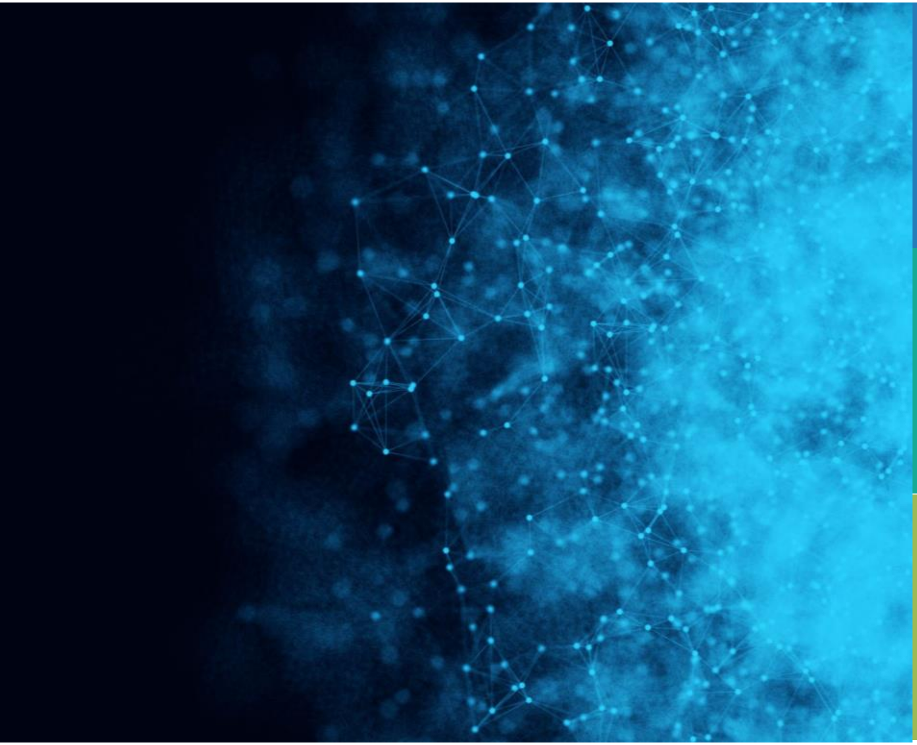
REST APIs

Application Integration

REST APIs



Simpler Alternative to SOAP

Developers frustrated with the complexities of SOAP embraced REST as an alternative. Simpler and looser standard that allows a client to access and manipulate data on a server resource.

REST Leverages Standard HTTP Behaviors

HTTP methods like GET, POST, PUT and DELETE are used to interact with resources. HTTP response codes like 200 or 403 are used for server replies. SOAP: remote procedure calls. REST: Resources

JSON

Although you can technically use any payload format like HTML or XML, JSON is almost always used for REST APIs.

REST Architecture

Client Server Architecture

Separation of concerns between the requesting process (client) and the API (server)..

Cacheability

HTTP caching approaches like proxy servers or content delivery networks should be supported.

Uniform Interface

Decoupled design where the server's representation of data is cleanly abstracted away from the interface exposed to the client. Payloads should be self-descriptive.

Statelessness

No context is saved between requests. Each request from a client to the server should contain all the data necessary to complete an action.

