BrandShelter

DNS API

BrandShelter

Table of contents

BrandShelter

Introduction

GraphQL API Endpoint

We offer a GraphQL endpoint to interact with the DNS API:
For the production environment:

```
https://secure.brandshelter.com/graphql
```

For the demo (OTE) environment:

```
https://demo.brandshelter.com/graphql
```

A great place to learn about GraphQL is here: Introduction to GraphQL | GraphQL

Authorization

Access Tokens

You can manage your access tokens in the Brandshelter frontend application.
A access token´s "secret" is needed to create a valid JWT which is used to access the DNS API.

https://secure.brandshelter.com/access_tokens

JWT (JSON Web Token)

To get a JWT suitable for access to the GraphQL API, you need a user with a valid, not expired access token.
Using the secret from that token, you can then create and sign a JWT with content:

- `jti` as a "globally unique" identifier, there are no restrictions here, it can be anything.
- `iss` and sub as the user's login name.
  `aud` as the string "BrandShelter".
- `nbf` must be a timestamp in the past. The "nbf" (not before) claim identifies the time before which the JWT is not valid.
- `iat` must be a timestamp in the past. The "iat" (issued at) claim identifies the time at which the JWT was issued.
- `exp` must be a timestamp in the future. The "exp" (expiration time) claim identifies the expiration time on or after which the JWT is invalid.

You then sign that JWT using the access token's "secret" with algorithm "HS512" (HMAC with SHA-512).
Doing so, you must make sure that some JWT headers are set appropriately:

- `kid` must be the user's login name and the access token's name (not the secret!) separated by a forward slash, e.g., `"john.doe@brandshelter.com/ExampleToken"`.
- `alg` must be "HS512".

BrandShelter

Example JWT header:

```
{
  "kid": "john.doe@brandshelter.com/ExampleToken",
   "alg": "HS512"
}
```

Example JWT payload:

```
{
   "iss": "john.doe@brandshelter.com",
   "sub": "john.doe@brandshelter.com",
   "aud": "BrandShelter",
   "nbf": 1680522050,
   "exp": 1872228410,
   "iat": 1680594434,
    "jti": "BrandShelter-7a589414-d004-46b4-a95f-50b763f678d3"
}
```

You can use JWT.IO to create or verify JWTs.

HTTP Bearer Authentication

For authentication you must send the JWT in the Authorization header when making requests to DNS API:

```
Authorization: Bearer <JWT>
```

Introspection & API documentation

The GraphQL introspection system allows to query information about the available schema. There are various tools and clients which are able to generate an API documentation from the live endpoint using a introspection query. You can use for example Altair GraphQL Client to easily generate and browse the API documentation.

API Schema

GraphQL has an introspection feature that allows you to retrieve the documentation, i.e. the API schema, directly from the API itself. However, for a better user experience, we recommend using a GraphQL client of your choice with a documentation browser.

Schema Query

To query the complete API schema something like this can be used.

BrandShelter

```
query IntrospectionQuery {
  __schema {
    queryType {
      name
    }
    mutationType {
      name
    }
    subscriptionType {
      name
    }
    types {
      ...FullType
    }
    directives {
      name
      description
      locations
      args {
        ...InputValue
      }
    }
  }
}

fragment FullType on __Type {
  kind
  name
  description
  fields(includeDeprecated: true) {
    name
    description
    args {
      ...InputValue
    }
    type {
      ...TypeRef
    }
    isDeprecated
    deprecationReason
  }
  inputFields {
    ...InputValue
  }
  interfaces {
    ...TypeRef
  }
  enumValues(includeDeprecated: true) {
    name
    description
```

```
      isDeprecated
      deprecationReason
    }
    possibleTypes {
      ...TypeRef
    }
  }

  fragment InputValue on __InputValue {
    name
    description
    type {
      ...TypeRef
    }
    defaultValue
  }

  fragment TypeRef on __Type {
    kind
    name
    ofType {
      kind
      name
      ofType {
        kind
        name
        ofType {
          kind
          name
          ofType {
            kind
            name
            ofType {
              kind
              name
              ofType {
                kind
                name
                ofType {
                  kind
                  name
                }
              }
            }
          }
        }
      }
    }
  }
```

BrandShelter

Browsing documentation with a GraphQL client

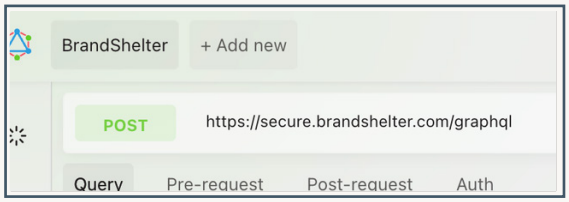
There are many GraphQL clients. In this example we use the Altair GraphQL Client.

