Factsheet

APIs – Data API

Data API

Data API

The Data API provides read-only access to raw and aggregated metric results, similar to that which you might find in SamKnows One's charting, in addition to the raw test results collected from our various measurement agents.

It can be used in two primary ways, through the SamKnows One UI, where you can then download CSVs of results; or the API directly for clients who wish to integrate our data into their internal platforms or automated systems.

SamKnows One UI

Within SamKnows One you have both a request builder, links to this documentation, and the ability to manage the Data API authentication key for your organisation.

The request builder allows you to select which of the three types of data you wish to export, then use the normal SamKnows One chart builder in order to either produce downloadable CSVs, or example requests for using the API directly, outside of the SamKnows One UI.

Types of data

We have two different types of data you can export using the Data API. Metric data and Test Data.

- Metric data is very similar to the data you might get if you do an export CSV from SamKnows One's charting functionality. It uses our specially derived metrics and with it you can fully utilise much of the power of SamKnows One including aggregation giving you averages, standard deviations, percentiles, confidence interval values and whilst also being able to use functionality such as prefiltering, normalisation, splitting data into multiple series and advanced filtering. You can view this as either unaggregated results (raw) or in aggregated form.
- Test data is the detailed raw unfiltered and unprocessed output from the tests that execute on various measurement agents (Whitebox, CPE, Mobile and Web). This data can be useful for doing a deep-dive into the data and is significantly more detailed but also a lot more complex to use.

We also have various metadata endpoints to aid your use of the API such as the various metrics and tests accessible to you, and the various pieces of metadata you can filter or split data by.

Development

Developer documentation is available below. URLs for production and development purposes will be provided upon commencement of a project. Authentication tokens are managed through the SamKnows One UI.

Performance and capacity

We actively manage capacity planning to ensure that the API can handle as many simultaneous connections/sessions as required by our clients.

Basic analytics queries (including the Data API and in SamKnows One web interface) for small date ranges and small data sets will complete within a few hundred milliseconds in almost all situations. Higher spikes in load

can occasionally cause higher response times but this is mitigated by careful monitoring and extensive capacity planning. Query times depend on the amount of measurement agents reporting and the date range being requested.

Logging

All analytics queries are logged internally by SamKnows for auditing and analytics purposes.

Overview

Metric data endpoints

- Metrics Data Endpoint (JSON) [POST /metric_data] - Fetch data for specific metrics
- Metrics Data Endpoint (CSV) [POST /metric_data.csv] - Generate a CSV and return a link for above endpoint
- Test Data Endpoint (JSON) [POST /test_data] - Fetch raw test data
- Test Data Endpoint (CSV) [POST /test_data.csv] - Generate a CSV and return a link for above endpoint

Metadata endpoints

- Metrics endpoint [GET /metrics] - Get a list of all metrics
- Tests endpoint [GET /tests] - Get a list of all tests
- Splittables endpoint [GET /splits] - Get a list of everything you can split by when looking at metric data
- Filterables endpoint [GET /filters] - Get a list of everything you can filter by, and links for auto-complete values when looking at metric data

Metric data endpoint

Request [POST /metric_data]

```
+ Request (application/json)
    + Headers
        Authorization: {token}
        Accept: application/json
    + Body
        {
          "metric": "httpget",
          "chartType": "aggregate",
          "aggregation": "hourly",
          "normalised": true,
          "prefilter": false,
          "splits": [
            "package"
          ],
          "filters": [
            {
              "id": "package",
              "filterType": "1",
              "filterValues": [
                "6737",
                "6659"
              ]
            }
          ],
          "time": {
            "from": "2017-05-04 23:00",
            "to": "2017-05-07 04:00"
          }
        }
```

(Please see page 8)

All fields are documented in more depth below.

