



- [API Reference](#)
- [-- REST API Resources --](#)
- [Authentication](#)
- [Feedback](#)
- [Data Sources & Data Streams](#)
- [Channels](#)
 - [Get channels](#)
- [-- Information you need to Know --](#)
- [Rates and Limits](#)
- [Response Codes](#)
 - [HTTP Responses](#)
 - [HTTP Status Codes](#)
 - [Error Response Fields](#)
- [Anodot Error Codes](#)

API Reference

API Base URLs

- Asia Pacific customers, Use <https://ap.anodot.com>
- India customers, Use <https://in.anodot.com>
- US customers, Use <https://api.anodot.com>
- Personalized URL scheme? Use <https://my-org.anodot.com>

Welcome to the Anodot API Reference Site. Use the Anodot APIs to access Anodot Objects.

The Anodot API endpoints can be used to:

- Post metrics to Anodot using Metric 2.0 and Metric 3.0 protocols.
- Post Events and Tags to Anodot.
- Get information on metrics, alerts, feedback, anomalies and other entities

i We keep updating this site with new endpoints and APIs.
Need help? Reach out at support@anodot.com and our support team will be happy to assist you.

-- REST API Resources --

The various API endpoints and their supported functions are listed from here onwards.

Authentication

Basic Authentication

Using the basic token in a REST call:

```
POST https://api.anodot.com/api/v1/metrics?token=${DC_Key}&protocol=anodot20
```

Basic authentication is used for:

- Sending metrics to Anodot.
- Calling Anodot legacy APIs.

Anodot basic authentication uses a **Data Collection Key**. Copy the **Data Collection Key** from the Token Management page and use it in your REST calls.

i We recommend using Access Token based Authentication for all purposes other than posting metrics. See details below.

Access Tokens

Step 2: Token Request:

```
curl -X POST \
https://app.anodot.com/api/v2/access-token \
-H 'Content-Type: application/json' \
-d '{"refreshToken": "722354b89ae60761f9fcxcxca613315c"}'
```

The Response contains the token

```
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjoiMDgwNjVlYjA3ZTI4ZTRlZGVlZjxkZGJMTJlNWUyYjU1MjNjMjA4KTAwNjBjODI1YjYwM2M0MGJkOThlMThlYjQ2ZDMyMmIxDi1lLCJpYXQ1OjE1NTM2ODU4OTksImV4cCI6IjE5MzZg50X0.TC"
```

Step 3: Using the token in the calls

```
curl -X GET \
https://app.anodot.com/api/v2/feedbacks \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjoiMDgwNjVlYjA3ZTI4ZTRlZGVlZjxkZGJMTJlNWUyYjU1MjNjMjA4KTAwNjBjODI1YjYwM2M0MGJkOThlMThlYjQ2ZDMyMmIxDi1lLCJpYXQ1OjE1NTM2ODU4OTksImV4cCI6IjE5MzZg50X0.' \
-d '{"startTime": 1578391000, "endTime": 1578392000}'
```

To use access token authentication, follow these 3 simple steps:

1. Define a token in the Token Management page within Anodot.
2. Use the token in an Authentication call and get the access token which is valid for 24 hours. Use the **/access-token** endpoint:

```
POST https://app.anodot.com/api/v2/access-token
```

3. Use the access token in your REST calls
Set the Authentication header of your call with the bearer token.

AUTHENTICATION RESPONSE CODES

Code	Description
200	The response includes a token
401	Token Mismatch

i Remember:

Access tokens are valid for 24 hours. Include periodic calls in your code to get an updated token.

Feedback

End Point **GET /api/v2/feedbacks**

Our users and trigger recipients give feedback (Not Interesting/Good Catch) to anomaly alert triggers via:

- The Alert console
- The trigger investigation page
- Directly from the received trigger, via Email, Slack and other channels that support it.

Use the feedback API endpoint to get the feedback instances given by your organization members, tune your alerts, map your anomalies to business cases and improve the use of Anodot by your teams.

Authentication type: Access Token Authentication.

Get Feedback

Request Example:

```
curl -X GET \
https://app.anodot.com/api/v2/feedbacks \
-H 'Content-Type: application/json' \
-H 'Authorization: Bearer ${TOKEN}'
-d '{"startTime": 1578391000, "endTime": 1578392000}'
```

REQUEST ARGUMENTS

Argument	Type	Description
startTime [Required]	Epoch	Time the feedback was given. Default value is "Now minus 24 hours"
endTime [Required]	Epoch	Time the feedback was given. Default value is "Now".

