

[Create workspace - To work on APIs](#)

[Delete WorkSpace](#)

[Create Collection - To have HTTP Request](#)

[CRUD API Testing](#)

[GET Request](#)

[GET Status](#)

[Set a Variable](#)

[GET List of Books](#)

[GET List of Books By Query Parameters](#)

[GET Book By ID](#)

[POST Request - Submit Order without Access Token](#)

[HTTP Status Codes Reference](#)

[● 1XX – Informational Responses](#)

[● 2XX – Success Responses](#)

[● 3XX – Redirection Responses](#)

[● 4XX – Client Errors](#)

[● 5XX – Server Errors](#)

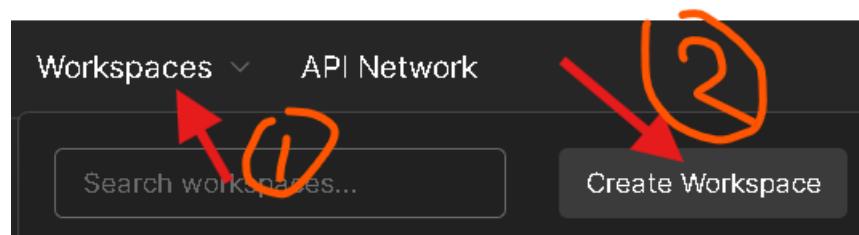
[POST Request\[Register API Client\] - To Access Token](#)

[POST Request - Submit Order with Access Token](#)

[PATCH Request - Update the Order](#)

[DELETE Request - Delete an Order](#)

## Create workspace - To work on APIs



**Create your workspace**

Workspace name  
Created Postman Workspace

Select workspace type  
 Internal Build and test APIs within your team.  
 Partner TRY FOR FREE Share APIs securely with trusted partners.  
 Public Make APIs accessible to the world.

Choose how you'd like to set up this workspace  
 Blank workspace I want to start from scratch.  
 From a Git repository I want to connect my existing Git project.  
 Choose a Folder

❖ Create with Postman AI  
Ask AI to set up collections, specs, and environments for your API.

Describe what you want to build (e.g., "Payments API").

