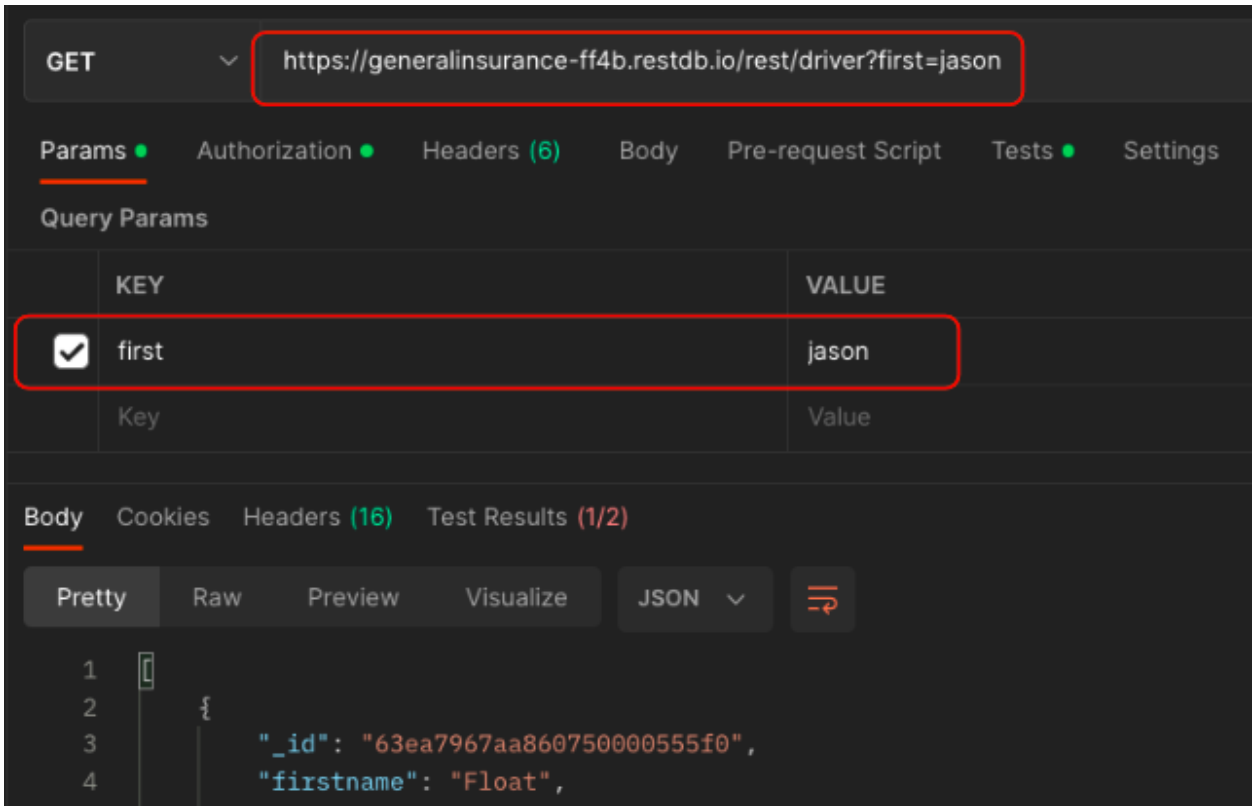
Understanding API Parameter?

A parameter is a value that you pass in to an API request that defines the specifics of the request. This could include criteria for filtering the data returned, authentication credentials, and other details that are specific to a particular API. By passing parameters in API requests, you can control what data you receive from the API and how it is presented. See below I am using parameters to search all records with first name Jason.



When making a request to an API, a request must be made with specific parameters to retrieve the desired data. A parameter is a value that you pass in to an API request that defines the specifics of the request. This could include criteria for filtering the data returned, authentication credentials, and other details that are specific to a particular API.

There are two types of parameters in API requests: query parameters and request body parameters.

Query parameters, also known as URL parameters or GET parameters, are a common way to pass data from a client to a server in an API request. They are key to many API operations and play an important role in API testing.