Response Example:

```
{
  "total": 1,
  "feedbacks": [
    {
      "id": "a9b6d1ea-70bd-4bab-b567-a16ce60a91f2",
      "type": "GOOD_CATCH",
      "comment": "Another close alert test",
      "createdTime": 1578391511,
      "userName": "test@anodot.com",
      "anomalyId": "http://test.anodot.com/#!/anomalies?ref=pd&tabs=main;0&activeTab=1&anomalies=;0(d863a79ec48b407a808379facc89df7e)&duration=;1(1)&durationScale=;minutes(minutes)&delta=;0(0)&deltaTy
      "alerts": [
        {
          "id": "http://test.anodot.com/#!/alerts/6d59552d-7d61-44ae-bf20-2d39555af8c7",
          "emailId": "d863a",
          "alertName": "NewAlert1 {{component}}",
          "startTime": 1578381300,
          "endTime": 1578390300,
          "status": "CLOSE",
          "alertOwner": "automation testing"
        }
      ]
    }
  ]
}
```

RESPONSE FIELDS

Field	Type	Description / Example
total	Number	Number of feedback instances included in the response
feedbacks[]	Array	An array of feedback instances
id	String (\$uuid)	Feedback id
type	String	Type of feedback. Possible values: * Good Catch * Not Interesting
comment	String	Optional comment, if provided by the user while giving the feedback.
createdTime	Epoch	Feedback creation time.
username	String	Email of the user who provided the feedback
anomalyId	String	Link to the anomaly page in the Anodot platform.
alerts[]	Array	An array of alerts related to the feedback instance.

ALERTS ARRAY FIELDS

Field	Type	Description / Example
id	String	Link to the ID of the Alert in the Anodot platform.
emailId	String	5 characters used to identify the email. These are also the first 5 chars of the AnomalyId.
alertName	String	Alert name in Anodot
startTime	Epoch	Alert start time
endTime	Epoch	[Optional] Alert end time, relevant if the alert is closed.
status	String	The alert status. Possible values: * OPEN * CLOSE.
alertOwner	String	The alert owner in Anodot. Possible values are: * User first and Last name. * Group name.

Data Sources & Data Streams

End Point prefix is **/api/v2/bc**

The two steps to connect to an external data source and stream metrics to Anodot:

1. In Anodot - Create an Anodot data source according to your source type: S3, a database, a Kinesis Stream, Google Ads or any other source from the list of available resources.
2. In Anodot or through an API - Create one or more data streams to relay the needed data as metrics.

Additional information is available in our Help center documentation

Use the Kinesis API to:

- Get the list of Data sources in your account
- Get the available data streams in your account
- Create a new data stream
- Pause / Resume a data stream
- Delete a data stream

Authentication type: Access Token Authentication.

Data Sources

End Point prefix is **/api/v2/bc/data-sources**

There are several options to get the configured data sources:

- Get all data sources
- Get all data sources of a certain type
- Get all data sources according to names
- Get a data source according to its id

GET data-sources

Request Example: Get All Data Sources

```
curl -X GET \
  "https://app.anodot.com/api/v2/bc/data-sources" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${TOKEN}"
```

Request Example: Get S3 Data Sources

```
curl -X GET \
  "https://app.anodot.com/api/v2/bc/data-sources?type=s3" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${TOKEN}"
```

Use this call to get all data sources defined in your account.
You can optionally get the data sources by type.

REQUEST ARGUMENTS

Argument	Type	Description
query [Optional]	String	If left empty, the response returns all data sources. To get data sources of a certain source type, use the relevant ENUM value in the query.