Aggregated chart type response (200)

```
{
    "code": "OK",
    "message": "Request successful",
    "data": [
     {
      "metricData": [
       {
        "metricValue": 23031.16,
        "dtime": "2018-01-03T00:00:00",
        "stdDev": 5718.58,
        "sampleCount": 159,
        "unitCount": 152,
        "ci90": 594.41,
        "ci95": 909.12,
        "ci99": 1194.75,
        "min": 10296,
        "max": 43483,
        "percentiles": {
         "99": 38968,
         "1": 12036,
         "25": 18734,
         "20": 18734,
         "5": 14526,
         "90": 30505,
```

```
            "50": 22667,
            "95": 33040,
            "75": 26641,
            "80": 26641,
            "10": 16054
           },
          },
          {
           "metricValue": 22738.24,
           "dtime": "2018-01-03T01:00:00",
           "stdDev": 5526.6,
           "sampleCount": 168,
           "unitCount": 164,
           "ci90": 553.04,
           "ci95": 845.85,
           "ci99": 1111.6,
           "min": 10097,
           "max": 39356,
           "percentiles": {
            "99": 38159,
            "1": 11250,
            "25": 19283,
            "20": 19283,
            "5": 14559,
            "90": 29712,
            "50": 22090,
            "95": 32495,
            "75": 26193,
            "80": 26193,
            "10": 15496
           },
          }
         ]
        }
       ]
      }
```


Within the metric data blob you will see the following key-value pairs:
(Please see pages 8-9)

The units vary from metric to metric, and this data is provided on the metrics endpoint, for example the httpget and httppost metrics that produce upload/download speeds are measured in bytes per second.

Raw data chart type response (200)


```
{
    "code": "OK",
    "message": "Request successful",
    "data": [
     {
      "metricData": [
       {
        "metricValue": 27426340,
        "dtime": "2018-01-08T00:00:20",
        "agent": "06a68b00-2c35-11a7-b11a-0aa47a6ababd",
        "unit": "827829"
       },
       {
        "metricValue": 27427038,
        "dtime": "2018-01-08T00:00:38",
        "agent": "06a68b00-2c35-11a7-b11a-0aa47a6ababd",
        "unit": "827829"
       }
      ]
     }
    ]
   }
```


Within the metric_data blob you will see the following key value pairs:
(Please see page 9)

Metrics

We have a huge variety of metrics available extracting useful data from our test results. Some metrics are only available when aggregated. Some metrics also have additional filters and splits available (such as httpget and httppost which you can filter and split by thread count or IP version). All the available metrics are detailed on the metrics endpoint.

Normalisation

Normalisation functionality is available when retrieving aggregate time series data. This is useful when plotting results by hour, but the test schedule does not run test results every hour across every agent. Without normalisation, users will often see a saw-toothing effect, with some hours having measurement agents with much better results than other hours simply due to agent distribution. Normalisation forces our analytics engine to look at the underlying test schedule for the metric in question, and to normalise results across the blocks of hours

that the tests should run over. This results in an even number of measurement agents and results being represented for all hours of the day, thus removing the spikes in the charts.

Filtering

You can filter the outputted data in a number of ways:

- Date / Time
- Prefilters
- Custom defined metadata
- Standardised metadata

All filters except date and time ranges can be inclusive (include only those specified) or exclusive (exclude those specified). Date and time ranges are always inclusive (only the range you specify).

Date time

You must specify at least the date range over which you wish to retrieve data from. Datetimes used in to/from fields can be described in the following formats, all dates and times refer to measurement agents' local time. They will be used exactly as stated with no rounding. The stated times would be therefore be included.

- YYYY-MM-DD HH:MM:SS (e.g. 2018-01-01 10:15:30)
- YYYY-MM-DD HH:MM (e.g. 2018-01-01 10:00)

Prefilters

Our prefilter functionality uses a database of manually specified prefilters that preclude metric results that are extremely unlikely or impossible in addition to utilising our machine learning predictive algorithms and models in order to try and filter out data which may be considered to be unreliable or 'bad' data.

Standardised metadata

There are a number of fields that are always available to filter by, for routers & Whiteboxes they are:

Field	Description
target_set	The SamKnows or client-specified target groups
package	The specified package/product the agent is on
country	The ability to filter down to results from a particular country
target	A specific target / test node
agent	Filtering down test results by Agent ID
unit	Filtering down test results by Unit ID
isp	The unit's associated ISP
base	The device model

For mobile they are:

Field	Description
model	Phone model
geo_country	The country we identify the device as being in
manufacturer	The device manufacturer
connection_type	Wi-Fi or Cellular, which connection type was used during the test
cellular_technology	The current cellular technology available during the test run e.g. 3G / 4G
carrier_name	The carrier name
operating_system_version	The version of the OS (Android or iOS version)

Chart types

The valid chart types are:

- raw - This will give you a row/object per data point
- aggregate - This will allow you to do aggregation, most commonly aggregating by an element of time

Aggregations

If you have specified aggregate as your chart type then you may also specify an aggregation.