Query parameters are added to the end of an API endpoint URL and are used to filter, sort, or otherwise manipulate the data returned by the API. For example, if you are making a request to retrieve information about a particular product, you could use a query parameter to specify the ID of the product you are interested in.
For example, if the API endpoint is "https://api.example.com/products", you could add a query parameter to specify the ID of the product you are interested in like this: "https://api.example.com/products?id=123".

Request body parameters, on the other hand, are passed in the body of an API request, typically in JSON format. They are used to send data to the API that the API needs to process the request, such as authentication credentials or information required to create a new resource.
For example, if you were creating a new product using the API, you would send the data for the new product in the request body, along with any other required parameters.

Some other common parameters that are frequently used in API requests include:

- API key: A unique identifier that is used to authenticate a client and provide access to the API.
- Limit and offset: Used to control the number of results returned by an API, with limit specifying the maximum number of results to return and offset specifying the starting point of the results.
- Sort order: Used to specify the order in which results are returned, such as ascending or descending.
- Fields: Used to specify which fields should be included in the response, reducing the amount of data returned and potentially speeding up the response time.

It's important to note that each API will have its own specific parameters and requirements, so it's always a good idea to consult the API documentation before making a request. Additionally, some APIs may have required parameters that must be included in every request, while others may have optional parameters that can be omitted if not needed.

Understanding the role of query parameters in API requests

Query parameters are used to provide additional information to the server and to customize the API response. They can be used to filter or sort data, to perform authentication and authorization, and to perform data validation. In API testing, it is important to understand the role of query parameters in each API request in order to test the API correctly.

Verifying required query parameters

Required query parameters are those that are necessary for the API request to be processed successfully. The server expects to receive these parameters in the request and if they are not provided, the request will fail. In API testing, it is important to verify that required query parameters are provided in the request and that the API returns an error if they are missing. This can be done by sending a request without the required parameters and verifying that the API returns an error.

Testing optional query parameters

Optional query parameters are those that are not required but may provide additional information to the server. In API testing, it is important to verify that the API can handle requests with and without optional query parameters. This can be done by sending requests with and without the optional parameters and verifying that the API returns the expected results. It is also important to verify that the API returns the expected results when optional query parameters are provided.

Testing default query parameters

Default query parameters are those that have a default value set by the API. In API testing, it is important to verify that the API uses the default value when the parameter is not provided in the request. This can be done by sending a request without the parameter and verifying that the API returns the expected results using the default value. It is also important to verify that the API uses the provided value when the parameter is included in the request. This can be done by sending a request with the parameter and verifying that the API returns the expected results using the provided value.

Verifying fixed query parameters

Fixed query parameters are those that have a fixed value and cannot be changed by the client. In API testing, it is important to verify that the API uses the fixed value for the parameter and that it cannot be changed in the request. This can be done by sending a request with the parameter set to a different value and verifying that the API uses the fixed value, not the provided value.

Testing validation query parameters

Validation query parameters are those that are used to validate the API request. The server uses these parameters to verify that the request is valid and that the client is authorized to access the API. In API testing, it is important to verify that the API returns an error if the validation query parameters are not

provided or are incorrect. This can be done by sending a request with incorrect or missing validation parameters and verifying that the API returns an error.

Verifying filter query parameters

Filter query parameters are used to filter the data returned by the API. They are commonly used to return a subset of data that meets certain criteria. In API testing, it is important to verify that the API returns the expected results when filter query parameters are provided. This can be done by sending requests with different filter parameters and verifying that the API returns the expected filtered data. It is also important to verify that the API returns the full data set when no filter parameters are provided.

Testing sort query parameters

Sort query parameters are used to sort the data returned by the API. They are commonly used to sort the data in ascending or descending order based on certain criteria. In API testing, it is important to verify that the API returns the data in the expected order when sort query parameters are provided. This can be done by sending requests with different sort parameters and verifying that the API returns the data in the expected order. It is also important to verify that the API returns the data in the default order when no sort parameters are provided.

