**P4 Warehouse REST API – knowledge document**

**Introduction**

The **P4 Warehouse API** is a REST‑style interface used by P4 Warehouse Cloud WMS/ERP software to integrate warehouses, customers, vendors, invoices and operational transactions. Requests are HTTPS calls to https://api.p4warehouse.com/<endpoint>. All calls are authenticated via an API key supplied in the ApiKey header. Responses are JSON documents representing business objects such as clients, customers, products, pick tickets and purchase orders.

This document is an overview of the API structure, not a replacement for the online Swagger documentation, and it summarises how the major endpoints are organised and what data they return. It is aimed at developers and integrators looking to consume the P4 Warehouse API using C#, but the concepts apply to any language.

**Note:** The examples shown here come from the publicly accessible Swagger endpoints. They may not reflect your exact data. For up‑to‑date details always refer to the Swagger definition.

**Authentication**

Every request must include an API key in the request header. In the Swagger UI the key is entered in the **Authorize** dialog. Once authorised, requests can be executed directly from the UI. Without a valid key the server returns 401 Unauthorized.

Example header (the actual key is not shown for security reasons):

vbnet

CopyEdit

ApiKey: <your‑api‑key>

**Major endpoint categories**

**Clients**

* **GET /clients** – returns a list of client companies configured in the system. Each client object includes:
  + id (GUID)
  + name (string) – the customer’s name
  + description (nullable string)
  + ssccCompanyId (string) – SSCC/GS1 company identifier

In the test data the call returned a single client record with name "casMed" and ssccCompanyId "0753529"A screenshot of a computer

AI-generated content may be incorrect.api.p4warehouse.com. Use the client ID on other endpoints (e.g., products, vendors or warehouses) to scope the returned data.

**Client invoices**

* **GET /client-invoices** – lists invoices issued to a client. Query parameters include page and pageSize, and the Accept header should be set to application/json. The response is an array of invoice objects. Although the example call returned an empty arrayA screenshot of a computer

  AI-generated content may be incorrect.api.p4warehouse.com, the **example value** in Swagger shows that an invoice contains fields such as:
  + id (GUID)
  + invoiceNumber (string)
  + clientId (GUID)
  + startPeriod, endPeriod, postingDate (dates)
  + subTotal, total (decimal)
  + dateCreated (date)

Invoice lines are not returned here; they are available on other endpoints.

**Customers**

* **GET /customers** – returns a list of customer records. Customers are the companies to whom goods are shipped. Each customer object includes:
  + id (GUID)
  + customerCode (customer identifier)
  + companyName (string) – legal company name
  + description (nullable string)
  + phone, email and contactPerson
  + taxId (string)
  + client (object) – the owning client with fields id, name, description, ssccCompanyId

Example data returned from the API shows customers like **HICSA PANAMA SA** (customerCode "HICSA", phone "234‑5656", email "info@hicsapanama.com"), and another customer **Barrdega** (customerCode "C‑0001", phone "20960996" etc.)A screenshot of a computer

AI-generated content may be incorrect.api.p4warehouse.com.

* + **GET /customers/{id}** – returns a single customer by its GUID. The response includes the same fields as the list endpoint along with the nested client object. When called with id 1f900e20‑73a2‑47a8‑9661‑48f1e311ed80 the server returned the customer **HICSA PANAMA SA** with customerCode "HICSA", companyName "HICSA PANAMA SA", phone "234‑5656", email "info@hicsapanama.com", contactPerson "Principal" and client name "CasaMed"A screenshot of a computer program

    AI-generated content may be incorrect.api.p4warehouse.com. Use this endpoint when you need the full details of a specific customer.

**Vendors**

* **GET /vendors** – lists vendors (suppliers). Optional query parameters are code (vendor code) and clientId. A vendor record contains:
  + id (GUID)
  + vendorCode (string) – internal code used by the client
  + companyName (string)
  + description (nullable string)
  + phone, email, contactPerson
  + client – details of the owning client (id, name, description, ssccCompanyId)

In the example response the API returned vendor records such as **NEW SUPPLIER SA** with vendorCode "SUP001", companyName "NEW SUPPLIER SA", email "supp@gmail.com" and phone "23505565"A screenshot of a computer

AI-generated content may be incorrect.api.p4warehouse.com. An example schema also shows that optional fields like contactPerson or description may be nullA screenshot of a computer

AI-generated content may be incorrect.api.p4warehouse.com.

