**What are the different types of HTTP status codes?**

HTTP status codes are grouped into five classes, each beginning with a number that represents the type of response. The classes are:

**TYPES OF ERRORS**

**1xx informational responses**

These codes indicate that the server has received the request and is processing it. They are primarily used to manage communication between the client and server during the early stages of a request-response cycle. Some examples of this type of code are:

**100 Continue:** This status code indicates that the initial part of the request has been received and the server would like the client to send the rest of it.

**101 Switching Protocols**: This status code is used to inform the client that the server is changing the protocol that is being used in the connection.

102 Processing: This status code is an interim response that indicates the server is still processing the request.

**2xx success responses**

These codes indicate that the client’s request was successfully received, understood, and processed by the server. Some of the most common 200 responses are:

**200 OK**: This status code indicates that the request was successful, and the server returned the requested data.

**201 Created**: This status code means that the request was successful, and the server created a new resource.

**204 No Content**: This status code indicates that the request was successful, but the server did not return any data.

**3xx redirection responses**

These codes indicate that the client needs to take additional actions to fulfill the request. They are often used when the requested resource has moved to a different location. Some examples include:

**301 Moved Permanently**: This status code indicates that the requested resource has been permanently moved to a new URL. Clients should respond by updating their bookmarks and links to point to the new URL, and search engines should update their indexes with the new location.

**303 See Other**: This status code indicates that the response is available at a different URL, and the client should perform a GET request to that URL to retrieve the resource.

**4xx client error responses**

These codes indicate that there was an issue with the client’s request, such as a mistyped URL or invalid credentials. The most common 4xx responses include:

**400 Bad Request**: This status code indicates that the request was malformed or invalid.

**401 Unauthorized**: This status code lets the client know that it is not authorized to access the requested resource.

**403 Forbidden**: This status code communicates that the client is authenticated but not authorized to access the requested resource.

**404 Not Found**: This status code indicates that the requested resource was not found on the server.

**409 conflict:** Indicates that something went wrong essentially this api client has already registered.

**5xx server error responses**

These codes, which indicate that the server encountered an error while trying to fulfill the client’s request, include:

**500 Internal Server Error**: This generic error code indicates the server encountered an unexpected condition that prevented it from fulfilling the request.

**502 Bad Gateway**: This status code indicates that a server acting as a gateway or proxy received an invalid response from an upstream server.

**503 Service Unavailable**: This status code is returned when the server is temporarily unable to handle the request. It’s often seen during periods of increased traffic or when the server is undergoing maintenance.

**TYPES OF METHODS**

What are the most common HTTP methods?

HTTP methods enable API clients to perform [CRUD](https://www.postman.com/api-glossary/" \l "crud) (Create, Read, Update, and Delete) actions on an API’s resources in a standardized and predictable way. The most commonly used HTTP methods are:

**GET**

The GET method is used to retrieve data on a server. Clients can use the GET method to access all of the resources of a given type, or they can use it to access a specific resource. For instance, a GET request to the /products endpoint of an e-commerce API would return all of the products in the database, while a GET request to the /products/123 endpoint would return the specific product with an ID of 123. GET requests typically do not include a request body, as the client is not attempting to create or update data.

**POST**

The POST method is used to create new resources. For instance, if the manager of an e-commerce store wanted to add a new product to the database, they would send a POST request to the /products endpoint. Unlike GET requests, POST requests typically include a request body, which is where the client specifies the attributes of the resource to be created. For example, a POST request to the /products endpoint might have a request body that looks like this:

{

"name": "Sneakers",

"color": "blue",

"price": 59.95,

"currency": "USD"

}

**PUT**

The PUT method is used to replace an existing resource with an updated version. This method works by replacing the entire resource (i.e., the specific product located at the /products/123 endpoint) with the data that is included in the request’s body. This means that any fields or properties not included in the request body are deleted, and any new fields or properties are added.

**PATCH**

The PATCH method is used to update an existing resource. It is similar to PUT, except that PATCH enables clients to update specific properties on a resource—without overwriting the others. For instance, if you have a product resource with fields for name, brand, and price, but you only want to update the price, you could use the PATCH method to send a request that only includes the new value for the price field. The rest of the resource would remain unchanged. This behavior makes the PATCH method more flexible and efficient than PUT.