Query parameters play a crucial role in API requests and are essential to many API operations. Understanding the different types of query parameters and their usage in API testing is important for ensuring the API is functioning correctly and delivering the expected results. API testers should verify that required query parameters are provided, that the API can handle optional and default query parameters, that fixed query parameters are used correctly, and that validation, filter, and sort query parameters return the expected results. By thoroughly testing query parameters, API testers can ensure that the API is reliable, secure, and delivering the expected results to clients.

Some additional parameter examples:

Query parameters are key-value pairs that are added to the end of a URL to pass data to a server. The structure of a query parameter is as follows:

http://www.example.com/path?key1=value1&key2=value2&key3=value3

In this example, the query parameters are key1, key2, and key3, and their respective values are value1, value2, and value3.

Here's a simple example of how query parameters can be used in a real-world scenario:
Suppose you want to search for a specific product on an e-commerce website. The URL for the search page might look like this:

https://www.example.com/search

To specify what you want to search for, you can add a query parameter to the URL:
https://www.example.com/search?q=table

In this case, the query parameter q has a value of table, so the server knows that you want to search for the keyword "table".

Suppose you want to search for books on Amazon. The URL for the search page on Amazon might look like this:

https://www.amazon.com/s?field-keywords=

To specify what you want to search for, you can add a query parameter to the URL:
https://www.amazon.com/s?field-keywords=harry+potter

In this case, the query parameter field-keywords has a value of harry+potter, so Amazon knows that you want to search for books related to "harry potter". The "+" sign is used to separate words in the query parameter value, which is known as URL encoding.

Here's Another example:

Suppose you want to search for hotels on a travel website. The URL for the search page might look like this:
https://www.travelsite.com/hotel/search

To specify the criteria for your hotel search, such as the location, check-in date, and number of nights, you can add query parameters to the URL:
https://www.travelsite.com/hotel/search?location=New+York&checkin=2022-12-01&nights=7

In this case, the query parameters are location, checkin, and nights, and their respective values are New+York, 2022-12-01, and 7. The travel website uses this information to display a list of hotels that meet your criteria.

Request Body Parameters

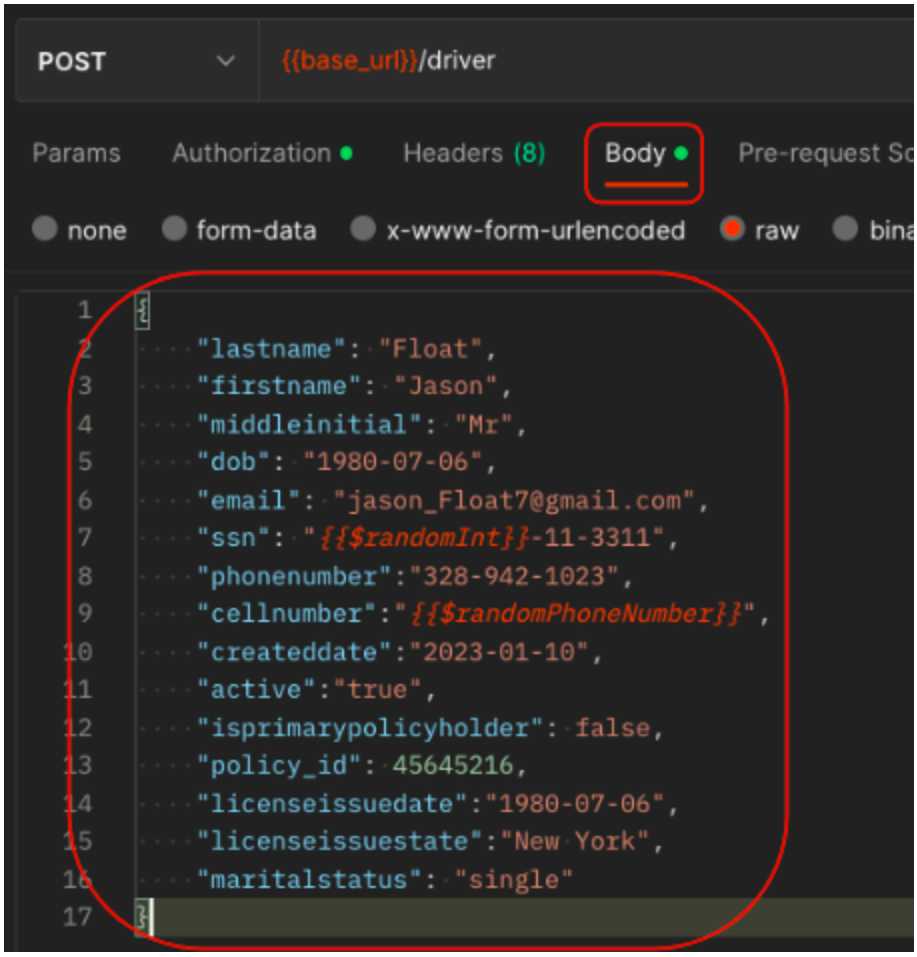
Request Body Parameters are the data that is sent to an API endpoint in the body of an HTTP request. These parameters are used to send additional information to the API endpoint so that it can process the request and return an appropriate response.