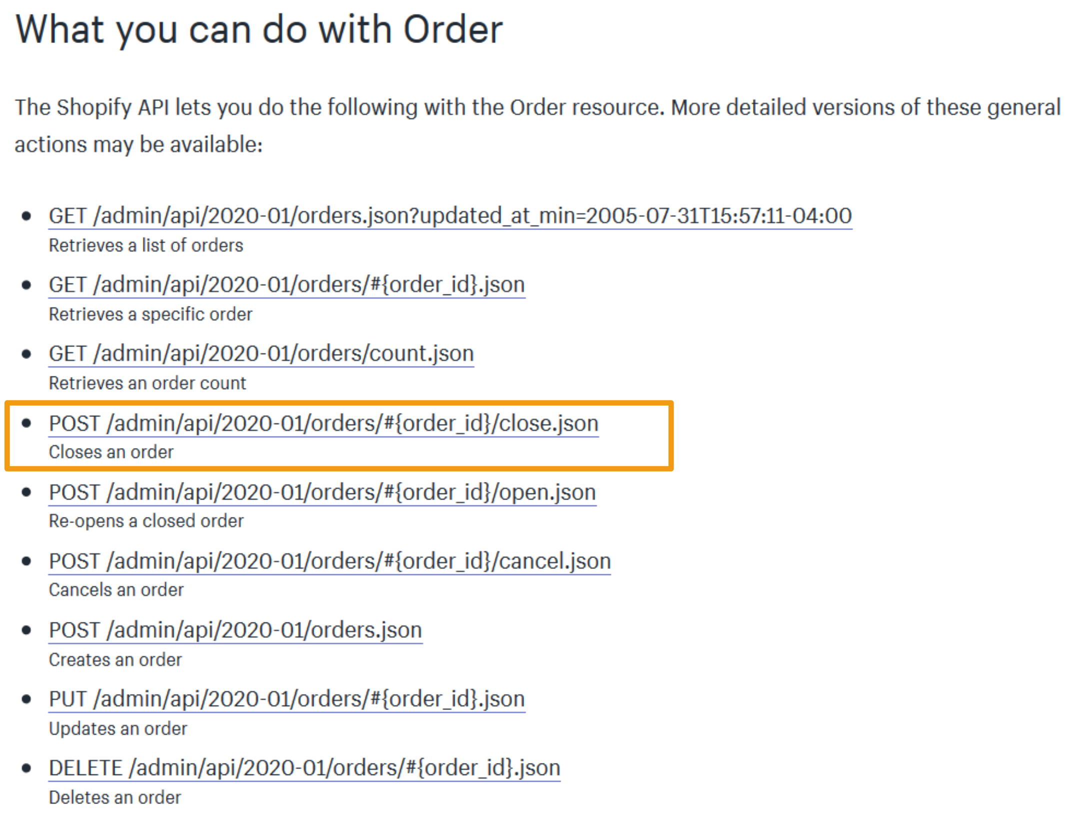
Layered System

It should be possible to add intermediary servers in between the client and server. Examples include API gateways or load balancers.

"CRUD" Operations

HTTP Verbs for CRUD

- Create – POST
- Read – GET
- Update – PUT
- Delete - DELETE



What you can do with Order

The Shopify API lets you do the following with the Order resource. More detailed versions of these general actions may be available:

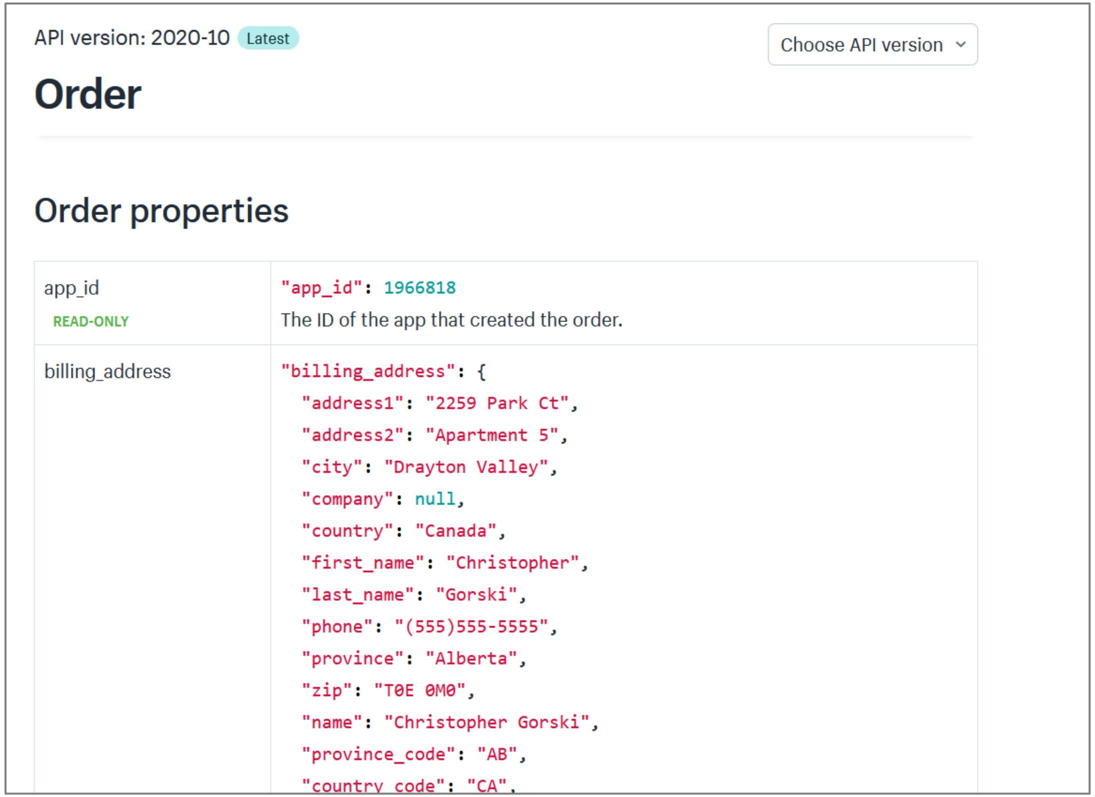
- GET /admin/api/2020-01/orders.json?updated_at_min=2005-07-31T15:57:11-04:00
  Retrieves a list of orders
- GET /admin/api/2020-01/orders/#{order_id}.json
  Retrieves a specific order
- GET /admin/api/2020-01/orders/count.json
  Retrieves an order count
- POST /admin/api/2020-01/orders/#{order_id}/close.json
  Closes an order
- POST /admin/api/2020-01/orders/#{order_id}/open.json
  Re-opens a closed order
- POST /admin/api/2020-01/orders/#{order_id}/cancel.json
  Cancels an order
- POST /admin/api/2020-01/orders.json
  Creates an order
- PUT /admin/api/2020-01/orders/#{order_id}.json
  Updates an order
- DELETE /admin/api/2020-01/orders/#{order_id}.json
  Deletes an order

