
`MIGRATION_ENRICHMENT_USE_QUEUE`	`false`	Use queue path on migration finalize (requires queue enabled)
`JINA_API_KEY`	—	Jina rerank API key (env only, never commit)

GET /api/inventory/parse-query

Deterministic query parsing (intent facets, name variations, ontology-expanded queries).

Query parameters

Param	Type	Required
q	string	Yes

Requires `x-tenant-id`.

POST /api/inventory/search-feedback

Record search feedback for active learning (`selection` , `confirmed_sale` , `not_found` , `admin_correction`). Max 50 items.

Requires `x-tenant-id`.

GET /api/inventory/search-metrics

Search quality metrics for the tenant (`days` query param, default 7, max 30).

Requires `x-tenant-id` .

GET /api/inventory/semantic-search

Natural language product search (embeddings). **HYBRID tenants:** returns base products only; `quantity` is aggregated sellable stock across purchase batches.

Query parameters

Param	Type	Required	Default
`q`	string	Yes	—
`limit`	number	No	10

Response 200

Array of search hits (product id, name, similarity, etc. — shape from embedding service).

Example

```
curl -s "https://erp.yallav.io/api/inventory/semantic-search?
q=red+brake+pads&limit=5" \
  -H "x-tenant-id: $TENANT_ID"
```

GET /api/inventory/resolve-barcode

Resolve a product by barcode string.

Query parameters

Param	Type	Required
`code`	string	Yes

Response 200

```
{ "product": { "id": "...", "productName": "...", "...": "..." } }
```

Errors

Status	Cause
404	Not found

POST /api/inventory/aliases/bulk-create

Persist alternate search text → `inventoryId` (max **50**).

Request

```
{
  "aliases": [
    {
      "aliasText": "mazda three",
      "inventoryId": "inv-uuid",
      "source": "user_correction"
    }
  ]
}
```

`source`	Values
Optional	`user_correction`, `agent`, `user_confirm`, `transaction_confirm`

Response 200

Per-alias results from `bulkCreateProductSearchAliases` (success/failure per row).

Integration flow

Diagram omitted in PDF — see https://erp.yallav.io/docs/erp-api-reference.pdf

---

Related

- categories.md (see categories) — required before create (`categoryId`)
- imports/products.md (see ../imports/products) — CSV row import
- sales/invoices.md (see ../sales/invoices) — `prepare-lines` uses product resolution

Categories API

Manage product categories. Base path: `/api/categories`.

Bulk patterns: conventions.md (see ../conventions).

Endpoint summary

Method	Path	Description
GET	`/api/categories`	List categories
POST	`/api/categories`	Create one category
GET	`/api/categories/{id}`	Get by ID
PUT	`/api/categories/{id}`	Update
DELETE	`/api/categories/{id}`	Soft delete
POST	`/api/categories/bulk-resolve`	Batch name lookup
POST	`/api/categories/bulk-create`	Batch create

GET /api/categories

List categories for the tenant.

Headers

- Session cookie **or** Bearer + `x-tenant-id`

Query parameters

Param	Type	Description
`search`	string	Filter by name
`skip`	number	Pagination offset

`take`	number	Page size
`sortBy`	string	Sort field
`sortOrder`	`asc` | `desc`	Sort direction

Response 200

```
{
  "data": [{ "id": "uuid", "name": "Electronics", "createdAt": "...",
"updatedAt": "..." }],
  "total": 42
}
```

Example

```
curl -s -b cookies.txt "https://erp.yallav.io/api/categories?
search=elec&take=20"
```

POST /api/categories

Create a single category.

Request body

Field	Type	Required
`name`	string	Yes (min 1 char)

Response 201

Category object (includes `id`, `name`).

Errors

Status	Cause
409	Name already exists

GET /api/categories/{id}

Response 200

Category object.

Errors

Status	Cause
404	Not found

PUT /api/categories/{id}

Request body

Field	Type	Required
`name`	string	Yes

DELETE /api/categories/{id}

Response 200

```
{ "success": true }
```

POST /api/categories/bulk-resolve

Resolve up to **50** category names. See entity bulk resolve (see ../conventions).

Request

```
{
  "lines": [{ "name": "Electronics", "nameVariations": ["Elec"] }]
}
```

Response 200

```
{
  "lines": [
    {
      "lineIndex": 0,
      "status": "exact",
      "exists": true,
      "id": "cat-uuid",
      "name": "Electronics"
    }
  ],
  "summary": { "exact": 1, "similar": 0, "none": 0 }
}
```

POST /api/categories/bulk-create

Create up to **50** categories.

Request

```
{
  "categories": [{ "name": "New Category" }]
}
```

Response 201

```
{
  "results": [{ "index": 0, "success": true, "category": { "id":
"uuid", "name": "New Category" } }],
  "summary": { "succeeded": 1, "failed": 0 }
}
```

Integration flow

Diagram omitted in PDF — see https://erp.yallav.io/docs/erp-api-reference.pdf

Resolve categories before creating products that require `categoryId`.

Manufacturers API

Manage product manufacturers. Base path: `/api/manufacturers`.

There is **no** `bulk-create` endpoint; use single `POST` or resolve-only bulk for lookups.

Endpoint summary

Method	Path	Description
GET	`/api/manufacturers`	List
POST	`/api/manufacturers`	Create one
GET	`/api/manufacturers/{id}`	Get by ID
PUT	`/api/manufacturers/{id}`	Update
DELETE	`/api/manufacturers/{id}`	Soft delete
POST	`/api/manufacturers/bulk-resolve`	Batch name lookup (max 50)

GET /api/manufacturers

Query parameters

Param	Type	Description
`search`	string	Name filter
`skip`, `take`	number	Pagination
`sortBy`, `sortOrder`	string	Sorting

Headers

Session or Bearer + `x-tenant-id`.

Response 200

Paginated list (`data` , `total`).

POST /api/manufacturers

Request body

Field	Type	Required
`name`	string	Yes

Response 201

Manufacturer object.

Errors

Status	Cause
409	Duplicate name

POST /api/manufacturers/bulk-resolve

Same contract as category bulk resolve (see categories). Response lines include `manufacturerId` and `manufacturerName` aliases.

