traceview Documentation

Release 0.2.0

Dan Riti

May 04, 2014

Contents

1 Installation **3**

2 Quick Start **5**

3 API Documentation **7**

3.1 Latency 11

Release v0.2.0.

Library for providing access to the TraceView API v2.

Please see the TraceView API Reference for more information.

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*)

The `TraceView` object.

Provides access to TraceView API resources.

Parameters **api_key** – The TraceView API access key.

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')
```

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']
```

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()

Get all hosts that have been traced.

Returns all hosts traced

Return type list

Usage:

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

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']
```

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.

users()

Get user information.

Returns user information

Return type list

Usage:

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

3.1 Latency

class `traceview.latency.`**`Client`**(**args*, ***kwargs*)

The `Client` class.

Get client side latency information.

Parameters **api_key** – The TraceView API access key.

`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`**(**args*, ***kwargs*)

The `Server` class.

Get server side latency information.

Parameters **api_key** – The TraceView API access key.

`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}
```