# API Specifications & Endpoint Inventory

## Purpose and Scope

This document defines the **publicly exposed HTTP/REST interfaces** for the ERP/IMS platform described in the preceding architecture and requirements documents. It inventories each API endpoint grouped by functional domain, describes request/response semantics, and summarizes key design conventions such as resource naming, versioning, pagination, filtering, authentication and error handling. The intent is to provide developers, integrators and auditors with a clear reference for how clients (including the Angular front‑end, partner systems and third‑party services) interact with the microservices.

The scope includes:

* Internal service APIs that expose CRUD operations on core business entities (e.g., products, inventory items, orders, purchase orders, vendors, lots/serials, returns, tax rates, general ledger entries and analytics).
* External integration endpoints for payments, shipping carriers, tax calculation, 3PL, vendor EDI and future Phase 2 modules.
* Conventions for versioning, authentication, authorization, pagination, filtering, error handling and response structure.

It does **not** prescribe implementation details or message bus topics (those are covered in the event architecture document), but references relevant business logic in the acceptance criteria and data model.

## Design Principles and Standards

### RESTful Guidelines

The APIs follow REST principles and adopt the design guidelines recommended by the Azure Architecture Center. According to Microsoft’s API design best practices, a RESTful web API should be platform‑independent, loosely coupled and use standard HTTP verbs to operate on resources【991336973704714†L304-L349】. Resources are uniquely identified by URIs (for example, /orders/1), and clients interact using standard HTTP methods such as GET, POST, PUT, PATCH and DELETE【991336973704714†L347-L348】. The APIs are **stateless**, meaning that each request contains all information needed to complete the operation【991336973704714†L350-L355】.

### Resource URIs and Naming Conventions

Resources are organized around business entities. URIs use **nouns**, not verbs, and plural nouns for collections (e.g., /products, /orders)【991336973704714†L379-L427】. Nested URIs are used sparingly; relationships between entities are typically represented by including an identifier in the path (e.g., /orders/{orderId}/items) or by returning links to related resources in the response body. The guidelines caution against deeply nested URIs and recommend hypermedia links (HATEOAS) for navigating associations【991336973704714†L420-L448】.

### Versioning

The platform uses **URI versioning** to indicate breaking changes. Each major version is prefixed to the base path, such as /api/v1/products. The Azure guidance notes that a web API may need to support multiple versions to maintain backward compatibility【991336973704714†L975-L992】. Version numbers increment when structural changes break clients; minor changes that add optional fields do not require a new version. Deprecated versions remain supported for a defined period.

### Pagination, Filtering and Sorting

Collection endpoints implement **pagination** using query parameters page (starting at 1) and pageSize (default 20, maximum 100). For example, GET /api/v1/products?page=2&pageSize=50 returns the second page of products. Sorting is accomplished with a sort parameter (e.g., sort=price,-name), and filtering uses query parameters such as status=shipped or minCost=100【991336973704714†L812-L833】. Field selection can be requested via a fields parameter to return only specified fields【991336973704714†L831-L837】. The API validates requested fields to ensure proper authorization.

### Data Format

All endpoints accept and return **JSON** encoded as UTF‑8 (Content-Type: application/json; charset=utf‑8). For bulk operations or file uploads (e.g., import of product images), multipart formats may be used. Responses include Location headers when creating resources, and support ETag headers for caching where appropriate.

### Authentication and Authorization

The APIs are secured with **OAuth2 bearer tokens** issued by Microsoft Entra ID. Each request must include an Authorization: Bearer <token> header. Services verify the token and enforce **role‑based access control (RBAC)** according to the RBAC matrix. For internal service‑to‑service calls, Azure Managed Identities are used.

### Error Handling & Status Codes

Endpoints adhere to standard HTTP status semantics. For example, GET requests return **200 OK** with a representation of the resource on success, **204 No Content** when no data is returned, and **404 Not Found** when the resource does not exist【991336973704714†L539-L548】. POST requests return **201 Created** on success and include the URI of the new resource in the Location header; they may return **400 Bad Request** if the request payload is invalid【991336973704714†L549-L583】. PUT and PATCH requests are idempotent; they return **200 OK**, **201 Created** or **204 No Content** on success and **409 Conflict** when the resource state prevents the update【991336973704714†L585-L621】. Errors include a JSON body with an error code, message, correlation ID and optional details.