Request example

```
{
  "lines": [{ "name": "Toyota" }]
}
```

Response example


```
{
  "lines": [
    {
      "lineIndex": 0,
      "status": "exact",
      "exists": true,
      "id": "mfg-uuid",
      "name": "Toyota",
      "manufacturerId": "mfg-uuid",
      "manufacturerName": "Toyota"
    }
  ],
  "summary": { "exact": 1, "similar": 0, "none": 0 }
}
```

Related

- `GET /api/inventory/lookups` returns all manufacturers (with categories and dealers) in one call.
- Optional on products: `manufacturerId` on inventory create/update.

Customers API

Customer (buyer) master data. Base path: `/api/customers`.

Bulk patterns: conventions.md (see ../conventions).

Endpoint summary

Method	Path	Description
GET	`/api/customers`	List customers
POST	`/api/customers`	Create one
GET	`/api/customers/{id}`	Get by ID
PUT	`/api/customers/{id}`	Update
DELETE	`/api/customers/{id}`	Soft delete
POST	`/api/customers/bulk-resolve`	Batch name lookup
POST	`/api/customers/bulk-create`	Batch create
GET	`/api/customers/receivables-summary`	Aggregate customer AR (total outstanding + top debtors)
GET	`/api/customers/{id}/account-payments`	Account balance summary
POST	`/api/customers/{id}/account-payments`	Record account payment

GET /api/customers

Headers

Session or Bearer + `x-tenant-id`. With `x-ai-agent: true`, `take` capped at 50.

Query parameters

Param	Type	Description
`search`	string	Name/company/mobile search
`skip`	number	Offset
`take`	number	Page size

Response 200


```
{
  "data": [
    {
      "id": "cust-uuid",
      "name": "Acme Retail",
      "companyName": "Acme LLC",
      "mobileNumber": "+962790000000"
    }
  ],
  "total": 80
}
```


Example

```
curl -s -b cookies.txt "https://erp.yallav.io/api/customers?
search=acme&take=20"
```

POST /api/customers

Request body

Field	Type	Required
`name`	string	Yes

`companyName`	string	No, nullable
`mobileNumber`	string	No, nullable

Response 201

Full customer object.

Errors

Status	Cause
409	Duplicate or invalid mobile

GET /api/customers/{id}

Response 200

Customer object.

PUT /api/customers/{id}

Same fields as create (partial update per service rules).

DELETE /api/customers/{id}

Soft delete. Response `{ "success": true }`.

POST /api/customers/bulk-resolve

Max **50** lines. See conventions (see ../conventions).

Request

```
{
  "lines": [{ "name": "Acme", "nameVariations": ["ACME Corp"] }]
}
```

Response

Lines include `customerId` and `customerName` aliases.

```
{
  "lines": [
    {
      "lineIndex": 0,
      "status": "exact",
      "exists": true,
      "id": "cust-uuid",
      "name": "Acme",
      "customerId": "cust-uuid",
      "customerName": "Acme"
    }
  ],
  "summary": { "exact": 1, "similar": 0, "none": 0 }
}
```

POST /api/customers/bulk-create

Request

```
{
  "customers": [
    { "name": "New Shop", "companyName": null, "mobileNumber": null }
  ]
}
```

Response 201

```
{
  "results": [
    { "index": 0, "success": true, "customer": { "id": "uuid", "name":
"New Shop" } }
  ],
  "summary": { "succeeded": 1, "failed": 0 }
}
```

GET /api/customers/receivables-summary

Aggregate accounts receivable across all customers (sum of open invoice `remainingBalance`).

Headers

Bearer or session + `x-tenant-id` (+ `x-user-id` for agent calls).

Response 200

```
{
  "totalOutstanding": 12500.5,
  "customerCountWithDebt": 12,
  "customers": [
    {
      "id": "cust-uuid",
      "name": "Acme Retail",
      "totalOutstanding": 3200
    }
  ]
}
```

- `totalOutstanding` — sum of all unpaid invoice balances for the tenant.
- `customerCountWithDebt` — number of customers with at least one open invoice balance.

- `customers` — up to 50 customers ordered by highest outstanding (for agent/UI drill-down).

Example

```
curl -s -H "Authorization: Bearer $TOKEN" -H "x-tenant-id: $TENANT" \
  "https://erp.yallav.io/api/customers/receivables-summary"
```

GET /api/customers/{id}/account-payments

Customer running balance / credit summary for on-account sales.

Response 200

Account summary from `accountPaymentService.getAccountSummary` (balance, credit available, recent activity — exact fields from service).

POST /api/customers/{id}/account-payments

Record a payment against customer account (multi-line payment schema).

Uses `accountPaymentSchema` from `src/lib/paymentSchemas` (payment lines, methods, optional `useCredit`, `date`, `notes`).

Response 200

Payment result from `recordAccountPayment`.

Errors

Status	Cause
400	Zod validation
404	Customer not found

Integration with invoices

Diagram omitted in PDF — see https://erp.yallav.io/docs/erp-api-reference.pdf

See sales/invoices.md (see ../sales/invoices).

Suppliers (Dealers) API

Purchase-side suppliers are stored as **dealers** in the data model. The UI label is “supplier”; API paths support both names.

Path alias: `/api/suppliers/*` re-exports `/api/dealers/*` handlers. Document once, call either path.

Bulk patterns: conventions.md (see ../conventions).

Endpoint summary

Method	Path (`/api/dealers` or `/api/suppliers`)	Description
GET	`.../`	List suppliers
POST	`.../`	Create one
GET	`.../{id}`	Get by ID
PUT	`.../{id}`	Update
DELETE	`.../{id}`	Soft delete
POST	`.../bulk-resolve`	Batch name lookup
POST	`.../bulk-create`	Batch create
GET	`.../{id}/account-payments`	Account summary
POST	`.../{id}/account-payments`	Record payment
POST	`.../{id}/account-payments/refund-credit`	Refund credit

GET /api/dealers

Same as `GET /api/suppliers`.

Headers

Session or Bearer + `x-tenant-id`. `x-ai-agent: true` caps `take` at 50.

Query parameters

Param	Type	Description
`search`	string	Filter
`skip` , `take`	number	Pagination
`sortBy` , `sortOrder`	string	Sort

Response 200


```
{
  "data": [
    {
      "id": "dealer-uuid",
      "name": "Main Parts Co",
      "companyName": null,
      "mobileNumber": null
    }
  ],
  "total": 15
}
```

POST /api/dealers

Request body

Field	Type	Required
`name`	string	Yes (labeled “supplier name” in validation messages)
`companyName`	string	No, nullable
`mobileNumber`	string	No, nullable

Response 201

Dealer (supplier) object.

