anodot

We monitor your business.

Using Anodot Metric 3.0 Protocol

February 2021

anodot

We monitor your business.

Introduction

What is the value of the Anodot Metric 3.0 Protocol?

- Using the Metric 3.0 protocol, the data which you send to Anodot is much more structured along dimensions and tags. The metrics all align to a pre-defined schema which enables better error handling, better investigation, etc.
- It is possible to post timestamp watermarks to indicate 'flushing directives' to Anodot (See below an explanation on how watermarks work and what are they good for).

High level flow

Creating a Schema

Working with the Metric 3.0 Protocol requires the user to define a schema before sending the metrics. Having this schema enables Anodot to better understand the structure of the metrics and various features are dependent on this schema declared. So the first step you will need to do before starting to send the metrics themselves is to create the schema using the `/stream-schema` API.
A schema has the following structure:

- The required measures and their attributes
    - Units
    - Aggregation
    - Counting method
- The required dimensions and attributes
    - Name
    - Fill policy

Sending Metrics matching the Schema

Sending metrics to Anodot is done using the `/metrics` API. The parameters are pretty much the same as the call in the 2.0 protocol, with the main difference being the 'body' of the call, where the data is sent according to the schema defined in the previous steps.

Send Watermarks

When Anodot gets data points, it collates them into ‘buckets’. The reason is to enable working on aggregated data points to make the detection of anomalies as efficient as possible. In order to make sure that we collect the data points into the correct ‘buckets’, Anodot needs to know when is the right time to ‘flush’ this collection of data points. When you are sending the data to Anodot via API, there is no way of the platform to know this and you have to explicitly state - “this time frame is done”. This concept is called ‘watermark’. A ‘watermark’ is a signal to Anodot that the data is ready to be processed. Sending this command basically tells Anodot - “no data samples of a later date will arrive” and it can stop collating the data in the ‘bucket’ and start the processing in the algorithm. Sending the watermark is done using the `/metrics/watermark` API.

We monitor your business.

Detailed Calls

Create Schema

Follow the Authentication process to obtain a bearer token

Request Description

Field	Mandatory	Description
version	Optional	String Describe the schema version
name	Mandatory	String / [200 chars] Schema name, will be later used as the stream name Must be unique
Dimensions []	Mandatory	List of dimensions, max 30 Each dimension must be unique Each dimension name is up to 30 chars
measurements[]	Mandatory	List of measurements, max 200
For each measurement:		
units	Optional	Provide the unit for the measure, this unit will be used in Anodot display
aggregation	Mandatory	Aggregation function for the measure. Valid values: - average - sum
countby	Mandatory	How to count the data points, Valid value: currently “None” is the only value supported (each bucket counts as “1”)
missingDimPolicy		How to treat missing dimension values in the posted data.
action		Valid values: - Fill - the empty dimension value will be filled with the value in the “fill” field (see below) - Fail - data point will be rejected
fill		String

		Fill value. Applicable if action field is set to fill

Request Example

```
curl --location --request POST
'https://app.anodot.com/api/v2/stream-schemas' \
--header 'Authorization: Bearer {{bearer-token}}' \
--header 'Content-Type: application/json' \
--data-raw '{
 "version": "1",
 "name": "schema name",
 "dimensions": [
   "OS",
   "GEO"
 ],
 "measurements": {
   "m1": {
     "units": "sec",
     "aggregation": "average",
     "countBy": "none"
   },
   "m2": {
     "units": "sec",
     "aggregation": "average",
     "countBy": "none"
   }
 },
 "missingDimPolicy": {
   "action": "fill",
   "fill": "dummy_val"
 }
}'
```

Response Description

The response contains the schema details and provides a schema id for you to use when posting metrics to this schema.

Response Example

```
{
  "schema": {
    "id": "111111-22222-3333-4444",
    "version": "1",
    "name": "schema name",
    "dimensions": [
      "OS",
      "GEO"
    ],
    "measurements": {
      "m1": {
        "units": "sec",
        "aggregation": "average",
        "countBy": "none"
      },
      "m2": {
        "units": "sec",
        "aggregation": "average",
        "countBy": "none"
      }
    },
    "missingDimPolicy": {
      "action": "fill",
      "fill": "dummy_val"
    }
  },
  "meta": {
    "createdTime": "1547630800000",
    "modifiedTime": "1547630800000"
    }
}
```

Get schemas

After you’ve created the schema, you can use this call get the list of all schemas. Follow the Authentication process to obtain a token

Request Example

```
curl --location --request GET
'https://app.anodot.com/api/v2/stream-schemas/schemas' \
--header 'Authorization: Bearer {{bearer-token}}'
```

Response Description

If the request is successful, you will get an array of “streamsSchemaWrapper” objects. Each of these objects has the following properties:

Field	Description
schema	Schema Object (see above for definition)
meta	Created and Modifed time of the schema.

