Sphere API Usage Guide

The API is structured to facilitate retrieval of multiple pages of data. The typical workflow involves initiating the query asynchronously, monitoring the query status until completion, and then iterating through the paged data until all results have been fetched. This process is illustrated in the `getDataAsync` method of the provided script.

It's also possible to start the query using a synchronous API endpoint (`blocklogs/query/start`). This synchronous method will initiate the query, wait for its completion, and return the first 999 rows of data. If more results are available from the query, a `next_token` will be included in the response.

The API swagger is available in the Sphere docs website. A sample script showing how to use the API is available in Sphere github repo.

Algorithm for Asynchronous API Usage

1. **INITIATE**: Initiate the query by calling the `blocklogs/query/startasync` endpoint. Provide the start time and end time in epoch milliseconds as part of the request. The API will respond with a unique query execution ID.
2. **WAIT**: Periodically check the query status by making requests to `blocklogs/query/{QUERY_EXECUTION_ID}/status`, continuing until a success status is returned or a predefined timeout limit is reached.
3. **FETCH**: Upon successful query completion, retrieve the data in pages by making requests to `blocklogs/query/{QUERY_EXECUTION_ID}/data`. Continue this process until all results have been returned, ensuring to include the `next_token` in the request body when it is available.

Python Code for the Algorithm

The following code is copied from a sample script in Sphere github repo. Refer to the README file for more information on how to use the script.

```
def getDataAsync(api, timestamp_start, timestamp_end):

    # 1. INITIATE: start the query
    query_result = api.query_start_async(timestamp_start, timestamp_end)
    query_execution_id = query_result['query_execution_id']

    # Wait for 5 seconds for the query to complete
    time.sleep(5)

    # 2. WAIT: Get execution status
    for i in range(1, 11):
        # get query execution
        query_status = api.query_status(
            query_execution_id=query_execution_id)
        query_execution_status = query_status['status']
```

```
        if query_execution_status == 'SUCCEEDED':
            break
        elif query_execution_status == 'FAILED':
            raise Exception(f"STATUS: {query_execution_status}")
        else:
            time.sleep(i)
    else:
        api.query_stop(
            query_execution_id=query_execution_id)
        raise Exception('TIME OVER')

    count = 0
    all_data = []
    next_token = None
    # 3. FETCH: Get the data
    while True:
        result = api.query_data(
            query_execution_id=query_execution_id,
            next_token=next_token)

        data = result['data']
        count = count + len(data)
        all_data += data
        next_token = result['next_token'] \
            if 'next_token' in result else None

        # If no more data exit the loop
        if not next_token:
            break

    return (all_data, query_execution_id, count)
```

Details of the Sphere API

Asynchronous Query Start

The asynchronous query start API, identified as `blocklogs/query/startasync`, initiates a `SELECT` query on the database with the conditions `start >= start_epoch_milliseconds AND end < end_epoch_milliseconds` and returns immediately with a query execution ID.

API Endpoint:

`POST https://api.getsphere.ai/blocklogs/query/startasync`

Example Request Payload:

```
{
    "start_epoch_milliseconds": 1690106400000,
```

```
    "end_epoch_milliseconds": 1690135200000
}
```

Example Response Payload:

```
{
    "query_execution_id": "2b71b179-bc41-4c16-946b-1a8b5d81f36e",
}
```

Synchronous Query Start

The synchronous query start API, `blocklogs/query/start`, initiates a similar `SELECT` query and waits for the query to finish. The response includes the first page of data, query execution ID, and a next token if more data is available.

API Endpoint:

```
POST https://api.getsphere.ai/blocklogs/query/start
```

Example Request Payload:

```
{
    "start_epoch_milliseconds": 1690106400000,
    "end_epoch_milliseconds": 1690135200000
}
```

Example Response Payload:

```
{
    "data": [
        [ROW OF DATA],
        [ROW OF DATA]
    ],
    "query_execution_id": "2b71b179-bc41-4c16-946b-1a8b5d81f36e",
}
```

Query Status

The query status API, `blocklogs/query/{QUERY_EXECUTION_ID}/status`, provides the status of a specific query, which can be `SUCCEEDED`, `FAILED`, or another status.

API Endpoint:

```
GET https://api.getsphere.ai/blocklogs/query/{QUERY_EXECUTION_ID}/status
```

Example Response Payload:

```
{
    "status": "SUCCEEDED",
}
```

Stop Query

The stop query API `blocklogs/query/{QUERY_EXECUTION_ID}/stop`, halts the execution of a query if it's still running.

API Endpoint:

`POST https://api.getsphere.ai/blocklogs/query/{QUERY_EXECUTION_ID}/stop`

Query Data API

The query data API, `blocklogs/query/{QUERY_EXECUTION_ID}/data`, retrieves data associated with a given query execution ID and an optional next token.

API Endpoint

`GET https://api.getsphere.ai/blocklogs/query/{QUERY_EXECUTION_ID}/data`

Example Request Payload:

```
{
    "next_token": "aslkjsfh4y02="
}
```

Example Response Payload:

```
{
    "data": [
        [ROW OF DATA],
        [ROW OF DATA]
    ],
    "query_execution_id": "2b71b179-bc41-4c16-946b-1a8b5d81f36e",
    "next_token": "jsjdvd239dn=",
}
```

Sphere API Data Schema

Item	Description	SQL Datatype	Null Value
id	Unique ID.	VARCHAR(255)	Not Applicable
epoc_in_usec	UTC millisecond date which transaction happened.	INT8	Not Applicable
ad_position	The position where the advertisement was placed. (e.g., VIDEO, BANNER, HEADER, etc)	VARCHAR(255)	
app_name	The name of the application or site where the advertisement was placed.	VARCHAR(255)	Empty String
domain	The domain name if it's a website.	VARCHAR(255)	
user_id	The User ID.	VARCHAR(255)	
platform_device_make	The device make.	VARCHAR(255)	
platform_device_type	The device type.	VARCHAR(255)	
platform_os	The device operating system.	VARCHAR(255)	
platform_os_version	The device operating system.	VARCHAR(255)	
platform_browser	The browser type.	VARCHAR(255)	-1
platform_browser_version	The browser version.	VARCHAR(255)	-1
geo_region	The state or territory of the user.	VARCHAR(255)	
ip_address	The IP address of the user.	VARCHAR(255)	
ad_size_bytes	The average ad size in bytes calculated independently and constent for a given ad type.	VARCHAR(255)	-1