Errors

Status	Cause
409	Name already exists

POST /api/dealers/bulk-resolve

Also available at `POST /api/suppliers/bulk-resolve`.

Request

```
{
  "lines": [{ "name": "Main Parts Co" }]
}
```

Response

Lines include `dealerId` and `dealerName` aliases (same as `id` / `name`).

```
{
  "lines": [
    {
      "lineIndex": 0,
      "status": "exact",
      "exists": true,
      "dealerId": "uuid",
      "dealerName": "Main Parts Co"
    }
  ],
  "summary": { "exact": 1, "similar": 0, "none": 0 }
}
```

POST /api/dealers/bulk-create

Body key is **`dealers`** (not `suppliers`), max **50**.

Request

```
{
  "dealers": [{ "name": "New Supplier Ltd" }]
}
```

Also: `POST /api/suppliers/bulk-create`.

Response 201

```
{
  "results": [
    { "index": 0, "success": true, "dealer": { "id": "uuid", "name":
"New Supplier Ltd" } }
  ],
  "summary": { "succeeded": 1, "failed": 0 }
}
```

Account payments

`GET` / `POST /api/dealers/{id}/account-payments` mirror customer account APIs with `side: 'dealer'` in the service.

Supplier alias paths: `/api/suppliers/{id}/account-payments`.

Usage in orders and inventory

- `POST /api/orders` optional `dealerId`
- Product optional `dealerId` on inventory create

- `GET /api/inventory/lookups` includes dealers list

See sales/orders.md (see ../sales/orders) and catalog/inventory.md (see ../catalog/inventory).

Invoices API

Sales invoices (accounts receivable). Base path: `/api/invoices`.

There is **no** `bulk-resolve` / `bulk-create` for invoices. Use **`prepare-lines`** to resolve customers and products, then **`POST /api/invoices`** to create.

Enums: conventions.md (see ../conventions).

Endpoint summary

Method	Path	Description
GET	`/api/invoices`	List invoices
POST	`/api/invoices`	Create invoice
GET	`/api/invoices/{id}`	Get invoice (via PUT route file pattern — use list or service)
PUT	`/api/invoices/{id}`	Update invoice
DELETE	`/api/invoices/{id}`	Soft delete
POST	`/api/invoices/prepare-lines`	Resolve customer + lines before create
POST	`/api/invoices/{id}/payments`	Record payment
POST	`/api/invoices/import/validate`	Validate import rows
POST	`/api/invoices/import`	Import invoices

Import: imports/invoices.md (see ../imports/invoices).

GET /api/invoices

Headers

Session or Bearer + `x-tenant-id`. Dev fallback `seed-tenant-001` if header invalid — not for production.

`x-ai-agent: true` caps `take` at 50.

Query parameters

Param	Type	Description
`search`	string	Text search
`skip`, `take`	number	Pagination
`customerId`	string	Filter by customer
`paymentStatus`	enum	`UNPAID`, `PARTIALLY_PAID`, `PAID`, `OVERPAID`
`paymentMethod`	enum	`CASH`, `BANK_TRANSFER`, `CHEQUE`
`fromDate`, `toDate`	string	Date range
`sortBy`, `sortOrder`	string	Sort

Response 200


```
{
  "data": [
    {
      "id": "inv-doc-uuid",
      "customerId": "cust-uuid",
      "total": 1500,
      "paymentStatus": "UNPAID",
      "invoiceDate": "2026-06-01T00:00:00.000Z"
    }
  ],
  "total": 200
}
```

POST /api/invoices

Create a sales invoice with line items referencing existing inventory IDs.

Request body

Field	Type	Required	Constraints
`customerId`	string	No	Nullable
`invoiceDate`	string	No	ISO date
`discount`	number	No	≥ 0
`amountPaid`	number	No	≥ 0 (initial payment)
`paymentMethod`	enum	No	`CASH`, `BANK_TRANSFER`, `CHEQUE`
`items`	array	Yes	Min 1 line
`items[].inventoryId`	string	Yes	Product ID
`items[].quantity`	number	Yes	Int, > 0
`items[].unitPrice`	number	Yes	≥ 0
`notes`	string	No	Nullable
`attachments`	string[]	No	URLs/paths

Response 201

```
{
  "invoice": {
    "id": "inv-doc-uuid",
    "customerId": "cust-uuid",
    "items": [],
    "total": 1500,
    "paymentStatus": "UNPAID"
  },
  "creditApplied": 0
}
```

`creditApplied` reflects customer account credit applied at creation.

Example

```
curl -s -X POST "https://erp.yallav.io/api/invoices" \
  -H "Authorization: Bearer $ERP_SERVICE_SECRET" \
  -H "x-tenant-id: $TENANT_ID" \
  -H "x-user-id: $USER_ID" \
  -H "Content-Type: application/json" \
  -d '{
    "customerId": "cust-uuid",
    "paymentMethod": "CASH",
    "items": [
      { "inventoryId": "prod-uuid", "quantity": 2, "unitPrice": 75 }
    ]
  }'
```

POST /api/invoices/prepare-lines

Resolve customer names and product lines (by code/name), optionally create missing lookups, and return an import plan. Max **50** lines (`MAX_PREPARE_INVOICE_LINES`).

Use this before `POST /api/invoices` when line data comes from CSV, OCR, or free text.

Request body

Field	Type	Required
`customerName`	string	No (default for all lines)
`lines`	array	Yes, 1–50

Each line (extends invoice import row schema):

Field	Type	Required
`customerName`	string	No (overrides top-level)
`productName`	string	Yes
`productCode`	string	No
`category`	string	No

`quantity`	number	Yes, ≥ 1
`unitPrice`	number	Yes, ≥ 0
`discount`	number	No (defaults 0)

Response 200


```
{
  "lines": [
    {
      "lineIndex": 0,
      "isValid": true,
      "errors": [],
      "data": {
        "productName": "Widget",
        "quantity": 2,
        "unitPrice": 50
      },
      "resolvedIds": {
        "customerId": "cust-uuid",
        "inventoryId": "prod-uuid",
        "categoryId": "cat-uuid"
      },
      "productStatus": "EXISTING",
      "exists": true,
      "matchedBy": "code",
      "stockToAdd": 0
    }
  ],
  "createdLookups": { "customers": [], "categories": [] },
  "importPlan": {
    "existingProductsUsed": 1,
    "newProductsToCreate": 0,
    "productsToRestock": 0,
    "totalStockToAdd": 0
  },
  "summary": { "valid": 1, "invalid": 0 },
  "message": "All lines ready"
}
```


`productStatus`	Meaning

`EXISTING`	Product matched in catalog
`NEW`	Will be created on import/commit

Example

```
curl -s -X POST "https://erp.yallav.io/api/invoices/prepare-lines" \
  -H "x-tenant-id: $TENANT_ID" \
  -H "Content-Type: application/json" \
  -d '{
    "customerName": "Acme Shop",
    "lines": [
      { "productName": "Widget A", "productCode": "W-1", "quantity": 1,
"unitPrice": 100 }
    ]
  }'
```

PUT /api/invoices/{id}

Update invoice header and line items per `InvoiceService` rules.