In API testing, it is important to validate that the Request Body Parameters are being sent correctly, and that the API endpoint is able to process them as expected. To do this, testers need to have a good understanding of the API specification, and the expected input parameters for each endpoint.

Here are the steps to test Request Body Parameters in an API:

1. Identify the endpoint: Determine the endpoint you want to test, and the type of HTTP request that needs to be sent (e.g. POST, PUT, etc.).
2. Review the API specification: Review the API specification to determine the expected Request Body Parameters for the endpoint, and their data types (e.g. string, integer, etc.).
3. Prepare the request body: Construct the request body using the expected parameters and their values. Ensure that the data being sent is in the correct format, and that it meets any constraints specified in the API specification (e.g. string length, numeric range, etc.).
4. Send the request: Use a tool such as Postman, cURL, or a custom code to send the HTTP request to the API endpoint with the prepared Request Body Parameters.
5. Validate the response: Verify that the response from the API endpoint meets the expected result. This may include checking the status code, the data returned in the response body, or any headers that are sent with the response.
6. Repeat the process: Repeat the above steps for any additional endpoints or combinations of parameters that need to be tested.

Below we are sending a body parameter in the form of json data to create a new driver using Driver Post endpoint.



It's important to note that the process of testing Request Body Parameters will vary depending on the API being tested, and the specific requirements of the test. Testers should always follow the API specification, and use a systematic approach to ensure that all parameters are tested thoroughly.

Validating Request Body Parameters:

One of the most important steps in testing Request Body Parameters is to validate that the parameters are being sent correctly and that the API endpoint can process them as expected. This includes checking the data type, format, and constraints of the parameters.
For example, if the API specification states that a parameter should be an integer between 1 and 100, it is important to test that the API endpoint correctly handles values that fall within this range, as well as values that are outside of it. This helps to ensure that the API endpoint is not vulnerable to data validation attacks, such as injection attacks or buffer overflows.

It is also important to test that the API endpoint can handle missing or incomplete parameters correctly. For example, if a required parameter is not provided in the request body, the API endpoint should return an error response with a descriptive message.

Validation of the response from the API endpoint is also a critical step in the testing process. This includes checking the status code, the data returned in the response body, and any headers that are sent with the response. The response should match the expected result based on the API specification, and any errors should be handled appropriately.

Tools for Testing Request Body Parameters:

There are several tools available for testing Request Body Parameters in APIs, including Postman, cURL, and custom code.

Postman is a popular tool for testing APIs. It allows testers to easily construct and send HTTP requests, and it provides an intuitive interface for viewing and validating the responses from the API endpoint.

cURL is a command-line tool for sending HTTP requests. It is often used by developers and testers to quickly test API endpoints from the command line.

Custom code can also be used to test Request Body Parameters in APIs. This approach provides the most flexibility, as testers can write custom code to test specific scenarios and requirements. This is often used when testing complex APIs, or when testing APIs that are not well documented.