Partner Catalog API Documentation

Overview

The Partner Catalog API provides programmatic access to curated product data from our store. This API allows partners to fetch product information, including details, variants, pricing, inventory, and images.

Base URL: URL WILL BE PROVIDED

API Version: `v1` **Content Type:** `application/json`

Authentication

All API requests require authentication using a Bearer token in the `Authorization` header.

Request Format

```
Authorization: Bearer YOUR_API_KEY
```

Getting Your API Key

Contact us to receive your unique API key. You will receive a partner key that is assigned a dedicated API key for secure access.

Example

```
curl -X GET "URL WILL BE PROVIDED " \
  -H "Authorization: Bearer your-api-key-here"
```

API Endpoints

1. List Products

Retrieve a paginated list of products from the Partner Catalog.

Endpoint: `GET /v1/catalog/products`

Query Parameters:

Parameter	Type	Required	Default	Description
`cursor`	string	No	-	Pagination cursor from previous response
`limit`	integer	No	25	Number of products per page (1-100)

Request Example:

```
curl -X GET "BASE_URL/v1/catalog/products?limit=50" \
  -H "Authorization: Bearer your-api-key-here"
```

Response Structure:

```
{
  "data": [
    {
      "id": "gid://.../Product/1234567890",
      "title": "Product Name",
      "handle": "product-handle",
      "status": "ACTIVE",
      "updatedAt": "2026-01-25T10:30:00Z",
      "vendor": "Brand Name",
      "productType": "Category",
      "tags": ["tag1", "tag2"],
      "descriptionHtml": "<p>Product description</p>",
      "featuredImage": {
        "url": "https://cdn.example.com/...",
        "altText": "Product image"
      },
      "images": [
        {
          "url": "https://cdn.example.com/...",
          "altText": "Product image"
        }
      ],
      "variants": [
        {
          "id": "gid://.../ProductVariant/9876543210",
          "sku": "SKU-123",
          "barcode": "1234567890123",
          "price": "29.99",
          "compareAtPrice": "39.99",
          "inventoryQuantity": 100,
          "inventoryPolicy": "DENY",
          "taxable": true,
          "inventoryItem": {
            "id": "gid://.../InventoryItem/1111111111",
            "requiresShipping": true,
            "measurement": {
              "weight": {
                "value": 1.5,
                "unit": "KILOGRAMS"
              }
            }
          }
        }
```

```
      ]
    }
  ],
  "pagination": {
    "hasNextPage": true,
    "nextCursor": "eyJsYXN0X2lkIjo..."
  },
  "meta": {
    "collection": "Partner Catalog",
    "count": 50
  }
}
```

---

2. Get Single Product

Retrieve detailed information for a specific product.

Endpoint: `GET /v1/catalog/products/{handle}` or `GET /v1/catalog/products?id={gid}`

Path Parameters:

Parameter	Type	Required	Description
`handle`	string	Yes*	Product handle (URL-friendly identifier)

Query Parameters:

Parameter	Type	Required	Description
`id`	string	Yes*	Product GID (e.g.,`gid://.../Product/1234567890`)

*Either `handle` (in path) or `id` (in query) is required.

Request Examples:

```
# Using product handle
curl -X GET "BASE_URL/v1/catalog/products/coconut-bar-soap" \
  -H "Authorization: Bearer your-api-key-here"

# Using product GID
curl -X GET "BASE_URL/v1/catalog/products?id=gid://.../Product/1234567890" \
  -H "Authorization: Bearer your-api-key-here"
```

Response Structure:


```
{
  "data": {
    "id": "gid://.../Product/1234567890",
    "title": "Product Name",
    "handle": "product-handle",
    "status": "ACTIVE",
    "updatedAt": "2026-01-25T10:30:00Z",
    "vendor": "Brand Name",
    "productType": "Category",
    "tags": ["tag1", "tag2"],
    "descriptionHtml": "<p>Product description</p>",
    "featuredImage": {
      "url": "https://cdn.example.com/...",
      "altText": "Product image"
    },
    "images": [
      {
        "url": "https://cdn.example.com/...",
        "altText": "Product image"
      }
    ],
    "variants": [
      {
        "id": "gid://.../ProductVariant/9876543210",
        "sku": "SKU-123",
        "barcode": "1234567890123",
        "price": "29.99",
        "compareAtPrice": "39.99",
        "inventoryQuantity": 100,
        "inventoryPolicy": "DENY",
        "taxable": true,
        "inventoryItem": {
          "id": "gid://.../InventoryItem/1111111111",
          "requiresShipping": true,
          "measurement": {
            "weight": {
              "value": 1.5,
              "unit": "KILOGRAMS"
            }
          }
        }
      }
    ]
  }
}
```

