traceview Documentation

Release 0.7.0

Dan Riti

June 13, 2016

Contents

1 Installation **3**

2 Quick Start **5**

3 API Documentation **7**

3.1 Latency 13

3.2 Total Requests 16

3.3 Formatters 16

Python Module Index **19**

Release v0.7.0.

Library for providing access to the TraceView API v2.

Please see the TraceView API Reference for more information.

Supports Python 2.7, 3.3, 3.4, 3.5

CHAPTER 1

Installation

To install python-traceview, simply:

```
$ pip install python-traceview
```

CHAPTER 2

Quick Start

Accessing information from TraceView is very simple.

Begin by importing the `traceview` module:

```
>>> import traceview
```

Now, let's initialize a TraceView object using your TraceView access (API) key. You can find this key on the **Organization Overview** page in TraceView:

```
>>> tv = traceview.TraceView('API KEY HERE')
```

Now, we have a `TraceView` object called `tv`. We can get all the information we need from this object.

For example, let's get all available applications setup within your TraceView account:

```
>>> tv.apps()
[u'Default', u'pyramid_web_app']
```

Nice, right? We can also get a server side latency summary for the `Default` application:

```
>>> tv.server.latency_summary(app='Default', time_window='hour')
{u'count': 2746.0, u'average': 213911.87181354698, u'latest': 35209.87654320987}
```

TraceView has traced 2746 requests in the last hour, with an average latency of 213ms. That's all well and good, but it's also only the start of what information you can get from TraceView.

CHAPTER 3

API Documentation

class `traceview.`**`TraceView`**(*api_key*, *formatter=None*)

The `TraceView` object.

Provides access to TraceView API resources.

Parameters

- **api_key** – The TraceView API access key.
- **formatter** (*func*) – (optional) Function to format API results. See the module `traceview.formatters`.

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
```

`actions`()

Get all actions that have been traced.

Returns all actions traced.

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.actions()
[u'admin', u'products', u'blog', u'settings', u'logout']
```

`annotation`(*message*, **args*, ***kwargs*)

Create an annotation.

Annotations are used to log arbitrary events into TraceView, which are used to understand the correlation between system events (i.e. code release, server restarts, etc) and performance trends.

Parameters

- **message** (*str*) – The annotation message.
- **appname** (*str*) – (optional) The application to associate the annotation with.
- **hostname** (*str*) – (optional) The host to associate the annotation with.
- **username** (*str*) – (optional) The user name to associate the annotation with.
- **layer** (*str*) – (optional) The layer name to associate the annotation with.

- **time** (*str*) – (optional) The time to associate the annotation with, in seconds since the epoch.

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.annotation('Code deployed', appname='production_web')
```

annotations (*appname=None*, **args*, ***kwargs*)

Get annotations.

Annotations are used to log arbitrary events into TraceView, which are used to understand the correlation between system events (i.e. code release, server restarts, etc) and performance trends.

The default time window is one week.

Parameters

- **appname** (*str*) – (optional) The application name to filter annotations by.
- **time_start** (*str*) – (optional) The start time for the time window, in milliseconds since the epoch.
- **time_end** (*str*) – (optional) The end time for the time window, in milliseconds since the epoch.

Return type list

Usage:

```
>>> import pprint
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> pprint.pprint(tv.annotations(appname='production_web'))
[{u'app': 3,
  u'host': u'prod-web.example.com',
  u'id': 123,
  u'message': u'Code deployed',
  u'time': 1409685758,
  u'username': u'dan'},
  ...
]
```

apps ()

Get all available applications.

Returns all available applications

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.apps()
[u'Default', u'flask_app']
```

assign (*hostname*, *appname*, **args*, ***kwargs*)

Assign a host to an existing application.

Please note that you cannot join host names to the *Default* application, as all hosts start there.

Parameters

- **hostname** (*str*) – The host name to assign to the application.
- **appname** (*str*) – The existing application name.
- **layer** (*str*) – (optional) The layer name to assign to the application.

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.assign(hostname='web-app-1234', appname='production_web')
```

browsers()

Get all browsers used by end users.

Returns all browsers used by end users

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.browsers()
[u'Chrome', u'Firefox', u'Links', u'Safari', u'Wii']
```

client = None

Get `Client` latency information.

controllers()

Get all controllers that have been traced.

