Bumba Exchange GraphQL API - Complete Documentation

🎉 Successfully Discovered Schema

This document contains the complete working schema for the Bumba Exchange GraphQL API.

📚 Table of Contents

1. API Endpoints
2. Authentication
3. Query Reference
4. Example Queries
5. Field Descriptions
6. Common Patterns

🔗 API Endpoints

Base URL: `https://exchange-api.bumba.global/graphql`

Method: `POST`

Headers:

```
Content-Type: application/json
Authorization: Bearer YOUR_JWT_TOKEN
```

🔐 Authentication

Authentication uses AWS Cognito JWT tokens. Include the token in the Authorization header:

```bash
Authorization: Bearer eyJraWQiOiJ...
```

📖 Query Reference

1. currencies - Get Available Currencies

Fields:

- `__typename` (String) - Type name
- `name` (String) - Currency name
- `type` (String) - Currency type

Example:

```graphql
{
  currencies {
    __typename
    name
    type
  }
}
```

---

2. accounts - Get User Accounts

Fields:

- `__typename` (String) - Type name
- `user_id` (String/ID) - User identifier
- `username` (String) - Account username
- `email` (String) - Account email
- `balance` (Number) - Account balance
- `created_at` (String/DateTime) - Account creation timestamp
- `updated_at` (String/DateTime) - Last update timestamp

Example:

```graphql
```

```
{
 accounts {
  __typename
  user_id
  username
  email
  balance
  created_at
  updated_at
 }
}
```

---

3. user - Get Current User Information

Fields:

- `__typename` (String) - Type name
- `user_id` (String/ID) - User identifier
- `username` (String) - Username
- `email` (String) - Email address
- `created_at` (String/DateTime) - Account creation timestamp
- `updated_at` (String/DateTime) - Last update timestamp

Example:

graphql

```
{
 user {
  __typename
  user_id
  username
  email
  created_at
  updated_at
 }
}
```

---

4. trades - Get Trade History

Fields:

- `__typename` (String) - Type name
- `user_id` (String/ID) - User identifier
- `price` (Number) - Trade price
- `quantity` (Number) - Trade quantity
- `created_at` (String/DateTime) - Trade timestamp
- `side` (String) - Trade side (BUY/SELL)
- `order_id` (String/ID) - Associated order ID

Example:

```graphql
{
  trades {
    __typename
    user_id
    price
    quantity
    created_at
    side
    order_id
  }
}
```

---

5. open_orders - Get Active Orders

Fields:

- `__typename` (String) - Type name
- `user_id` (String/ID) - User identifier
- `price` (Number) - Order price
- `quantity` (Number) - Order quantity
- `total` (Number) - Total order value
- `created_at` (String/DateTime) - Order creation timestamp

- `updated_at` (String/DateTime) - Last update timestamp
- `status` (String) - Order status (OPEN, PARTIALLY_FILLED, etc.)
- `type` (String) - Order type (LIMIT, MARKET, etc.)
- `side` (String) - Order side (BUY/SELL)
- `order_id` (String/ID) - Order identifier

Example:

```graphql
{
  open_orders {
    __typename
    user_id
    order_id
    price
    quantity
    total
    status
    type
    side
    created_at
    updated_at
  }
}
```

---

6. closed_orders - Get Order History

Fields:

- `__typename` (String) - Type name
- `user_id` (String/ID) - User identifier
- `price` (Number) - Order price
- `quantity` (Number) - Order quantity
- `total` (Number) - Total order value
- `created_at` (String/DateTime) - Order creation timestamp
- `updated_at` (String/DateTime) - Last update timestamp

- `status` (String) - Order status (FILLED, CANCELLED, etc.)
- `type` (String) - Order type (LIMIT, MARKET, etc.)
- `side` (String) - Order side (BUY/SELL)
- `order_id` (String/ID) - Order identifier

Example:

```graphql
{
  closed_orders {
    __typename
    user_id
    order_id
    price
    quantity
    total
    status
    type
    side
    created_at
    updated_at
  }
}
```

💡 Field Descriptions

Common Fields Across Types

- **user_id**: Unique identifier for the user
- **created_at**: ISO 8601 timestamp of when the record was created
- **updated_at**: ISO 8601 timestamp of when the record was last modified
- **price**: Numeric value representing price (likely in quote currency)
- **quantity**: Numeric value representing amount (likely in base currency)
- **total**: Calculated as price × quantity
- **side**: Trading direction - typically "BUY" or "SELL"
- **status**: Current state of the order/trade

- **type**: Order type - typically "LIMIT", "MARKET", "STOP", etc.
- **order_id**: Unique identifier for orders

🔄 Common Patterns

Query Multiple Endpoints

```graphql
{
  user {
    user_id
    username
    email
  }

  open_orders {
    order_id
    price
    quantity
    status
  }

  trades {
    price
    quantity
    side
    created_at
  }
}
```

Get Dashboard Overview

```graphql
```

```
{
  user {
    username
    email
  }

  accounts {
    balance
    username
  }

  open_orders {
    order_id
    price
    quantity
    side
    status
  }
}
```

🚀 Usage Examples

Using cURL

bash

```
curl -X POST https://exchange-api.bumba.global/graphql \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -d '{
    "query": "{ user { username email } }"
  }'
```

Using Python

python

```
import requests

url = "https://exchange-api.bumba.global/graphql"
headers = {
    "Content-Type": "application/json",
    "Authorization": "Bearer YOUR_TOKEN"
}

query = """
{
  user {
    username
    email
  }
  open_orders {
    order_id
    price
    quantity
    status
  }
}
"""

response = requests.post(url, json={"query": query}, headers=headers)
print(response.json())
```

Using JavaScript/Node.js

javascript

```
const fetch = require('node-fetch');

const url = 'https://exchange-api.bumba.global/graphql';
const query = `
  {
    user {
      username
      email
    }
    open_orders {
      order_id
      price
      quantity
      status
    }
  }
`;

fetch(url, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer YOUR_TOKEN'
  },
  body: JSON.stringify({ query })
})
.then(res => res.json())
.then(data => console.log(data));
```

⚠️ Important Notes

1. **Rate Limiting**: Be respectful with API calls. Implement appropriate delays between requests.
2. **Token Expiry**: JWT tokens expire. Monitor the `exp` field in your token and refresh as needed.
3. **Field Types**: Some numeric fields may return as strings. Parse accordingly.
4. **Pagination**: These queries may return all results. Check if the API supports pagination parameters.
5. **Mutations**: This documentation only covers queries. Mutations (create, update, delete) were not discovered but may exist.

🔍 What's Not Available

The following queries were tested but **do NOT exist** in this API:

- `me`, `viewer`, `currentUser`, `profile`
- `balances`, `wallet`, `wallets` (use `accounts` instead)
- `markets`, `market`
- `vaults` (query exists but no fields could be discovered)
- `users` (query exists but no fields could be discovered)

📝 Notes

- The API uses **snake_case** for field names (e.g., `user_id`, `created_at`)
- Some queries use **snake_case** for query names (e.g., `open_orders`, `closed_orders`)
- Introspection is **disabled** on this API for security reasons
- This schema was discovered through systematic field testing

---

🤝 Contributing

If you discover additional fields or queries, please document them here!

Last Updated: November 4, 2025 **API Version:** Unknown (introspection disabled)