https://help.shopify.com/en/api/reference/orders/order

Retrieving a Single Resource

GET orders

https://api.shopify.com/admin/api/2020-01/orders/#123


API version: 2020-10 Latest

Choose API version

Order

Order properties

app_id READ-ONLY	"app_id": 1966818 The ID of the app that created the order.
billing_address	see below

```
"billing_address": {
  "address1": "2259 Park Ct",
  "address2": "Apartment 5",
  "city": "Drayton Valley",
  "company": null,
  "country": "Canada",
  "first_name": "Christopher",
  "last_name": "Gorski",
  "phone": "(555)555-5555",
  "province": "Alberta",
  "zip": "T0E 0M0",
  "name": "Christopher Gorski",
  "province_code": "AB",
  "country_code": "CA",
```

https://help.shopify.com/en/api/reference/orders/order#show-2020-01

Retrieving a List of Resources

GET /orders

https://api.shopify.com/admin/api/2020-01/orders?status=open

Additional Search Parameters:

Created and updated dates, financial status, and fulfillment status

Paging

https://api.shopify.com/admin/api/2020-01/orders?limit=100&page=2&sort=last_modified

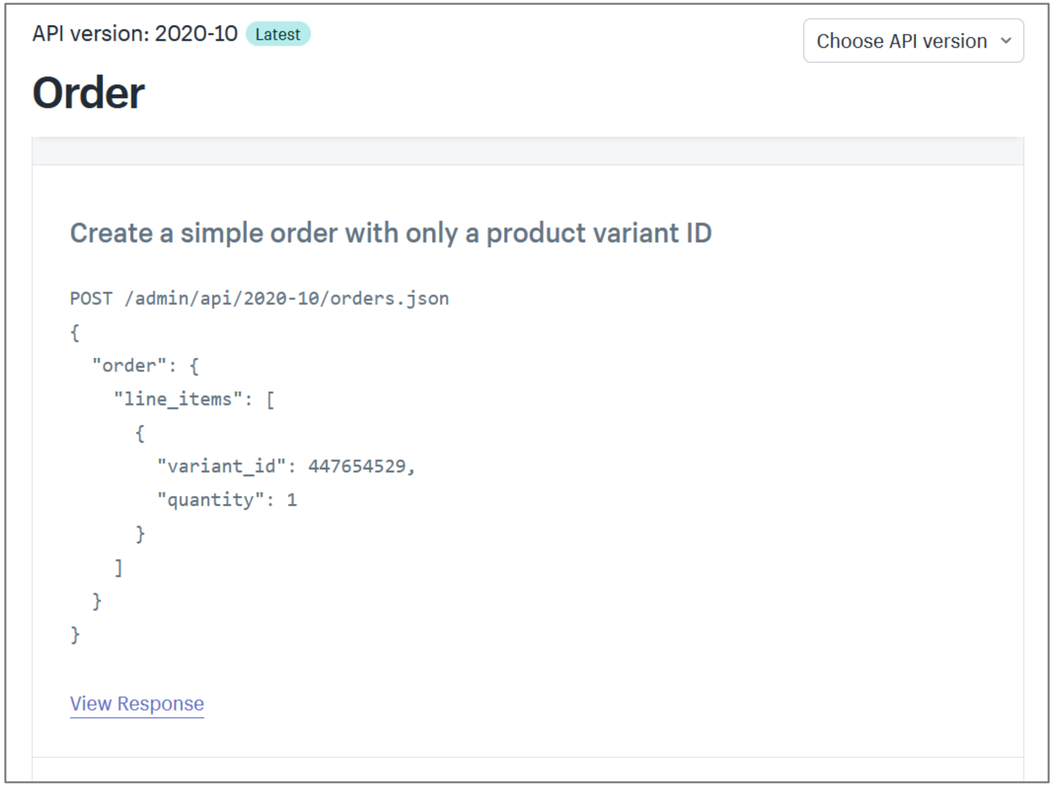
Creating a Resource

POST /orders

https://api.shopify.com/admin/api/2020-01/orders

HTTP Response Codes

- 201 – Record successfully added
- 400 – Error adding your order



Order

Create a simple order with only a product variant ID

```
POST /admin/api/2020-10/orders.json
{
  "order": {
    "line_items": [
      {
        "variant_id": 447654529,
        "quantity": 1
      }
    ]
  }
}
```