Returns all controllers traced

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.controllers()
[u'admin', u'products', u'blog', u'settings', u'logout']
```

delete(*host_id*, **args*, ***kwargs*)

Deprecated since version 0.6.0: Use `delete_host` instead.

Delete an existing host.

Parameters **host_id** (*str*) – The id of the host to delete.

Returns indicates if host was successfully deleted.

Return type boolean

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.delete(host_id='123')
True
```

delete_app(*app_name*, **args*, ***kwargs*)

Delete an existing app.

Parameters **app_name** (*str*) – The name of the app to delete.

Returns indicates if app was successfully deleted.

Return type boolean

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.delete_app(app_name='APP_123')
True
```

delete_host (*host_id*, **args*, ***kwargs*)

Delete an existing host.

Parameters **host_id** (*int*) – The id of the host to delete.

Returns indicates if host was successfully deleted.

Return type boolean

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.delete_host(host_id=123)
True
```

domains ()

Get all domains that have been traced.

Returns all domains traced

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.domains()
[u'example.com', u'www.example.com', u'mail.example.com']
```

error_rates (*app*, **args*, ***kwargs*)

Get the error rate for an application.

Each item in the items list is a pair of values (timestamp, error_rate). The error rate describes the number of traces with one or more errors, per total number of traces.

Parameters **app** (*str*) – The application name.

Returns timeseries data of the application's error rate

Return type dict

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.error_rates('Default')
{u'fields': u'timestamp,error_rate', u'items': [[1399082880.0, 0], [1399082910.0, 0], ...]}
```

hosts (*appname=None*, **args*, ***kwargs*)

Get all hosts that have been traced.

Parameters **appname** (*str*) – (optional) The application name to filter hosts by.

Returns all hosts traced

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.hosts()
[{u'last_trace': None, u'last_heartbeat': 1429033545, u'first_heartbeat': 1428060977, u'name
```

instrumentation(*host_id*)

Get instrumentation version information for a host.

Parameters **host_id** (*str*) – The id of the host.

Returns instrumentation version information for a host

Return type list

Usage:

```
>>> import pprint
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> pprint.pprint(tv.instrumentation(host_id=1))
[{u'name': u'tracelyzer',
  u'release_date': 1374537600,
  u'update_required': True,
  u'version': u'1.1.1'},
  ...
]
```

layers(*app*, **args*, ***kwargs*)

Get all recent layers for an application.

The default time window for reported layers is 1 day.

Parameters

- **app** (*str*) – The app name to list layers.
- **since_time** (*int*) – (optional) The start of the time window as a UTC timestamp in milliseconds.

Returns all available apps

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.layers('Default')
[u'PHP', u'cURL', u'lighttpd', u'php_mysql', u'php_mysqli']
```

licenses()

Get the current number of hosts reporting traces and the number of hosts licensed to the organization.

Returns licensing information for organization.

Return type dict

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.licenses()
{u'hosts_used': 5, u'hosts_limit': 10}
```

metrics()

Get all available host metrics that have been collected.

Returns all avaiable host metrics being collected.

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.metrics()
[u'cpu_user_frac:all', u'load', u'mem_apps', u'mem_cached', u'mem_swap', u'mem_totalused', .
```

organization()

Get organization information.

Returns organization information

Return type dict

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.organization()
{u'fullname': u'the example organization', u'name': u'example'}
```

regions()

Get all geographical region codes of end users.

Regions codes are ISO 3166-1 and ISO 3166-2 codes for all regions collected in RUM. Currently, country codes (ISO-3166-1) are available worldwide, and state codes (ISO-3166-2) are available in the US and Canada.

Returns all geographical region codes of end users

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.regions()
[u'CA', u'CA-BC', u'MX', u'RU', u'US', u'US-RI', ...]
```

server = None

Get `Server` latency information.

total_requests = None

Get `TotalRequests` information.

users()

Get user information.