LIST OF SOURCE TYPES

ENUM	Description
google_storage	Google Cloud Storage
google_analytics	Google Analytics
google_ads	Google Ads
bigquery	Google BigQuery DWH
adobe	Adobe Analytics
local_file	File Upload
s3	AWS S3
athena	AWS S3 Parquet data source powered with Athena
salesforce	Salesforce CRM
mysql	MySQL DB
psql	PostgreSQL DB
mssql	Microsoft SQL Server DB
databricks	Databricks
mariadb	Maria DB
redshift	AWS Redshift DWH
snowflake	Snowflake DWH
oracle	Oracle DB
mparticle	mParticle listener
kinesis	AWS Kinesis Stream

Response Example:

```
[
  {
    "id": "CLF6E0is1",
    "name": "demo.csv 1584355051733",
    "type": "local_file"
  },
  {
    "id": "CS3Jwrk4gf",
    "name": "Demo S3 1584005873272",
    "type": "s3"
  }
]
```

RESPONSE FIELDS

Field	Type	Description / Example
id	String	Data source id. This id can be used in future calls to stream creation.

Field	Type	Description / Example
name	String	Data Source name. Used for human readability.
type	String	The source type. Possible values are listed in source types table above.

GET data-sources/find

Use this call to get a list of data sources according to their name.

The string you include in the search query will be used for a case insensitive search in the data source names.

The result will be a list of data sources matching the search.

Request Example: Get Data Sources containing 'A' in their name

```
curl -X GET \
  "https://app.anodot.com/api/v2/bc/data-sources/find?searchQuery=s3" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${TOKEN}"
```

REQUEST ARGUMENTS

Argument	Type	Description
searchQuery	String	The query string is used to search the available the data source names

Response Example: an S3 data source and its parameters

```
{
  {
    "path" : "",
    "region" : "us-east-1",
    "modifyTime" : 1584285442,
    "id" : "CS3xxxBfrk4gf",
    "bucket" : "demo",
    "createTime" : 1584005874,
    "type" : "s3",
    "name" : "S3_anodot-bc Data Source 1584005873272"
  }
}
```

RESPONSE FIELDS

Response: List of data sources.

There are mutual fields appearing for all source types.

Each type has some fields relevant to that type.

Field	Type	Description / Example
id	string	The data source unique id
type	string [ENUM]	One of the data source types listed
name	string	Data source name as it appears in the Anodot App.
modifyTime	epoch	epoch time the data source was updated
createTime	epoch	epoch time the data source was created
id, type, name, modifyTime, createTime	Various	Additional fields according to data source type. e.g. authentication type, database name, database host, region, bucket name and more

GET data-sources/id

Request Example: Get Data Source by id

```
curl -X GET \
  "https://app.anodot.com/api/v2/bc/data-sources/CPGLrAfdxxxFm" \
```

```
-H 'Content-Type: application/json' \  
-H "Authorization: Bearer ${TOKEN}"
```

Use this call to get a single data source according to its id.
The result will be a data source object matching the id.

REQUEST ARGUMENTS

Argument	Type	Description
id	String	Data source id to retrieve.

Response Example: A PostgreSQL data source

```
{  
  "name" : "Postgres demo Data Source 1584551534093",  
  "dbHost" : "demo.anodot.com",  
  "type" : "psql",  
  "modifyTime" : 1584551534,  
  "id" : "CPGLrAfdxxxFm",  
  "verifyServerCertificate" : false,  
  "createTime" : 1584551534,  
  "dbName" : "demo_api",  
  "dbPort" : 5432,  
  "useSSL" : true,  
  "userName" : "demodbuser"  
}
```

RESPONSE FIELDS

Response: A data source object, including all fields according to its type.

Field	Type	Description / Example
id	string	The data source unique id
type	string [ENUM]	One of the data source types
name	string	Data source name as it appears in the Anodot App.
modifyTime	epoch	epoch time the data source was updated
createTime	epoch	epoch time the data source was created
Key-Ök	Various	Additional fields according to data source type. e.g. authentication type, database name, database host, region, bucket name and more

Data Streams

End Point prefix is **/api/v2/bc/data-streams**

- Get the available data streams in your account
- Create a new data stream
- Pause / Resume a data stream
- Delete a data stream

GET data-streams

Request Example: Get all data streams basic information

```
curl -X GET \  
"https://app.anodot.com/api/v2/bc/data-streams" \  
-H 'Content-Type: application/json' \  
-H "Authorization: Bearer ${TOKEN}"
```

Response Example:

```
{  
  "id": "SSSrLxxxxxxSC",  
}
```

```

    "name": "postgres test",
    "type": "psql"
  },
  {
    "id": "555yxxxxxgfg",
    "name": "S3 Test",
    "type": "s3"
  },
  {
    "id": "555xxxxzYUP00",
    "name": "Google Ads Test",
    "type": "google_ads"
  },
  {
    "id": "5554xxxxxQqn",
    "name": "My Data Test",
    "type": "local_file"
  },
  {
    "id": "555xxxxxxxCoR",
    "name": "S3 Parquet Test",
    "type": "athena"
  }
]

