GWOSC

Release 0.7.1.dev6+gc545053

Duncan Macleod

Apr 20, 2023

CONTENTS

A python interface to the Gravitational-Wave Open Science Center archive.

CHAPTER

ONE

INSTALLATION

`gwosc` can be installed via pip:

```
python -m pip install gwosc
```

or Conda:

```
conda install -c conda-forge gwosc
```

CHAPTER

TWO

DOCUMENTATION

2.1 Querying dataset information

gwosc.datasets includes functions to query for available datasets.

To search for all available datasets:

```
>>> from gwosc import datasets
>>> datasets.find_datasets()
['GW150914', 'GW151226', 'GW170104', 'GW170608', 'GW170814', 'GW170817', 'LVT151012', 'O1
↪', 'S5', 'S6']
>>> datasets.find_datasets(detector='V1')
['GW170814', 'GW170817']
>>> datasets.find_datasets(type='run')
['O1', 'S5', 'S6']
```

To query for the GPS time of an event dataset (or vice-versa):

```
>>> datasets.event_gps('GW170817')
1187008882.43
>>> datasets.event_at_gps(1187008882)
'GW170817'
```

Similar queries are available for observing run datasets:

```
>>> datasets.run_segment('O1')
(1126051217, 1137254417)
>>> datasets.run_at_gps(1135136350)  # event_gps('GW151226')
'O1'
```

To run an event query filtered by merger parameters:

```
>>> from gwosc.datasets import query_events
>>> query_events(select=["mass-1-source <= 3.0"])
['GW170817-v3', 'GW190425-v1', 'GW190425-v2', 'GW190425_081805-v3']
```

Functions

dataset_type(dataset[, host])	Returns the type of the named dataset
event_at_gps(gps[, host, tol])	Returns the name of the open-data event matching the GPS time
event_detectors(event[, catalog, version, host])	Returns the `set` of detectors that observed an event
event_gps(event[, catalog, version, host])	Returns the GPS time of an open-data event
event_segment(event[, detector, catalog, ...])	Returns the GPS `[start, stop)` interval covered by an event dataset
find_datasets([detector, type, segment, ...])	Find datasets available on the given GW open science host
query_events(select[, host])	Return a list of events filtered by the parameters in `select`
run_at_gps(gps[, host])	Returns the name of the open-data run dataset matching the GPS time
run_segment(run[, host])	Returns the GPS `[start, stop)` interval covered by a run dataset

2.1.1 dataset_type

gwosc.datasets.**dataset_type**(*dataset*, *host='https://gwosc.org'*)

Returns the type of the named dataset

Parameters

- **dataset** (str) – The name of the dataset to match
- **host** (str, optional) – the URL of the GWOSC host to query

Returns

type – the type of the dataset, one of 'run', 'event', or 'catalog'

Return type

str

Raises

ValueError – if this dataset cannot be matched

Examples

```
>>> from gwosc.datasets import dataset_type
>>> dataset_type('O1')
'run'
```

2.1.2 event_at_gps

gwosc.datasets.**event_at_gps**(*gps*, *host='https://gwosc.org'*, *tol=1*)

Returns the name of the open-data event matching the GPS time

This function will return the first event for which `|eventgps - gps| < = tol`.

Parameters

- **gps** (float) – The GPS time to locate
- **host** (str, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org
- **tol** (float, optional) – the search window (in seconds), default: 1

Returns

event – the name of the matched event

Return type

str

Raises

ValueError – if no events are found matching the GPS time

Examples

```
>>> from gwosc.datasets import event_at_gps
>>> event_at_gps(1187008882)
'GW170817'
>>> event_at_gps(1187008882, tol=.1)
ValueError: no event found within 0.1 seconds of 1187008882
```

2.1.3 event_detectors

gwosc.datasets.**event_detectors**(*event*, *catalog=None*, *version=None*, *host='https://gwosc.org'*)