* + **GET /vendors/{id}** – retrieves a single vendor by its GUID. The returned object has the same structure as the list endpoint. In a test call with id 002f250c‑78a8‑437e‑9fae‑59bfc1220187 the API returned a vendor with code SUP001, companyName "NEW SUPPLIER SA", email "supp@gmail.com", phone "23555655" and nested client details (for example, client name "CasMed" and ssccCompanyId "0753329")A screenshot of a computer

    AI-generated content may be incorrect.api.p4warehouse.com. This endpoint is useful for retrieving up‑to‑date vendor information when you already know the vendor’s id.

**Warehouses**

* **GET /warehouses** – returns a list of warehouses for the authenticated client. Optional query parameter warehouseCode filters by the warehouse code. Each warehouse has:
  + id (GUID)
  + warehouseCode (string) – code such as 101, 102, etc.

The call returned a warehouse with id "5eb36ee0-93ae-4e08-bbd9-bad8303e48b7" and warehouseCode "101"A screenshot of a computer

AI-generated content may be incorrect.api.p4warehouse.com. The example schema in Swagger shows the same structure with placeholder valuesA screenshot of a computer

AI-generated content may be incorrect.api.p4warehouse.com.

**Products**

* **GET /products** – returns product definitions (SKUs). Products belong to a client and are identified by sku and id. Important fields:
  + id (GUID)
  + sku (string) – the stock keeping unit or item number
  + description (string)
  + upc (nullable string) – global product code, optional
  + barcodeType (enum) – e.g., Code128
  + barcodeValue (string) – the barcode printed on the product
  + referenceNumber (string) – may map to an internal or supplier part number
  + client – nested object with id, name, description, ssccCompanyId

Example data retrieved via the API showed several products:

* + A thermal paper roll (sku "905-0027775-00"), description “Thermal paper 4d R12…”, barcodeType "Code128", and barcodeValue equal to the SKUA screenshot of a computer

    AI-generated content may be incorrect.api.p4warehouse.com.
  + A product sku "7950206" with description “BSS BAG NGP 500ML – LAM” and no upcA screenshot of a computer

    AI-generated content may be incorrect.api.p4warehouse.com.

The schema also includes an example with blank strings for each fieldA screenshot of a computer

AI-generated content may be incorrect.api.p4warehouse.com.

* + **GET /products/{id}** – intended to return a single product by id. During testing this endpoint consistently returned 406 Not Acceptable when using the default text/plain media type. Changing the Accept header to application/json still resulted in validation errors or “not found” responses for some of the test IDsA screenshot of a computer

    AI-generated content may be incorrect.api.p4warehouse.com. Some products in the sample data have IDs that do not conform to a proper GUID format, which prevents them from being requested individually. For now, retrieving product details is more reliable via the list endpoint (/products) and filtering the results in code.

**Inventory**

* **GET /inventory** – returns current inventory quantities. Query parameters are warehouseId and clientId. Without filtering the response may be empty if there is no inventory set upA screenshot of a computer

  AI-generated content may be incorrect.api.p4warehouse.com. Inventory objects include:
  + id (GUID)
  + product – product object (as above)
  + warehouse – warehouse object
  + quantityOnHand and quantityAvailable (integers or decimals)

Inventory may also include lot numbers, serial numbers and expiration dates depending on configuration.

**Adjustments**

* **POST /adjustments** – executes an inventory adjustment. The request body must conform to the AdjustmentQuery schema (warehouse ID, product, quantity, reason, etc.). The Swagger documentation describes the AdjustmentQuery fields and indicates that the endpoint returns a list of Adjustment objects upon success[api.p4warehouse.com](https://api.p4warehouse.com/swagger/v1/swagger.json#:~:text=,).

**Pick tickets**

* **GET /pick-tickets** – retrieves open pick tickets. The response is a list of documents used to fulfil orders. A pick ticket includes numerous fields:
  + id (GUID)
  + pickTicketNumber (string) – unique ticket number
  + referenceNumber (string) – reference for integration
  + poNumber (string) – purchase order number
  + comments (string)
  + pickingInstructions, packagingInstructions, shippingInstructions (strings) – instructions to the warehouse
  + requiredDate, cancelDate (dates)
  + freightType, routeNumber, freightTerms, carrier, shippingService, shipCode (strings)
  + warehouse – nested warehouse object with id and warehouseCode
  + customer – nested customer object with id, customerCode, companyName and contact details

The example pick ticket in Swagger includes these fields and shows that cross‑docking and shipping label images can be part of the recordA screenshot of a computer program

AI-generated content may be incorrect.api.p4warehouse.com.