```

Use this call to get all data streams in your account with basic information

RESPONSE FIELDS

The response includes a basic list of data streams, the fields for each data stream are:

Field	Type	Description
id	string	Data stream unique ID
name	string	Data stream name.
type	string [c f i /]	Data stream type. See source type list

GET data-streams/find

Request Example: Get Data Streams containing **XYA** in their name

```

curl -X GET \
  "https://app.anodot.com/api/v2/bc/data-streams/find?searchQuery=test" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${TOKEN}"

```

Use this call to get a list of data streams according to the search query.

The string you include in the search query will be used for a case insensitive search in the data stream names.

The result will be a list of data streams matching the search.

- i** The result is a list of full data stream objects. Use these structures in the creation calls to create new streams with proper configuration.

REQUEST ARGUMENTS

Argument	Type	Description
searchQuery	String	The query string is used to search the available the data stream names

Response Example: an S3 data stream and its parameters

```

{
  {
    "timeDefinition": {
      "timeColumnIdx": 5,
      "timePattern": "epoch_seconds"
    },
    "schema": {
      "columns": [
        {
          "targetType": "gauge",
          "id": "5655-cd1f2cff1c1d",

```



```
    "sourceColumn" : "0",
    "name" : "id",
    "hidden" : false,
    "type" : "metric"
  },
  {
    "type" : "metric",
    "hidden" : false,
    "sourceColumn" : "1",
    "name" : "number1",
    "targetType" : "gauge",
    "id" : "0838-fc15de8c75c8"
  },
  {
    "hidden" : false,
    "type" : "metric",
    "sourceColumn" : "2",
    "name" : "number2",
    "id" : "74e6-090be30f786f",
    "targetType" : "gauge"
  },
  {
    "hidden" : false,
    "type" : "metric",
    "id" : "4505-7814afd0f9fa",
    "targetType" : "gauge",
    "sourceColumn" : "3",
    "name" : "number3"
  },
  {
    "name" : "cloths",
    "sourceColumn" : "4",
    "id" : "9472-9a1b7e338ef6",
    "type" : "dimension",
    "hidden" : false
  }
],
"sourceColumns" : [
  {
    "id" : "0",
    "index" : 0
  },
  {
    "index" : 1,
    "id" : "1"
  },
  {
    "index" : 2,
    "id" : "2"
  },
  {
    "index" : 3,
    "id" : "3"
  },
  {
    "index" : 4,
    "id" : "4"
  }
]
},
"modifyTime" : 1584355552,
"paused" : false,
"hasDimreduce" : false,
"metrics" : [
  0,
  1,
  2,
  3
],
"dataSourceId" : "CS3JwVbfxxgxf",
"missingDimPolicy" : {
  "action" : "fill",
  "fill" : "unknown"
},
"fileNamePattern" : "yyyyMMddHH",
"state" : "running",
"path" : "folder6f537479-xxxx-43f2-9ce3-66b0294b7532",
"maxMissingFiles" : 0,
"type" : "s3",
"createTime" : 1584355524,
"fileNameSuffix" : "_data.csv",
"fileNamePrefix" : "data3_",
"name" : "test",
"timeZone" : "UTC",
"status" : "ok",
"pollingInterval" : "daily",
"fileFormat" : {
  "shouldReplaceEmptyCells" : true,
  "decimalPointChar" : ".",
  "hasHeader" : true,

```

```
    "delimiter" : ",",
    "ignoreEmptyLines" : false,
    "quoteChar" : ""
  },
  "historicalDateRange" : {
    "constRange" : "m3"
  },
  "id" : "SSSq07uxxyvI",
  "dimensions" : [
    4
  ]
}
]
```

RESPONSE FIELDS

The response contains a list of complete data stream objects

GET data-streams/:id

Request Example: Get Data Stream by id

```
curl -X GET \
  "https://app.anodot.com/api/v2/bc/data-streams/{id}" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer s{TOKEN}"
```

Use this call to get a single data stream according to its id.
The result will be a data stream object matching the id.