1. Download the client at https://altairgraphql.dev/. Ensure to download the desktop version. Altair GraphQL Cloud uses a webservice inbetween that will lead to cross-origin resource sharing (CORS) errors and thus won't work.

2. Install the client

3. Configure your client

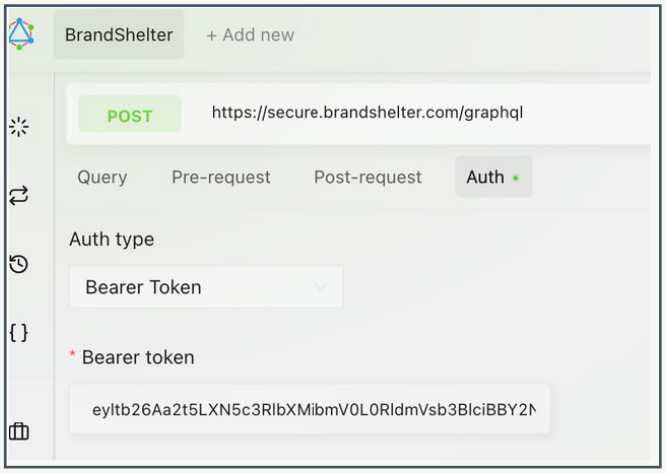
a) Enter the respective API endpoint URL