Returns user information

Return type list

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.users()
[{u'admin': True, u'name': u'Jane Doe', u'email': u'jdoe@example.com'}, { ... }]
```

3.1 Latency

class `traceview.latency.`**`Client`**(*api*)

The `Client` class.

Get client side latency information.

`latency_series`(*app*, **args*, ***kwargs*)

Get a timeseries line of the applications latency and volume.

Each timeseries point is a triple of (timestamp, volume, latency).

Parameters

- **app** (*str*) – The app name.
- **time_window** (*str*) – (optional) The time window ('hour', 'day', or 'week') to filter on.
- **time_end** (*str*) – (optional) The end time for the time window.
- **domain** (*str*) – (optional) The domain to filter on.
- **url** (*str*) – (optional) The url path to filter on.
- **layer** (*str*) – (optional) The application layer to filter on.
- **controller** (*str*) – (optional) The controller to filter on.
- **action** (*str*) – (optional) The action to filter on.
- **browser** (*str*) – (optional) The browser to filter on.
- **region** (*str*) – (optional) The region to filter on.

Returns timeseries data of the application latency and volume

Return type dict

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.client.latency_series('Default')
{u'fields': u'timestamp,volume,avg_latency', u'items': [[1399090230.0, 0, None], ...]}
```

`latency_summary`(*app*, **args*, ***kwargs*)

Get a summary of the latency and volume traced.

Parameters

- **app** (*str*) – The app name.
- **time_window** (*str*) – (optional) The time window ('hour', 'day', or 'week') to filter on.
- **time_end** (*str*) – (optional) The end time for the time window.
- **domain** (*str*) – (optional) The domain to filter on.

- **url** (*str*) – (optional) The url path to filter on.
- **layer** (*str*) – (optional) The application layer to filter on.
- **controller** (*str*) – (optional) The controller to filter on.
- **action** (*str*) – (optional) The action to filter on.
- **browser** (*str*) – (optional) The browser to filter on.
- **region** (*str*) – (optional) The region to filter on.

Returns timeseries data of the application latency and volume

Return type dict

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.client.latency_summary('Default')
{u'count': 93.0, u'average': 14503082.720430108, u'latest': None}
```

class `traceview.latency.`**`Server`**(*api*)

The `Server` class.

Get server side latency information.

`latency_by_layer`(*app*, **args*, ***kwargs*)

Get timeseries data grouped by application layers.

Parameters

- **app** (*str*) – The app name.
- **time_window** (*str*) – (optional) The time window ('hour', 'day', or 'week') to filter on.
- **time_end** (*str*) – (optional) The end time for the time window.
- **domain** (*str*) – (optional) The domain to filter on.
- **url** (*str*) – (optional) The url path to filter on.
- **layer** (*str*) – (optional) The application layer to filter on.
- **controller** (*str*) – (optional) The controller to filter on.
- **action** (*str*) – (optional) The action to filter on.

Returns timeseries data of the application latency and volume

Return type dict

Usage:

```
>>> import pprint
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.server.latency_by_layer('Default')
>>> pprint.pprint(tv.server.latency_by_layer('Default'))
[{u'layer': u'PHP',
  u'timeseries': {u'fields': u'timestamp,volume,avg_latency',
                  u'items': [[1399089540.0, 10, 0], ...]}},
 ...
]
```

latency_series (*app*, **args*, ***kwargs*)

Get a timeseries line of the applications latency and volume.

Each timeseries point is a triple of (timestamp, volume, latency).

Parameters

- **app** (*str*) – The app name.
- **time_window** (*str*) – (optional) The time window ('hour', 'day', or 'week') to filter on.
- **time_end** (*str*) – (optional) The end time for the time window.
- **domain** (*str*) – (optional) The domain to filter on.
- **url** (*str*) – (optional) The url path to filter on.
- **layer** (*str*) – (optional) The application layer to filter on.
- **controller** (*str*) – (optional) The controller to filter on.
- **action** (*str*) – (optional) The action to filter on.

Returns timeseries data of the application latency and volume

Return type dict

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.server.latency_series('Default')
{u'fields': u'timestamp,volume,avg_latency', u'items': [[1399089120.0, 27.0, 226074.07407407
```

latency_summary (*app*, **args*, ***kwargs*)

Get a summary of the latency and volume traced.