REQUEST ARGUMENTS

Argument	Type	Description
id	String	Data stream id to retrieve.

Response Example: A Google Ads data stream

```
{
  "status" : "ok",
  "createTime" : 1584878085,
  "schema" : {
    "sourceColumns" : [
      {
        "name" : "Cost",
        "id" : "Cost"
      },
      {
        "id" : "Clicks",
        "name" : "Clicks"
      },
      {
        "id" : "Impressions",
        "name" : "Impressions"
      },
      {
        "name" : "Conversions",
        "id" : "Conversions"
      }
    ],
    "columns" : [
      {
        "id" : "32dd-8f1cbb6d3e48",
        "hidden" : false,
        "name" : "Cost",
        "metricTags" : {
          "unit" : "USD"
        },
        "type" : "metric",
        "targetType" : "counter",
        "transform" : {
          "parameters" : [
            "0.000001"
          ],
          "name" : "scale",
          "input" : [
            {
              "sourceColumn" : "Cost"
            }
          ]
        }
      }
    ]
  }
}
```

```

    }
  },
  {
    "name": "Clicks",
    "sourceColumn": "Clicks",
    "hidden": false,
    "id": "79d8-fa375dd34f90",
    "targetType": "counter",
    "type": "metric"
  },
  {
    "sourceColumn": "Impressions",
    "name": "Impressions",
    "id": "8214-d826d1746a93",
    "hidden": false,
    "type": "metric",
    "targetType": "counter"
  },
  {
    "type": "metric",
    "targetType": "counter",
    "id": "5489-090f6c31b882",
    "hidden": false,
    "sourceColumn": "Conversions",
    "name": "Conversions"
  },
  {
    "name": "Dummy Dimension",
    "hidden": false,
    "id": "dfcb-4aaf2c5188b1",
    "transform": {
      "parameters": [
        "1"
      ],
      "name": "dummy",
      "input": []
    },
    "type": "dimension"
  }
]
},
"timezone": "America/Los_Angeles",
"missingDimPolicy": {
  "action": "fill",
  "fill": "unknown"
},
"metrics": [
  "Clicks",
  "Conversions",
  "Cost",
  "Impressions"
],
"pollingInterval": "daily",
"paused": false,
"clientCustomerId": "7559118320",
"type": "google_ads",
"state": "running",
"id": "SSSjMxxfXgdL",
"hasDimreduce": false,
"pollingResolution": "days",
"delayMinutes": 60,
"modifyTime": 1584879047,
"historicalDateRange": {
  "constRange": "m1"
},
"reportType": "ACCOUNT_PERFORMANCE_REPORT",
"name": "test",
"dataSourceId": "CAW5nPxxxxwt4",
"dimensions": [],
"basedOnTemplateId": "44"
}

```

RESPONSE FIELDS

The response contains the complete data stream object.

POST data streams

Request Example: Create a PostgreSQL data Stream

```

curl -X POST \
  "https://app.anodot.com/api/v2/bc/data-streams" \
  -H 'Authorization: Bearer ${TOKEN}' \

```

```

-H 'Content-Type: application/json' \
-d '{
  "tableName": "bc_stream",
  "dimensions": [
    "status",
    "resolution"
  ],
  "timestampColumn": "create_time",
  "historicalDateRange": {
    "constRange": "d3"
  },
  "maxBackFillIntervals": 3,
  "dataSourceId": "CPGLrAfd3YpFm",
  "timeZone": "UTC",
  "metrics": [
    "is_paused"
  ],
  "schema": {
    "sourceColumns": [
      {
        "name": "is_paused",
        "id": "is_paused"
      },
      {
        "name": "status",
        "id": "status"
      },
      {
        "id": "resolution",
        "name": "resolution"
      }
    ],
    "columns": [
      {
        "name": "is_paused",
        "targetType": "counter",
        "type": "metric",
        "hidden": false,
        "id": "b888-6cd7a274c69a",
        "sourceColumn": "is_paused"
      },
      {
        "sourceColumn": "status",
        "id": "8e11-b51f18268970",
        "type": "dimension",
        "hidden": false,
        "name": "status"
      },
      {
        "name": "resolution",
        "id": "3ec1-fa87d9bf32a4",
        "sourceColumn": "resolution",
        "type": "dimension",
        "hidden": false
      }
    ]
  },
  "pollingInterval": "hourly",
  "delayMinutes": 60,
  "schemaName": "public",
  "name": "postgres test api 2",
  "customQuery": false,
  "missingDimPolicy": {
    "action": "ignore",
    "fill": "unknown"
  },
  "type": "psql",
  "timestampType": "TIMESTAMP"
}