Everyone in your team can access ▾  
Cancel **Create Workspace**

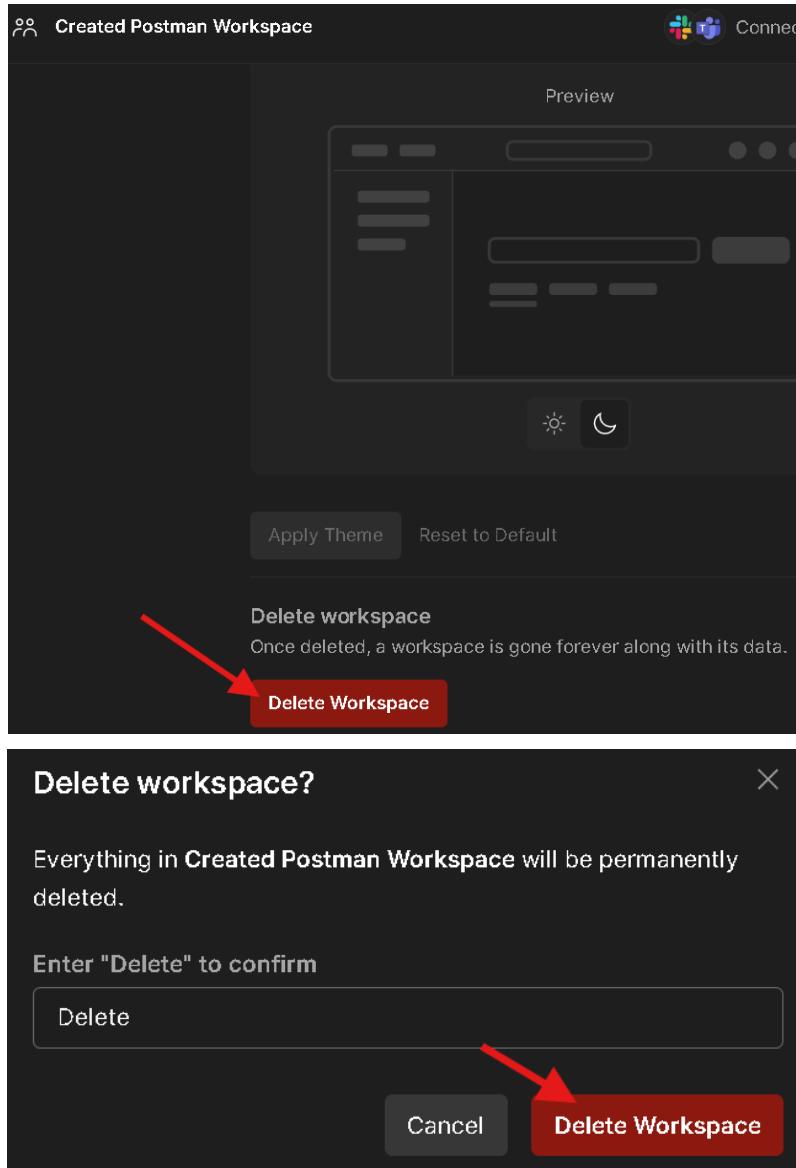
## Delete WorkSpace

**Created Postman Workspace**

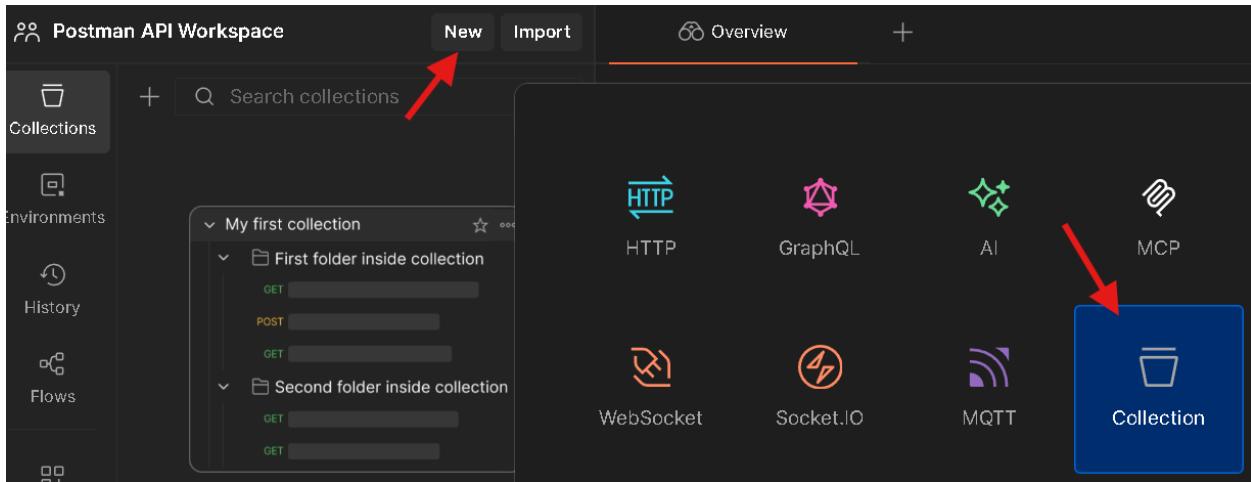
Collections + Search collections  
HTTP Request Collections GET Get Request

Overview Updates Settings

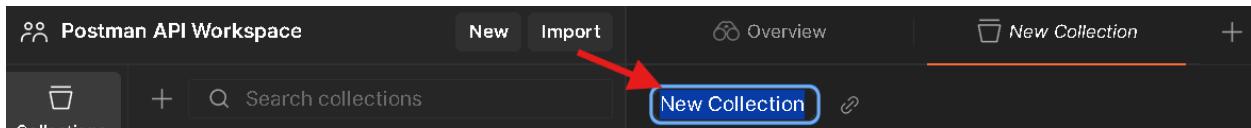
*Scroll down*



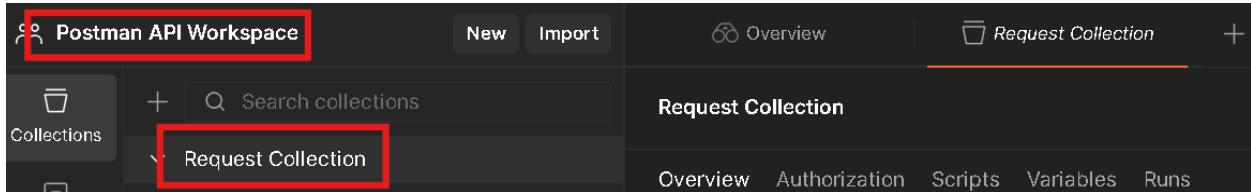
## Create Collection - To have HTTP Request