Parameters

- **app** (*str*) – The app name.
- **time_window** (*str*) – (optional) The time window ('hour', 'day', or 'week') to filter on.
- **time_end** (*str*) – (optional) The end time for the time window.
- **domain** (*str*) – (optional) The domain to filter on.
- **url** (*str*) – (optional) The url path to filter on.
- **layer** (*str*) – (optional) The application layer to filter on.
- **controller** (*str*) – (optional) The controller to filter on.
- **action** (*str*) – (optional) The action to filter on.

Returns timeseries data of the application latency and volume

Return type dict

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.server.latency_summary('Default')
{u'count': 2402.0, u'average': 271437.13572023314, u'latest': 19530.61224489796}
```

3.2 Total Requests

class `traceview.total_request.`**`TotalRequests`**(*api*)

`series`(*app*, **args*, ***kwargs*)

Get the total requests for an application.

Each item in the items list is a pair of values (timestamp, total_requests). total_requests is the number of requests to your application during that time period.

Parameters

- **app** (*str*) – The application name.
- **time_window** (*str*) – (optional) The time window ('hour', 'day', or 'week') to filter on.
- **time_end** (*str*) – (optional) The end time for the time window.

Returns timeseries data of the application's total requests

Return type dict

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.total_requests.series('APP NAME HERE')
{u'fields': u'timestamp,total_requests', u'items': [[1444650840.0, 583.0], [1444650870.0, 59
```

`summary`(*app*, **args*, ***kwargs*)

Get a summary of the applications total requests.

Parameters

- **app** – The application name.
- **time_window** (*str*) – (optional) The time window ('hour', 'day', or 'week') to filter on.
- **time_end** (*str*) – (optional) The end time for the time window.

Returns summary of the application's total requests

Return type dict

Usage:

```
>>> import traceview
>>> tv = traceview.TraceView('API KEY HERE')
>>> tv.total_requests.summary('APP NAME HERE')
{u'reqs_per_time_period': u'19.61/sec', u'total_requests': 70579.0}
```

3.3 Formatters

`traceview.formatters`

This module contains functions used to format TraceView API results.

`traceview.formatters.`**`tuplify`**(*results*, *class_name='Result'*)

Formats API results into `namedtuple` objects. Supports tuplifying results that are either timeseries data or objects (dicts).

Parameters

- **results** – TraceView API results.
- **class_name** (*str*) – (optional) Prefix string for name of the namedtuple.

Usage:

```
>>> import traceview
>>> from traceview.formatters import tuplify
>>> tv = traceview.TraceView('API KEY HERE', tuplify)
>>> tv.total_requests.summary('APP NAME HERE')
ResultTuple(reqs_per_time_period=u'19.53/sec', total_requests=70293.0)
```

t

traceview.formatters, 16

A

actions() (traceview.TraceView method), 7
annotation() (traceview.TraceView method), 7
annotations() (traceview.TraceView method), 8
apps() (traceview.TraceView method), 8
assign() (traceview.TraceView method), 8

B

browsers() (traceview.TraceView method), 9

C

Client (class in traceview.latency), 13
client (traceview.TraceView attribute), 9
controllers() (traceview.TraceView method), 9

D

delete() (traceview.TraceView method), 9
delete_app() (traceview.TraceView method), 9
delete_host() (traceview.TraceView method), 10
domains() (traceview.TraceView method), 10

E

error_rates() (traceview.TraceView method), 10

H

hosts() (traceview.TraceView method), 10

I

instrumentation() (traceview.TraceView method), 11

L

latency_by_layer() (traceview.latency.Server method), 14
latency_series() (traceview.latency.Client method), 13
latency_series() (traceview.latency.Server method), 14
latency_summary() (traceview.latency.Client method), 13
latency_summary() (traceview.latency.Server method), 15
layers() (traceview.TraceView method), 11
licenses() (traceview.TraceView method), 11

M

metrics() (traceview.TraceView method), 12

O

organization() (traceview.TraceView method), 12

R

regions() (traceview.TraceView method), 12

S

series() (traceview.total_request.TotalRequests method), 16
Server (class in traceview.latency), 14
server (traceview.TraceView attribute), 12
summary() (traceview.total_request.TotalRequests method), 16

T

total_requests (traceview.TraceView attribute), 12
TotalRequests (class in traceview.total_request), 16
TraceView (class in traceview), 7
traceview.formatters (module), 16
tuplify() (in module traceview.formatters), 16

U

users() (traceview.TraceView method), 12