```

Request Example: Create a Kinesis data Stream

```

curl -X POST \
  "https://app.anodot.com/api/v2/bc/data-streams" \
  -H 'Authorization: Bearer ${TOKEN}' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "kinesis api test 1",
  "dataSourceId": "CKIST8xxxI4Vp",
  "type": "kinesis",
  "pollingInterval": "m5",
  "dimensions": ["serverId.id", "checksum.offset"],
  "metrics": ["load"],
  "schema": {
    "columns": [
      {
        "sourceColumn": "load",

```

```

      "id": "b8af-14c99d63d6c1",
      "name": "load",
      "type": "metric",
      "targetType": "gauge",
      "hidden": false
    },
    {
      "sourceColumn": "serverId.id",
      "id": "b8af-14c99d63d6c2",
      "name": "serverId.id",
      "type": "dimension",
      "hidden": false
    },
    {
      "sourceColumn": "checksum.offset",
      "id": "b8af-14c99d63d6c3",
      "name": "checksum.offset",
      "type": "dimension",
      "hidden": false
    }
  ],
  "sourceColumns": [
    {"id": "load", "path": "load"},
    {"id": "serverId.id", "path": "serverId.id"},
    {"id": "checksum.offset", "path": "checksum.offset"}
  ]
},
"recordType": "single_json",
"filters": [{"path": "clazz", "value": "HeartbeatMessage"}],
"delayMinutes": 5,
"timeDefinition": {"path": "checksum.time", "timePattern": "epoch_millis"}
}'

```

Use this call to create a data-stream. The call needs to contain the entire data-stream object, including the link to parent data-source.

Remember:

The parent data source should be defined within the Anodot App.
Use the GET data-sources call to retrieve its id and use the id as the datasourcelid in the stream you create.

REQUEST ARGUMENTS

The request is a complete data stream object.
See below a partial list of stream components.

- **Context:** Where is the information taken from.
 - All types: Data Source Id, Data stream name, type
 - S3: Path in the bucket, file prefix and suffix, filename pattern
 - RDB : Schema name, Table name
- **Schedule:** Data stream timing properties
 - Collection interval
 - Timezone
 - Delay
 - History span to collect
 - Backfill policy
- **Schema:** The measures and dimensions collected by the stream
 - Measures: name, aggregation type, unit
 - Dimensions: name

A Pro Tip:

To get the relevant data stream object structure, do a GET data-stream on an existing live stream of the same type in your account.
You will be able to see the complete object structure in the output.

SCHEDULE ARGUMENTS

Field	Description & Values
pollingInterval	The possible frequencies to collect data: m1, m5, m10, m15, m30, hourly, h2, h3, h4, h6, h8, h12, daily [m=minute, h=hour]
Historical Range	The history to collect: h1, h4, d1, d3, w1, m1, m3, m6, y1 [h=last hour, w=last week, m=last month, y=last year]

Polling interval restrictions per type:

- **Google Analytics:**
 - polling interval: hourly, daily
 - pollingResolution: hours, days

- Allowed combinations hourly&hours OR daily&days
- **BigQuery:** hourly, daily
- **Salesforce:** hourly, daily
- **Adobe Analytics:** hourly, daily
- **Kinesis:** m1, m5, m10, m15, m30, hourly
- **SQL typed:** m1, m5, m15, hourly, h2, h3, h4, h6, h8, h12, daily
- **S3, GCS:** Also consider the fileDataPattern

Historical time span restrictions per type:

- **Kinesis:** no history

Polling interval and Historical span Allowed combinations:

Polling interval	Historical spans allowed
m1	h1, h4, d1, d3, w1
m5, m10, m15, m30	h1, h4, d1, d3, w1, m1
hourly, h2, h3, h4, h6, h8, h12	d1, d3, w1, m1, m3, m6, y1
daily	d1, d3, w1, m1, m3, m6, y1, y2, y5, y10, y15, y20

RESPONSE FIELDS

The response contains the created data stream object.

