# Documentation for `getAllAgents` API Endpoint

The `getAllAgents` API endpoint is a part of the `swarms.world` application, designed to fetch all agent records from the database. This endpoint is crucial for retrieving various agents stored in the `swarms_cloud_agents` table, including their metadata such as name, description, use cases, language, requirements and tags. It provides an authenticated way to access this data, ensuring that only authorized users can retrieve the information.

## Purpose

The primary purpose of this API endpoint is to provide a method for clients to fetch a list of agents stored in the `swarms_cloud_agents` table, with the ability to filter by name, tags, language, requirement package and use cases. It ensures data integrity and security by using an authentication guard and handles various HTTP methods and errors gracefully.

## API Endpoint Definition

### Fetch All Agents

#### Endpoint URL

```
https://swarms.world/get-agents
```

#### HTTP Method

```
GET
```

### Request Headers

Header	Type	Required	Description
Authorization	String	Yes	Bearer token for API access

### Query Parameters

- **name** (optional): A substring to match against the agent name. The query is case-insensitive.
- **tag** (optional): A comma-separated list of tags to filter agents by. The query matches any of the provided tags, and is case-insensitive.
- **language** (optional): A substring to match against the language the agent is written in. The query is case-insensitive.
- **use_case** (optional): A substring to match against the use case titles within the `use_cases` array. The query is case-insensitive.
- **req_package** (optional): A substring to match against the requirement packaages within the `requirements` array. The query is case-insensitive.

#### Response

##### Success Response (200)

Returns an array of agents.

```json
[
  {
    "id": "string",
    "name": "string",
    "description": "string",
    "language": "string",
    "agent": "string",
    "use_cases": [
      {
        "title": "string",
        "description": "string"
      }
    ],
    "requirements": [
      {
        "package": "string",
        "installation": "string"
      }
    ],
    "tags": "string"
  },
  ...
```

]

##### Error Responses

- **405 Method Not Allowed**

  ```json
  {
    "error": "Method <method> Not Allowed"
  }
  ```

- **500 Internal Server Error**

  ```json
  {
    "error": "Could not fetch agents"
  }
  ```

### Fetch Agent by ID

#### Endpoint URL

```
```

https://swarms.world/get-agents/[id]

```

#### HTTP Method

```
GET
~~~

Request Headers

Header	Type	Required	Description
Authorization	String	Yes	Bearer token for API access

Response

Success Response (200)

Returns a single agent by ID.

```json
{
  "id": "string",
  "name": "string",
  "description": "string",
~~~

```
  "language": "string",
  "agent": "string",
  "use_cases": [
    {
      "title": "string",
      "description": "string"
    }
  ],
  "requirements": [
    {
      "package": "string",
      "installation": "string"
    }
  ],
  "tags": "string"
}
```

##### Error Responses

- **404 Not Found**

  ```json
  {
    "error": "Agent not found"
  }
  ```

- **500 Internal Server Error**

  ```json
  {
    "error": "Could not fetch agent"
  }
  ```

### Request Handling

1. **Method Validation**: The endpoint only supports the `GET` method. If a different HTTP method is used, it responds with a `405 Method Not Allowed` status.

2. **Database Query**:

   - **Fetching All Agents**: The endpoint uses the `supabaseAdmin` client to query the `swarms_cloud_agents` table. Filters are applied based on the query parameters (`name`, `tag`, `language`, `req_package` and `use_case`).
   - **Fetching an Agent by ID**: The endpoint retrieves a single agent from the `swarms_cloud_agents` table by its unique ID.

3. **Response**: On success, it returns the agent data in JSON format. In case of an error during the database query, a `500 Internal Server Error` status is returned. For fetching by ID, if the agent is not found, it returns a `404 Not Found` status.

### Code Example

#### JavaScript (Node.js)

```javascript
import fetch from "node-fetch";

// Fetch all agents with optional filters
const getAgents = async (filters) => {
  const queryString = new URLSearchParams(filters).toString();
  const response = await fetch(
    `https://swarms.world/get-agents?${queryString}`,
    {
      method: "GET",
      headers: {
        "Content-Type": "application/json",
        Authorization: "Bearer {apiKey}",
      },
    }
  );

  if (!response.ok) {
    throw new Error(`Error: ${response.statusText}`);
  }
  const data = await response.json();

  console.log(data);

};


// Fetch agent by ID

const getAgentById = async (id) => {

  const response = await fetch(`https://swarms.world/get-agents/${id}`, {

    method: "GET",

    headers: {

      "Content-Type": "application/json",

      Authorization: "Bearer {apiKey}",

    },

  });


  if (!response.ok) {

    throw new Error(`Error: ${response.statusText}`);

  }


  const data = await response.json();

  console.log(data);

};