**DELETE**

The DELETE method is used to remove data from a database. When a client sends a DELETE request, it is requesting that the resource at the specified URL be removed. For example, a DELETE request to the /products/123 endpoint will permanently remove the product with an ID of 123 from the database. Some APIs may leverage authorization mechanisms to ensure that only clients with the appropriate permissions are able to delete resources.

Which HTTP methods are safe?

Safe HTTP methods facilitate read-only operations, which means they do not create or alter the API’s resources. GET is the most commonly used safe method, but the HEAD method—which is used to retrieve only the headers of a resource—is also safe.

* **GET Request:**To retrieve or fetch data
* **POST Request:**To create and update data
* **PUT Request;**To update data
* **DELETE Request:**For deleting data

**Authentication Methods in Postman**

## Various types of Authentication Methods in Postman:

**Table of Content**

* [No auth](https://www.geeksforgeeks.org/what-are-authentication-methods-supported-in-postman/" \l "1-no-auth)
* [API auth](https://www.geeksforgeeks.org/what-are-authentication-methods-supported-in-postman/" \l "2-api-auth)
* [Bearer token](https://www.geeksforgeeks.org/what-are-authentication-methods-supported-in-postman/" \l "3-bearer-token)
* [JWT bearer](https://www.geeksforgeeks.org/what-are-authentication-methods-supported-in-postman/" \l "4-jwt-bearer)
* [Basic auth](https://www.geeksforgeeks.org/what-are-authentication-methods-supported-in-postman/" \l "5-basic-auth)

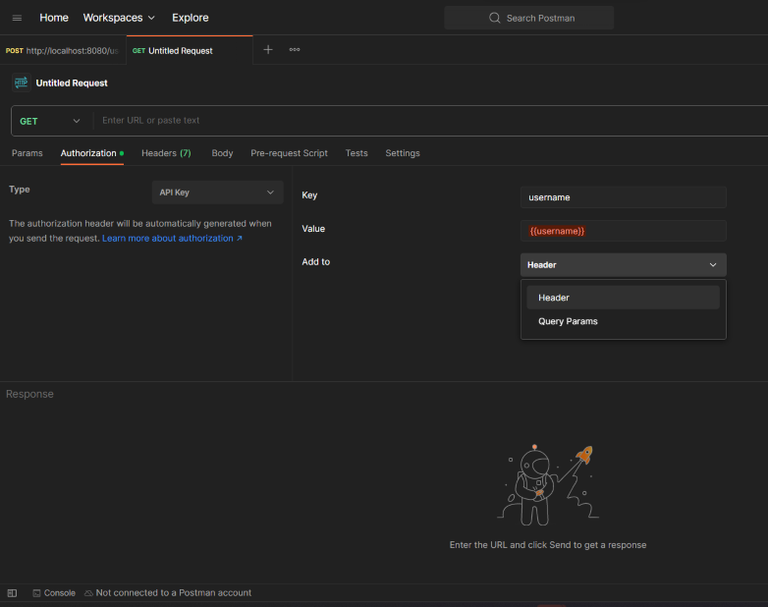
Let’s deep dive and explore top five Authentication type in detail:

## 1. No auth:

Postman will not shoot authorization details with a request unless you define an authtype. By default the No Auth type is selected as authtype in Postman, however if it is not selected you can choose No Auth from the Authorization tab > Type dropdown list, If your request does not need authorization.

## 2. API auth:

With **API key auth**, you ship a key-fee pair to the API both inside the request headers or question parameters. In the request Authorization tab, select API Key from the Type listing. Enter your key name and price, and pick out either Header or Query Params from the Add to dropdown listing. You can keep your values in variables for extra security. Postman will append the applicable facts in your request Headers or the URL query string.



*API Key in Postman*

## 3. Bearer token:

**Bearer tokens** allow requests to authenticate using an get entry to key, along with a [JSON Web Token (JWT)](https://www.geeksforgeeks.org/json-web-token-jwt/" \t "https://www.geeksforgeeks.org/what-are-authentication-methods-supported-in-postman/_blank). The token is a text string, blanketed inside the request header. In the request Authorization tab, choose Bearer Token from the Type dropdown list. In the Token area, enter your API key price. For introduced security, save it in a variable and reference the variable by call.