### Security and Compliance

All endpoints must abide by the security requirements document (authentication, encryption, auditing). Sensitive information (payment tokens, PII, hazardous‑material classifications) is never returned in plain text. Audit logs capture all API calls for compliance purposes. Rate limiting and throttling are applied per token to mitigate abuse and DoS attacks.

## Endpoint Inventory

The following tables summarise each endpoint per functional domain. Descriptions, parameters and responses are intentionally concise; developers should consult the OpenAPI/Swagger definitions for detailed schemas. Endpoints requiring elevated roles are noted in the **Auth** column.

### Product Service

| Method & Path | Description | Parameters / Body | Response (simplified) | Auth |
| --- | --- | --- | --- | --- |
| **GET** /api/v1/products | List products with pagination, sorting and filtering by category, SKU or search term. | Query: page, pageSize, sort, search, categoryId | Array of product summary objects (id, sku, name, price, status), total count. | Authenticated |
| **POST** /api/v1/products | Create a new product or kit/bundle. | JSON body with product attributes (SKU, name, description, categoryId, dimensions, hazardousClass, unitConversions, price). | **201 Created** with full product object and Location header. | ProductAdmin role |
| **GET** /api/v1/products/{productId} | Retrieve a product’s details, including kit/bundle components and default warehouse stock. | Path: productId | Full product object (attributes, pricing tiers, kit components, tags). | Authenticated |
| **PUT** /api/v1/products/{productId} | Update an existing product; supports re‑pricing and kit component changes. | JSON body with updated fields. | Updated product object or **204 No Content**. | ProductAdmin role |
| **DELETE** /api/v1/products/{productId} | Soft delete or archive a product (removes from sale but retains data). | Path: productId | **204 No Content**. | ProductAdmin role |
| **GET** /api/v1/products/{productId}/inventory | View per‑location inventory quantities and backorder status. | Path: productId; Query: locationId optional. | List of inventory items by location, including available, reserved, safety stock. | InventoryViewer role |
| **GET** /api/v1/products/search | Search products by keyword with fuzzy matching. | Query: q (string) | Array of matching products (id, name, sku, relevance score). | Authenticated |

### Inventory Service

| Method & Path | Description | Parameters / Body | Response (simplified) | Auth |
| --- | --- | --- | --- | --- |
| **GET** /api/v1/inventory/locations | List warehouses, cross‑docks and 3PL facilities. | Query: type (warehouse, crossdock, 3pl) | Array of location objects (id, name, address, type, capacity). | InventoryViewer role |
| **GET** /api/v1/inventory/items | List inventory items across locations with filters. | Query: productId, locationId, lotNumber, serialNumber, status | Array of items (id, productId, locationId, quantity, lot, serial, expiry). | InventoryViewer role |
| **POST** /api/v1/inventory/receipts | Record a receipt (ASN or manual). Creates inventory items and updates purchase order status. | JSON body: vendorId, purchaseOrderId (optional), locationId, lines [productId, quantity, lotNumber, serialNumbers, expiryDate]. | **201 Created** with receipt object and created items. | InventoryManager role |
| **POST** /api/v1/inventory/transfers | Transfer stock between locations or adjust bins. | JSON body: fromLocationId, toLocationId, items [productId, quantity, lot/serial], transferDate. | **201 Created** with transfer record. | InventoryManager role |
| **POST** /api/v1/inventory/cycle-counts | Initiate a cycle count for a location or product category. | JSON body: locationId, productCategoryId, countDate. | **201 Created** with cycle count ID. | InventoryManager role |
| **GET** /api/v1/inventory/cycle-counts/{countId} | Retrieve cycle count results and variances. | Path: countId | Cycle count object (id, status, countedItems, variances, adjustments). | InventoryManager role |

