Partner Portal GraphQL API

This document provides information on how to use the GraphQL API for the Partner Portal module.

Available Queries

The module provides two main GraphQL queries:

- `partner`: Retrieve a single partner by ID or slug
- `partners`: Retrieve a list of partners with filtering, sorting, and pagination options

Querying a Single Partner

You can query a single partner by ID or slug using the `partner` query:

```
{
  partner(id: 1) {
    partner_id
    name
    slug
    logo
    description
    website
    contact_email
    is_active
  }
}
```

Or by slug:

```
{
  partner(slug: "partner-name") {
    partner_id
    name
    slug
    logo
    description
    website
    contact_email
    is_active
  }
}
```

Note: You must provide either `id` or `slug` parameter, but not both. If both are provided, the `id` parameter takes precedence.

Querying a List of Partners

You can query a list of partners using the `partners` query:

```
{
  partners(
    pageSize: 10
    currentPage: 1
```

```
  ) {
    items {
      partner_id
      name
      slug
      logo
      description
      website
      contact_email
      is_active
    }
    total_count
    page_info {
      page_size
      current_page
      total_pages
    }
  }
}
```

Filtering Partners

You can filter partners by various attributes:

```
{
  partners(
    filter: {
      name: { like: "%company%" }
      is_active: { eq: true }
    }
  ) {
    items {
      partner_id
      name
      slug
    }
    total_count
  }
}
```

Available Filter Fields

The following fields can be used for filtering:

- `name`: Partner name
- `slug`: Partner URL key
- `website`: Partner website URL
- `is_active`: Partner active status

Available Filter Conditions

- `eq`: Equals
- `neq`: Not equals
- `like`: Contains (uses SQL LIKE)
- `gt`: Greater than
- `lt`: Less than
- `gteq`: Greater than or equals
- `lteq`: Less than or equals

- `in`: In an array of values
- `nin`: Not in an array of values

OR Filtering

You can also perform OR filtering:

```
{
  partners(
    filter: {
      or: {
        name: { like: "%company%" }
        website: { like: "%example.com%" }
      }
    }
  ) {
    items {
      partner_id
      name
      website
    }
    total_count
  }
}
```

Sorting Partners

You can sort partners by name or ID:

```
{
  partners(
    sort: {
      name: ASC
    }
  ) {
    items {
      partner_id
      name
    }
    total_count
  }
}
```

Available Sort Fields

- `name`: Sort by partner name
- `partner_id`: Sort by partner ID

Available Sort Directions

- `ASC`: Ascending order
- `DESC`: Descending order

Pagination

You can paginate the results using `pageSize` and `currentPage`:

```
{
  partners(
    pageSize: 5
    currentPage: 2
  ) {
    items {
      partner_id
      name
    }
    total_count
    page_info {
      page_size
      current_page
      total_pages
    }
  }
}
```

Pagination Parameters

- `pageSize`: Number of items per page (default: 20)
- `currentPage`: Current page number (default: 1)

Response Structure

Partner Query Response

The `partner` query returns a single Partner object with the following fields:

- `partner_id`: The unique identifier for the partner
- `name`: The partner's name
- `slug`: The partner's URL key
- `logo`: The path to the partner's logo image
- `description`: The partner's description
- `website`: The partner's website URL
- `contact_email`: The partner's contact email address
- `is_active`: Whether the partner is active (boolean)

Partners Query Response

The `partners` query returns a Partners object with the following structure:

- `items`: An array of Partner objects
- `total_count`: The total number of partners matching the criteria
- `page_info`: Pagination information
    - `page_size`: Number of items per page
    - `current_page`: Current page number
    - `total_pages`: Total number of pages

Example Response

```
{
  "data": {
    "partners": {
      "items": [
```

```
        {
          "partner_id": 1,
          "name": "Example Partner",
          "slug": "example-partner",
          "logo": "wholesale/partner/example-logo.png",
          "description": "Description of the partner",
          "website": "https://example.com",
          "contact_email": "contact@example.com",
          "is_active": true
        }
      ],
      "total_count": 1,
      "page_info": {
        "page_size": 20,
        "current_page": 1,
        "total_pages": 1
      }
    }
  }
}
```

Security Considerations

Note that only active partners (`is_active = true`) are returned in the GraphQL API responses. This is enforced by the resolvers to ensure that only published partner data is available through the API.

Implementation Details

The GraphQL API is implemented using the following components:

- **Schema**: Defined in `etc/schema.graphqls`
- **Resolvers**:
  - `Wholesale\PartnerPortal\Model\Resolver\Partner`: Handles the `partner` query
  - `Wholesale\PartnerPortal\Model\Resolver\Partners`: Handles the `partners` query

Both resolvers use the `PartnerRepository` to fetch data, ensuring consistent business logic across the module.

Error Handling

The API will return appropriate GraphQL errors in the following cases:

- When requesting a partner that doesn't exist
- When requesting a partner that exists but is inactive
- When providing invalid pagination parameters (e.g., negative page numbers)
- When providing invalid filter or sort parameters

Example error response:

```
{
  "errors": [
    {
      "message": "Partner with id \"999\" does not exist.",
      "locations": [
```

```
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "partner"
      ]
    }
  ],
  "data": {
    "partner": null
  }
}
```

Testing

You can test the GraphQL API using the GraphQL playground in your Magento installation or by using tools like Postman or Insomnia. The endpoint is typically:

```
https://your-magento-url/graphql
```

A simple test script is available in `docs/graphql_test.php` that demonstrates how to query the API programmatically.