Custom Metrics

SPM lets you extend standard performance metrics reports. SPM exposes APIs and provides libraries that let you send custom metrics (any numerical data, not just performance metrics) into SPM and graph it along other reports.

NOTE:

- To be able to use Custom Metrics, you need a Sematext account. If you don't already have it, you can create it here, it's free, no credit card needed. After you have Sematext account, create an SPM App to which Custom Metrics will be sent.
- If you have already created some SPM Apps under your account in the past, you can send Custom Metrics to any of them.
- If you just registered, you can create SPM Apps by following the steps after Sematext account registration, or by clicking directly here.

Terminology

- A "metric" is a distinct triple of (metric name, filter1 name, filter2 name). Some examples:
  - *(“cpuIdle”, **server=“server1”**, **null**) vs. **(“cpuIdle”,** null**,** server=“server1”**)* – these are 2 distinct metrics
  - *(“**cpuSteal**”, server=“server1”, null) vs. **(“**cpuIdle**”, server=“server1”, null)* – these are 2 distinct metrics
  - *(“diskRead”, server=“server1”, device=“/dev/**xxx**”) vs. (“diskRead”, server=“server1”, device=“/dev/**yyy**”)* – this is the same 1 metric with two different values for filter2 named “device”
  - *(“diskWrite”, server=“server1”, device=“/dev/**xxx**”) vs. (“diskWrite”, server=“server1”, device=“/dev/**yyy**”)* – this is the same 1 metric with two different values for filter2 named “device”
  - *(“**diskRead**”, server=“server1”, device=“/dev/xxx”) vs. (“**diskWrite**”, server=“server1”, device=“/dev/xxx”)* – these are 2 distinct metrics

- A "data point" is the value associated with some (metric, filter1, filter2) combination. Some examples:
  - *(“cpuIdle”, server=“server1”, null) = 90*
  - *(“cpuIdle”, server=“server1”, null) = 90* – this is a distinct data point for the same metric as in the previous line

Introduction

A data point has the following attributes:

- timestamp - time since “Epoch” in milliseconds
- name - name of metric
- value - double value
- aggregation type - min, max, avg, sum
- filter1 value - string value for filter1
- filter2 value - string value for filter2

Custom metrics reports display values aggregated using ‘aggregation type’. For example, values 1.0, 2.0, and 3.0 sent during the same minute with average (‘avg’) aggregation type will be represented by a single point on a graph, with value 2.0 for that minute.

Filter1 and filter2 values can be used to organize complex metrics. For example, one can use them to track the number of registered users by gender and account type. In registered users example data points can be specified as:

```
{
   "name" : "registered-users-count",
   "value" : 42,
   "aggregation" : "sum",
   "filter1" : "user.gender=male",
   "filter2" : "account.type=free"
}

{
   "name" : "registered-users-count",
   "value" : 24,
   "aggregation" : "sum",
   "filter1" : "user.gender=female",
   "filter2" : "account.type=free"
}

{
   "name" : "registered-users-count",
   "value" : 10,
   "aggregation" : "sum",
   "filter1" : "user.gender=female",
   "filter2" : "account.type=paid"
}
```

In reports, the selected values in one filter are combined with an ‘OR’, while selected values in different values are ‘ANDed’. For example, to display the number of paying female users, in filter1 the value ‘female’ should be selected, and in filter2 value ‘paid’ should be selected.

We recommend to use 'parameter.name=value' format to define filter values.

Counting Metrics

Each SPM plan allows for a different number of metrics and a different number of datapoints per calendar month. Please see SPM Plans page.

To stay within your plan limits pay attention to the number of distinct metrics. A metric is a (metric name, filter1, filter2) triple. For example:

- if you are sending 10 distinct metric names, 5 distinct values for filter1, and 2 distinct values for filter2, you will be using a total of 10*5*2=100 metrics
- if you are sending 10 distinct metric names, 5 distinct values for filter1, and 10 distinct values for filter2, you will be using a total of 10*5*10=500 metrics

Max Data Input Rate

The API allows up to 3,600,000,000 (3.6 billion) data points per hour. When this limit is exceeded a 429 HTTP response code is returned.