Postman will append the token cost to the textual content Bearer inside the required format to the request Authorization header as follows:

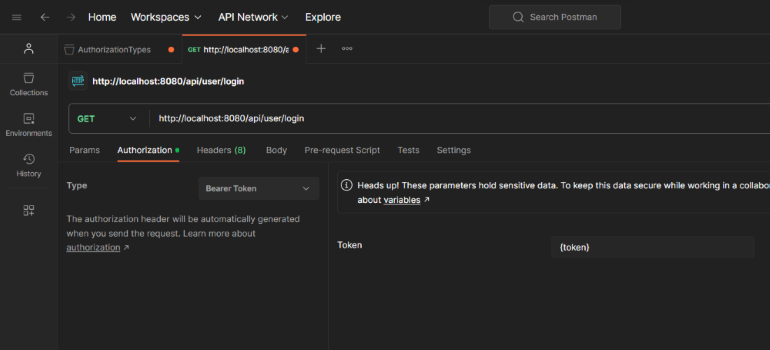
Bearer <Your API key>

### To add a bearer token in Postman, you can do the following:

* Create a variable,
* Call the Login API,
* Set variables,
* Read variables when calling the API.

### To pass a bearer token in a header, you can do the following:

1. Make an HTTP GET or POST request,
2. Send your bearer token with the Authorization: Bearer {token} HTTP header.



*Bearer Token*

## 4. JWT bearer:

A JWT Bearer grant type is used when the client wants to receive access tokens without transmitting sensitive information. It can also be used with trusted clients to gain access to user resources without user authorization.

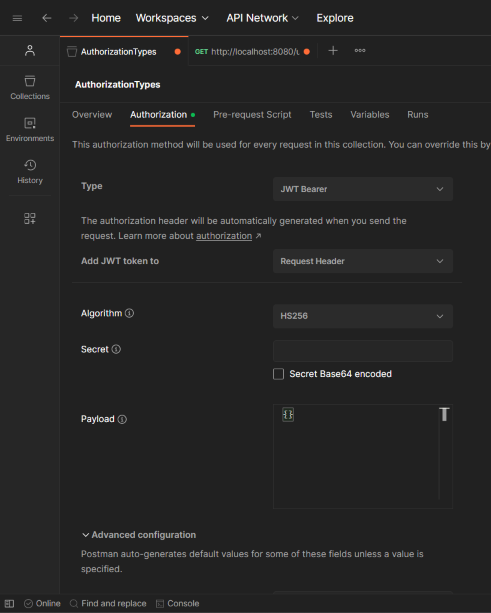
### In the request Authorization tab, select JWT Bearer from the Type dropdown list.

Algorithm – Select an algorithm to use for the JWT token. Supported algorithms consist of:

* **HS** – HMAC with SHA
* **RS** – RSA (RSASSA-PKCS1-v1\_5) with SHA
* **ES** – ECDSA with SHA
* **PS** – RSA (RSASSA-PSS) with SHA
* **Secret** – The secret that’s used with the HMAC-SHA set of rules.
* **Secret Base64 encoded** – If the key is encoded in the base-sixty four format.
* **Private key** – The non-public key for signing the token for RS, ES, and PS algorithms. Select Select file to add a personal key in PKCS #8 format.
* **Payload** – Enter the payload records in your JWT token, in JSON layout.

In the Advanced configuration segment, you can additionally configure the following gadgets. If you don’t configure them, they are generated routinely.

* **Header prefix** – An optional prefix to apply at the start of headers. This header prefix is a part of the request and not part of JWT.
* **Headers** – Any custom headers you furthermore mght need to send in the JWT token. Headers concerning the chosen algorithm are routinely delivered.



*JWT Bearer*

## 5. Basic auth:

Basic Auth is the most basic type of authentication in Postman. It requires a username and password to access the API.

### In the request Authorization tab, select Basic Auth from the Type dropdown list.

* Enter your API username and password in the Username and Password fields.
* For extra security, store these in variables.