Data Models

Product Object

Field	Type	Description
`id`	string	Unique product identifier (GID)
`title`	string	Product title
`handle`	string	URL-friendly product identifier
`status`	string	Product status:`ACTIVE`, `ARCHIVED`, or `DRAFT`
`updatedAt`	string	ISO 8601 timestamp of last update
`vendor`	string	Product vendor/brand
`productType`	string	Product category/type
`tags`	array[string]	Product tags
`descriptionHtml`	string	HTML product description
`featuredImage`	object	Primary product image
`images`	array[object]	All product images
`variants`	array[Variant]	Product variants

Variant Object

Field	Type	Description
`id`	string	Variant unique identifier (GID)
`sku`	string	Stock Keeping Unit (may be null)
`barcode`	string	Product barcode (may be null)
`price`	string	Current price (decimal as string)
`compareAtPrice`	string	Original/compare price (may be null)
`inventoryQuantity`	integer	Available inventory count
`inventoryPolicy`	string	`DENY` or `CONTINUE` (sell when out of stock)
`taxable`	boolean	Whether variant is taxable
`inventoryItem`	object	Inventory item details

Inventory Item Object

Field	Type	Description
`id`	string	Inventory item identifier
`requiresShipping`	boolean	Whether item requires shipping
`measurement.weight.value`	number	Weight value
`measurement.weight.unit`	string	Weight unit (e.g.,`KILOGRAMS`, `POUNDS`)

Image Object

Field	Type	Description
`url`	string	Full image URL
`altText`	string	Image alt text (may be null)

Pagination

The list products endpoint uses cursor-based pagination for efficient data retrieval.

How It Works

1. **First request:** Omit the `cursor` parameter to start from the beginning.
2. **Next pages:** Use the `nextCursor` value from the previous response (`pagination.nextCursor`).
3. **More data?** Check `pagination.hasNextPage`; if `true`, call the API again with `cursor=pagination.nextCursor`.

Example Flow

```
# First page
GET /v1/catalog/products?limit=25

# Response includes:
{
  "pagination": {
    "hasNextPage": true,
    "nextCursor": "eyJsYXN0X2lkIjo..."
  }
}

# Next page (use nextCursor as cursor)
GET /v1/catalog/products?limit=25&cursor=eyJsYXN0X2lkIjo...
```

Best Practices

- Use a `limit` between 25–100 for optimal performance.
- Always check `hasNextPage` before requesting the next page.
- Store `nextCursor` if you need to resume pagination later.

Error Handling

The API uses standard HTTP status codes to indicate success or failure.

Status Codes

Code	Description
`200`	Success
`400`	Bad Request (invalid parameters)
`401`	Unauthorized (missing or invalid API key)
`403`	Forbidden (product not in Partner Catalog)
`404`	Not Found (product doesn't exist)
`500`	Internal Server Error

Error Response Format

```
{
  "error": "Error message describing what went wrong"
}
```

Common Errors

401 Unauthorized

```
{
  "error": "Unauthorized: Invalid or missing API key"
}
```

Solution: Verify your `Authorization` header includes a valid Bearer token.

403 Forbidden

```
{
  "error": "Forbidden: Product is not in Partner Catalog collection"
}
```

Solution: The requested product is not available in the Partner Catalog. Only products in the curated collection are accessible.

404 Not Found

```
{
  "error": "Product not found: product-handle"
}
```

Solution: Verify the product handle or GID is correct and the product exists.

400 Bad Request

```
{
  "error": "Invalid limit parameter (must be 1-100)"
}
```

Solution: Ensure query parameters are within valid ranges.