**Name the collection**



**Finally Workspace and Collection to keep request is created**



## CRUD API Testing

**CRUD = Create, Read, Update, Delete**

It means testing all basic operations of an API.

Operation	HTTP Method	Purpose
Create	POST	Add new data
Read	GET	Fetch data
Update	PUT / PATCH	Modify data
Delete	DELETE	Remove data

## GET Request

### GET Status

The screenshot shows the Postman API Workspace interface. At the top, there are 'New' and 'Import' buttons. Below them is a search bar labeled 'Search collections'. A red arrow points to the 'Request Collection' section, which is currently empty. It features a '+' button, a star icon, and a three-dot menu icon. To the right, there is a large 'Add request' button.

Rename the HTTP(Hypertext Transfer Protocol) - GET Request

This screenshot shows the Postman API Workspace with the 'New Request' tab selected. The 'Request Collection' section is visible, and a red arrow points to the 'New Request' button, which is highlighted with a blue border. Below the collection list, there is a search bar and a dropdown menu set to 'GET'.

### Notes

This screenshot displays the Simple Books API documentation. On the left, the title 'Simple Books API' is shown, along with a brief description: 'This API allows you to reserve a book.' Below that, it says 'The API is available at <https://simple-books-api.click>'. On the right, there is a 'Endpoints' section with a 'Status' endpoint listed: 'GET /status'. A note below it states: 'Returns the status of the API.'



Write -> Save -> Send

The screenshot shows the Postman API Workspace with a completed GET request. The URL 'https://simple-books-api.click/status' is entered in the request field. A red arrow points to the 'Send' button, which is highlighted with a blue border. Other buttons like 'Save' and 'Share' are also visible.

Body  200 OK

{ } JSON ▾ Preview Visualize | ▾

```
1 {  
2   "status": "OK"  
3 }
```

## Set a Variable

Select the API

HTTP Request Collection / GET Status

GET https://simple-books-api.click/status

Right click on it

HTTP Request Collections / Get Request

GET https://simple-books-api.click/status

Docs Params Authoriza Cut

Set as variable

Set as new variable

Name: BaseURL

Value: https://simple-books-api.click

Scope: Collection

{{{BaseURL}}}/status

To Look collection Variables: Click target collection -> Variables

Request Collection

GET Status

Overview Authorization Scripts Variables Runs

Variable	Value
BaseUrl	https://simple-books-api.click

## GET List of Books

### Notes

## List of books

GET /books

Returns a list of books.

Optional query parameters:

- type: fiction or non-fiction
- limit: a number between 1 and 20.

*Let's get list of books*

*Add a request to collection*

▼ Request Collection

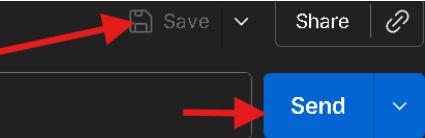
GET GET Status

Add request

*Rename it*



HTTP Request Collection / **GET List of Books**



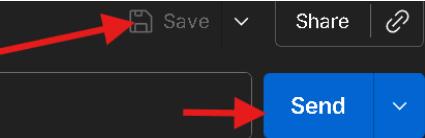
*Provide -> save -> Send*

HTTP Request Collection / **GET List of Books**

GET {{BaseUrl}} /books

Save Share

Send



Body JSON

```

21      "id": 6,
22      "name": "Viscount Who Loved Me",
23      "type": "fiction",
24      "available": true
25    },
26    {
27      "id": 2,
28      "name": "Just as I Am",
29      "type": "non-fiction",
30      "available": false
31    },
32    {
33      "id": 5,
34      "name": "Untamed",
35      "type": "non-fiction",
36      "available": true
37    }
38  ]

