+++ title = "Data source HTTP API " description = "Grafana Data source HTTP API" keywords = ["grafana", "http", "documentation", "api", "data source"] aliases = ["/docs/grafana/latest/http_api/datasource/"] +++

Data source API

> *If you are running Grafana Enterprise and have [Fine-grained access control]({{< relref "../enterprise/access-control/_index.md" >}}) enabled, for some endpoints you would need to have relevant permissions. Refer to specific resources to understand what permissions are required.*

Get all data sources

`GET /api/datasources`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope
datasources:read	datasources:*

Examples

Example Request:

```
GET /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```
HTTP/1.1 200
Content-Type: application/json

[
   {
     "id": 1,
     "orgId": 1,
     "uid": "H8joYFVGz"
     "name": "datasource_elastic",
     "type": "elasticsearch",
     "typeLogoUrl":
"public/app/plugins/datasource/elasticsearch/img/elasticsearch.svg",
     "access": "proxy",
     "url": "http://mydatasource.com",
     "password": "",
     "user": "",
     "database": "grafana-dash",
     "basicAuth": false,
```

```
    "isDefault": false,
    "jsonData": {
        "esVersion": 5,
        "logLevelField": "",
        "logMessageField": "",
        "maxConcurrentShardRequests": 256,
        "timeField": "@timestamp"
    },
    "readOnly": false
  }
]
```

Get a single data source by Id

`GET /api/datasources/:datasourceId`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope
datasources:read	datasources:* datasources:id:* datasources:id:1 (single data source)

Examples

Example Request:

```
GET /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
```

```
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
  "isDefault": false,
  "jsonData": {
    "graphiteType": "default",
    "graphiteVersion": "1.1"
  },
  "secureJsonFields": {},
  "version": 1,
  "readOnly": false
}
```

Get a single data source by UID

`GET /api/datasources/uid/:uid`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope
datasources:read	datasources:* datasources:uid:* datasources:uid:kLtEtcRGk (single data source)

Examples

Example request:

```
GET /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
```

```
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
  "isDefault": false,
  "jsonData": {
    "graphiteType": "default",
    "graphiteVersion": "1.1"
  },
  "secureJsonFields": {},
  "version": 1,
  "readOnly": false
}
```

Get a single data source by Name

`GET /api/datasources/name/:name`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope
datasources:read	datasources:* datasources:name:* datasources:name:test_datasource (single data source)

Examples

Example Request:

```
GET /api/datasources/name/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```
HTTP/1.1 200
Content-Type: application/json

{
  "id": 1,
  "uid": "kLtEtcRGk",
  "orgId": 1,
  "name": "test_datasource",
```

```
  "type": "graphite",
  "typeLogoUrl": "",
  "access": "proxy",
  "url": "http://mydatasource.com",
  "password": "",
  "user": "",
  "database": "",
  "basicAuth": false,
  "basicAuthUser": "",
  "basicAuthPassword": "",
  "withCredentials": false,
  "isDefault": false,
  "jsonData": {
    "graphiteType": "default",
    "graphiteVersion": "1.1"
  },
  "secureJsonFields": {},
  "version": 1,
  "readOnly": false
}
```

Get data source Id by Name

`GET /api/datasources/id/:name`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope
datasources.id:read	datasources:* datasources:name:* datasources:name:test_datasource (single data source)

Examples

Example Request:

```
GET /api/datasources/id/test_datasource HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```
HTTP/1.1 200
Content-Type: application/json

{
```

```
  "id":1
}
```

Create a data source

`POST /api/datasources`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope
datasources:create	n/a

Examples

Example Graphite Request:

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name":"test_datasource",
  "type":"graphite",
  "url":"http://mydatasource.com",
  "access":"proxy",
  "basicAuth":false
}
```

Example Graphite Response:

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": false,
    "basicAuthUser": "",
```

```
    "basicAuthPassword": "",
    "withCredentials": false,
    "isDefault": false,
    "jsonData": {},
    "secureJsonFields": {},
    "version": 1,
    "readOnly": false
  },
  "id": 1,
  "message": "Datasource added",
  "name": "test_datasource"
}
```

> ***Note:*** *By defining* `password` *and* `basicAuthPassword` *under* `secureJsonData` *Grafana encrypts them securely as an encrypted blob in the database. The response then lists the encrypted fields under* `secureJsonFields`.