The following aggregations are available that aggregate by time in chronological order:

- hourly
- two_hourly

- four_hourly
- six_hourly
- daily
- weekly
- monthly
- quarterly

The following aggregations are available that use an element of time (e.g. the date or the hour of the day)

- hour_of_day
- day_of_week
- day_of_month

The following other aggregations are available:

- total - This aggregates everything in a split group into a single objects/rows

Splits

Splits allow you to, when aggregating, separate out data into different data series. So for example, if you are doing an hour of day aggregation and split by package you might see:

- Time: 4pm - 5pm, Package: 38x10, Average: 37.4
- Time: 4pm - 5pm, Package: 76x25, Average: 75.4
- Time: 5pm - 6pm, Package: 38x10, Average: 37.1
- Time: 5pm - 6pm, Package: 76x25, Average: 74.1
- Time: 6pm - 7pm, Package: 38x10, Average: 36.2
- Time: 6pm - 7pm, Package: 76x25, Average: 74.2
- Time: 7pm - 8pm, Package: 38x10, Average: 36.3
- Time: 7pm - 8pm, Package: 76x25, Average: 74.3
- Time: 8pm - 9pm, Package: 38x10, Average: 37.1
- Time: 8pm - 9pm, Package: 76x25, Average: 75.1

You can find out the metadata you can split by from the split metadata endpoint.

Test data endpoint

Request [POST /test_data]

+ Request (application/json)

+ Headers

Authorization: {token}
Accept: application/json

+ Body

```
{
   "test": "httpget",
   "date": {
      "from": "2017-05-04",
      "to": "2017-05-07"
   }
}
```

FieldDescriptionExampletestThis is the key that refers to the test you wish to retrieve data ondnsdateThe date range you wish to retrieve data for (UTC)"from": "2018-01-01","to": "2018-01-05"

You can find the value for the panel id in SamKnows One. This essentially corresponds to your data source / data subscription. Other fields are documented in more depth below.

Response 200


```
{
    "code": "OK",
    "message": "Request successful",
    "data": [
     {
         "unit_id": 32477,
         "agent_id": "91f7f827-2c33-11e7-
b97c-0cc47a6beaccc",
         "dtime": "2018-01-03 12:23:30.000",
         "base": "skwb8",
         "ddate": "2018-01-03",
         "target": "n11-the1.samknows.com",
         "address": "77.75.107.173",
         "fetch_time": 10000503,
         "bytes_total": 205720256,
         "bytes_sec": 20570991,
         "bytes_sec_interval": 20570991,
         "warmup_time": 5000238,
         "warmup_bytes": 96839378,
```

```
        "sequence": 0,
        "threads": 1,
        "tcp_retransmissions": 36,
        "successes": 1,
        "failures": 0,
        "ip_version": 4,
        "target_group": "Off-net"
      }
    ]
  }
```


Within the data objects you will see all the various columns and values we store. These values will vary from test to test, and you can find descriptions and tests available provided on the tests endpoint.

The following fields you will see across many tests and are therefore not documented per test:

FieldDescriptionExamplebaseThe device modelskwb8ddateThe date the test took place on2018-01-01dtimeThe datetime the test ended2018-01-03 12:23:30.000agent_idThe agent ID of the reporting agent06a68b00-2c35-11a7-b11a-0aa47a6ababdunit_idThe unit ID of the reporting agent827829

Date range filtering

Date and time ranges are always inclusive (only the range you specify). You must specify a date range. Data Formats: Dates used in to/from fields can be described in any of the following formats, they must always be UTC. When used a from, it will be rounded down (e.g. 2018-01 would include from 2018-01-01 00:00:00). When used as a to, it will be rounded up (e.g. 2018-04 would include up to 2018-04-31 23:59:59). The stated dates/days would therefore be included.

- YYYY-MM-DD (e.g. 2018-01-01)
- YYYYMMDD (e.g. 20180101)
- YYYY-MM (e.g. 2018-01)
- YYYY (e.g. 2018-01-01)
- +x days (e.g. +5 days meaning 5 days in the future)
- -x days (e.g. -5 days meaning 5 days ago)
- last|next|this DAY (e.g. last Monday)
- now

Metadata endpoints

Request [GET /metrics]

This will get an array of the metrics (for metric data) available to you and their units.