Requires `x-tenant-id` .

Item edit — inventory (variation) quantities

When `items` is sent:

1. Stock previously sold on this invoice is **restored** to each variation (`invoiceItem.inventoryId`).
2. New line quantities are applied. Lines with the same `id` and **quantity** keep the **same variation row** (price-only edits do not re-run FIFO).
3. Changed quantities or new products run **FIFO** deduction (HYBRID: oldest in-stock variations in the product group).
4. Net effect: variation quantities reflect the new sold amounts; stock is not lost across edits.

Line `unitPrice` updates change invoice revenue only — catalog `sellingPrice` is not modified.

Request body (items excerpt)

Field	Type	Notes
`items[].id`	string	Existing `InvoiceItem` id when editing a line
`items[].inventoryId`	string	Base or variation product id
`items[].quantity`	number	Int, > 0
`items[].unitPrice`	number	≥ 0

DELETE /api/invoices/{id}

Soft delete invoice.

POST /api/invoices/{id}/payments

Record payment on an existing invoice.

Request body (single payment)

Field	Type	Required
`amount`	number	Yes, ≥ 0
`paymentMethod`	enum	No
`date`	string	No
`reference`	string	No
`notes`	string	No
`chequeId`	string	No
`chequeInfo`	object	No

Request body (composite)

```
{
  "type": "composite",
  "date": "2026-06-03",
  "notes": "Split payment",
  "lines": [
    { "amount": 500, "paymentMethod": "CASH" },
    { "amount": 500, "paymentMethod": "BANK_TRANSFER", "reference":
"TRX-1" }
  ]
}
```

Parsed by `parsePaymentBody` in `src/lib/paymentSchemas.ts`.

Response 200

```
{ "invoice": { "id": "...", "paymentStatus": "PAID", "...": "..." } }
```

Integration flow

Diagram omitted in PDF — see https://erp.yallav.io/docs/erp-api-reference.pdf

---

Related

- parties/customers.md (see ../parties/customers)
- catalog/inventory.md (see ../catalog/inventory)
- imports/invoices.md (see ../imports/invoices)

Orders API

Purchase orders (stock inbound from suppliers). Base path: `/api/orders`.

Endpoint summary

Method	Path	Description
GET	`/api/orders`	List orders
POST	`/api/orders`	Create purchase order
GET	`/api/orders/{id}`	Get order
PUT	`/api/orders/{id}`	Update
DELETE	`/api/orders/{id}`	Soft delete
POST	`/api/orders/{id}/deliver`	Mark delivered (stock in)
POST	`/api/orders/{id}/payments`	Record payment
POST	`/api/orders/import/validate`	Validate import rows
POST	`/api/orders/import`	Import orders

Import: imports/orders.md (see ../imports/orders).

GET /api/orders

Headers

Session or Bearer + `x-tenant-id`. Dev `seed-tenant-001` fallback — not for production.

`x-ai-agent: true` caps `take` at 50.

Query parameters

Param	Type	Description

`search`	string	Filter
`skip` , `take`	number	Pagination
`dealerId`	string	Supplier filter
`manufacturerId`	string	Filter
`paymentStatus`	enum	Payment status
`orderStatus`	enum	`PENDING` , `DELIVERED`
`fromDate` , `toDate`	string	Date range
`sortBy` , `sortOrder`	string	Sort

Response 200

Paginated `{ data, total }` with order summaries.

POST /api/orders

Create a purchase order.

Request body

Field	Type	Required	Constraints
`orderName`	string	No	Label
`dealerId`	string	No	Supplier ID, nullable
`manufacturerId`	string	No	Nullable
`deliveryDate`	string	Yes	ISO date
`orderStatus`	enum	No	`PENDING` , `DELIVERED`
`amountPaid`	number	No	≥ 0
`discount`	number	No	≥ 0
`items`	array	Yes	Min 1
`items[].inventoryId`	string	Yes	Product
`items[].quantity`	number	Yes	Int, > 0

`items[].purchasePrice`	number	Yes	≥ 0
`items[].sellingPrice`	number	No	≥ 0
`items[].discount`	number	No	≥ 0
`items[].bonusQuantity`	number	No	Int, ≥ 0
`notes`	string	No	Nullable
`attachments`	string[]	No	

Response 201


```
{
  "order": {
    "id": "order-uuid",
    "dealerId": "dealer-uuid",
    "orderStatus": "PENDING",
    "items": []
  },
  "creditApplied": 0
}
```

Example

```
curl -s -X POST "https://erp.yallav.io/api/orders" \
  -H "x-tenant-id: $TENANT_ID" \
  -H "x-user-id: $USER_ID" \
  -H "Content-Type: application/json" \
  -d '{
    "dealerId": "dealer-uuid",
    "deliveryDate": "2026-06-10",
    "items": [
      {
        "inventoryId": "prod-uuid",
        "quantity": 10,
        "purchasePrice": 25,
        "sellingPrice": 40
      }
    ]
  }'
```

POST /api/orders/{id}/deliver

Mark order as delivered and apply stock increases per business rules.

Request

No body required.

Response 200

```
{ "order": { "id": "...", "orderStatus": "DELIVERED", "...": "..." } }
```

Requires `x-tenant-id` and recommended `x-user-id`.

POST /api/orders/{id}/payments

Same payment patterns as invoices (`parsePaymentBody` : single or composite lines). See invoices.md (see invoices).

PUT /api/orders/{id}

Update order per service validation.

DELETE /api/orders/{id}

Soft delete.