Rate Limits

To ensure fair usage and system stability, the API implements rate limiting:

- **Rate Limit:** 100 requests per minute per API key
- **Burst Limit:** 10 requests per second

Rate Limit Headers

Responses include rate limit information in headers:

```
X-RateLimit-Limit: 100
X-RateLimit-Remaining: 95
X-RateLimit-Reset: 1643123456
```

Handling Rate Limits

If you exceed the rate limit, you'll receive a `429 Too Many Requests` response:

```
{
  "error": "Rate limit exceeded. Please retry after 60 seconds."
}
```

Best Practices:

- Implement exponential backoff for retries
- Cache responses when possible
- Batch requests when appropriate
- Monitor rate limit headers

Webhooks

Receive real-time notifications when catalog data changes. When a change is detected, we send a **POST** request to your webhook URL with a JSON body.

Events

Event	When it fires
Update	Product details changed (title, price, etc.)

Event	When it fires
Delete	Product removed from catalog
Inventory	Stock quantity changed

Setup

Contact your account manager with:

- Your webhook URL (must be HTTPS)
- Events you want (update, delete, inventory)

You will receive a **webhook secret** to verify that requests really come from us.

Payload Example

Each POST body is JSON. Example for a product update:

```
{
  "id": 1234567890,
  "title": "Updated Product Name",
  "handle": "product-handle",
  "vendor": "Brand Name",
  "status": "active",
  "updated_at": "2026-01-25T10:30:00Z"
}
```

Verification

Every webhook request includes an **`X-Webhook-Signature`** header (HMAC-SHA256, base64-encoded). Use the **raw request body** (as received, before parsing JSON) and your webhook secret to verify the signature.

Example in Python:

```
import hmac
import hashlib
import base64

def verify_webhook(raw_body_bytes, signature_header, secret):
    expected = base64.b64encode(
        hmac.new(secret.encode(), raw_body_bytes, hashlib.sha256).digest()
    ).decode()
    return hmac.compare_digest(expected, signature_header)
```

Code Examples

Python

```python
import requests

API_BASE = 'BASE_URL'  # e.g. https://your-app.example.com
API_KEY = 'your-api-key-here'

def list_products(cursor=None, limit=25):
    url = f'{API_BASE}/v1/catalog/products'
    params = {'limit': limit}
    if cursor:
        params['cursor'] = cursor

    response = requests.get(
        url,
        params=params,
        headers={'Authorization': f'Bearer {API_KEY}'}
    )
    response.raise_for_status()
    return response.json()

def get_product(handle):
    url = f'{API_BASE}/v1/catalog/products/{handle}'
    response = requests.get(
        url,
        headers={'Authorization': f'Bearer {API_KEY}'}
    )
    response.raise_for_status()
    return response.json()

# Example usage
products = list_products(limit=50)
print(f"Found {len(products['data'])} products")

if products['pagination']['hasNextPage']:
    next_page = list_products(
        cursor=products['pagination']['nextCursor'],
        limit=50
    )
    print(f"Next page: {len(next_page['data'])} products")
```

Best Practices

1. Caching

Cache product data locally to reduce API calls and improve performance:

- Cache product details for 5-15 minutes
- Cache product lists for 1-5 minutes
- Invalidate cache on webhook notifications

2. Error Handling

Always implement proper error handling:

- Check HTTP status codes
- Handle rate limit errors with retry logic
- Log errors for debugging
- Implement exponential backoff for retries

3. Pagination

- Always check `hasNextPage` before fetching the next page
- Use appropriate `limit` values (25-100)
- Store `endCursor` if you need to resume pagination

4. Performance

- Make requests in parallel when fetching multiple products
- Use appropriate `limit` values to balance request count and payload size
- Monitor rate limit headers to avoid throttling

5. Security

- Never expose your API key in client-side code
- Store API keys securely (environment variables, secret managers)
- Rotate API keys periodically
- Verify webhook signatures if using webhooks

Support

Getting Help

- **Email:** feras.khouri@jafarshop.com

Reporting Issues

When reporting issues, please include:

- API endpoint and request details
- Response status code and body
- Timestamp of the request
- Your API key identifier (not the key itself)

License

This API is provided for authorized partners only. Unauthorized access or use is prohibited.