```

### GET List of Books By Query Parameters

Create request -> Write -> save -> Send

Request Collection / GET List of Books by Query

HTTP Method: GET

URL: {{BaseUrl}} /books?type=fiction&limit=2

Params Tab (highlighted):

- Docs
- Params**
- Authorization
- Headers (7)
- Body
- Scripts
- Settings

Buttons: Save, Share, Send (highlighted)

During Write it automatically adds

Params Tab (highlighted):

Key	Value
type	fiction
limit	2

Where type if type of book, limit is number of books to watch

Body JSON

```

3      "id": 1,
4      "name": "The Russian",
5      "type": "fiction",
6      "available": true
7    },
8    {
9      "id": 3,
10     "name": "The Vanishing Half",
11     "type": "fiction",

```

## GET Book By ID

**Similarly, let's GET Book by ID**

Add request -> Write -> Save -> Send

The screenshot shows the Postman interface with a dark theme. On the left, under 'Request Collection', there are three items: 'GET Status', 'GET List of Books', and 'GET Book by ID'. A red arrow points from the bottom of the 'Request Collection' list to the 'GET Book by ID' item. In the main workspace, a 'GET' request is selected with the URL {{BaseUrl}} /books/2. Another red arrow points from the URL field to the 'Send' button at the top right. The 'Send' button is highlighted in blue. Below the URL, the response status is shown as '200 OK' with a timestamp of '517 ms'. At the bottom, there are tabs for Body, Params, Authorization, Headers (7), Body, Scripts, and Settings.

## POST Request - Submit Order without Access Token

### Notes

#### Submit an order

POST /orders

Allows you to submit a new order. Requires authentication.

The request body needs to be in JSON format and include the following properties:

- bookId - Integer - Required
- customerName - String - Required

#### Example

```
POST /orders/
Authorization: Bearer <YOUR TOKEN>

{
  "bookId": 1,
  "customerName": "John"
}
```

The response body will contain the order Id.

*Create -> Write -> Save -> Send*

The screenshot shows the Postman interface with a collection named "Request Collection". A POST request titled "POST Submit Order" is selected. The URL is set to `POST {{BaseUrl}}/orders`. The "Authorization" tab is highlighted. A red arrow points from the URL field to the URL bar. Another red arrow points from the "Authorization" tab to the "Authorization" section in the request details. A third red arrow points from the "Send" button to the "Send" button in the top right corner.

### Results

The results panel shows a 401 Unauthorized response. The body is displayed in JSON format:

```
1 {  
2 |   "error": "Missing Authorization header."  
3 }
```

## HTTP Status Codes Reference

### ● 1XX – Informational Responses

Code	Name	Description
100	Continue	<i>Client should continue with the request</i>
101	Switching Protocols	<i>Server agrees to switch protocols</i>
102	Processing	<i>Server is processing the request (WebDAV)</i>

### ● 2XX – Success Responses

Code	Name	Description
200	OK	<i>Request succeeded and data returned</i>
201	Created	<i>Resource successfully created</i>
202	Accepted	<i>Request accepted but processing not complete</i>
203	Non-Authoritative Information	<i>Meta-information from third party</i>
204	No Content	<i>Request succeeded, no response body</i>

---

205	<i>Reset Content</i>	<i>Client should reset the document view</i>
206	<i>Partial Content</i>	<i>Partial GET successful (range request)</i>

---

## 3XX – Redirection Responses

<b>Code</b>	<b>Name</b>	<b>Description</b>
300	<i>Multiple Choices</i>	<i>Multiple options for the resource</i>
301	<i>Moved Permanently</i>	<i>Resource permanently moved to new URL</i>
302	<i>Found</i>	<i>Temporarily moved to another URL</i>
303	<i>See Other</i>	<i>Response found at different URL</i>
304	<i>Not Modified</i>	<i>Cached version should be used</i>
307	<i>Temporary Redirect</i>	<i>Temporary redirect (same method)</i>
308	<i>Permanent Redirect</i>	<i>Permanent redirect (same method)</i>