Integration flow

Diagram omitted in PDF — see https://erp.yallav.io/docs/erp-api-reference.pdf

---

Related

- parties/suppliers.md (see ../parties/suppliers)
- catalog/inventory.md (see ../catalog/inventory)
- imports/orders.md (see ../imports/orders)

Import API Overview

Bulk data onboarding uses a **validate → commit** pattern for products, invoices, and purchase orders. Generic CSV mapping helpers support dashboard/AI uploads stored in Redis.

Two-step domain import

Diagram omitted in PDF — see https://erp.yallav.io/docs/erp-api-reference.pdf

Step	Purpose
Validate	Parse rows, resolve names to IDs, report per-row errors and pending lookups
Import	Persist documents and optionally create missing catalog rows

Each validate endpoint accepts a JSON array of row objects (same shape as CSV columns after mapping). Large imports may batch internally (e.g. 100 rows per batch, safety cap ~5000).

Domain import endpoints

Domain	Validate	Import
Products	`POST /api/inventory/import/validate`	`POST /api/inventory/import`
Invoices	`POST /api/invoices/import/validate`	`POST /api/invoices/import`
Orders	`POST /api/orders/import/validate`	`POST /api/orders/import`

Detail:

- products.md (see products)
- invoices.md (see invoices)
- orders.md (see orders)

Generic CSV helpers (`/api/import/csv`)

Used when a file was uploaded via the dashboard chat CSV flow (stored under `uploadId` in Redis). Integrators with their own CSV pipeline can still call these after obtaining an `uploadId` from the upload endpoint (dashboard-only; not fully documented here).

Method	Path	Description
POST	`/api/import/csv/profile`	Inspect upload headers and samples
POST	`/api/import/csv/suggest-mapping`	Suggest column → field mapping
POST	`/api/import/csv/apply-plan`	Apply mapping and return normalized rows

POST /api/import/csv/profile

Body: `{ "uploadId": "string" }`

Response: Profile object (headers, sample rows, warnings) from Redis.

Errors: **404** if upload expired or missing.

POST /api/import/csv/suggest-mapping

Body:

Field	Type	Required
`uploadId`	string	Yes
`profile`	string	No
`contextKeys`	string[]	No

Response: Suggested column mapping (shape from `suggestCsvUploadMapping`).

POST /api/import/csv/apply-plan

Body:

Field	Type	Required
`uploadId`	string	Yes
`profile`	string	No

`columnMapping`	object	No
`context`	object	No
`defaults`	object	No
`includeUnmapped`	boolean	No

Response 200:

```
{
  "uploadId": "abc",
  "rowCount": 120,
  "rows": [ { "...": "normalized row" } ]
}
```

Pass `rows` to the appropriate `import/validate` endpoint.

Authentication

All import routes require `x-tenant-id` (session or Bearer). Long-running imports set `maxDuration = 300` seconds on the route.

Recommended integrator path (no dashboard upload)

1. Map your CSV columns locally to the row schema in products.md (see products), invoices.md (see invoices), or orders.md (see orders).
2. `POST .../import/validate` with `{ "rows": [ ... ] }`.
3. Fix invalid rows from per-row `errors`.
4. `POST .../import` with validated payload (often `{ "rows": validRows }` or structure returned by validate — see each module doc).

For interactive name resolution before invoice create, prefer `POST /api/invoices/prepare-lines` (max 50 lines) instead of full import.

Product Import API

Import products from structured rows (CSV mapped to JSON). Paths under `/api/inventory/import`.

Overview: overview.md (see overview).

POST /api/inventory/import/validate

Validate rows and resolve category, manufacturer, and supplier names to IDs.

Headers

- `x-tenant-id` required

Request body

Field	Type	Required
`rows`	array	Yes

Optional query/body pagination for large files (see route): `offset`, `limit` in body for chunked validation responses.

Max rows: 5000 per request (`MAX_SAFETY_CAP`).

Row schema (each element of `rows`)

Field	Type	Required	Constraints
`productName`	string	Yes	
`category`	string	Yes	Category name (resolved or created on import)
`purchasePrice`	number	Yes	> 0
`sellingPrice`	number	Yes	> 0
`quantity`	number	Yes	≥ 0

`productCode`	string	No	SKU
`manufacturer`	string	No	Name
`dealer`	string	No	Supplier name
`shelfLocation`	string	No	
`stockAlertThreshold`	number	No	≥ 0

Per-row validation result

```
{
  "isValid": true,
  "errors": [],
  "data": { "productName": "Widget", "category": "Parts", "...": "..."
},
  "resolvedIds": {
    "categoryId": "cat-uuid",
    "manufacturerId": "mfg-uuid",
    "dealerId": "dealer-uuid"
  },
  "pendingLookups": {
    "category": false,
    "manufacturer": false,
    "dealer": false
  }
}
```

`pendingLookups.*: true` means the name will be created on import if not `dryRun`.

Response 200 (aggregate)

Includes `results` (per row), `summary` counts, `pendingLookups` lists, and pagination metadata when chunking.

Errors

Status	Cause
400	Missing `rows`, too many rows, invalid row shape

Example

```
curl -s -X POST "https://erp.yallav.io/api/inventory/import/validate" \
  -H "x-tenant-id: $TENANT_ID" \
  -H "Content-Type: application/json" \
  -d '{
    "rows": [
      {
        "productName": "Widget A",
        "productCode": "W-1",
        "category": "Electronics",
        "purchasePrice": 10,
        "sellingPrice": 15,
        "quantity": 100
      }
    ]
  }'
```

POST /api/inventory/import

Commit validated rows (creates products and pending categories/manufacturers/dealers as needed).

Request

Typically `{ "rows": [ ... ] }` with the same row shape; use rows marked valid from validate step. Route may accept `validRows` or full validation output — send the structure your validate response documents for commit.

Requires `x-tenant-id` and recommended `x-user-id`.

Response 200 / 201

Import summary with created/updated counts and per-row outcomes (see `bulkInventoryImport` service).

Notes

- Long-running: `maxDuration = 300` seconds.
- Duplicate product codes may fail per row without aborting the whole batch.

- Prefer bulk-create (see ../catalog/inventory) for up to 50 known `categoryId` rows without CSV.

Related

- catalog/inventory.md (see ../catalog/inventory)
- import/csv profile (see ../imports/overview) for column mapping before validate

Invoice Import API

Bulk create sales invoices from rows. Paths under `/api/invoices/import`.