* + **GET /pick-tickets/{id}** – returns a single pick ticket by its GUID. The returned object contains the same fields described above plus arrays for totes and lines. In a test call with id 69fad488‑970d‑4257‑b9b6‑af4d682bfc93 the API returned a pick ticket with pickTicketNumber "3", referenceNumber and poNumber as empty strings, warehouse { id "5eb36ee0‑39ae‑4e08‑bbd9‑bad8303e4b87", warehouseCode "101" } and customer details for **HICSA PANAMA SA** (customer id 1f900e20‑73a2‑47a8‑9661‑48f1e311ed80, companyName "HICSA PANAMA SA", phone "234‑5656", email "info@hicsapanama.com", contact person "Principal")A screenshot of a computer program

    AI-generated content may be incorrect.api.p4warehouse.com. The arrays totes and lines were empty in this particular record. Use this endpoint when you need to drill into a specific pick ticket.

**Production orders**

* **GET /production-orders** – returns production orders for manufacturing or assembly processes. The response body may be empty if none existA screenshot of a computer

  AI-generated content may be incorrect.api.p4warehouse.com. According to the example, a production order has:
  + id (GUID)
  + productionOrderNumber (string)
  + referenceNumber (string)
  + comments (string)
  + warehouse – id and warehouse code

Posting or updating production orders is performed via the corresponding POST/PUT endpoints.

**Purchase orders**

* **GET /purchase-orders** – lists inbound purchase orders. The response can be filtered and is empty if no orders are presentA screenshot of a computer

  AI-generated content may be incorrect.api.p4warehouse.com. The **example value** in Swagger shows the typical fields:
  + id (GUID)
  + purchaseOrderNumber (string)
  + referenceNumber, reference1, reference2, reference3 (strings) – various reference numbers from upstream systems
  + appointmentNumber and appointmentDate (for scheduling deliveries)
  + requiredDate, closeDate (dates)
  + comments (string)
  + vendor – nested vendor object (see Vendors section) with contact details and client informationA screenshot of a computer

    AI-generated content may be incorrect.api.p4warehouse.com

Purchase orders also have lines (items) and statuses that are handled on separate endpoints (POST/PUT/DELETE).

**Customer returns**

* **GET /customer-returns** – retrieves customer return documents. Each return includes a reference number, return date, status, reason codes and the associated customer. Additional endpoints allow posting new returns or updating existing ones. (Detailed schema is available in Swagger but was not executed during this exploration.)

**Error handling**

Standard HTTP status codes are used:

* **200 OK** – request succeeded. Data returned in response body.
* **400 Bad Request** – invalid input (e.g., missing required fields, invalid UUID format). The response may include details of validation errors.
* **401 Unauthorized** – missing or incorrect API key.
* **404 Not Found** – record with the specified ID was not found.
* **406 Not Acceptable** – Accept header requested a format that the server cannot provide (for example text/plain on an endpoint that only supports application/json).
* **500 Internal Server Error** – unexpected server error; contact support.

**Usage recommendations**

* **Always specify the correct Accept header**. For JSON responses use application/json. The Swagger UI defaults to text/plain; adjust it to avoid 406 responses.
* **Filter results when possible**. Some endpoints support pagination (page/pageSize) or filtering by codes or IDs. Passing filters reduces payload size and improves performance.
* **Observe required fields** in POST/PUT requests. The Swagger schema definitions describe which properties are required for creation.
* **Use GUIDs consistently**. Many endpoints require the id of clients, customers, warehouses, products, etc. These IDs are GUIDs returned by the corresponding GET endpoints.
* **Handle null values gracefully**. Optional fields such as description, contactPerson, upc or barcodeValue may be null. Your integration should not assume values exist.

**Conclusion**

The P4 Warehouse API is a broad set of REST endpoints that model key business entities for warehouse management—clients, customers, vendors, warehouses, products, inventory, pick tickets, production orders, and purchase orders. Each entity is represented as a JSON object with clearly defined properties and relationships to other entities.

From a C# integration perspective, you can build strong‑typed models corresponding to the schemas above. Use HttpClient to send requests with the ApiKey header and parse responses into your models. Always validate inputs and handle error codes.

For additional operations like creating adjustments, posting new purchase orders or handling returns, consult the complete Swagger documentation and follow the same patterns described here. This document should provide a solid foundation for understanding the structure and data returned by the P4 Warehouse API.

**Creating, updating and deleting records**

While many endpoints are read‑only, the API also exposes POST, PUT and DELETE verbs for creating, updating and removing data. These write operations require valid request bodies and often return only the id of the affected record. Below are the most commonly used write endpoints and their expected payloads.