### Order Service

| Method & Path | Description | Parameters / Body | Response (simplified) | Auth |
| --- | --- | --- | --- | --- |
| **GET** /api/v1/orders | List orders with pagination, filter by status, customer, date range. | Query: page, pageSize, status, customerId, startDate, endDate | Array of order summary objects (id, orderNumber, customer, status, orderDate, total). | OrderViewer role |
| **POST** /api/v1/orders | Create an order from cart or CSR entry. Includes items, shipping, billing and payment info. | JSON body: customerId, items [productId, quantity, price], shippingMethod, shippingAddress, billingAddress, promotions. | **201 Created** with full order object and order number. | Authenticated (B2C) or CSR role |
| **GET** /api/v1/orders/{orderId} | Retrieve order details, including items, payments, shipments and notes. | Path: orderId | Full order object. | OrderViewer or customer (owner) |
| **PUT** /api/v1/orders/{orderId} | Update order status or modify items prior to fulfillment; CSR can override (with proper role). | JSON body: status changes (e.g., confirmed, backordered), updates to shipping/billing, replacement items. | Updated order object. | CSR role |
| **POST** /api/v1/orders/{orderId}/payments | Capture a payment or request authorization via Payment Service. | JSON body: paymentMethodId, amount, currency, token. | Payment transaction object (id, status, authorizationCode). | OrderManager role |
| **GET** /api/v1/orders/{orderId}/shipments | Retrieve shipments associated with an order. | Path: orderId | Array of shipment objects (trackingNumber, carrier, status, labelUrl). | OrderViewer role |
| **POST** /api/v1/orders/{orderId}/shipments | Create a shipment; triggers rate shopping and label purchase via Shipping Service. | JSON body: items to ship, carrierId, serviceType, origin, destination. | **201 Created** with shipment object. | OrderManager role |
| **POST** /api/v1/orders/{orderId}/cancel | Cancel an order prior to shipment; handles refunds via Payment Service. | Path: orderId; Body: reason code. | Order status updated to cancelled and refund transaction details. | CSR or customer (if not shipped) |
| **POST** /api/v1/orders/{orderId}/returns | Initiate a return/RMA for an order. | Body: items [orderLineId, quantity], reasonCode, returnMethod. | **201 Created** with RMA record. | CSR or customer (returns portal) |

### Purchasing and Vendor Service

| Method & Path | Description | Parameters / Body | Response (simplified) | Auth |
| --- | --- | --- | --- | --- |
| **GET** /api/v1/purchase-orders | List purchase orders by status, vendor, date. | Query: page, pageSize, status, vendorId, startDate, endDate | Array of purchase orders (id, number, vendor, status, total). | PurchasingViewer role |
| **POST** /api/v1/purchase-orders | Create a purchase order with one or more line items. | JSON body: vendorId, shipToLocationId, lines [productId, quantity, unitCost, desiredDeliveryDate]. | **201 Created** with purchase order object. | PurchasingManager role |
| **GET** /api/v1/purchase-orders/{poId} | Retrieve purchase order details; includes approval history, lines and receipts. | Path: poId | Purchase order object. | PurchasingViewer role |
| **PUT** /api/v1/purchase-orders/{poId} | Update purchase order details prior to approval or dispatch. | Body: modifications to lines, terms or delivery dates. | Updated purchase order object. | PurchasingManager role |
| **POST** /api/v1/purchase-orders/{poId}/approve | Approve or reject a purchase order; triggers vendor notification. | Body: approved (boolean), optional comment. | Updated status with approval metadata (approverId, timestamp). | PurchasingApprover role |
| **POST** /api/v1/purchase-orders/{poId}/receive | Record partial or complete receipt of a PO; updates inventory. | Body: receipt lines [lineId, quantityReceived, lotNumber, serialNumbers, expiryDate]. | Receipt record and updated PO status. | InventoryManager role |
| **GET** /api/v1/vendors | List vendors with filters and sorting. | Query: page, pageSize, search, rating | Array of vendor objects (id, name, rating, leadTime, minOrderQty). | PurchasingViewer role |
| **POST** /api/v1/vendors | Create new vendor; includes contact information and vendor evaluation fields. | JSON body: name, address, contactEmail, phone, leadTimeDays, minimumOrderQuantity, rating. | **201 Created** with vendor object. | PurchasingManager role |
| **GET** /api/v1/vendors/{vendorId} | Retrieve vendor details, including scorecards. | Path: vendorId | Vendor object with performance metrics (on‑time delivery %, quality %, lastPODate). | PurchasingViewer role |
| **PUT** /api/v1/vendors/{vendorId} | Update vendor information or scorecard attributes. | Body: updated vendor fields. | Updated vendor object. | PurchasingManager role |

### Lot & Serial Tracking Service

| Method & Path | Description | Parameters / Body | Response (simplified) | Auth |
| --- | --- | --- | --- | --- |
| **GET** /api/v1/lots | List lot numbers and expiration dates for products. | Query: productId, locationId, status | Array of lot objects (lotNumber, productId, quantity, manufactureDate, expiryDate, status). | InventoryViewer role |
| **GET** /api/v1/lots/{lotNumber} | Retrieve details of a specific lot, including linked inventory items and orders. | Path: lotNumber | Lot object with history of receipts, transfers and consumption. | InventoryViewer role |
| **GET** /api/v1/serials | List serialized items for high‑value tools. | Query: productId, locationId, status | Array of serial objects (serialNumber, productId, status, warrantyExpiry). | InventoryViewer role |
| **GET** /api/v1/serials/{serialNumber} | Get details for a serialized item and its service history. | Path: serialNumber | Serial object with purchase date, repairs and warranty information. | InventoryViewer role |