Returns the set of detectors that observed an event

Parameters

- **event** (str) – the name of the event to query
- **host** (str, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org
- **version** (int, None, optional) – the version of the data release to use, defaults to the highest available version

Returns

detectors – the set of detectors for which data file URLs are included in the data release

Return type

set

Examples

```
>>> from gwosc.datasets import event_detectors
>>> event_detectors("GW150914")
{'H1', 'L1'}
```

2.1.4 event_gps

gwosc.datasets.**event_gps**(*event*, *catalog=None*, *version=None*, *host='https://gwosc.org'*)

Returns the GPS time of an open-data event

Parameters

- **event** (str) – the name of the event to query
- **catalog** (str, optional) – name of catalogue that hosts this event
- **version** (int, None, optional) – the version of the data release to use, defaults to the highest available version
- **host** (str, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns

gps – the GPS time of this event

Return type

float

Examples

```
>>> from gwosc.datasets import event_gps
>>> event_gps('GW170817')
1187008882.43
>>> event_gps('GW123456')
ValueError: no event dataset found for 'GW123456'
```

2.1.5 event_segment

gwosc.datasets.**event_segment**(*event*, *detector=None*, *catalog=None*, *version=None*, *host='https://gwosc.org'*)

Returns the GPS `[start, stop)` interval covered by an event dataset

Parameters

- **event** (str) – the name of the event
- **detector** (str, optional) – prefix of GW detector
- **catalog** (str, optional) – name of catalogue that hosts this event
- **version** (int, None, optional) – the version of the data release to use, defaults to the highest available version
- **host** (str, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns

start, end – the GPS `[start, end)` interval covered by this run dataset

Return type
int

Examples

```
>>> from gwosc.datasets import event_segment
>>> event_segment("GW150914")
segment(1126257415, 1126261511)
```

2.1.6 find_datasets

gwosc.datasets.**find_datasets**(*detector=None*, *type=None*, *segment=None*, *match=None*, *catalog=None*, *version=None*, *host='https://gwosc.org'*)

Find datasets available on the given GW open science host

Parameters

- **detector** (str, optional) – prefix of GW detector
- **type** (str, optional) – type of datasets to restrict, one of 'run', 'event', or 'catalog'
- **segment** (2-tuple of int, None, optional) – a GPS [start, stop) interval to restrict matches to; datasets will match if they overlap at any point; this is not used to filter catalogs
- **match** (str, optional) – regular expression string against which to match datasets
- **host** (str, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns
datasets – the names of all matched datasets, possibly empty

Return type
list of str

Examples

(Correct as of 2018-03-14)

```
>>> from gwosc.datasets import find_datasets
>>> find_datasets()
['GW150914', 'GW151226', 'GW170104', 'GW170608', 'GW170814', 'GW170817',
 'LVT151012', 'O1', 'S5', 'S6']
>>> find_datasets(detector='V1')
['GW170814', 'GW170817']
>>> find_datasets(type='event')
['GW150914', 'GW151226', 'GW170104', 'GW170608', 'GW170814', 'GW170817',
 'LVT151012']
```

2.1.7 query_events

gwosc.datasets.**query_events**(*select*, *host='https://gwosc.org'*)

Return a list of events filtered by the parameters in `select`

Parameters

- **select** (`list` of `str`) – A list of strings where each element is a range constrain on the event parameters. All ranges should have inclusive ends (<= and => operators).
- **host** (`str`, optional) – the URL of the GWOSC host to query

Examples

```
>>> from gwosc.datasets import query_events
>>> query_events(
...     select=[
...         "mass-1-source >= 1.4",
...         "200 >= luminosity-distance >= 100",
...     ]
... )
['GW190425-v1', 'GW190425-v2', 'GW190425_081805-v3']
```

Notes

Operators:

- <= (or =<)
- => (or >=)

Parameters:

