

Local-Store Inventory API

Upload a catalog's product inventory and serve priced shopping lookups.

Audience: Catalog integration developers **Base URL:** <https://reader.digitalquire.com> **Auth:** Per-catalog bearer token

Overview

The **local-store inventory** model lets a catalog owner upload a self-contained product inventory as JSON. Each item carries its full product data (name, price, image, etc.), so a shopping lookup is a direct read — no live scrape, no third-party fetch. Two endpoints make up the integration: one to **upload** the inventory, one the reader uses to **look up** a product by SKU.

- PUT** `/api/inventory/<pubID>` — upload (full-replace) the catalog's inventory. Requires a bearer token.
- GET** `/api/price/<pubID>?sku=` — the reader's product lookup. Returns a normalized product.

Inventory is stored as one JSON object per publication in object storage at `commerce/inventory/{clientID}/{pubID}.json` and cached in-process. An upload fully replaces the file and refreshes the cache immediately.

Data Model

Inventory file

The upload body is a strict-JSON object. **Comments are not allowed** — a JSON parser must accept it verbatim.

```
// strict JSON — the // annotations below are for this doc only
{
  "schemaVersion": 1,           // format version
  "currency": "USD",            // file-level default for items that omit it
  "updatedAt": "2026-10-26T00:00:00Z", // optional; stamped server-side if absent
  "items": [ /* one or more InventoryItem, see below */ ]
}
```

Field	Type	Required	Notes
schemaVersion	number	optional	Format version. Currently 1 .
currency	string	optional	File-level default currency applied to items that omit currency .
updatedAt	string	optional	ISO 8601. Server stamps one if absent; echoed back in the response.
items	array	required	Non-empty array of InventoryItem. An empty array is rejected (400).

InventoryItem

A superset of the lookup response. Only `sku` and `name` are required; unknown fields (e.g. `order_num`) are stored verbatim and ignored by the lookup.

Field	Type	Required	Notes
<code>sku</code>	string	required	The lookup key. Must be non-empty.
<code>name</code>	string	required	Product name.
<code>price</code>	number string	optional	A number, or a currency string like <code>"\$34.99"</code> . A finite value > 0 yields a priced line; otherwise the reader shows a request-for-quote.
<code>currency</code>	string	optional	Overrides the file default. Cascade: item $\rightarrow$ file $\rightarrow$ USD.
<code>unitOfMeasure</code>	string	optional	e.g. <code>each</code> , <code>case</code> .
<code>image</code>	string	optional	Absolute image URL.
<code>description</code>	string	optional	Short product description.
<code>available</code>	string	optional	e.g. <code>In Stock</code> .
<code>url</code>	string	optional	Product page URL.

Normalized product (lookup response)

Shape returned by `GET /api/price`. `found` is `true` exactly when a finite `price > 0` was resolved.

```
{
  "found": true,
  "sku": "170857",
  "name": "PVC Dinghy Rub Rail White 10'",
  "price": 34.99,
  "currency": "USD",
  "unitOfMeasure": "each",
  "image": "https://cdn.example.com/170857.jpg",
  "description": "...",
  "available": "In Stock",
  "url": "https://shop.example.com/products/170857"
}
```

A missing SKU, a missing file, or a non-positive price returns `{ "found": false }`, which the reader renders as a request-for-quote line.

Upload Endpoint

PUT /api/inventory/<pubID>

Request

- **Path:** pubID — the publication identifier for the catalog.
- **Header:** Authorization: Bearer <token> (required).
- **Header:** Content-Type: application/json .
- **Body:** the inventory file (strict JSON).
- **Semantics:** full replace. Re-sending the same body is idempotent.

Example (Postman / curl)

```
PUT https://reader.digitalquire.com/api/inventory/120205201854
Authorization: Bearer dq_inv_7f3c...
Content-Type: application/json

{
  "schemaVersion": 1,
  "currency": "USD",
  "items": [
    { "sku": "170857", "name": "PVC Dinghy Rub Rail White 10'",
      "price": 34.99, "order_num": "AD3455" }
  ]
}
```

Success response 200

```
{
  "ok": true,
  "pubID": "120205201854",
  "itemCount": 1,
  "duplicateSkus": [],
  "updatedAt": "2026-10-26T12:00:00Z"
}
```

Status codes

Code	Meaning	When
200	Accepted & stored	Valid token + valid body. Returns the upload summary.
400	Bad request	Not valid JSON, empty/missing items , or an item missing sku / name . No write occurs.
401	Unauthorized	Missing or malformed Authorization: Bearer header.

Code	Meaning	When
403	Forbidden	Token does not match this publication's token (or none issued).
413	Payload too large	Body exceeds the configured maximum (default 10 MB).

Checks run in order: size (413) → bearer present (401) → token matches this publication (403) → body valid (400) → write (200). An invalid payload is rejected before anything is stored.

Authentication

The upload API is secured by a **per-catalog bearer token**:

- Presented as `Authorization: Bearer <token>`. Tokens look like `dq_inv_<random>`.
- Scoped to a single publication. A token issued for one `pubID` cannot write any other.
- Stored only as a salted hash and compared in constant time; the plaintext is shown once at issue and never stored or logged.
- Issued, rotated, and revoked by a Digital Quire operator. Contact your Digital Quire representative to obtain a token.

Keep the token secret. Treat it like a password. Transmit only over HTTPS. If it leaks, request a rotation — the old token stops working immediately.

Lookup Endpoint

GET `/api/price/<pubID>?sku=<sku>`

Used by the reader to resolve a product for a SKU. No auth required; it reads the catalog's uploaded inventory. Always responds `200` — a miss is expressed in the body as `{ "found": false }` rather than an error status.

```
GET https://reader.digitalquire.com/api/price/120205201854?sku=170857
→ 200 { "found": true, "sku": "170857", "price": 34.99, ... }
```

Integration Checklist

1. Obtain your `pubID` and a bearer token from your Digital Quire representative.
2. Build your inventory file as strict JSON with a non-empty `items` array; each item needs `sku` and `name`.
3. **PUT** it to `/api/inventory/<pubID>` with the bearer header. Confirm a `200` and the `itemCount`.
4. Spot-check a SKU via `GET /api/price/<pubID>?sku=` and confirm `found: true` with the expected price.
5. Re-upload whenever prices or availability change — each upload fully replaces the prior file.