# Swarms Telemetry API Documentation

This documentation covers the API for handling telemetry data. The API is implemented using Next.js, Supabase for data storage, and Zod for request validation. The handler processes incoming telemetry data, validates it, and stores it in a Supabase database. The handler also includes robust error handling and retries for database insertions to ensure data reliability.

## Endpoint

- **URL:** `/api/telemetry`
- **Method:** `POST`
- **Content-Type:** `application/json`
- **Description:** Receives telemetry data and stores it in the Supabase database.

## Request Schema

The API expects a JSON object in the request body that matches the following schema, validated using Zod:

Field Name	Type	Required	Description
`data`	`any`	No	Telemetry data payload.
`swarms_api_key`	`string`	No	API key associated with the swarms framework.
`status`	`string`	No	Status of the telemetry data. Default is `'received'`.
`processing_time`	`string`	No	Time taken to process the telemetry data.

## Response

### Success Response

- **Status Code:** `200 OK`
- **Content-Type:** `application/json`
- **Body:**

```json
{
    "message": "Telemetry data received and stored successfully"
}
```

### Error Responses

- **Status Code:** `400 Bad Request`
- **Content-Type:** `application/json`
- **Body:**

```json
{
    "error": "Invalid data format",
    "details": [
        // Zod validation error details
    ]
```

}

- **Status Code:** `405 Method Not Allowed`
- **Content-Type:** `application/json`
- **Body:**

```json
{
    "error": "Method Not Allowed"
}
```

- **Status Code:** `500 Internal Server Error`
- **Content-Type:** `application/json`
- **Body:**

```json
{
    "error": "Internal Server Error",
    "details": "Error message"
}
```

## Example Usage

### Python (Using `requests` Library)

```python
import requests

url = "https://swarms.world/api/telemetry"
headers = {
    "Content-Type": "application/json"
}
data = {
    "data": {"example_key": "example_value"},
    "swarms_api_key": "your_swarms_api_key",
    "status": "received",
    "processing_time": "123ms"
}

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

print(response.status_code)
print(response.json())
```

### Node.js (Using `axios` Library)

```javascript
const axios = require('axios');
const url = 'https://swarms.world/api/telemetry';
const data = {
    data: { example_key: 'example_value' },
    swarms_api_key: 'your_swarms_api_key',
    status: 'received',
    processing_time: '123ms'
};

axios.post(url, data)
    .then(response => {
        console.log(response.status);
        console.log(response.data);
    })
    .catch(error => {
        console.error(error.response.status);
        console.error(error.response.data);
    });
```

### Go (Using `net/http` and `encoding/json`)

```go
package main

import (
    "bytes"

    "encoding/json"

    "fmt"

    "net/http"

)


func main() {

    url := "https://swarms.world/api/telemetry"

    data := map[string]interface{}{

        "data":             map[string]interface{}{"example_key": "example_value"},

        "swarms_api_key":  "your_swarms_api_key",

        "status":          "received",

        "processing_time": "123ms",

    }

    jsonData, err := json.Marshal(data)

    if err != nil {

        fmt.Println("Error marshaling JSON:", err)

        return

    }


    req, err := http.NewRequest("POST", url, bytes.NewBuffer(jsonData))

    if err != nil {

        fmt.Println("Error creating request:", err)

        return

    }
    req.Header.Set("Content-Type", "application/json")
    client := &http.Client{}
    resp, err := client.Do(req)
    if err != nil {
        fmt.Println("Error making request:", err)
        return
    }
    defer resp.Body.Close()

    fmt.Println("Response status:", resp.Status)
}
```

### cURL Command

```bash
curl -X POST https://swarms.world/api/telemetry \
-H "Content-Type: application/json" \
-d '{
    "data": {"example_key": "example_value"},
    "swarms_api_key": "your_swarms_api_key",
    "status": "received",
    "processing_time": "123ms"
}'
```

### Supabase Table Structure

The Supabase table (presumably `swarms_framework_schema`) should have the following columns:

- **`data`**: JSONB or TEXT - Stores the telemetry data payload.
- **`swarms_api_key`**: TEXT - Stores the API key associated with the data.
- **`source_ip`**: TEXT - Stores the IP address of the request source.
- **`status`**: TEXT - Stores the status of the data processing.
- **`processing_time`**: TEXT - Stores the time taken to process the telemetry data.

## References and Further Reading

- [Next.js API Routes Documentation](https://nextjs.org/docs/api-routes/introduction)
- [Supabase JavaScript Client](https://supabase.com/docs/reference/javascript/supabase-client)
- [Zod Schema Validation](https://zod.dev/)
- [OpenAPI Specification](https://swagger.io/specification/)

This documentation is designed to be thorough and provide all the necessary details for developers to effectively use and integrate with the telemetry API.