Response Example

```
[
    {
        "streamSchemaWrapper": {
```

```
        "schema": {
            "id": "0MCop0RjSsWk3W-QGmpnWA",
            "version": "1",
            "name": "Anodot Usage 002 - Hourly",
            "dimensions": [
                "Event_Action",
                "Event_Category",
                "Event_Label",
                "Page_path_level_2",
                "Page_path_level_3",
                "Page_path_level_4",
                "Version"
            ],
            "measurements": {
                "Total_Events": {
                    "aggregation": "sum",
                    "countBy": "none"
                },
                "Pageviews": {
                    "aggregation": "sum",
                    "countBy": "none"
                }
            },
            "missingDimPolicy": {
                "action": "ignore"
            }
        },
        "meta": {
            "createdTime": 1533540645730,
            "modifiedTime": 1533540645730
            }
        },
        "schemaCubesWrapper": {}
    }
```

Post Metrics

General

1. Authentication: Use the same token as you use with Metric 2.0 protocol
2. Body: Array of data samples. Maximal number of entries is 10K per request.

Request Description

Field	**Mandatory**	**Description**
schemaId	Mandatory	Id received from the “create schema” call
timestamp	Mandatory	Integer - The data sample’s timestamp Unix epoch time in seconds At most one hour in the future
dimensions{}	Mandatory	Key-value pairs of properties
measurements{}	Mandatory	Decimal double precision number. Without a thousands separator
tags{}	Optional	List of tags attached to the measures Key value pairs of Tags are metadata of the metric and do not affect its uniqueness

Request Example

```
curl --location --request POST
'http://app.anodot.com/api/v1/metrics?protocol=anodot30&token={{data-
token}} \
```

We monitor your business.

```
--header 'Content-Type: application/json' \
--data-raw '[
 {
   "schemaId": "111111-22222-3333-4444",
   "timestamp": “143876178”,
   "dimensions": {
     "Geo": "US",
     "Device": "Mobile",
     "ProductCategory": "Shoes"
   },
   "measurements": {
     "measure1": “10”,
     "measure2": “25.5”
   },
   "tags": {
     "ActiveCampaignID": [
       “1234”
     ],
     "AccountManagers": [
       "JohnDoe",
       "MaryJane"
     ]
   }
 }
]'
```

anodot
We monitor your business.

Sending Stream Watermark

A watermark timestamp commits that no data samples with timestamps less than or equal to it will be sent. For authentication use the same token used for posting the data points.

Request Description

Field	Mandatory	Description
schemaId	Mandatory	Id received from the “create schema” call
watermark	Mandatory	Integer - timestamp of the watermark. (Unix epoch time in seconds)

Request Example

```
curl --location --request POST
'http://app.anodot.com/api/v1/metrics/watermark?protocol=anodot30&tok
en={{data-token}} \
--header 'Content-Type: application/json' \
--data-raw '{
 "schemaId": "111111-22222-3333-4444",
 "watermark": “143877000”
}
'
```

Note:
To close the bucket and send the data for processing you need to send a beginning of the next bucket as a watermark timestamp. For example, you have hourly data and you’ve just sent data points with `2020-06-01 06:00` timestamp. To commit it to processing you need to send `2020-06-01 07:00` watermark (the beginning of the next hourly bucket). Note that flushing the

data is a scheduled process and you will probably need to wait some time (around 15 minutes) until you see a data point in the UI after sending the watermark.

FAQ

Some questions we've gathered on using this API

Is the schema mandatory?

- Yes it is. The schema defines the structure of the metrics expected to be received when sending them using Metric 3.0 protocol. Using the schema enables benefits down the road, such as advanced analysis and investigation, to enable easier root cause analysis of your anomalies.

If I send a request in the wrong structure?

- The "post metrics" request is composed of an array with up to 10K data samples. It is possible that some entries will be faulty and will be rejected. Please analyze the response to view the faults.

What does the watermark do?

- The first watermark of the stream represents that historical (retroactive) data was consumed.
- Every other watermark notifies the platform that another "chunk" of information is ready for processing, and there is no need to wait for additional information in that span. This implies composite metrics / alerts can be calculated on the spot, and cube analysis can be conducted as well.

Is using the watermark mandatory?

- No, but it is highly recommended to use it. Even though Anodot can process the data samples using the current flushing mechanism, the watermark provides precise processing indicators which result in even more accurate anomaly detection and alerting. It's Anodot - only better!