Example Graphite Request with basic auth enabled:

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "test_datasource",
  "type": "graphite",
  "url": "http://mydatasource.com",
  "access": "proxy",
  "basicAuth": true,
  "basicAuthUser": "basicuser",
  "secureJsonData": {
    "basicAuthPassword": "basicpassword"
  }
}
```

Example Response with basic auth enabled:

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
```

```
    "user": "",
    "database": "",
    "basicAuth": true,
    "basicAuthUser": "basicuser",
    "basicAuthPassword": "",
    "withCredentials": false,
    "isDefault": false,
    "jsonData": {},
    "secureJsonFields": {
      "basicAuthPassword": true
    },
    "version": 1,
    "readOnly": false
  },
  "id": 102,
  "message": "Datasource added",
  "name": "test_datasource"
}
```

Example CloudWatch Request:

```
POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "name": "test_datasource",
  "type": "cloudwatch",
  "url": "http://monitoring.us-west-1.amazonaws.com",
  "access": "proxy",
  "jsonData": {
    "authType": "keys",
    "defaultRegion": "us-west-1"
  },
  "secureJsonData": {
    "accessKey": "Ol4pIDpeKSA6XikgOl4p",
    "secretKey": "dGVzdCBrZXkgYmxlYXNlIGRvbid0IHN0ZWFs"
  }
}
```

Update an existing data source

`PUT /api/datasources/:datasourceId`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope

datasources:write	datasources:* datasources:id:* datasources:id:1 (single data source)

Examples

Example Request:

```
PUT /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk

{
  "id":1,
  "orgId":1,
  "name":"test_datasource",
  "type":"graphite",
  "access":"proxy",
  "url":"http://mydatasource.com",
  "password":"",
  "user":"",
  "database":"",
  "basicAuth":true,
  "basicAuthUser":"basicuser",
  "secureJsonData": {
    "basicAuthPassword": "basicpassword"
  },
  "isDefault":false,
  "jsonData":null
}
```

Example Response:

```
HTTP/1.1 200
Content-Type: application/json

{
  "datasource": {
    "id": 1,
    "orgId": 1,
    "name": "test_datasource",
    "type": "graphite",
    "typeLogoUrl": "",
    "access": "proxy",
    "url": "http://mydatasource.com",
    "password": "",
    "user": "",
    "database": "",
    "basicAuth": true,
    "basicAuthUser": "basicuser",
```

```
    "basicAuthPassword": "",
    "withCredentials": false,
    "isDefault": false,
    "jsonData": {},
    "secureJsonFields": {
      "basicAuthPassword": true
    },
    "version": 1,
    "readOnly": false
  },
  "id": 102,
  "message": "Datasource updated",
  "name": "test_datasource"
}
```

> ***Note:*** *Similar to creating a data source, `password` and `basicAuthPassword` should be defined under `secureJsonData` in order to be stored securely as an encrypted blob in the database. Then, the encrypted fields are listed under `secureJsonFields` section in the response.*

Delete an existing data source by id

`DELETE /api/datasources/:datasourceId`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope
datasources:delete	datasources:* datasources:id:* datasources:id:1 (single data source)

Examples

Example Request:

```
DELETE /api/datasources/1 HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```
HTTP/1.1 200
Content-Type: application/json

{"message":"Data source deleted"}
```

Delete an existing data source by UID

`DELETE /api/datasources/uid/:uid`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope
datasources:delete	datasources:* datasources:uid:* datasources:uid:kLtEtcRGk (single data source)

Examples

Example request:

```
DELETE /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example response:

```
HTTP/1.1 200
Content-Type: application/json

{
    "message": "Data source deleted",
    "id": 1
}
```

Delete an existing data source by name

`DELETE /api/datasources/name/:datasourceName`

Required permissions

See note in the [introduction]({{< ref "#data-source-api" >}}) for an explanation.

Action	Scope
datasources:delete	datasources:* datasources:name:* datasources:name:test_datasource (single data source)

Examples

Example Request:

```
DELETE /api/datasources/name/test_datasource HTTP/1.1
Accept: application/json
```

```
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUlY2RnVKZTFVaDFsNFZXdE9ZWmNrMkZYbk
```

Example Response:

```
HTTP/1.1 200
Content-Type: application/json

{
  "message":"Data source deleted",
  "id": 1
}
```

Data source proxy calls

`GET /api/datasources/proxy/:datasourceId/*`

Proxies all calls to the actual data source.

Query a data source

Queries a data source having a backend implementation.

`POST /api/ds/query`

> ***Note:*** *Grafana's built-in data sources usually have a backend implementation.*

Example request for the Test data source:

```
POST /api/ds/query HTTP/1.1
Accept: application/json
Content-Type: application/json