---

## 4XX – Client Errors

<b>Code</b>	<b>Name</b>	<b>Description</b>
400	<i>Bad Request</i>	<i>Server cannot understand request</i>
401	<i>Unauthorized</i>	<i>Authentication required or failed</i>
403	<i>Forbidden</i>	<i>Server refuses to authorize</i>
404	<i>Not Found</i>	<i>Resource not found</i>
405	<i>Method Not Allowed</i>	<i>Method not allowed for URL</i>
406	<i>Not Acceptable</i>	<i>Resource not acceptable per headers</i>
407	<i>Proxy Authentication Required</i>	<i>Client must authenticate with proxy</i>
408	<i>Request Timeout</i>	<i>Server timed out waiting</i>
409	<i>Conflict</i>	<i>Conflict with current server state</i>

<b>410</b>	<i>Gone</i>	<i>Resource no longer available</i>
<b>411</b>	<i>Length Required</i>	<i>Content-Length missing</i>
<b>412</b>	<i>Precondition Failed</i>	<i>Request header conditions failed</i>
<b>413</b>	<i>Payload Too Large</i>	<i>Request entity too large</i>
<b>414</b>	<i>URI Too Long</i>	<i>URI too long</i>
<b>415</b>	<i>Unsupported Media Type</i>	<i>Media format not supported</i>
<b>417</b>	<i>Expectation Failed</i>	<i>Server cannot meet request expectation</i>
<b>429</b>	<i>Too Many Requests</i>	<i>Too many requests in short time</i>

---

## ● 5XX – Server Errors

<b>Code</b>	<b>Name</b>	<b>Description</b>
<b>500</b>	<i>Internal Server Error</i>	<i>Unexpected server condition</i>
<b>501</b>	<i>Not Implemented</i>	<i>Method not supported</i>
<b>502</b>	<i>Bad Gateway</i>	<i>Invalid response from upstream server</i>
<b>503</b>	<i>Service Unavailable</i>	<i>Server overloaded or down</i>
<b>504</b>	<i>Gateway Timeout</i>	<i>No timely response from upstream</i>
<b>505</b>	<i>HTTP Version Not Supported</i>	<i>HTTP version not supported</i>

## API Authentication

To submit or view an order, you need to register your API client.

POST /api-clients/

The request body needs to be in JSON format and include the following properties:

- `clientName` - String
- `clientEmail` - String

Example

```
{  
  "clientName": "Postman",  
  "clientEmail": "valentin@example.com"  
}
```

The response body will contain the access token. The access token is valid for 7 days.

Possible errors

Status code 409 - "API client already registered." Try changing the values for `clientEmail` and `clientName` to something else.

Access token is like a temporary password, and using for the request

### POST Request[Register API Client] - To Access Token

Create -> Name it -> Write URL & Body raw -> Save -> Send

The screenshot shows the Postman interface with a collection named "Request Collection". A POST request titled "POST Register API Client" is selected. The URL is set to `POST {{BaseUrl}}/api-clients`. The "Body" tab is selected, showing a raw JSON payload:

```
1 {  
2   "clientName": "PostmanAPI",  
3   "clientEmail": "PostmanAPI@example.com"  
4 }
```

Red arrows point to the "Send" button, the "Body raw" field, and the "Params" tab in the header.

*Body raw Json =*

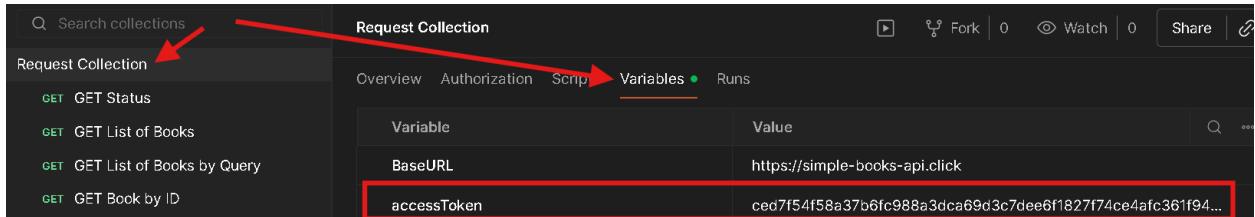
```
"clientName": "PostmanAPI",  
"clientEmail": "PostmanAPI@example.com"
```

