API Documentation: `POST /ntraffic`

Overview

This document outlines the usage of the `/ntraffic` endpoint. This endpoint is used to submit traffic-related data. Clients are required to ensure that the request, including both the header and body, does not exceed 6 KB in total.

Endpoint Details

- **URL**: `http://localhost:8080/ntraffic`
- **Method**: `POST`
- **Content-Type**: `application/json`
- **Request Size Limit**: 6 KB (including both header and body)

Request Structure

Request Headers

The request header must include the following information:

- **Client-ID**: A unique identifier for the client making the request. This is mandatory for each request.

Example:

```
{
  "Client-ID": "your-client-id"
}
```

Request Body

The request body should contain the traffic parameters. [You can define the parameters here or reference another section.]

Example:

```
{
  "value1",
  "value2",
  ...
}
```

Examples

PHP (using cURL)

```
php
Copy code
$clientId = 'your-client-id';
$data = ['key1' => 'value1', 'key2' => 'value2'];

$ch = curl_init('http://localhost:8080/ntraffic');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($data));
curl_setopt($ch, CURLOPT_HTTPHEADER, [
    'Content-Type: application/json',
    'Client-ID: ' . $clientId
]);

$response = curl_exec($ch);
curl_close($ch);

echo $response;
```

Python (using `requests` library)

```
python
Copy code
import requests

url = 'http://localhost:8080/ntraffic'
headers = {'Client-ID': 'your-client-id'}
data = {'key1': 'value1', 'key2': 'value2'}

response = requests.post(url, json=data, headers=headers)
print(response.text)
```

C++ (using `libcurl`)

```
cpp
Copy code
#include <iostream>
#include <curl/curl.h>

int main() {
    CURL *curl;
    CURLcode res;
    const char *url = "http://localhost:8080/ntraffic";
    const char *data = "{\"key1\":\"value1\", \"key2\":\"value2\"}";
    const char *clientId = "your-client-id";

    curl_global_init(CURL_GLOBAL_DEFAULT);
    curl = curl_easy_init();
    if(curl) {
        struct curl_slist *headers = NULL;
        headers = curl_slist_append(headers, "Content-Type: application/json");
        headers = curl_slist_append(headers, std::string("Client-ID: " +
std::string(clientId)).c_str());

        curl_easy_setopt(curl, CURLOPT_URL, url);
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, data);
        curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);
```

```
        res = curl_easy_perform(curl);
        if(res != CURLE_OK)
            fprintf(stderr, "curl_easy_perform() failed: %s\n",
curl_easy_strerror(res));

        curl_easy_cleanup(curl);
    }
    curl_global_cleanup();

    return 0;
}
```

Java (using `HttpURLConnection`)

```
java
Copy code
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class PostRequest {
    public static void main(String[] args) throws Exception {
        String url = "http://localhost:8080/ntraffic";
        String clientId = "your-client-id";
        String jsonInputString = "{\"key1\": \"value1\", \"key2\": \"value2\"}";

        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection) obj.openConnection();
        con.setRequestMethod("POST");
        con.setRequestProperty("Content-Type", "application/json");
        con.setRequestProperty("Client-ID", clientId);

        con.setDoOutput(true);
        try(OutputStream os = con.getOutputStream()) {
            byte[] input = jsonInputString.getBytes("utf-8");
            os.write(input, 0, input.length);
        }

        int responseCode = con.getResponseCode();
        System.out.println("Response Code : " + responseCode);
    }
}
```

Node.js (using `axios`)

```
javascript
Copy code
const axios = require('axios');

const url = 'http://localhost:8080/ntraffic';
const data = { key1: 'value1', key2: 'value2' };
const config = {
  headers: {
    'Client-ID': 'your-client-id',
    'Content-Type': 'application/json'
  }
};

axios.post(url, data, config)
  .then((response) => {
    console.log(response.data);
```

```
  })
  .catch((error) => {
    console.error(error);
  });
```

Perl (using `HTTP::Tiny`)

```
perl
Copy code
use strict;
use warnings;
use HTTP::Tiny;
use JSON;

my $url = 'http://localhost:8080/ntraffic';
my $client_id = 'your-client-id';
my $data = { key1 => 'value1', key2 => 'value2' };

my $http = HTTP::Tiny->new();
my $response = $http->post_form(
    $url,
    encode_json($data),
    {
        headers => {
            'Content-Type' => 'application/json',
            'Client-ID'    => $client_id
        }
    }
);

print $response->{content};
```

Ruby (using `Net::HTTP`)

```
ruby
Copy code
require 'net/http'
require 'uri'
require 'json'

url = URI.parse('http://localhost:8080/ntraffic')
request = Net::HTTP::Post.new(url)
request['Content-Type'] = 'application/json'
request['Client-ID'] = 'your-client-id'
request.body = { key1: 'value1', key2: 'value2' }.to_json

response = Net::HTTP.start(url.hostname, url.port) do |http|
  http.request(request)
end

puts response.body
```

Error Handling

- **400 Bad Request**: This error is returned if the request size exceeds 6 KB or if required parameters are missing.
- **401 Unauthorized**: This error is returned if the `Client-ID` is missing or invalid.
- **500 Internal Server Error**: This error indicates a server-side problem.

Additional Notes

- Ensure that the total size of the request, including headers and body, does not exceed 6 KB. If the request exceeds this limit, the server will return a `400 Bad Request` error.
- For detailed information about traffic parameters, refer to the Traffic Parameters section.

Conclusion

This document provides the essential information required to interact with the `/ntraffic` endpoint. Please refer to the example codes for different programming languages to ensure proper implementation.