For ≤50 interactive lines, consider `POST /api/invoices/prepare-lines` instead (sales/invoices.md (see ../sales/invoices)).

POST /api/invoices/import/validate

Headers

`x-tenant-id` required.

Query parameters

Param	Default	Description
`dryRun`	`true`	Set `dryRun=false` to persist pending customer/category lookups during validation

Request body

```
{
  "rows": [
    {
      "customerName": "Acme Shop",
      "productName": "Widget",
      "productCode": "W-1",
      "category": "Parts",
      "quantity": 2,
      "unitPrice": 50,
      "discount": 0
    }
  ]
}
```

Field	Type	Required
`rows`	array	Yes

Row fields

Field	Type	Required
`productName`	string	Yes
`quantity`	number	Yes, ≥ 1
`unitPrice`	number	Yes, ≥ 0
`customerName`	string	No
`productCode`	string	No
`category`	string	No
`discount`	number	No (default 0)

Response 200


```
{
  "results": [
    {
      "isValid": true,
      "errors": [],
      "data": { "...": "..." },
      "resolvedIds": {
        "customerId": "cust-uuid",
        "inventoryId": "prod-uuid",
        "categoryId": "cat-uuid"
      },
      "importPlan": {
        "productStatus": "EXISTING",
        "matchedBy": "code",
        "stockToAdd": 0
      }
    }
  ],
  "createdLookups": {},
  "importPlan": {
    "existingProductsUsed": 1,
    "newProductsToCreate": 0,
    "productsToRestock": 0,
    "totalStockToAdd": 0
  },
  "summary": { "valid": 1, "invalid": 0 },
  "message": "..."
}
```


Example

```
curl -s -X POST "https://erp.yallav.io/api/invoices/import/validate?
dryRun=true" \
  -H "x-tenant-id: $TENANT_ID" \
  -H "Content-Type: application/json" \
  -d '{"rows":
[{"customerName":"Acme","productName":"Widget","quantity":1,"unitPrice":1
```

POST /api/invoices/import

Commit import after successful validation.

Request

Validated rows or wrapper expected by `invoiceService` / import route (typically rows with resolved IDs and valid flags from validate step).

Requires `x-tenant-id`, `x-user-id` recommended.

Response

Created invoice summaries and per-row errors for partial failure.

Notes

- May create customers, categories, and products when `pendingLookups` were true during validate with `dryRun=false`.
- Stock adjustments follow `importPlan.stockToAdd` for existing products.

Flow

Diagram omitted in PDF — see https://erp.yallav.io/docs/erp-api-reference.pdf

---

Related

- sales/invoices.md (see ../sales/invoices)
- imports/overview.md (see overview)

Order Import API

Bulk create purchase orders from rows. Paths under `/api/orders/import`.

POST /api/orders/import/validate

Headers

`x-tenant-id` required.

Request body

```
{
  "rows": [
    {
      "orderName": "PO-2026-001",
      "deliveryDate": "2026-06-15",
      "orderStatus": "PENDING",
      "productCode": "SKU-1",
      "productName": "Widget",
      "quantity": 10,
      "purchasePrice": 25,
      "bonusQuantity": 0,
      "dealer": "Main Supplier",
      "manufacturer": "Acme Mfg"
    }
  ]
}
```

Row schema

Field	Type	Required	Constraints
`deliveryDate`	string	Yes	
`productCode`	string	Yes	
`quantity`	number	Yes	≥ 1

`purchasePrice`	number	Yes	≥ 0
`orderName`	string	No	
`orderStatus`	string	No	
`productName`	string	No	
`bonusQuantity`	number	No	≥ 0
`dealer`	string	No	Supplier name
`manufacturer`	string	No	

Products are matched against catalog by code (and name). Dealers/manufacturers resolved by name similar to product import.

Response 200

Per-row `results` with `isValid`, `errors`, `resolvedIds`, `pendingLookups`, plus batch `summary` and optional `pendingLookups` aggregate lists.

Max **5000** rows per request (safety cap).

Example

```
curl -s -X POST "https://erp.yallav.io/api/orders/import/validate" \
  -H "x-tenant-id: $TENANT_ID" \
  -H "Content-Type: application/json" \
  -d '{
    "rows": [
      {
        "deliveryDate": "2026-06-15",
        "productCode": "SKU-1",
        "quantity": 5,
        "purchasePrice": 20
      }
    ]
  }'
```

POST /api/orders/import

Commit purchase orders from validated rows.

Requires `x-tenant-id`. May create products/suppliers when validation marked pending lookups.

`maxDuration = 300` seconds.

Response

Order creation summary with per-group or per-row success/failure (see `OrderService` import implementation).

Related

- sales/orders.md (see ../sales/orders)
- parties/suppliers.md (see ../parties/suppliers)
- imports/overview.md (see overview)

Analytics API

Read-only reporting and dashboard metrics. Base path: `/api/analytics`.

Endpoint summary

Method	Path	Description
GET	`/api/analytics/dashboard-bundle`	All dashboard widgets in one call
GET	`/api/analytics/{model}`	Single analytics model

GET /api/analytics/dashboard-bundle

Aggregated dashboard data for a date range (defaults to current calendar month).

Headers

Session or Bearer + `x-tenant-id`. Dev fallback `seed-tenant-001` — not for production.

Query parameters

Param	Type	Default	Description
`from`	string	Start of current month	ISO date
`to`	string	End of current month	ISO date
`take`	number	`10`	Top-selling item count

Response 200

```
{
  "widgets": { "...": "KPI widgets" },
  "trend": [ { "date": "...", "sales": 0, "...": "..." } ],
  "topSelling": [ { "inventoryId": "...", "name": "...", "qty": 0 } ],
  "lowStock": [ { "...": "..." } ],
  "expiring": [ { "...": "..." } ],
  "recommendations": [ { "...": "AI hints" } ],
  "dateRange": { "from": "2026-06-01T...", "to": "2026-06-30T..." }
}
```

Exact widget keys come from `AnalyticsService.getDashboardWidgets`.

Example

```
curl -s "https://erp.yallav.io/api/analytics/dashboard-bundle?
from=2026-01-01&to=2026-06-30&take=15" \
  -H "x-tenant-id: $TENANT_ID"
```

GET /api/analytics/{model}

Fetch one analytics dataset.