**Vendors – POST/PUT**

Vendors can be created and updated via the /vendors endpoint:

* **POST /vendors** – creates a new vendor. The request body is a JSON document with the vendor details. Required fields are companyName (string), email (string), phone (string), contactPerson (string), description (string) and vendorCode (string). The optional clientId field must be a valid client GUID; if omitted the vendor is created without a client association. A successful call returns HTTP 200 and a response body containing the GUID of the new vendor, for example { "id": "1668af71‑c25a‑4450‑a5e6‑29645d5b61ad" }A screenshot of a computer

  AI-generated content may be incorrect.api.p4warehouse.com. If clientId is supplied but not a valid GUID, the server returns 400 Bad Request with a validation error.
* **PUT /vendors** – updates an existing vendor. The request body must include the vendor’s id along with any fields to change. Fields not present are left unchanged. In testing, updating a vendor’s name, phone and description returned HTTP 200 and the same vendor id in the response bodyA screenshot of a computer

  AI-generated content may be incorrect.api.p4warehouse.com.

There is no DELETE method for vendors; records are normally deactivated rather than removed.

**Vendor creation JSON template**

json

CopyEdit

{

"companyName": "string",

"email": "string",

"phone": "string",

"contactPerson": "string",

"description": "string",

"clientId": "uuid", // optional: omit or supply a valid client id

"vendorCode": "string"

}

**Vendor update JSON template**

json

CopyEdit

{

"id": "uuid",

"companyName": "string",

"email": "string",

"phone": "string",

"contactPerson": "string",

"description": "string"

}

**Purchase orders – POST/PUT/DELETE**

Purchase orders represent inbound shipments from vendors. They can be created, updated or deleted using the following endpoints:

* **POST /purchase-orders** – creates a new purchase order. The request body accepts a rich object with fields such as:
  + comments (string) – free‑form comments.
  + referenceNumber, reference1, reference2, reference3 (strings) – external reference numbers.
  + appointmentNumber (string) and appointmentDate (date) – scheduling information.
  + requiredDate and closeDate (dates).
  + vendorId (GUID) – supplier for the order.
  + warehouseId (GUID) – warehouse where goods will be received.
  + purchaseOrderNumber (string) – your internal PO number.
  + vendor (object) – optional nested vendor information. When a vendorId is supplied the nested vendor data is ignored; when no vendorId is provided the nested object must contain full vendor details (companyName, email, phone, contactPerson, description and a client object).

The example in Swagger shows a complete payload with all fields populatedA screenshot of a computer

AI-generated content may be incorrect.api.p4warehouse.com. On success the server returns HTTP 200 with the new purchase order’s id.

* **PUT /purchase-orders** – updates an existing purchase order. The request body mirrors the POST payload but must include the id of the order. Only the supplied fields are updated; unspecified properties remain unchanged.
* **DELETE /purchase-orders** – deletes a purchase order by its code and client. Two query parameters are required: code (the purchase order number) and clientId (GUID of the owning client). If a matching order exists it is removed and the server returns HTTP 204 No Content. If the order is not found, the response is 404 Not Found.
* **DELETE /purchase-orders/{id}** – deletes a purchase order using its GUID. On success the server returns HTTP 204 No Content. Attempting to delete a non‑existent record returns 404 Not Found.

**Purchase order creation JSON template**

json

CopyEdit

{

"purchaseOrderNumber": "string", // your internal PO number

"vendorId": "uuid", // supplier; supply either vendorId or the nested vendor object

"warehouseId": "uuid", // warehouse receiving the order

"referenceNumber": "string",

"reference1": "string",

"reference2": "string",

"reference3": "string",

"appointmentNumber": "string",

"appointmentDate": "2025-07-24T22:00:00Z",

"requiredDate": "2025-07-24T22:00:00Z",

"closeDate": "2025-07-25T22:00:00Z",

"comments": "string",

"vendor": {

"companyName": "string",

"email": "string",

"phone": "string",

"contactPerson": "string",

"description": "string",

"id": "uuid",

"client": {

"id": "uuid",

"name": "string",

"description": "string",

"ssccCompanyId": "string"

}

}

}

**Purchase order update JSON template**

json

CopyEdit

{

"id": "uuid",

"purchaseOrderNumber": "string",

"vendorId": "uuid",

"warehouseId": "uuid",

"referenceNumber": "string",

"reference1": "string",

"reference2": "string",

"reference3": "string",

"appointmentNumber": "string",

"appointmentDate": "2025-07-24T22:00:00Z",

"requiredDate": "2025-07-24T22:00:00Z",

"closeDate": "2025-07-25T22:00:00Z",

"comments": "string"

}

**Batch upload JSON template**

json

CopyEdit

[

{

"purchaseOrderNumber": "string",

"vendorId": "uuid",

"warehouseId": "uuid",

"referenceNumber": "string",

"reference1": "string",

"reference2": "string",

"reference3": "string",

"appointmentNumber": "string",

"appointmentDate": "2025-07-24T22:00:00Z",

"requiredDate": "2025-07-24T22:00:00Z",

"closeDate": "2025-07-25T22:00:00Z",

"comments": "string"

}

]

The API also provides **POST /purchase-orders/upload** to upload a batch of pending purchase orders. This endpoint accepts an array of purchase order objects and processes them in bulk.