+ Request (application/json)

    + Headers

      Authorization: {token}
      Accept: application/json

Response 200


```
{
    "code": "OK",
    "message": "Request successful",
    "data": [
      {
        "key": "httpget",
        "unit": "mbps"
      },
      {
        "key": "httpget_retrans",
        "unit": "count"
      },
      {
        "key": "jitterDown",
        "unit": "ms"
      },
      {
        "key": "dnsResponse",
        "unit": "ms"
      },
      {
        "key": "dnsFail",
        "unit": "%"
      },
      {
        "key": "webgetAvg",
        "unit": "s"
      }
    ]
  }
```


Request [GET /tests]

This will get an array of the tests (for test data) available to you and their measurement agents.

+ Request (application/json)

+ Headers

Authorization: {token}
Accept: application/json

Response 200 (for units)

```
{
    "code": "OK",
    "message": "Request successful",
    "data": [
      "httpget",
      "httppost",
      "dns",
      "netflix",
      "ping",
      "jitter",
      "latency",
      "webget",
      "youtube",
      "network_usage",
      "traceroute"
    ]
  }
```

Response 200 (for mobile)

```
{
    "code": "OK",
    "message": "Request successful",
    "data": [
      "mobile_download",
      "mobile_latency",
      "mobile_upload",
      "mobile_www",
      "mobile_youtube"
    ]
  }
```

CSVs

Examples throughout this document reference the JSON format, but by appending .csv to the endpoint you can receive raw CSV output:

- Metrics Data Endpoint (CSV) [POST /metric_data.csv] - Generate a CSV and return a link for above endpoint
- Test Data Endpoint (CSV) [POST /test_data.csv] - Generate a CSV and return a link for above endpoint

If you would prefer, we can upload the CSV to Amazon AWS S3 and you can download it from there. To do so, use one of the following two endpoints:

- Metrics Data Endpoint (JSON) [POST /metric_data_download] - Fetch data for specific metrics
- Test Data Endpoint (JSON) [POST /test_data_download] - Fetch raw test data

Metric data endpoint

Request [POST /metric_data]:

Field	Description	Type	Example	Required?
metric	The key for the metric you wish to fetch. All the available metrics are detailed on the metrics endpoint.	string	dnsFail	Yes
chartType	The chart type, raw or aggregate.	string	aggregate	Yes
aggregation	If you selected an aggregate chart type, what to aggregate by	string	hourly	Required for aggregate
normalised	Toggle for enabling normalisation, see below for more information.	boolean	true	Yes
prefilter	Toggle for enabling prefiltering, see below for more information.	boolean	true	Yes
splits	Array of pieces of metadata to split by	list	["package","isp"]	No
filters	Filters are detailed in more depth below, please see there for more information	list of objects	[{"id": "package","filterType": "1","filterValues": ["6737"]}]	No
time	The local time range to return data from	object consisting of to/from strings	{"from": "2017-05-04 15:00","to": "2017-05-07 03:00"}	No

Field	Description	Example
metricValue	The primary metric value, it is usually an average for most metrics	74768.46
dtime	The datetime representation of the beginning of the period represented by this data point (time series aggregates only)	2018-01-03T05:00:00 or 2018-01-03
aggregatedNumber	When aggregating by something that's not time e.g. hour of day, this is the value of the group	1
stdDev	Population Standard Deviation	9620041.6
sampleCount	Sample count (of tests)	1016
agentCount	Number of measurement agents that reported contributing results	663
ci90	90th Confidence Interval	478783.08
ci95	95th Confidence Interval	732278.46

Field	Description	Example
ci99	99th Confidence Interval	962348.39
min	Minimum	0
max	Maximum	117136375
median	Median	4671160
percentiles	Array of common percentile values	"99": 38159, "1": 11250, "25": 19283, "5": 14559, "90": 29712, "50": 22090, "95": 32495, "75": 26193, "10": 15496
{split}	Columns containing the values of data when it has been split	38x10

Field	Description	Example
metricValue	The raw metric value. See Metric Types	74768.46
dtime	The datetime the test ended	2018-01-03T00:00:00
agent	The agent ID of the reporting agent	06a68b00-2c35-11a7-b11a-0aa47a6ababd
unit	The unit ID of the reporting agent	827829
{split}	Columns containing the values of data when it has been split	38x10