

Data source API



- [Requirements](#)
- [Endpoints](#)
- [Get all data sources](#)
 - [Examples](#)
- [Get a single data source by uid](#)
 - [Examples](#)
- [Get a single data source by name](#)
 - [Examples](#)
- [Get data source Id by name](#)
 - [Examples](#)
- [Create a data source](#)
 - [Examples](#)
- [Update an existing data source](#)
 - [Examples](#)
- [Delete an existing data source by uid](#)
 - [Examples](#)
- [Delete an existing data source by name](#)
 - [Examples](#)
- [Data source proxy calls](#)
- [Check data source health](#)
 - [Examples](#)
- [Fetch data source resources](#)
 - [Examples](#)
- [Query a data source](#)
 - [Status codes](#)

```
{{< docs/shared lookup="developers/deprecated-apis.md" source="grafana"
version="<GRAFANA_VERSION>" >}}
```

Requirements

If you are running Grafana Enterprise, for some endpoints you'll need to have specific permissions. Refer to [Role-based access control permissions](#) for more information.

Endpoints

Method	URI	Summary
GET	/api/datasources	Get all data sources
GET	/api/datasources/uid/:uid	Get a single data source by uid
GET	/api/datasources/name/:name	Get a single data source by name (deprecated)
GET	/api/datasources/id/:name	Get data source Id by name (deprecated)
POST	/api/datasources	Create a data source
PUT	/api/datasources/uid/:uid	Update an existing data source
DELETE	/api/datasources/uid/:uid	Delete an existing data source by uid
DELETE	/api/datasources/name/:datasourceName	Delete an existing data source by name (deprecated)
GET	/api/datasources/proxy/uid/:uid/*	Data source proxy calls
GET	/api/datasources/uid/:uid/health	Check data source health
GET	/api/datasources/uid/:uid/resources/*	Fetch data source resources
POST	/api/ds/query	Query a data source

Get all data sources

GET /api/datasources

{{< admonition type="warning" >}} This API currently doesn't handle pagination. The default maximum number of data sources returned is 5000. You can change this value in the default.ini file. {{< /admonition >}}

Required permissions

See note in the [introduction](#) 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 <SERVICE_ACCOUNT_TOKEN>
```

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": {
      "logLevelField": "",
      "logMessageField": "",
```

```
        "maxConcurrentShardRequests": 256,  
        "timeField": "@timestamp"  
    },  
    "readOnly": false  
}  
]
```

Get a single data source by uid

GET /api/datasources/uid/:uid

Required permissions

See note in the [introduction](#) 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 <SERVICE_ACCOUNT_TOKEN>
```

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

{{< admonition type="warning" >}} This API is deprecated and will be removed in a future release. {{< /admonition >}}

Required permissions

See note in the [introduction](#) 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 <SERVICE_ACCOUNT_TOKEN>
```

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

{{< admonition type="warning" >}} This API is deprecated and will be removed in a future release. {{< /admonition >}}

Required permissions

See note in the [introduction](#) 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 <SERVICE_ACCOUNT_TOKEN>
```

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](#) 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 <SERVICE_ACCOUNT_TOKEN>

{
  "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"
}

```

{{< admonition type="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`. {{< /admonition >}}

Example Graphite Request with basic auth enabled:

```

POST /api/datasources HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer <SERVICE_ACCOUNT_TOKEN>

{
  "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 <SERVICE_ACCOUNT_TOKEN>

```
{
  "name": "test_datasource",
  "type": "cloudwatch",
  "url": "http://monitoring.us-west-1.amazonaws.com",
  "access": "proxy",
  "jsonData": {
    "authType": "keys",
    "defaultRegion": "us-west-1"
  },
  "secureJsonData": {
    "accessKey": "0l4pIDpeKSA6Xikg0l4p",
    "secretKey": "dGVzdCBBrZXkgYmxlYXNlIGRvbid0IHN0ZWFs"
  }
}
```

Update an existing data source

PUT /api/datasources/uid/:uid

Required permissions

See note in the [introduction](#) for an explanation.

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

Examples

Example Request:

```
PUT /api/datasources/uid/kLtEtcRGk HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer <SERVICE_ACCOUNT_TOKEN>

{
  "id":1,
```

```
"uid": "uid",
"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
}
```

Note that the UID cannot be modified.

Example Response:

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

{
  "datasource": {
    "id": 1,
    "uid": "uid",
    "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"
}
```

{{< admonition type="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. {{< /admonition >}}

Delete an existing data source by uid

DELETE /api/datasources/uid/:uid

Required permissions

See note in the [introduction](#) 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 <SERVICE_ACCOUNT_TOKEN>
```

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
```

{{< admonition type="warning" >}} This API is deprecated and will be removed in a future release. {{< /admonition >}}

Required permissions

See note in the [introduction](#) 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 <SERVICE_ACCOUNT_TOKEN>

Example Response:

HTTP/1.1 200

Content-Type: application/json

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

Data source proxy calls

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

Proxies all calls to the actual data source identified by the `uid`.

Check data source health

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

Makes a call to the health endpoint of data source identified by the given `uid`. This is not mandatory - every plugin author has to [implement support for health checks](#) in their plugin themselves.

Examples

Example Request:

```
GET api/datasources/uid/P8045C56BDA891CB2/health HTTP/1.1
```

Accept: application/json

Content-Type: application/json

Authorization: Bearer <SERVICE_ACCOUNT_TOKEN>

Example Response:

```
HTTP/1.1 200
```

```
Content-Type: application/json
```

```
{
  "message": "1. Successfully queried the CloudWatch metrics API.\n2. Successfully queried the\nCloudWatch logs API.",
  "status": "OK"
}
```

Fetch data source resources

```
GET /api/datasources/uid/:uid/resources/*
```

Makes a call to the resources endpoint of data source identified by the given `uid`.

Examples

Example Request:

```
GET api/datasources/uid/P8045C56BDA891CB2/resources/dimension-keys?region=us-east-2&namespace=AWS%2FEC2&dimensionFilters=%7B%7D&metricName=CPUUtilization HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer <SERVICE_ACCOUNT_TOKEN>
```

Example Response:

```
HTTP/1.1 200
```

```
Content-Type: application/json
```

```
[
  {
    "text": "AutoScalingGroupName",
    "value": "AutoScalingGroupName",
    "label": "AutoScalingGroupName"
  },
  {
    "text": "ImageId",
```

```

    "value": "ImageId",
    "label": "ImageId"
  },
  {
    "text": "InstanceId",
    "value": "InstanceId",
    "label": "InstanceId"
  },
  {
    "text": "InstanceType",
    "value": "InstanceType",
    "label": "InstanceType"
  }
]

```

Query a data source

Queries a data source having a backend implementation.

POST /api/ds/query

{{< admonition type="note" >}} Grafana's built-in data sources usually have a backend implementation. {{< /admonition >}}

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]
  ]
}
]
}
}
}
```

Status codes

Code	Description
200	All data source queries returned a successful response.
400	Bad request due to invalid JSON, missing content type, missing or invalid fields, etc. Or one or more data source queries were unsuccessful. Refer to the body for more details.
403	Access denied.
404	Either the data source or plugin required to fulfil the request could not be found.
500	Unexpected error. Refer to the body and/or server logs for more details.