Accessing the GraphQL API

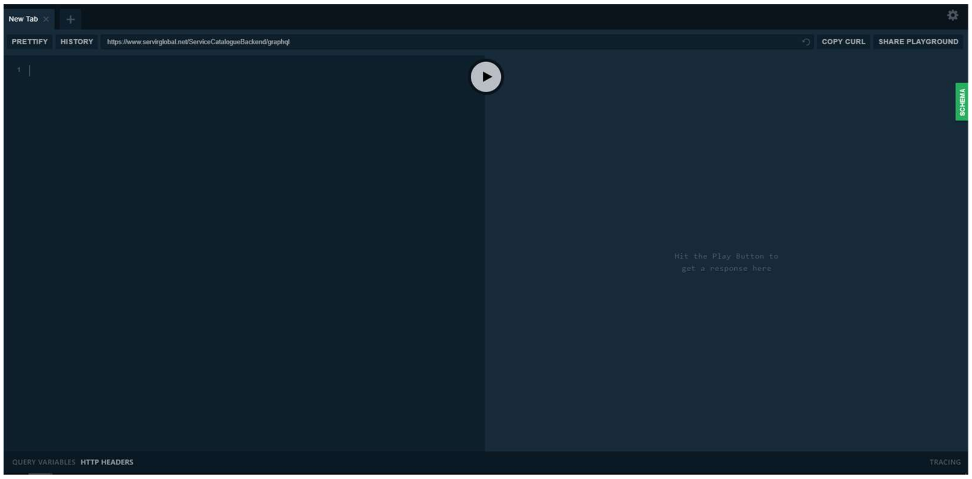
Servir

Table of Contents

The service catalogue can be queried using a GraphQL API. Below documentation describes the API and how you can use it from your apps.

1 GraphQL playground

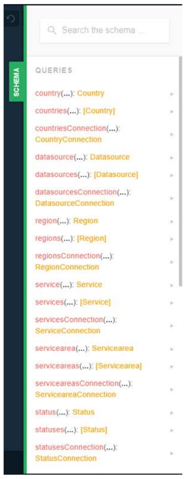
Go to https://www.servirglobal.net/ServiceCatalogueBackend/playground , you will see a friendly UI to test your queries


New Tab
PRETTIFY
HISTORY
https://www.servirglobal.net/ServiceCatalogueBackend/graphql
COPY CURL
SHARE PLAYGROUND
SCHEMA
Hit the Play Button to get a response here
QUERY VARIABLES
HTTP HEADERS
TRACING

Note: make sure you set the URL at the top of the playground tool to https://www.servirglobal.net/ServiceCatalogueBackend/graphql or you will see a "Server cannot be reached" error.

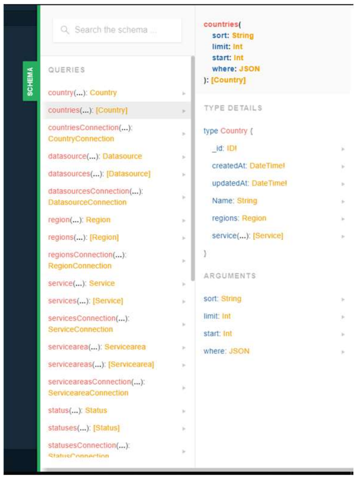
1.1 How to query data using the GraphQL playground

You can see all the queries available in the schema section on the right of GraphQL playground



```
country(...): Country
countries(...): [Country]
countriesConnection(...): CountryConnection
datasource(...): Datasource
datasources(...): [Datasource]
datasourcesConnection(...): DatasourceConnection
region(...): Region
regions(...): [Region]
regionsConnection(...): RegionConnection
service(...): Service
services(...): [Service]
servicesConnection(...): ServiceConnection
servicearea(...): Servicearea
serviceareas(...): [Servicearea]
serviceareasConnection(...): ServiceareaConnection
status(...): Status
statuses(...): [Status]
statusesConnection(...): StatusConnection
```

Clicking a query in the list you will see details of the arguments needed and the details of the type the query will return



```
QUERIES
country(...): Country
countries(...): [Country]
countriesConnection(...):
CountryConnection
datasource(...): Datasource
datasources(...): [Datasource]
datasourcesConnection(...):
DatasourceConnection
region(...): Region
regions(...): [Region]
regionsConnection(...):
RegionConnection
service(...): Service
services(...): [Service]
servicesConnection(...):
ServiceConnection
servicearea(...): Servicearea
serviceareas(...): [Servicearea]
serviceareasConnection(...):
ServiceareaConnection
status(...): Status
statuses(...): [Status]
statusesConnection(...):
StatusConnection
```