// Example usage

getAgents({

  name: "example",

  tag: "tag1,tag2",
  use_case: "example",
  language: "langauge",
  req_package: "package_name",
}).catch(console.error);
getAgentById("123").catch(console.error);
```

#### Python

```python
import requests

API_KEY = "{apiKey}"

# Fetch all agents with optional filters
def get_agents(filters):
    query_string = "&".join([f"{key}={value}" for key, value in filters.items()])
    url = f"https://swarms.world/get-agents?{query_string}"
    headers = {
        "Content-Type": "application/json",
        "Authorization": f"Bearer {API_KEY}",
    }
    response = requests.get(url, headers=headers)

    if not response.ok:
        raise Exception(f"Error: {response.reason}")
    data = response.json()

    print(data)

    return data


# Fetch agent by ID

def get_agent_by_id(agent_id):

    url = f"https://swarms.world/get-agents/{agent_id}"

    headers = {

        "Content-Type": "application/json",

        "Authorization": f"Bearer {API_KEY}",

    }

    response = requests.get(url, headers=headers)


    if not response.ok:

        raise Exception(f"Error: {response.reason}")


    data = response.json()

    print(data)

    return data


# Example usage

try:

    get_agents({

        "name": "example",

        "tag": "tag1,tag2",
        "use_case": "example",
        "language": "language",
        "req_package": "package_name",
    })
except Exception as e:
    print(e)

try:
    get_agent_by_id("123")
except Exception as e:
    print(e)
```

#### cURL

```sh
# Fetch all agents with optional filters
curl -X GET "https://swarms.world/get-agents?name=example&tag=tag1,tag2&use_case=example&language=language&req_package=package_name" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer {apiKey}"

# Fetch agent by ID
curl -X GET "https://swarms.world/get-agents/123" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer {apiKey}"
```

Go

```go
package main

import (
 "encoding/json"
 "fmt"
 "net/http"
 "net/url"
 "os"
)

func getAgents(filters map[string]string) error {
 query := url.Values{}
 for key, value := range filters {
  query.Set(key, value)
 }

 url := fmt.Sprintf("https://swarms.world/get-agents?%s", query.Encode())
 req, err := http.NewRequest("GET", url, nil)
 if err != nil {
  return err
 }


 req.Header.Set("Content-Type", "application/json")

 req.Header.Set("Authorization", "Bearer {apiKey}")


 client := &http.Client{}

 resp, err := client.Do(req)

 if err != nil {

  return err

 }

 defer resp.Body.Close()


 if resp.StatusCode != http.StatusOK {

  return fmt.Errorf("error: %s", resp.Status)

 }


 var data interface{}

 if err := json.NewDecoder(resp.Body).Decode(&data); err != nil {

  return err

 }


 fmt.Println(data)

 return nil

}


func getAgentById(id string) error {
```

```
url := fmt.Sprintf("https://swarms.world/get-agents/%s", id)
req, err := http.NewRequest("GET", url, nil)
if err != nil {
 return err
}

req.Header.Set("Content-Type", "application/json")
req.Header.Set("Authorization", "Bearer {apiKey}")

client := &http.Client{}
resp, err := client.Do(req)
if err != nil {
 return err
}
defer resp.Body.Close()

if resp.StatusCode != http.StatusOK {
 return fmt.Errorf("error: %s", resp.Status)
}

var data interface{}
if err := json.NewDecoder(resp.Body).Decode(&data); err != nil {
 return err
}

fmt.Println(data)
```

```
 return nil
}
func main() {
 filters := map[string]string{
  "name":        "example",
  "tag":         "tag1,tag2",
  "use_case":    "example",
  "language":    "language",
  "req_package": "package_name",
 }

    getAgents(filters)
    getAgentById("123")
}
```

#### Attributes Table

Attribute	Type	Description
id	String	Unique identifier for the agent
name	String	Name of the agent
description	String	Description of the agent
agent	String	The actual agent
lanuage	String	The code language of the agent
use_cases	Array	Use cases for the agent

requirements	Array	Requirements for the agent
tags	String	Tags associated with the agent

## Additional Information and Tips

- Handle different error statuses appropriately to provide clear feedback to users.
- Consider implementing rate limiting and logging for better security and monitoring.

## References and Resources

- [Next.js API Routes](https://nextjs.org/docs/api-routes/introduction)
- [Supabase Documentation](https://supabase.com/docs)
- [Node Fetch](https://www.npmjs.com/package/node-fetch)
- [Requests Library (Python)](https://docs.python-requests.org/en/latest/)
- [Go net/http Package](https://pkg.go.dev/net/http)

This documentation provides a comprehensive guide to the `getAllAgents` API endpoint, including usage examples in multiple programming languages and detailed attribute descriptions.