*Results:*

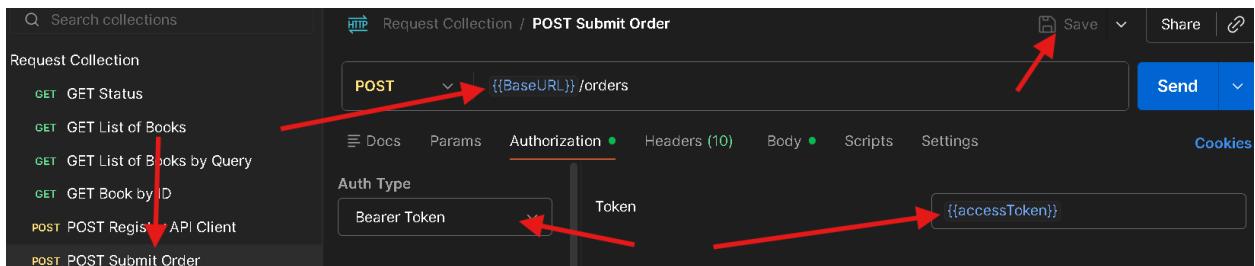
The results panel shows a successful `201 Created` response with a response time of `376 ms` and a response size of `826 B`. The response body is displayed in JSON format:

```
1 {  
2   "accessToken": "ced7f54f58a37b6fc988a3dca69d3c7dee6f1827f74ce4afc361f94ea9c9e62e"  
3 }
```

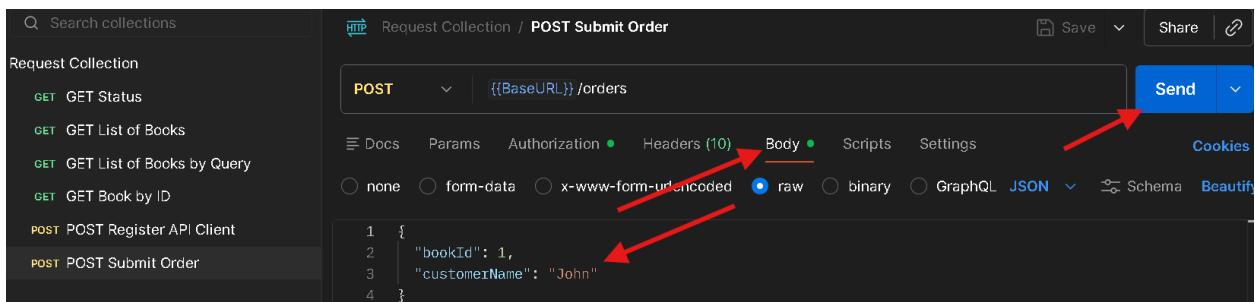
**Save token as Collection Variable**



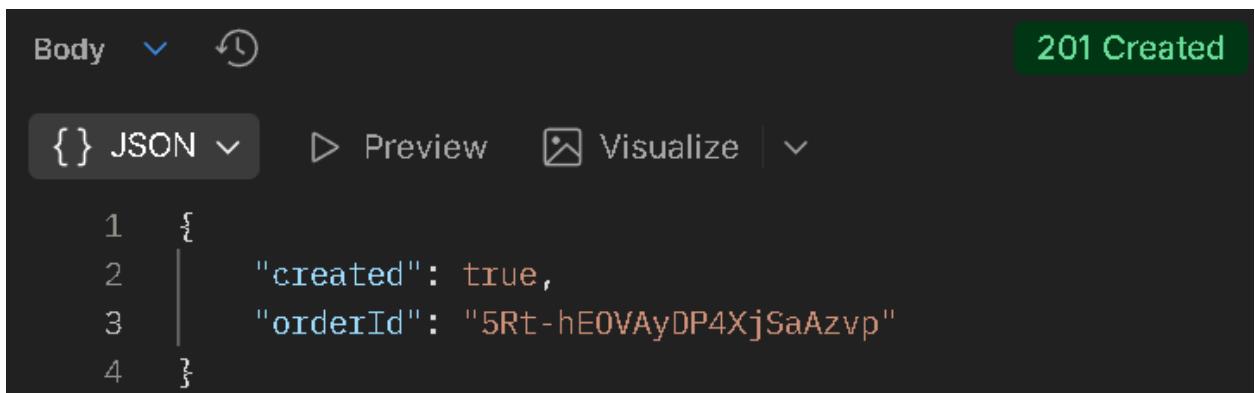
## POST Request - Submit Order with Access Token