Path parameter `model`

model	Description
`dashboard`	Dashboard widgets only
`sales`	Sales detail with summary
`profits`	Profit detail
`recommendations`	AI recommendations
`top-selling`	Top selling items

`trend`	Sales trend series
`low-stock`	Low stock alerts
`expiring`	Near-expiry products
`expenses-detail`	Expense breakdown

Invalid model → **400** `{ "message": "Invalid model" }`.

Query parameters

Param	Type	Description
`from`, `to`	string	Date range (optional; some models use month default)
`categoryId`	string	Filter
`manufacturerId`	string	Filter
`inventoryId`	string	Filter
`customerId`	string	Filter
`search`	string	Text filter
`take`	number	Limit (top-selling)
`months`	number	Trend months

Response 200

JSON shape depends on `model` (array or object from `AnalyticsService`).

Examples

```
# Sales detail
curl -s "https://erp.yallav.io/api/analytics/sales?from=2026-01-
01&to=2026-06-30" \
  -H "x-tenant-id: $TENANT_ID"

# Low stock (no date range required)
curl -s "https://erp.yallav.io/api/analytics/low-stock" \
  -H "x-tenant-id: $TENANT_ID"
```

Errors

Status	Body
400	Invalid model
500	`{ "message": "Internal server error" }`

Related

Read-only treasury: treasury.md (see treasury).

Treasury API (Read-only)

Company cash and bank position plus movement history. Documented routes are **read-only** for integrators.

Base path: `/api/accountant/treasury`.

Endpoint summary

Method	Path	Description
GET	`/api/accountant/treasury/summary`	Balances and setup status
GET	`/api/accountant/treasury/transactions`	Paginated movements

Write routes (cash-in, transfer, owner withdrawal, vouchers) exist for the dashboard but are outside this developer subset.

GET /api/accountant/treasury/summary

Headers

`x-tenant-id` required.

Query parameters

Param	Type	Description
`asOfDate`	string	Snapshot date
`fromDate`	string	Period start (owner withdrawals in period)
`toDate`	string	Period end

Response 200


```
{
  "balances": {
    "cashOnHand": 10000,
    "bankBalance": 25000,
    "total": 35000
  },
  "ownerWithdrawalsInPeriod": 500,
  "hasPeriodFilter": true,
  "setupCompleted": true,
  "openingCash": 5000,
  "openingBank": 20000
}
```


Exact `balances` keys follow `treasuryMovementService.getSummaryMetrics`.

Example

```
curl -s "https://erp.yallav.io/api/accountant/treasury/summary?
fromDate=2026-06-01&toDate=2026-06-30" \
  -H "x-tenant-id: $TENANT_ID"
```

GET /api/accountant/treasury/transactions

Paginated treasury voucher / movement list.

Headers

`x-tenant-id` required. `x-ai-agent: true` caps `take` at 50 (default take 50).

Query parameters

Param	Type	Default	Description
`skip`	number	`0`	Offset

`take`	number	`50`	Page size
`fromDate`	string	—	Filter start
`toDate`	string	—	Filter end
`type`	enum	—	`TreasuryVoucherType`
`source`	enum	—	`TreasuryVoucherSource`

Response 200


```
{
  "data": [
    {
      "id": "voucher-uuid",
      "type": "CASH_IN",
      "amount": 1000,
      "date": "2026-06-01T00:00:00.000Z",
      "source": "MANUAL",
      "notes": null
    }
  ],
  "total": 42
}
```


Field names match `treasuryMovementService.listTransactions`.

Example

```
curl -s "https://erp.yallav.io/api/accountant/treasury/transactions?
take=20&fromDate=2026-06-01" \
  -H "x-tenant-id: $TENANT_ID"
```

Errors

Status	Body

401	`{ "error": "Unauthorized" }`
500	`{ "error": "<message>" }`

Related

Payment methods on invoices/orders: conventions.md (see conventions).

Negative cash / bank balances

Cash and bank **ledger** balances (Cash & Bank dashboard) may go **negative** when a payment or manual outflow exceeds available funds. The operation completes with a `cash_negative` or `bank_negative` warning rather than blocking.

For **Chart of Accounts / balance sheet** presentation, negative ledger amounts are **reclassified**:

- Asset leaves `cash` / `bank` show `max(0, ledger)`.
- Liability leaves `cash_overdraft` / `bank_overdraft` show `abs(min(0, ledger))`.

Treasury journal entries are unchanged; reclassification is COA resolver presentation only.

Opening balance setup (dashboard)

`GET/POST /api/accountant/treasury/setup` — opening cash, bank, and **opening capital** (equity offset for legacy books before lock).

Balance sheet at lock (matches Chart of Accounts capital resolver):

```
targetCapitalCoa = totalAssets − totalLiabilities − cumulativeRetained + withdrawals
requiredOpeningCapital = targetCapitalCoa − openingCash − openingBank − capitalFundedAssets − manualSuspenseEquity
```

- **Capital-funded fixed assets** (`AssetFundingSource.CAPITAL`) already increase the Capital COA leaf; the wizard subtracts them so opening capital is the **additional** equity only.

- Opening cash and bank are separate assets and also count toward equity.
- If capital-funded assets exceed `targetCapitalCoa`, save is blocked (`capitalOverAllocated`) — review asset funding tags or manual suspense.
- `GET` returns `preview`, `capitalFundedAssets`, `targetCapitalCoa`, `requiredOpeningCapital`, `capitalOverAllocated`.

`GET /api/accountant/treasury/summary` includes `balanceSheet` (`assets`, `liabilities`, `equity`, `gap`, `balanced`) when setup is complete.

`POST /api/accountant/treasury/setup` may return `remediation` when a `clean_match` double-count is auto-corrected after lock.

GET /api/accountant/treasury/reconcile-equity

Preview tenant self-service equity reconciliation (Tier B). Session auth required.

Response 200


```
{
  "balanceSheetGap": -50,
  "currentOpeningCapital": 54271,
  "projectedOpeningCapital": 54221,
  "delta": -50,
  "remediationStatus": "unresolvable",
  "canApply": true,
  "capitalOverAllocated": false,
  "explanation": "Opening capital will be adjusted…",
  "treasurySetupCompleted": true
}
```


`canApply` is false when `capitalOverAllocated` or treasury setup is incomplete.

POST /api/accountant/treasury/reconcile-equity

Apply recommended opening-capital adjustment after user confirmation. Adjusts equity offset only (not cash, bank, inventory, or invoices).