### Returns (RMA) Service

| Method & Path | Description | Parameters / Body | Response (simplified) | Auth |
| --- | --- | --- | --- | --- |
| **GET** /api/v1/returns | List return merchandise authorizations (RMAs) with status filters. | Query: page, pageSize, status, customerId, startDate, endDate | Array of returns (id, rmaNumber, orderId, status, returnDate). | CSR or ReturnsViewer |
| **POST** /api/v1/returns | Create an RMA for one or more order lines. | JSON body: orderId, lines [orderLineId, quantity], reasonCode, returnMethod (ship, drop‑off). | **201 Created** with RMA object (id, rmaNumber, status). | CSR or customer (returns portal) |
| **GET** /api/v1/returns/{rmaId} | Retrieve RMA details, including items, status, inspection results and disposition. | Path: rmaId | RMA object. | CSR or ReturnsViewer |
| **PUT** /api/v1/returns/{rmaId}/disposition | Update disposition of returned items (restock, refurbish, scrap). | Body: lines [rmaLineId, dispositionCode], notes. | Updated RMA status. | ReturnsManager role |
| **POST** /api/v1/returns/{rmaId}/refund | Process refund or store credit for returned items via Payment Service. | Body: refund amount, method (original, store credit). | Refund transaction object. | ReturnsManager role |

### Shipping & Rate Shopping Service

| Method & Path | Description | Parameters / Body | Response (simplified) | Auth |
| --- | --- | --- | --- | --- |
| **GET** /api/v1/shipping/carriers | List supported carriers and default services. | — | Array of carriers (id, name, supportedServiceTypes). | ShippingViewer role |
| **GET** /api/v1/shipping/rates | Compare shipping rates for a parcel across carriers and service levels. | Query: originZip, destinationZip, weight, dimensions, hazardous (boolean), deliveryDate | Array of rate options (carrier, serviceType, cost, estimatedDeliveryDate). | ShippingViewer role |
| **POST** /api/v1/shipping/labels | Purchase shipping labels; automatically stores label and tracking number. | JSON body: orderId or manual shipment, carrierId, serviceType, packages [weight, dimensions, contents]. | **201 Created** with shipment object (labelUrl, trackingNumber). | ShippingManager role |
| **GET** /api/v1/shipping/track/{trackingNumber} | Track a shipment’s current status. | Path: trackingNumber | Tracking status (status, location, timestamp). | ShippingViewer role |

### Tax & Accounting Service

| Method & Path | Description | Parameters / Body | Response (simplified) | Auth |
| --- | --- | --- | --- | --- |
| **GET** /api/v1/taxes/rates | Retrieve tax rates for a region (state, county, city). | Query: state, county, city, postalCode, effectiveDate | Tax rate object (jurisdiction, rate, effectiveDate). | AccountingViewer |
| **POST** /api/v1/taxes/calculate | Calculate tax for an order or invoice. | JSON body: destination (state, county), lines [productId, quantity, price], customerTaxExemptionId | Tax calculation result (totalTax, breakdown by jurisdiction). | OrderManager role |
| **GET** /api/v1/gl-entries | List general ledger entries by date range, account or transaction reference. | Query: page, pageSize, startDate, endDate, accountId | Array of GL entries (id, accountId, description, debit, credit, date). | Accountant role |
| **POST** /api/v1/gl-entries | Create a journal entry (debit/credit) manually or via integration. | JSON body: entries [accountId, debit, credit, memo], reference (PO, order, invoice). | **201 Created** with entry object. | Accountant role |
| **GET** /api/v1/accounts | Retrieve chart of accounts with hierarchy. | — | Array of account objects (id, code, name, type, parentId). | Accountant role |

### Analytics & Reporting Service

| Method & Path | Description | Parameters / Body | Response (simplified) | Auth |
| --- | --- | --- | --- | --- |
| **GET** /api/v1/analytics/dashboard | Returns KPIs and summary metrics for procurement, inventory accuracy, order fulfilment, vendor performance and sales. | Query: period (e.g., month, quarter), metric (optional) | JSON object with key metrics (inventoryAccuracy, fillRate, onTimeDelivery %, grossMargin, taxLiability). | ExecutiveViewer or AnalyticsViewer role |
| **GET** /api/v1/analytics/reports/{reportId} | Download prebuilt reports (PDF/CSV) or run ad‑hoc queries. | Path: reportId; Query: format (pdf, csv) | Binary or base64 encoded report. | Authorized roles according to report classification |

### External Integration Endpoints

The platform integrates with external services using specific endpoints or connectors. These endpoints are either proxies exposing internal services or client APIs to third parties. For each integration, ensure that sensitive credentials are stored in secure secrets and that API calls are encrypted.