**Body includes Make sure JSON is chosen**



**Results**



```

1 {
2   "created": true,
3   "orderId": "5Rt-hE0VAYDP4XjSaAzvp"
4 }

```

**Order is created if its in the stock.**

**For ID = 2**

```
1  {
2    "bookId": 2,
3    "customerName": "John"
4 }
```

Body 404 Not Found

{ } JSON ▾ ▶ Preview ⚡ Debug with AI ▾

```
1  {
2    "error": "This book is not in stock. Try again later."
3 }
```

### Proof

Request Collection + ⌛

- GET GET Status
- GET GET List of Books
- GET GET List of Books by Query
- GET GET Book by ID
- POST POST Register API Client
- POST POST Submit Order

GET {{BaseUrl}} /books

Docs Params Authorization • Headers (8)

none  form-data  x-www-form-urlencoded

Body

{ } JSON ▾ ▶ Preview ⚡ Visualize ▾

```
25  },
26  {
27    "id": 2,
28    "name": "Just as I Am",
29    "type": "non-fiction",
30    "available": false
31 }
```



## PATCH Request - Update the Order

Get Token from submit order

**POST** POST Submit Order

**PATCH** PATCH Update Order

```

2   |   "bookId": 1,
3   |   "customerName": "PostmanAPI"
4   |

```

Body { } JSON ▶ Preview Visualize

```

1   {
2   |   "created": true,
3   |   "orderId": "cuYLSe0ZIAEqMrNB309Qr"
4   |

```

Create -> Write -> Params -> Write token

Request Collection

- GET GET Status
- GET GET List of Books
- GET GET List of Books by Query
- GET GET Book by ID
- POST POST Register API Client
- POST POST Submit Order
- PATCH PATCH Update Order**

**PATCH** {{BaseUrl}} /orders/:orderId

Params

Key	Value	Description	Bulk Edit
orderId	cuYLSe0ZIAEqMrNB309Qr	Description	

Path Variables

Auth Type

Bearer Token {{accessToken}}

**PATCH** {{BaseUrl}} /orders/:orderId

Params Authorization Headers (10) Body Scripts Settings

Auth Type

Bearer Token {{accessToken}}

**PATCH** {{BaseUrl}} /orders/:orderId

Params Authorization Headers (10) Body Scripts Settings

Content-Type

none form-data x-www-form-urlencoded  raw  binary  GraphQL **JSON**

```

1   {
2   |   "bookId": 1,
3   |   "customerName": "new PostmanAPI"
4   |

```

The screenshot shows a Postman collection named "Request Collection". A PATCH request titled "PATCH Update Order" is selected. The URL is set to `{{BaseUrl}}/orders/:orderId`. The "Params" tab is active, showing a path variable `orderId` with the value `cuYLSe0ZlAEqMrNB309Qr`. The response status is shown as "204 No Content".

*To check the order*

The screenshot shows a GET request titled "GET Order" from the same collection. The URL is `{{BaseUrl}}/orders/:orderId`. The "Params" tab is active, showing the same `orderId` value. Red arrows point to the URL and the parameter value.

The screenshot shows the "Authorization" tab for the "GET Order" request. It is configured with "Bearer Token" and the token value `{{accessToken}}`. A red arrow points to the token input field.

*Then it actually changes*

{ } JSON ▾

Preview Visualize | ▾

```

1  {
2      "id": "cuYLSe0ZTAEqMrNB3090r",
3      "bookId": 1,
4      "customerName": "new PostmanAPI",
5      "quantity": 1,
6      "timestamp": 1768394300358
7  }

```

## DELETE Request - Delete an Order

Request Collection

- GET GET Status
- GET GET List of Books
- GET GET List of Books by Query
- GET GET Book by ID
- POST POST Register API Client
- POST POST Submit Order
- PATCH PATCH Update Order
- GET GET Order
- DEL DELETE Order

HTTP Request Collection / **DELETE Order**

**DELETE** ↘ {{BaseUrl}}/orders/:orderId

Docs	Params	Authorization	Headers (8)	Body	Scripts	Settings	Cookies
	Key Value Description						
	Key Value Description						