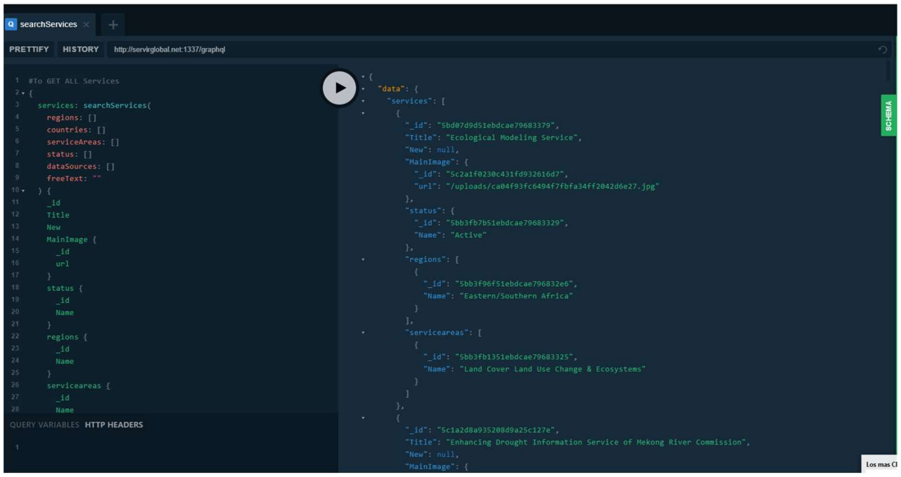
```
countries(
  sort: String
  limit: Int
  start: Int
  where: JSON
): [Country]

TYPE DETAILS

type Country {
  _id: ID!
  createdAt: DateTime!
  updatedAt: DateTime!
  Name: String
  regions: Region
  service(...): [Service]
}

ARGUMENTS

sort: String
limit: Int
start: Int
where: JSON
```

You can use the playground to run queries against the API.

An example of a query used in the service catalogue (you can copy and paste it and click on the "play" icon at the middle of the screen):

```
#To GET ALL Services and returns the id, title, main image, status, the
regions list with the id and the name and the service areas list with the
id and the name.
{
  services: searchServices(
    regions: []
    countries: []
    serviceAreas: []
    status: []
    dataSources: []
    freeText: ""
  ) {
    _id
    Title
    New
    MainImage {
      _id
      url
    }
    status {
      _id
      Name
    }
    regions {
      _id
      Name
    }
    serviceareas {
      _id
      Name
    }
  }
}
```

And you will see the result of your query on the right



```
#To GET ALL Services
{
  services: searchServices(
    regions: []
    countries: []
    serviceAreas: []
    status: []
    dataSources: []
    freeText: ""
  ) {
    _id
    Title
    New
    MainImage {
      _id
      url
    }
    status {
      _id
      Name
    }
    regions {
      _id
      Name
    }
    serviceareas {
      _id
      Name
```

```
{
  "data": {
    "services": [
      {
        "_id": "5bd07d9d51ebdcae79683379",
        "Title": "Ecological Modeling Service",
        "New": null,
        "MainImage": {
          "_id": "5c2a1f0230c431fd932616d7",
          "url": "/uploads/ca04f93fc6494f7fbfa34ff2042d6e27.jpg"
        },
        "status": {
          "_id": "5bb3fb7b51ebdcae79683329",
          "Name": "Active"
        },
        "regions": [
          {
            "_id": "5bb3f96f51ebdcae796832e6",
            "Name": "Eastern/Southern Africa"
          }
        ],
        "serviceareas": [
          {
            "_id": "5bb3fb1351ebdcae79683325",
            "Name": "Land Cover Land Use Change & Ecosystems"
          }
        ]
      },
      {
        "_id": "5c1a2d8a935208d9a25c127e",
        "Title": "Enhancing Drought Information Service of Mekong River Commission",
        "New": null,
        "MainImage": {
```

2 Authenticated API calls

Parts of the API require the client to be authenticated. In order to do queries using GraphQL for queries with policies (security) implemented in the API query response, you will need to do an authentication POST to Strapi to get the authorization token and then use that in your API calls.

2.1 Get an authorization token

This is an example of how to call the API and get the authorization token from an authentication POST:

```
$(function() {

        $.ajax({
          type: 'POST',
          url:
'https://www.servirglobal.net/ServiceCatalogueBackend/auth/local',
          data: {
          identifier: 'username', //Replace with your Strapi username
          password: 'password' //Replace with your Strapi password
          }
        }).done(function(auth) {
         console.log('Well done!');
          console.log('User profile', auth.user);
         console.log('AUTHORIZATION TOKEN', auth.jwt); //THIS IS THE
AUTHORIZATION TOKEN
        }).fail(function(error) {
         console.log('An error occurred:', error);
        });

});
```

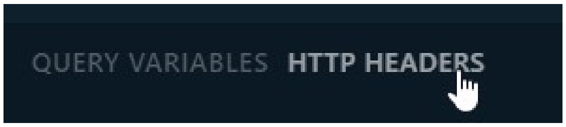
Note: this example call uses jQuery.