Body

```
{
  "confirm": true,
  "expectedOpeningCapital": 54271
}
```

`expectedOpeningCapital` must match current config (optimistic concurrency). Returns **409** if stale.

Response 200

```
{
  "success": true,
  "applied": true,
  "newOpeningCapital": 54221,
  "balanceSheetBalanced": true,
  "balanceSheetGap": 0
}
```

Fleet remediation (opening-capital double-count)

```
npx tsx scripts/remediate-unfunded-inventory.ts --all
npx tsx scripts/remediate-unfunded-inventory.ts --all --apply   # clean_match tenants only
npx tsx scripts/remediate-unfunded-inventory.ts --tenant-id=<uuid> --apply --force
npm run remediate:opening-capital:fleet          # assess drift (two-phase pre-filter)
OPENING_CAPITAL_AUTO_REMEDIATE=true npm run remediate:opening-capital:fleet:apply
```

CSV report under `reports/` tags each tenant: `clean_match` | `partial_match` | `unresolvable` | `already_balanced`. Only `clean_match` is auto-applied unless `--force` on a single tenant after manual review. Fleet `--apply` requires

`OPENING_CAPITAL_AUTO_REMEDIATE=true`.

Tier B (tenant UI): Cash & Bank, Chart of Accounts, and Financial Year pre-close show **Reconcile opening capital** when setup is complete and the balance sheet has a gap.

Super-admin remediation

`PATCH /api/admin/tenants/{id}/opening-capital` — body `{ "delta": -10 }` adjusts stored opening capital (equity) without moving cash/bank. Use to fix legacy tenants where supplier payments left the balance sheet out of balance.

Purchase orders and balance sheet

Orders are created **unpaid** (AP = total) even when `amountPaid` is supplied; payments are recorded afterward so AP clears when cash leaves. Paid-at-create orders therefore stay on the balance sheet: inventory increases and AP is cleared by the payment.

Settings API

Tenant configuration and catalog matching profile.

Endpoint summary

Method	Path	Description
GET	`/api/settings`	Tenant settings
PUT	`/api/settings`	Update tenant settings
GET	`/api/tenant/settings/catalog-profile`	Catalog profile JSON
PATCH	`/api/tenant/settings/catalog-profile`	Patch catalog profile

GET /api/settings

Headers

- `x-tenant-id` from session middleware **or** resolved from session cookie if header missing/invalid
- `x-ai-agent: true` triggers `ensureTenantBusinessProfile` before read

Response 200

Tenant settings object from `settingsService.getTenantSettings`, including thresholds and AI limits, for example:

Field	Type	Description
`deadStockDaysThreshold`	number	Dead stock alert days
`expiryRiskDaysThreshold`	number	Expiry warning window
`decliningSalesPercentage`	number	Alert threshold %
`expenseSpikePercentage`	number	Alert threshold %

`overduePaymentsThreshold`	number	Days
`stockAlertThresholdDefault`	number	Default low-stock threshold
`defaultLanguage`	string	Locale code
`aiAgentRules`	string	Nullable custom agent rules
`dailyAllowedAiAnalyzer`	number	Quota
`dailyAllowedWhatsapp`	number	Quota
`maxAiLoops`	number	Agent loop cap
`invoiceTitle`	string | null	Optional invoice header title override
`invoiceSubtitle`	string | null	Optional invoice subtitle override
`invoiceLogoUrl`	string | null	Tenant-scoped media URL (`/api/media/{tenantId}/...`)
`tenantName`	string	Joined from `Tenant.name` (read-only)

Errors

Status	Body
401	`{ "message": "Unauthorized" }`
404	`{ "message": "Settings not found" }`

Example

```
curl -s -b cookies.txt "https://erp.yallav.io/api/settings"
```

PUT /api/settings

Update one or more settings fields (partial update).

Request body (all optional)

Field	Type	Constraints
`deadStockDaysThreshold`	number	Int, ≥ 0
`expiryRiskDaysThreshold`	number	Int, ≥ 0
`decliningSalesPercentage`	number	0–100
`expenseSpikePercentage`	number	0–100
`overduePaymentsThreshold`	number	≥ 0
`stockAlertThresholdDefault`	number	Int, ≥ 0
`defaultLanguage`	string	2–5 chars
`aiAgentRules`	string	Max 5000, nullable
`dailyAllowedAiAnalyzer`	number	Int, ≥ 0
`dailyAllowedWhatsapp`	number	Int, ≥ 0
`maxAiLoops`	number	Int, ≥ 1
`invoiceTitle`	string | null	Max 120 chars; blank → `null` (falls back to company name on invoices)
`invoiceSubtitle`	string | null	Max 80 chars; blank → `null`
`invoiceLogoUrl`	string | null	Max 500 chars; must be `/api/media/{tenantId}/...` for current tenant

Only sent fields are updated.

Admin-only: `invoiceTitle`, `invoiceSubtitle`, and `invoiceLogoUrl` require `ADMIN` or `SUPER_ADMIN` role. Non-admins receive **403**.

Logo URL validation: Cross-tenant media paths are rejected with **400**.

Response 200

Updated settings object.

Errors

Status	Body
400	`{ "message": "Invalid settings data", "errors": { ... } }` or logo URL not tenant-scoped
403	`{ "message": "Forbidden" }` — non-admin branding or accounting update

GET /api/tenant/settings/catalog-profile

Catalog matching profile used by inventory bulk resolve (token expansions, co-occurrence, etc.).

Headers

Session or `x-tenant-id`.

Response 200

```
{
  "catalogProfileJson": {
    "version": 1,
    "tokenExpansions": {},
    "cooccurrence": {}
  }
}
```

Empty object when unset.

PATCH /api/tenant/settings/catalog-profile

Merge a partial patch into stored `catalogProfileJson`.

Request body

Valid catalog profile patch per `validateCatalogProfilePatch` (`src/services/tenant/catalogProfileTypes.ts`). Structure is tenant-specific graph data — integrators should treat as opaque JSON unless building advanced catalog tools.

Response 200

```
{
  "catalogProfileJson": { "...": "merged profile" }
}
```

Errors

Status	Body
400	`{ "message": "Invalid catalog profile patch" }`
401	Unauthorized

Related

- catalog/inventory.md (see catalog/inventory) — bulk resolve uses catalog profile
- README.md (see README) — authentication