

---

# **python\_moztelemetry Documentation**

*Release 0.3.9.13*

**Mozilla Firefox Data Platform**

**Mar 08, 2018**



---

# Contents

---

<b>1</b>	<b>API</b>	<b>3</b>
1.1	Dataset . . . . .	3
1.2	get_pings() (deprecated) . . . . .	5
1.3	Using Spark RDDs . . . . .	5
1.4	Experimental APIs . . . . .	5
<b>2</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>



A simple library to fetch and analyze data collected by the Mozilla Telemetry service. Objects collected by Telemetry are called `pings`. A ping has a number of properties (aka `dimensions`) and a payload. A session of Telemetry data analysis/manipulation typically starts with a query that filters the objects by one or more dimensions. This query can be expressed using either an orm-like api, *Dataset* or a simple function, *get\_pings()* (*deprecated*).



## 1.1 Dataset

**class** `moztelemetry.dataset.Dataset` (*bucket, schema, store=None, prefix=None, clauses=None, selection=None*)

Represents a collection of objects on S3.

A Dataset can have zero, one or many filters, which are refined using the *where* method. The result of refining a Dataset is a Dataset itself, so it's possible to chain multiple *where* clauses together.

The actual data retrieval is triggered by the *records* method, which returns a Spark RDD containing the list of records retrieved. To call *records* a SparkContext object must be provided.

Usage example:

```
bucket = 'test-bucket'
schema = ['submissionDate', 'docType', 'platform']

records = Dataset(bucket, schema) \
    .select(
        'clientId',
        os_name='environment.system.os.name',
        first_paint='payload.simpleMeasurements.firstPaint',
        // Take the first 2 stacks for each thread hang.
        stack_list='payload.threadHangStats[].hangs[].stack[0:2]'
    ).where(
        docType='main',
        appUpdateChannel='nightly',
        submissionDate=lambda x: x.startswith('201607'),
    ).records(sc)
```

For convenience Dataset objects can be created using the factory method *from\_source*, that takes a source name (e.g. 'telemetry') and returns a new Dataset instance. The instance created will be aware of the list of dimensions, available on its *schema* attribute for inspection.

**static from\_source()**

Create a Dataset configured for the given source\_name

This is particularly convenient when the user doesn't know the list of dimensions or the bucket name, but only the source name.

Usage example:

```
records = Dataset.from_source('telemetry').where(
    docType='main',
    submissionDate='20160701',
    appUpdateChannel='nightly'
)
```

**records** (*sc, limit=None, sample=1, seed=42, decode=None, summaries=None*)

Retrieve the elements of a Dataset

**Parameters**

- **sc** – a SparkContext object
- **limit** – maximum number of objects to retrieve
- **decode** – an optional transformation to apply to the objects retrieved
- **sample** – percentage of results to return. Useful to return a sample of the dataset. This parameter is ignored when *limit* is set.
- **seed** – initialize internal state of the random number generator (42 by default). This is used to make the dataset sampling reproducible. It can be set to None to obtain different samples.
- **summaries** – an iterable containing a summary for each item in the dataset. If None, it will be computed calling the summaries dataset.

**Returns** a Spark rdd containing the elements retrieved

**select** (*\*properties, \*\*aliased\_properties*)

Specify which properties of the dataset must be returned

Property extraction is based on JMESPath expressions. This method returns a new Dataset narrowed down by the given selection.

**Parameters**

- **properties** – JMESPath to use for the property extraction. The JMESPath string will be used as a key in the output dictionary.
- **aliased\_properties** – Same as properties, but the output dictionary will contain the parameter name instead of the JMESPath string.

**summaries** (*sc, limit=None*)

Summary of the files contained in the current dataset

Every item in the summary is a dict containing a key name and the corresponding size of the key item in bytes, e.g.:: { 'key': 'full/path/to/my/key', 'size': 200 }

**Parameters** **limit** – Max number of objects to retrieve

**Returns** An iterable of summaries

**where** (*\*\*kwargs*)

Return a new Dataset refined using the given condition

**Parameters** **kwargs** – a map of *dimension* => *condition* to filter the elements of the dataset. *condition* can either be an exact value or a callable returning a boolean value. If *condition* is a value, it is converted to a string, then sanitized.

## 1.2 get\_pings() (deprecated)

```
moztelemetry.spark.get_pings(sc, app=None, build_id=None, channel=None,
                             doc_type='saved_session', fraction=1.0, schema=None,
                             source_name='telemetry', source_version='4',
                             submission_date=None, version=None)
```

Returns a RDD of Telemetry submissions for a given filtering criteria.

### Parameters

- **sc** – an instance of SparkContext
- **app** – an application name, e.g.: “Firefox”
- **channel** – a channel name, e.g.: “nightly”
- **version** – the application version, e.g.: “40.0a1”
- **build\_id** – a build\_id or a range of build\_ids, e.g.: “20150601000000” or (“20150601000000”, “20150610999999”)
- **submission\_date** – a submission date or a range of submission dates, e.g.: “20150601” or (“20150601”, “20150610”)
- **source\_name** – source name, set to “telemetry” by default
- **source\_version** – source version, set to “4” by default
- **doc\_type** – ping type, set to “saved\_session” by default
- **schema** – (deprecated) version of the schema to use
- **fraction** – the fraction of pings to return, set to 1.0 by default

## 1.3 Using Spark RDDs

Both `Dataset` and `get_pings` return the data as a `Spark RDD`. Users can then use the `RDD api` to further shape or transform the dataset.

## 1.4 Experimental APIs

```
moztelemetry.zepplin.show(fig, width=600)
```

Renders a Matplotlib figure in Zeppelin.

### Parameters

- **fig** – a Matplotlib figure
- **width** – the width in pixel of the rendered figure, defaults to 600

Usage example:

```
import matplotlib.pyplot as plt
from moztelemetry.zepelin import show

fig = plt.figure()
plt.plot([1, 2, 3])
show(fig)
```

## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



**m**

`moztelemetry.dataset`, 3



## D

Dataset (class in moztelemetry.dataset), 3

## F

from\_source() (moztelemetry.dataset.Dataset static method), 3

## G

get\_pings() (in module moztelemetry.spark), 5

## M

moztelemetry.dataset (module), 3

## R

records() (moztelemetry.dataset.Dataset method), 4

## S

select() (moztelemetry.dataset.Dataset method), 4

show() (in module moztelemetry.zepplin), 5

summaries() (moztelemetry.dataset.Dataset method), 4

## W

where() (moztelemetry.dataset.Dataset method), 4