Path Variables

Key	Value	Description	Bulk Edit
orderId	cuYLSe0ZTAEqMrNB3090r	Description	

HTTP Request Collection / **DELETE Order**

**DELETE** ↘ {{BaseUrl}}/orders/:orderId

Docs	Params	Authorization	Headers (8)	Body	Scripts	Settings
		Auth Type				

Auth Type

Bearer Token ↘

Token → {{accessToken}}

HTTP Request Collection / **DELETE Order**

**DELETE** `{{BaseUrl}}/orders/:orderId` **Send**

Docs Params Authorization **Headers (8)** Body Scripts Settings Cookies

Auth Type  
Bearer Token

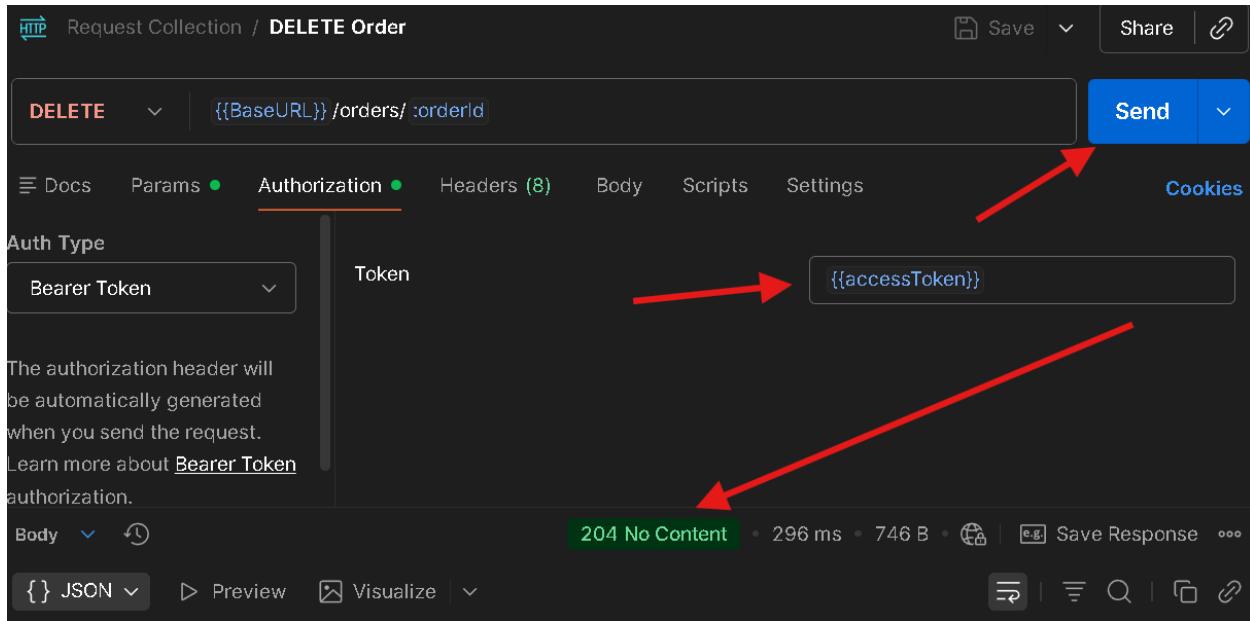
The authorization header will be automatically generated when you send the request.  
Learn more about [Bearer Token](#) authorization.

Token `accessToken`

Body `204 No Content` • 296 ms • 746 B • [Save Response](#)

{ } JSON ▾ ▶ Preview Visualize

**Check Is it deleted**



HTTP Request Collection / **GET Order**

**GET** `{{BaseUrl}}/orders/:orderId` **Send**

Docs Params Authorization **Headers (8)** Body Scripts Settings Cookies

Key	Value	Description	...	Bulk Edit
orderId	cuYLSe0ZIAEqMrNB309Qr	Description		

Body `404 Not Found` • 319 ms • 798 B • [Save Response](#)

{ } JSON ▾ ▶ Preview [Debug with AI](#)

```
1 {  
2 | "error": "No order with id cuYLSe0ZIAEqMrNB309Qr."  
3 }
```