Custom Metrics API

SPM provides REST API for sending data points for custom metrics:

http://spm-receiver.sematext.com/receiver/custom/receive.[format]?token=[spm app token]

The format can be either 'json' or 'raw'.

Applications' tokens are listed at https://apps.sematext.com/ui/monitoring

JSON format

URL: http://spm-receiver.sematext.com/receiver/custom/receive.json?token=[spm app token]

Content-type: application/json

Method: POST

JSON body:

Field name	Field type	Constraints
datapoints	array of ‘data point’	elements count <= 100

Data point:

Field name	Description	Field type	Constraints	Required
timestamp	Time in milliseconds since “Epoch”	Long		No
name	Metric name	String	length > 0 and <= 255	Yes
value	Data point value	Double		Yes
aggregation	Aggregation type	String		

“min”, “max”, “avg”, “sum”

Yes

filter1

Value for filter1

String

length > 0 and <= 255

No

filter2

Value for filter2

String

length > 0 and <= 255

No

Example:

```
{
   "datapoints" : [
      {
         "timestamp" : 1369669889927,
         "name" : "registered-users-count",
         "value" : 1,
         "aggregation" : "sum",
         "filter1" : "user.gender=male",
         "filter2" : "account.type=free"
      },
      {
         "name" : "registered-users-count",
         "value" : 1,
         "aggregation" : "sum",
         "filter1" : "user.gender=female",
         "filter2" : "account.type=paid"
      }
 ]
}
```

To send metrics using curl, save example above to a file named datapoints.json and submit it with the following call, using your own token:

```
curl -v -H 'Content-type: application/json' -d @datapoints.json http://spm-receiver.sematext
```

One-liner example:

```
curl -H 'Content-type: application/json' -d '{"datapoints" : [{"name": "registered-users", '
```

Raw format

URL: http://spm-receiver.sematext.com/receiver/custom/receive.raw?token=[spm app token]

Content-type: text/plain

Method: POST

Each line is tab delimited sequence of fields:

(timestamp)\t(name)\t(value)\t(aggregation type)\t(filter1)\t(filter2)

Lines are separated by new line character (‘\n’). Limit is 100 lines per request.

Field name	Description	Field type	Constraints	Required
timestamp	Time in milliseconds since “Epoch”	Long		No
name	Metric name	String	length > 0 and <= 255	Yes
value	Data point value	Double		

Yes

aggregation

Aggregation type

String

“min”, “max”, “avg”, “sum”

Yes

filter1

Value for filter1

String

length > 0 and <= 255

No

filter2

Value for filter2

String

length > 0 and <= 255

No

Example:

```
1369671381221 registered-users-count  1   sum user.gender=male    account.type=free
1369671381221 registered-users-count  1   sum user.gender=female  account.type=paid
```

To post these data using curl, save the example above (ensure tabs are preserved) to a file named datapoints.raw and use the following call with your own token:

```
curl -v -H 'Content-type: text/plain' -d @datapoints.raw http://spm-receiver.sematext.com/re
```

One-liner example:

```
curl -H 'Content-type: text/plain' -d "`echo -e "\tregistered-users\t1\tsum"`" http://spm-re
```

Java API

Sematext-metrics is an open source Java library for sending custom metrics from Java applications. Please refer to README for a quick start.

Coda Hale (aka Yammer) Metrics Reporter

Sematext-metrics-reporter is a Coda Metrics library Reporter for sending metrics to SPM that uses sematext-metrics under the hood.

Ruby API

Sematext-metrics is a gem for sending custom metrics from Ruby applications.

Node.js API

spm-metrics-js is an npm module for sending custom metrics from Node.js applications. Github repo: https://github.com/sematext/spm-metrics-js.

.Net API

Sematext-Metrics is an open source library for sending custom metrics from a .Net application. It is a .Net Standard 2.0 library so it will work with both .Net Core and .Net Framework applications.