Once you get the authorization token you can add it as Authorization header on your API call like this:

Authorization: "Bearer <AuthorizationToken>"

2.2 Using Authorization token in GraphQL playground

To set this authorization token in GraphQL playground you need to click on "HTTP HEADERS" at the bottom of the page



Then you can paste the authorization token in this way:

```
{
  "Authorization": "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1Yjc1YWU1NTI1OWJiYjA2Njg5
OTg1MjAiLCJpYXQiOjE1MzUwNDU0ODMsImV4cCI6MTUzNzYzNzQ4M30.vKoBjY9H3JMEdjVcQ
3YbwwQpYh7DKD3LmIk7Kn0vwfM"
}
```

2.3 Authorization error

Note: If you see an error response like this (with the word **"Forbidden"**):

```
{
  "data": {
    "articles": null
  },
  "errors": [
    {
      "message": "Forbidden",
      "locations": [
        {
          "line": 2,
          "column": 3
        }
      ],
      "path": [
        "articles"
      ]
    }
  ]
}
```

this means that you need to do the query with an authorization token (see the next section for more details).

3 Examples of queries used in service catalogue

3.1 GET ALL Services

Returns the id, title, main image, status, the regions list with the id and the name and the service areas list with the id and the name. (Does not need authorization token)

```
{
  services: searchServices(
    regions: [],
    countries: [],
    serviceAreas: [],
    status: [],
    dataSources: [],
    freeText: ""
  ) {
    _id
    Title
    New
    MainImage {
      _id
      url
    }
    status {
      _id
      Name
    }
    regions {
      _id
      Name
    }
    serviceareas {
      _id
      Name
    }
  }
}
```

3.2 GET All filter options

Gets the filter options ("Narrow by section" items in service catalogue) with ID and Name (Does not need authorization token)

```
 {
  Region: allRegions(limit: 10) {
    _id
    Name
    country {
      _id
      Name
    }
  }
  ServiceArea: allServiceAreas(start: 0, sort: "Name asc") {
    _id
    Name
  }
  Status: allStatus(start: 0, sort: "Name asc") {
    _id
    Name
  }
  DataSource: allDataSources(start: 0, sort: "Name asc") {
    _id
    Name
  }
}
```

3.3 GET Services filtered filtered by region

Gets services filtered by **multiple regions**, service area, status and data sources. **You can copy the Id's needed here from the query that gets all filter options or getting them under each content item in the API admin http://www.servirglobal.net:1337/admin/.** You can add multiple Id's for each content item in the same way as regions in this example. (Does not need authorization token)

```
{
  services: searchServices(
    regions: [
      "5bb3f96951ebdcae796832e5",
      "5bb3f96f51ebdcae796832e6",
      "5bb3f97451ebdcae796832e7"
    ],
    countries: [],
    serviceAreas: ["5bb3fb1351ebdcae79683325"],
    status: ["5bb3fb7b51ebdcae79683329"],
    dataSources: ["5bb3fc4b51ebdcae7968333a"],
    freeText: ""
  ) {
    _id
    Title
    New
    MainImage {
      _id
      url
    }
    status {
      _id
      Name
    }
    regions {
      _id
      Name
    }
    serviceareas {
      _id
      Name
    }
  }
}
```

3.4 Free text search

Gets service list by free text words. (Does not need authorization token)

```
{
  services: searchServices(
    regions: [],
    countries: [],
    serviceAreas: [],
    status: [],
    dataSources: [],
    freeText: "ecological"
  ) {
    _id
    Title
    New
    MainImage {
      _id
      url
    }
    status {
      _id
      Name
    }
    regions {
      _id
      Name
    }
    serviceareas {
      _id
      Name
    }
  }
}
```

3.5 GET Service details

Gets service details by Id. (Does not need authorization token)

```
{
  services: getServiceById(id: "5bd07d9d51ebdcae79683379") {
    _id
    Title
    Description
    Context
    Summary
    PeriodofPerformance
    MainImage {
      url
    }
    status {
      Name
    }
    regions {
      Name
    }
    countries {
      Name
    }
    serviceareas {
      Name
    }
    dataSources {
      Name
    }
    Developer
    Tools
    Data
    UseCases
    TrainingMaterials
    News
    UsersAndStakeholders
  }
}
```

3.6 GET Featured services by Region

Change the Id for regions as needed. (Does not need authorization token)

```
{
  services: searchServices(
    featured: true,
    regions: ["5bb3f96f51ebdcae796832e6"],
    countries: [],
    serviceAreas: [],
    status: [],
    dataSources: [],
    freeText: ""
  ) {
    _id
    Title
    Summary
    MainImage {
      _id
      url
    }
  }
}
```

4 How to do an API call using jquery.

Code example on how to call the API with one of the query examples using jQuery.

```
$.ajax({
          type: 'POST',
          url:
'https://www.servirglobal.net/ServiceCatalogueBackend/graphql',
          data: {
            operationName: null,
            query: " { Region : allRegions(limit:10) { _id  Name  country
{    _id    Name } } ServiceArea: allServiceAreas(start:0, sort: \"Name
asc\") { _id Name }   Status: allStatus(start:0, sort: \"Name asc\") {
_id Name }   DataSource :allDataSources(start:0, sort: \"Name asc\") {
_id Name }}",
            variables: {}
          }
        }).done(function(data) {
          console.log(data);
        }).fail(function(error) {
          console.log('An error occurred:', error);
        });
```