View Response

https://help.shopify.com/en/api/reference/orders/order#show-2020-01

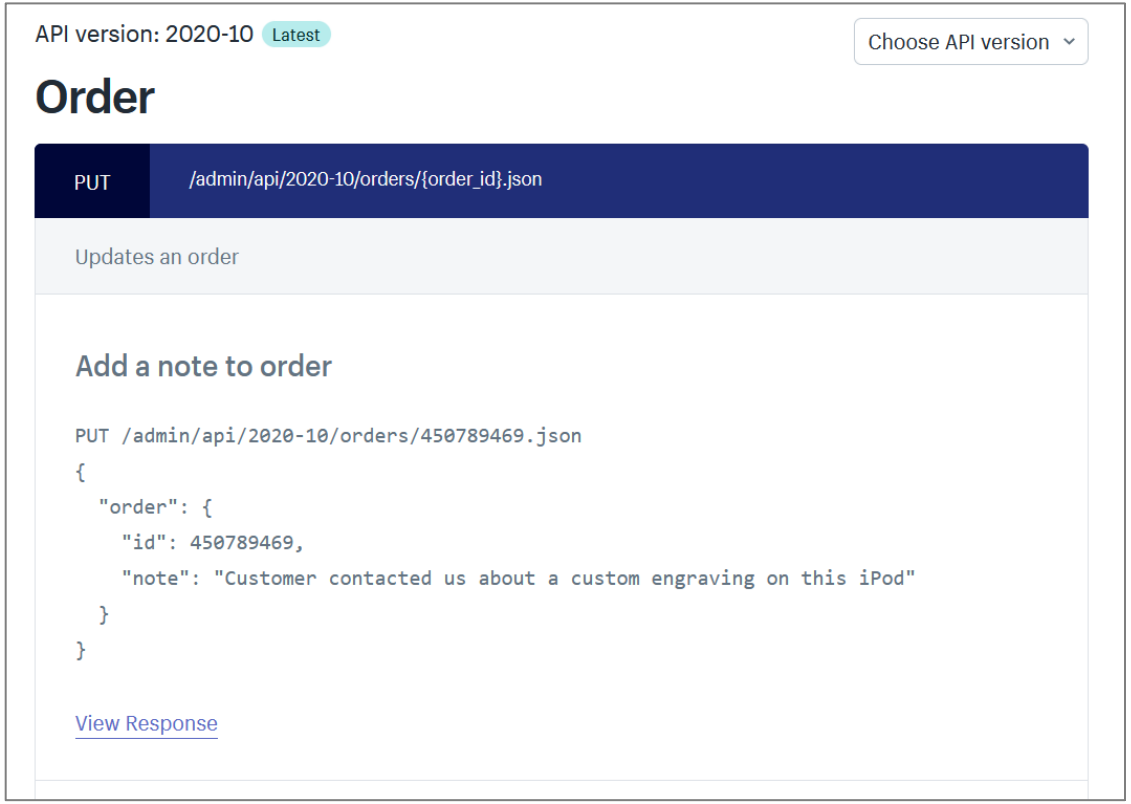
Updating a Resource

PUT /orders

https://api.shopify.com/admin/api/2020-01/orders/#123

HTTP Response Codes

- 200 – Record successfully added
- 400 – Error updating your order



Order

PUT /admin/api/2020-10/orders/{order_id}.json

Updates an order

Add a note to order

```
PUT /admin/api/2020-10/orders/450789469.json
{
  "order": {
    "id": 450789469,
    "note": "Customer contacted us about a custom engraving on this iPod"
  }
}
```

View Response

https://help.shopify.com/en/api/reference/orders/order#show-2020-01

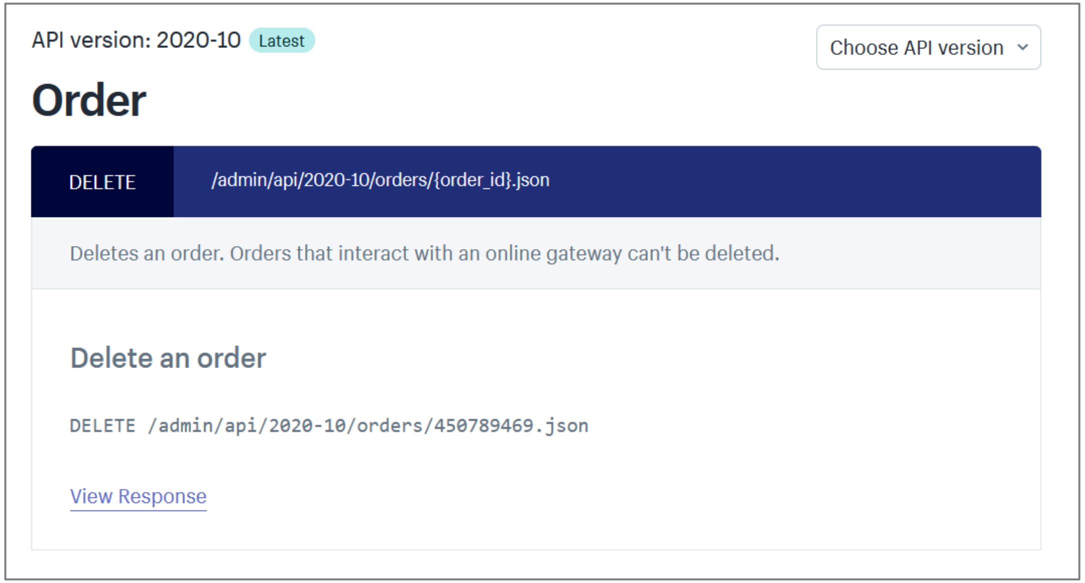
Deleting a Resource

DELETE /orders

https://api.shopify.com/admin/api/2020-01/orders/#123

HTTP Response Codes

- 200 – Record successfully added
- 400 – Error updating your order


API version: 2020-10 Latest

Choose API version

Order

DELETE /admin/api/2020-10/orders/{order_id}.json

Deletes an order. Orders that interact with an online gateway can't be deleted.