#### Payment Service Integration

* **POST** /api/v1/payments – Process a payment or refund through the configured payment gateway. Body includes orderId, amount, currency, paymentToken or reference; returns transaction status and authorization details. Handles 3DSecure flows as needed. Authentication: PaymentProcessor role.
* **GET** /api/v1/payments/{paymentId} – Retrieve payment details (status, captured amount, failure reason). Used to display to customer or reconcile accounts.

#### Shipping Carrier APIs

* Internal shipping endpoints call third‑party carrier APIs (UPS, FedEx, USPS). For each carrier, a dedicated integration microservice encapsulates the carrier’s REST/SOAP interface. These services expose internal endpoints (e.g., /integrations/shipping/fedex/rates, /integrations/shipping/fedex/labels) that are not public but documented for devops.

#### Tax Calculation Service

* **POST** /api/v1/integrations/tax/calculate – Proxy to the external tax service to obtain accurate tax rates and return JSON with county‑level breakdown. The body includes origin and destination addresses, product tax codes and exemptions.

#### 3PL & EDI Integration (Phase 2)

* **POST** /api/v1/integrations/3pl/orders – Send shipment orders to third‑party logistics providers. Body contains orderId, items, shipFrom/shipTo addresses, service level and pickup date. Response includes 3PL reference number.
* **GET** /api/v1/integrations/3pl/shipments/{reference} – Retrieve status and tracking information from the 3PL.
* **POST** /api/v1/integrations/edi/{vendorId} – Send or receive EDI messages (e.g., purchase orders, ASNs, invoices) with large vendors. Supports X12 transaction sets 850, 856 and 810. Body contains EDI envelope. Response includes acknowledgment codes.

### Phase 2 Endpoints (Future Enhancements)

Phase 2 functions will extend the API once features such as demand forecasting, promotion engine, contract pricing, light manufacturing, customer service console and new analytics dashboards are implemented. The following outlines prospective endpoints to be designed in the second phase:

* **Forecasting Service** – GET /api/v2/forecast/demand returns statistical demand forecasts by SKU and location; query accepts historical horizon, algorithm, seasonality indicators. POST /api/v2/forecast/run triggers a forecast generation job.
* **Promotion & Pricing Service** – POST /api/v2/promotions to create promotional campaigns; GET /api/v2/promotions/{promotionId} to retrieve details; POST /api/v2/pricing/contracts to set contract pricing for B2B customers.
* **Kitting / Light Manufacturing** – POST /api/v2/assembly-orders to create assembly or kitting orders; GET /api/v2/assembly-orders/{id} to monitor status; integrates with bill‑of‑materials.
* **Customer Service Console** – GET /api/v2/customer-service/cases to list support cases; POST /api/v2/customer-service/cases to open a new case; PUT /api/v2/customer-service/cases/{id} to update case status, notes or SLA timers.
* **EDI & 3PL enhancements** – Additional transaction sets (e.g., EDI 997 acknowledgments) and 3PL event webhooks for pickup, out‑for‑delivery and delivery confirmations.

## Conclusion

The ERP/IMS API surfaces a rich set of endpoints to support product master management, multi‑warehouse inventory, omnichannel order processing, procurement, vendor management, traceability, shipping, returns, tax calculation, accounting and analytics. Adhering to the RESTful conventions described by Microsoft’s API design guidelines—such as using resource‑based URIs, HTTP methods, versioning schemes and standardized status codes—makes the API intuitive and maintainable【991336973704714†L304-L349】【991336973704714†L539-L583】. Pagination, filtering and field selection parameters provide flexibility while protecting performance【991336973704714†L812-L833】. As the platform evolves, the versioning strategy and Phase 2 endpoints ensure backward compatibility and room for new capabilities. All endpoints must follow the security requirements, employ least‑privilege access and produce comprehensive audit logs to maintain compliance and traceability.