**Pick tickets and production orders**

Pick tickets (outbound shipments) and production orders can also be posted or updated via their respective endpoints:

* **POST /pick-tickets/upload** – uploads one or more new pick tickets into the system. The body is an array of pick ticket objects with the same structure as the GET responses (ticket number, references, warehouse, customer, lines and instructions). Successful calls return HTTP 200 and an array of created ticket IDs.

**Pick ticket upload JSON template**

json

CopyEdit

[

{

"pickTicketNumber": "string",

"referenceNumber": "string",

"poNumber": "string",

"comments": "string",

"pickingInstructions": "string",

"packagingInstructions": "string",

"shippingInstructions": "string",

"requiredDate": "2025-07-25T00:00:00Z",

"cancelDate": "2025-07-30T00:00:00Z",

"freightType": "string",

"routeNumber": "string",

"freightTerms": "string",

"carrier": "string",

"shippingService": "string",

"shipCode": "string",

"warehouseId": "uuid",

"customerId": "uuid",

"lines": [

{

"productId": "uuid",

"quantity": 10,

"uom": "string",

"description": "string"

}

]

}

]

* **PUT /pick-tickets** – updates an existing pick ticket. The request body must include id plus any fields to modify (picking instructions, ship code, etc.).

**Pick ticket update JSON template**

json

CopyEdit

{

"id": "uuid",

"comments": "string",

"pickingInstructions": "string",

"packagingInstructions": "string",

"shippingInstructions": "string",

"requiredDate": "2025-07-25T00:00:00Z",

"cancelDate": "2025-07-30T00:00:00Z",

"freightType": "string",

"routeNumber": "string",

"freightTerms": "string",

"carrier": "string",

"shippingService": "string",

"shipCode": "string"

}

* **POST /production-orders** – creates a production order. The body contains productionOrderNumber, referenceNumber, comments and the target warehouseIdA screenshot of a computer

  AI-generated content may be incorrect.api.p4warehouse.com. The server responds with the newly created order’s id.

**Production order creation JSON template**

json

CopyEdit

{

"productionOrderNumber": "string",

"referenceNumber": "string",

"comments": "string",

"warehouseId": "uuid"

}

* **PUT /production-orders** – updates an existing production order. The payload mirrors the POST body and must include the id field.

**Production order update JSON template**

json

CopyEdit

{

"id": "uuid",

"productionOrderNumber": "string",

"referenceNumber": "string",

"comments": "string",

"warehouseId": "uuid"

}

**Adjustments and returns**

The API includes endpoints for inventory adjustments and customer returns:

* **POST /adjustments** – creates an inventory adjustment. The request body is an AdjustmentQuery object containing the warehouse ID, product ID, quantity, reason and optional notes. The response is a list of adjustments created[api.p4warehouse.com](https://api.p4warehouse.com/swagger/v1/swagger.json#:~:text=,).

**Adjustment JSON template**

json

CopyEdit

{

"warehouseId": "uuid",

"productId": "uuid",

"quantity": 10,

"reason": "string",

"notes": "string"

}

* **POST /adjustments/upload** – uploads a set of adjustments in bulk. The body is an array of AdjustmentQuery objects.
* **POST /customer-returns** – records a new customer return. The body contains fields such as referenceNumber, returnDate, customerId, reasonCode and line details. The API responds with the created return’s ID. Updates can be performed via **PUT /customer-returns**, and returns can be deleted using **DELETE /customer-returns/{id}**.

**Customer return JSON template**

json

CopyEdit

{

"referenceNumber": "string",

"returnDate": "2025-07-24T00:00:00Z",

"customerId": "uuid",

"reasonCode": "string",

"lines": [

{

"productId": "uuid",

"quantity": 5,

"uom": "string",

"condition": "string"

}

]

}

These write operations enable a complete integration loop: new master data (vendors, customers, products) can be created or updated, operational documents like purchase orders and pick tickets can be posted or modified, and exceptions such as adjustments and returns can be recorded. Always refer to the Swagger schemas for required fields and test in a non‑production environment before automating writes.