Delete an order

```
DELETE /admin/api/2020-10/orders/450789469.json
```

View Response

https://help.shopify.com/en/api/reference/orders/order#show-2020-01

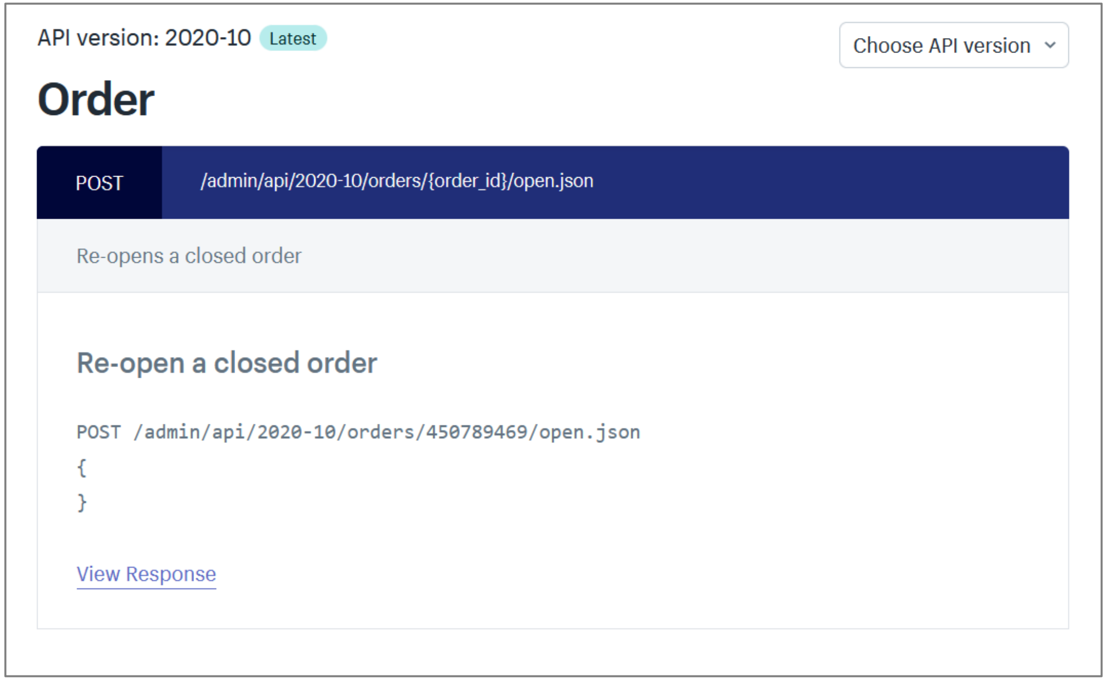
Non-CRUD Operations

POST /orders

https://api.shopify.com/admin/api/2020-01/orders/#123/open

HTTP Response Codes

- 200 – Record successfully added
- 400 – Error updating your order



API version: 2020-10 Latest

Choose API version

Order

POST /admin/api/2020-10/orders/{order_id}/open.json

Re-opens a closed order

Re-open a closed order

```
POST /admin/api/2020-10/orders/450789469/open.json
{
}
```

View Response

https://help.shopify.com/en/api/reference/orders/order#show-2020-01

Content Types

Content Negotiation

The process of selecting the best representation for a given response when multiple representations are available.

Client HTTP Headers

Content-Type : text/plain
Content-Type : application/xml
Content-Type : application/json
Content-Type : text/html
Content-Type : image/gif
Content-Type : image/jpeg

Rate Limiting

Rate Limiting
Rate limiting controls how often clients can make calls to API endpoints.

Throttling
Server will keep track of the number of requests over a certain period of time and "throttle" the client

Rate Limit Methods
Maximum number of calls that can be made within a certain period of time. Quotas. Leaky bucket algorithm.

Why Rate Limits?
Without rate limits, clients could exhaust server resources by executing a large number of API calls.

HTTP 429 Response
When rate limits are exceeded a 429 – Too many requests status code is normally returned.

Tracking Quotas
Some APIs provide status about rate limit consumption in the HTTP headers for each API response. This way the client can check how much of the quote they have consumed as of the last call.

https://en.wikipedia.org/wiki/Leaky_bucket

Avoiding Rate Limit Problems

1. Be careful with client applications that poll a particular API on a scheduled basis.
2. Check for the HTTP response code of 429 (or whatever the API returns for exceeding quota).
3. Avoid retrying errors in a code loop. When a non-success response is detected by the server, the client should not repeatedly retry the call without pay attention to the cause or the error condition
4. Use caching when possible

CORS

https://www.site1.com → https://www.site2.com

JavaScript API Call

https://geekflare.com/cors-basics/