Get data stream total metric count

Request Example:

```
curl -X GET \
  "https://app.anodot.com/api/v2/bc/data-streams/metrics/total?streamId={id}&timeRange=day" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${TOKEN}"
```

Use this call to get the number of metrics created by an active stream in a designated time range.

REQUEST ARGUMENTS

Argument	Type	Description
streamId	String	Data stream id to retrieve. Alternatively, you can use the stream name.
streamName	String	Full data stream name to retrieve.
timeRange	String [ENUM]	Time range to collect the number of metrics. Allowed values: "day", "week"

Response Example:

```
{
  "total" : 100
}
```

RESPONSE FIELDS

Field	Type	Description / Example
total	Int	Number of metric created by the data stream in the given time range.

Pause or Resume a data Stream

Request Example:

```
curl -X PUT \
  "https://app.anodot.com/api/v2/bc/data-streams/{id}/state/{action}" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${TOKEN}"
```

Use this call to pause a working data stream, or resume a paused stream.

REQUEST ARGUMENTS

Argument	Type	Description
id	String	Id of the data stream you wish to pause or resume
action	String [ENUM]	The action to perform on the data stream. Valid values: "pause", "resume"

Response Example:

```
{
  "name" : "test",
  "id" : "SSSGxxxDegnQEv",
  "paused" : true
}
```

RESPONSE FIELDS

Field	Type	Description / Example
name	String	Data stream name
id	String	Data stream id
paused	boolean	true - Data stream is paused, false - Data stream activity is resumed

Delete data Stream

Request Example:

```
curl -X DELETE \
  "https://app.anodot.com/api/v2/bc/data-streams/{id}" \
  -H 'Content-Type: application/json' \
  -H "Authorization: Bearer ${TOKEN}"
```

Use this call to delete a stream based on its id.

REQUEST ARGUMENTS

Argument	Type	Description
id	String	Id of the data stream you wish to delete
deleteMetrics [optional]	boolean	Delete the metrics created by the stream. Default = true.

Response Example

If the operation succeeded - you get no response.

Failure Response Example:

```
{
  "path" : "/api/v2/bc/data-streams/SSSGYfxxxob7",
  "message" : "object not found in underlying storage: stream SSSGYfxxxob7 was not found for user {account id}",
  "additionalInfo" : null,
  "name" : "HttpError",
  "andErrorCode" : 11002,
  "status" : 404
}
```

RESPONSE

If the operation succeeded - you get no response.

In case the operation failed, you will receive the failure reason

End Point **GET /api/v2/channels**

Channels are the outgoing integrations used to send alerts to their destinations. Anodot supports multiple channel types.

Use the channel API endpoint to get the channels available in your account and link them to alerts you create using the alerts API.

Authentication type: Access Token Authentication.

Get channels

Request Example: GET All channels in the account

```
curl -X GET \
https://app.anodot.com/api/v2/channels \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${TOKEN}"
```

Request Example: GET All channels of type slack

```
curl -X GET \
https://app.anodot.com/api/v2/channels?type=slack \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${TOKEN}"
```

Request Example: GET All channels of type slack containing "alert" in the name

```
curl -X GET \
https://app.anodot.com/api/v2/channels?type=slack&name=alert \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer ${TOKEN}"
```

REQUEST ARGUMENTS

Argument	Type	Description
type [Optional]	string	Limit the response to this channel type. Use single, lowercase word. Possible values are listed in the channel list below.
name [Optional]	string	Limit the response to anodot channel names containing this string

CHANNEL LIST

Type	Description
email	Create an email and send to participants in the email distribution list
webhook	Send the alert as a JSON object to a webhook server
slack	Post the alert as a slack message on a slack channel
pagerduty	Create an event in a PagerDuty service
jira	Open the alert as ticket in a JIRA project
opsgenie	Create an OpsGenie alert
msteams	Post the alert as an MS Teams message on an MS Teams channel

Response Example:

```
[
  {
    "id": "2b7465e5-dbda-46f5-ae67-xxxxxyyyyy",
    "name": "Test",
    "type": "email"
  },
  {
    "id": "558ef7a2-7064-49be-873d-xxxxxyyyyy",
    "name": "test slack",
    "type": "slack"
  }
]
```


RESPONSE FIELDS

The response is a list of channels

Field	Type	Description / Example
id	String (\$uuid)	channel id. Use this id when you create alerts via API and need to provide a destination channel.
type	String	Channel type. See possible values in channel list
name	String	Channel name.