{
   "queries":[
      {
         "refId":"A",
         "scenarioId":"csv_metric_values",
         "datasource":{
            "uid":"PD8C576611E62080A"
         },
         "format": "table"
         "maxDataPoints":1848,
         "intervalMs":200,
         "stringInput":"1,20,90,30,5,0",
      }
   ],
   "from":"now-5m",
   "to":"now"
}
```

JSON Body schema:

- **from/to** – Specifies the time range for the queries. The time can be either epoch timestamps in milliseconds or relative using Grafana time units. For example, `now-5m`.
- **queries** – Specifies one or more queries. Must contain at least 1.
- **queries.datasource.uid** – Specifies the UID of data source to be queried. Each query in the request must have a unique `datasource`.
- **queries.refId** – Specifies an identifier of the query. Defaults to "A".
- **queries.format** – Specifies the format the data should be returned in. Valid options are `time_series` or `table` depending on the data source.
- **queries.maxDataPoints** - Species the maximum amount of data points that a dashboard panel can render. Defaults to 100.
- **queries.intervalMs** - Specifies the time series time interval in milliseconds. Defaults to 1000.

In addition, specific properties of each data source should be added in a request (for example **queries.stringInput** as shown in the request above). To better understand how to form a query for a certain data source, use the Developer Tools in your browser of choice and inspect the HTTP requests being made to `/api/ds/query`.

Example Test data source time series query response:


```
{
  "results": {
    "A": {
      "frames": [
        {
          "schema": {
            "refId": "A",
            "fields": [
              {
                "name": "time",
                "type": "time",
                "typeInfo": {
                  "frame": "time.Time"
                }
              },
              {
                "name": "A-series",
                "type": "number",
                "typeInfo": {
                  "frame": "int64",
                  "nullable": true
                }
              }
            ]
          },
          "data": {
            "values": [
              [1644488152084, 1644488212084, 1644488272084, 1644488332084,
1644488392084, 1644488452084],
              [1, 20, 90, 30, 5, 0]
            ]
          }
```

```
        }
      ]
    }
  }
}
```

Deprecated resources

The following resources have been deprecated. They will be removed in a future release.

Query a data source by ID

> ***Warning:*** *This API is deprecated since Grafana v8.5.0 and will be removed in a future release. Refer to the new data source query API.*

Queries a data source having a backend implementation.

`POST /api/tsdb/query`

> ***Note:*** *Grafana's built-in data sources usually have a backend implementation.*

Example Request:

```
POST /api/tsdb/query HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "from": "1420066800000",
  "to": "1575845999999",
  "queries": [
    {
      "refId": "A",
      "intervalMs": 86400000,
      "maxDataPoints": 1092,
      "datasourceId": 86,
      "rawSql": "SELECT 1 as valueOne, 2 as valueTwo",
      "format": "table"
    }
  ]
}
```

JSON Body schema:

- **from/to** – Specifies the time range for the queries. The time can be either epoch timestamps in milliseconds or relative using Grafana time units. For example, `now-5m`.
- **queries.refId** – Specifies an identifier of the query. Defaults to "A".
- **queries.format** – Specifies the format the data should be returned in. Valid options are `time_series` or `table` depending on the data source.
- **queries.datasourceId** – Specifies the data source to be queried. Each `query` in the request must have a unique `datasourceId`.

- **queries.maxDataPoints** - Species the maximum amount of data points that a dashboard panel can render. Defaults to 100.
- **queries.intervalMs** - Specifies the time series time interval in milliseconds. Defaults to 1000.

In addition, specific properties of each data source should be added in a request. To better understand how to form a query for a certain data source, use the Developer Tools in your browser of choice and inspect the HTTP requests being made to `/api/tsdb/query`.

Example request for the MySQL data source:

```
POST /api/tsdb/query HTTP/1.1
Accept: application/json
Content-Type: application/json

{
  "from": "1420066800000",
  "to": "1575845999999",
  "queries": [
    {
      "refId": "A",
      "intervalMs": 86400000,
      "maxDataPoints": 1092,
      "datasourceId": 86,
      "rawSql": "SELECT\n  time,\n  sum(opened) AS \"Opened\",\n  sum(closed) AS
\"Closed\"\nFROM\n  issues_activity\nWHERE\n  $__unixEpochFilter(time) AND\n  period
= 'm' AND\n  repo IN('grafana/grafana') AND\n  opened_by IN('Contributor','Grafana
Labs')\nGROUP BY 1\nORDER BY 1\n",
      "format": "time_series"
    }
  ]
}
```

Example MySQL time series query response:

```
HTTP/1.1 200
Content-Type: application/json

{
  "results": {
    "A": {
      "refId": "A",
      "meta": {
        "rowCount": 0,
        "sql": "SELECT\n  time,\n  sum(opened) AS \"Opened\",\n  sum(closed) AS
\"Closed\"\nFROM\n  issues_activity\nWHERE\n  time >= 1420066800 AND time <=
1575845999 AND\n  period = 'm' AND\n  repo IN('grafana/grafana') AND\n  opened_by
IN('Contributor','Grafana Labs')\nGROUP BY 1\nORDER BY 1\n"
      },
      "series": [
        {
          "name": "Opened",
```

```
          "points": [
            [
              109,
              1420070400000
            ],
            [
              122,
              1422748800000
            ]
          ]
        },
        {
          "name": "Closed",
          "points": [
            [
              89,
              1420070400000
            ]
          ]
        }
      ]
    }
  }
}
```