b) Enter your bearer token

i. Click the "Auth" button

ii. Enter your bearer token into the field



4. Optionally do a quick simple query to check whether everything is working

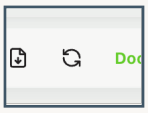
a) Click on the "Query" button

b) Enter this query

```
{
  currentAccount {
    clientNumber
  }
}
```

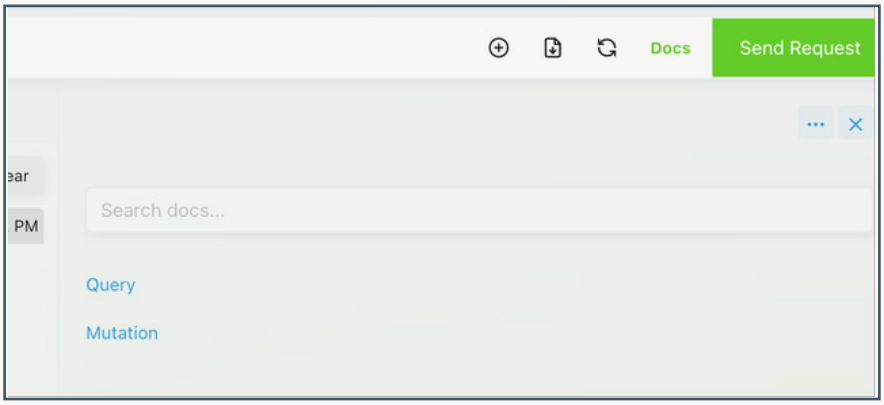
BrandShelter

5. Click the “Reload docs” button

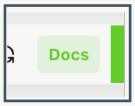


The round icon in the middle

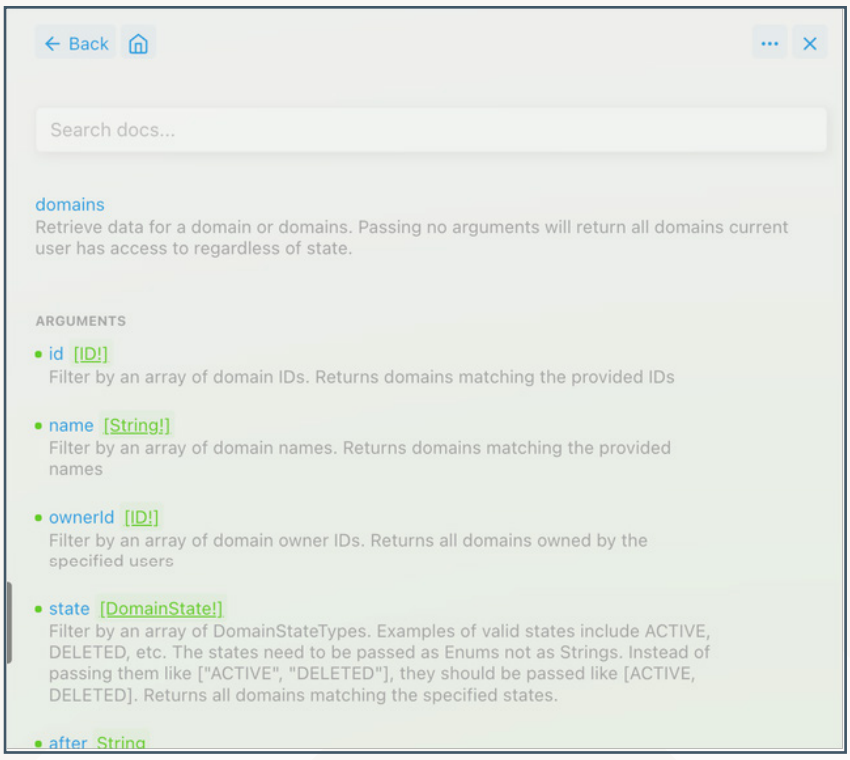
6. You should see the links for queries and mutations in the documentation pane.



If the documentation pane is not shown, click on the “Docs” button.



7. You can now click through links to see details about all queries, mutations and types.



BrandShelter

Examples

Retrieve name and email address of the user which is associated with the bearer token you're using:

```
query CurrentUser {
    currentUser {
        firstname
        lastname
        email
    }
}
```

Retrieve `id` , `name` and `zoneType` of the DNS zone for `test-domain-1001.net` .

```
query DnsZone {
    dnsZone(domainName: "test-domain-1001.net") {
        id
        name
        zoneType
    }
}
```

Retrieve `id` , `name` and `zoneType` of your accessible DNS zones:

```
query DnsZones {
    dnsZones {
        nodes {
            id
            name
            zoneType
        }
    }
}
```

Retrieve data of the resource records of the DNS zone for `test-domain-1001.net` :

```
query DnsRecordsByZone {
    dnsRecordsByZone(domainName: "test-domain-1001.net") {
        bindSyntax
        errors
        fqdn
        host
        id
        locked
        rdata
        ttl
        type
    }
}
```

BrandShelter

Retrieve data for resource records of type `A` and `AAAA` and web forwardings for the DNS zone `test-domain-1002.net`:

```
query DnsRecordsByZone {
    dnsRecordsByZone(domainName: "test-domain-1002.net", filters: { recordTypes: ["X-WEB-FWD", "A", "AAAA"] }) {
        host
        id
        locked
        rdata
        ttl
        type
    }
}
```

Create a new MX resource record for the DNS zone with `id=2` and retrieve `id` and bind representation of the created record:

```
mutation CreateDnsRecord {
  createDnsRecord(
    input: { type: "MX", dnsZoneId: 2, rdata: "6 mail.foo-1.com." }
  ) {
    id
    bindSyntax
  }
}
```

Update the DNS resource record with `id=54` with the provided data and retrieve `id` and bind representation of the updated record:

```
mutation UpdateDnsRecord {
  updateDnsRecord(
    input: { id: 54, type: "MX", dnsZoneId: 2, rdata: "6 mail.foo-1.com." }
  ) {
    id
    bindSyntax
  }
}
```

Delete the DNS resource record with `id=2076`:

```
mutation DeleteDnsRecord {
  deleteDnsRecord(id: 2076) {
    ids
  }
}
```

BrandShelter

Publish the DNS zone `test-domain-1001.net` and retrieve it's name:

```
mutation PublishDnsZone {
    publishDnsZone(domainName: "test-domain-1001.net") {
      name
    }
}
```

Retrieve `name`, `state` and `expiresAt` of the first 10 domains with state active or `deletion_pending`:

```
{
     domains(
       state: [active, deletion_pending],
        first: 10
     ) {
       nodes {
          name
          state
          expiresAt
       }
       pageInfo {
          endCursor
          hasNextPage
          hasPreviousPage
          startCursor
       }
       totalCount
    }
}
```