-- Information you need to Know --

Rates and Limits

APIs are limited by:

The default rate limit for Anodot end-points is 500 calls per minute. Specific end-points have different limits, please see the table below.

i Please adhere to the rate limitations to prevent data loss and error handling.

End Point	Rate Limit (RPM)
Feedback	GET - 50 RPM
Stream	POST/PUT - 1 RPM GET - 10 RPM DELETE - 15 RPM

The overall number of entities is limited according to the terms of use or your specific contract.

Response Codes

HTTP Responses

Anodot APIs use HTTP status codes to indicate the success or failure of a request.

An error indicates that the service did not successfully handle your request. In addition to the status code, the response may contain a JSON object with an errors array containing more detailed error messages (see Error response format below).

If the service is able to handle your request, but some issues are present (e.g. one of the metric sent has an illegal timestamp but all the rest are ok), the HTTP status code will indicate success and the response body will contain the expected result with the addition of errors array containing detailed error messages.

HTTP Status Codes

Error Code	Meaning
200	OK
400	Bad Request
401	Unauthorized

Error Code	Meaning
403	Forbidden
404	Not Found
500	Server error

Error Response Fields

Error Response Example

```

{
  "errors": [
    {
      "index": "failed sample index",
      "error": "<error code>",
      "description": "error description"
    }
  ]
}

```


Field	Description
errors	An array of errors received from the request
index	In case a batch of records was sent in the request, the index directs to the faulty entry
error	Error code
description	The error Description

The error code list

Anodot Error Codes

Error Code	Description
1001	Json parsing error
1002	General error
1004	API calls rate exceeded the maximum allowed
1005	Number of streams exceeded the maximum allowed
1006	Number of incomplete streams exceeded the maximum allowed
1007	Number of Data Sources exceeded the maximum allowed
2001	Token is not active
2002	Too many concurrent requests
2003	Metrics per seconds limit reached
2004	Metric properties are not defined. Must have at least one property.
2005	Metric properties limit exceeded. Metric may contain up to: {xxx} properties.
2006	Undefined property key. Metric property key must contain at least 1 character.
2007	Property: "{xxx}" key length exceeded. maximum key length is {yyy} characters.
2008	Undefined property value. Metric property value must contain at least 1 character.
2009	Property: "{xxx}" value length exceeded. Maximum value length is {yyy} characters .

Error Code	Description
2010	Property key: "{xxx}" contains illegal characters. "." character and space characters are not allowed.
2011	Property value: "{xxx}" contains illegal characters. "." character and space characters are not allowed.
2012	Property: "what" is undefined. "what" is a mandatory property that specified what is being measured by this metric.
2013	Metric tags limit exceeded. Metric may contain up to: {xxx} tags.
2014	Undefined tag key. Metric tag key must contain at least 1 character.
2015	Tag: "{xxx}" key length exceeded. Maximum key length is {yyy} characters.
2016	Undefined tag value. Metric tag value must contain at least 1 character.
2017	Tag: "{xxx}" value length exceeded. Maximum value length is {yyy} characters.
2018	Tag key: "{xxx}" contains illegal characters. "." character and space characters are not allowed.
2019	Tag value: "{xxx}" contains illegal characters. "." character and space characters are not allowed.
2020	Metric timestamp is undefined. Timestamp must be an epoch time in seconds.
2021	Metric timestamp: "{xxx}" is not a valid epoch time. Timestamp must be an epoch time in seconds.
2022	Metric timestamp: "{xxx}" is in the future. Timestamp must be an epoch time in seconds.
2023	Metric timestamp: "{xxx}" is negative. Timestamp must be an epoch time in seconds.
2024	Metric value is undefined. The "value" must be a valid decimal double precision number.
2025	Metric value: "{xxx}" is not a valid decimal number. The "value" must be a valid decimal double precision number.
2026	Metric value is NaN or Infinite
2027	Metric name is null or empty
2028	Metric name length exceeded the maximum allowed: {xxx}.
2029	Invalid protocol - supported protocols are: {xxx}.
2030	Invalid target type: "{xxx}". valid values are: {yyy}.

 The list is partial. Response codes are added for new and existing endpoints.