- ``gps-time`,
- `mass-1-source`,
- `mass-2-source`,
- `network-matched-filter-snr`,
- `luminosity-distance`,
- `chi-eff`,
- `total-mass-source`,
- `chirp-mass`,
- `chirp-mass-source`,
- `redshift`,
- `far`,
- `p-astro`,
- `final-mass-source`

For a full description of all parameters see https://www.gwosc.org/apidocs/#event5

2.1.8 run_at_gps

gwosc.datasets.**run_at_gps**(*gps*, *host='https://gwosc.org'*)

Returns the name of the open-data run dataset matching the GPS time

This function will return the first event for which `start <= gps < end`

Parameters

- **gps** (`float`) – The GPS time to locate
- **host** (`str`, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns

run – the name of the matched observing run

Return type

`str`

Raises

`ValueError` – if no datasets are found matching the GPS time

Examples

```
>>> from gwosc.datasets import run_at_gps
>>> run_at_gps(1135136350)
'O1'
>>> run_at_gps(0)
ValueError: no run dataset found containing GPS 0
```

2.1.9 run_segment

gwosc.datasets.**run_segment**(*run*, *host='https://gwosc.org'*)

Returns the GPS `[start, stop)` interval covered by a run dataset

Parameters

- **run** (`str`) – the name of the run dataset to query
- **host** (`str`, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns

start, end – the GPS `[start, end)` interval covered by this run dataset

Return type

`int`

Examples

```
>>> from gwosc.datasets import run_segment
>>> run_segment('O1')
segment(1126051217, 1137254417)
>>> run_segment('Oh dear')
ValueError: no run dataset found for 'Oh dear'
```

2.2 Querying for data file URLs

gwosc.locate provides functions to determine the file URLs containing data for a specific dataset.

You can search for remote data URLS based on the event name:

```
>>> from gwosc.locate import get_event_urls
>>> get_event_urls('GW150914')
['https://gwosc.org/eventapi/json/GWTC-1-confident/GW150914/v3/H-H1_GWOSC_4KHZ_R1-
↪1126259447-32.hdf5',
 'https://gwosc.org/eventapi/json/GWTC-1-confident/GW150914/v3/H-H1_GWOSC_4KHZ_R1-
↪1126257415-4096.hdf5',
 'https://gwosc.org/eventapi/json/GWTC-1-confident/GW150914/v3/L-L1_GWOSC_4KHZ_R1-
↪1126259447-32.hdf5',
 'https://gwosc.org/eventapi/json/GWTC-1-confident/GW150914/v3/L-L1_GWOSC_4KHZ_R1-
↪1126257415-4096.hdf5']
```

You can down-select the URLs using keyword arguments:

```
>>> get_event_urls('GW150914', detector='L1', duration=32)
['https://gwosc.org/eventapi/json/GWTC-1-confident/GW150914/v3/L-L1_GWOSC_4KHZ_R1-
↪1126259447-32.hdf5']
```

You can search for remote data URLs based on the GPS time interval as follows:

```
>>> from gwosc.locate import get_urls
>>> get_urls('L1', 968650000, 968660000)
['https://gwosc.org/archive/data/S6/967835648/L-L1_LOSC_4_V1-968646656-4096.hdf5',
 'https://gwosc.org/archive/data/S6/967835648/L-L1_LOSC_4_V1-968650752-4096.hdf5',
 'https://gwosc.org/archive/data/S6/967835648/L-L1_LOSC_4_V1-968654848-4096.hdf5',
 'https://gwosc.org/archive/data/S6/967835648/L-L1_LOSC_4_V1-968658944-4096.hdf5']
```

By default, this method will return the paths to HDF5 files for the 4 kHz sample-rate data, these can be specified as keyword arguments. For full information, see *get_urls()*.

Functions

get_urls(detector, start, end[, dataset, ...])	Fetch the URLs from GWOSC regarding a given GPS interval
get_run_urls(run, detector, start, end[, ...])	Fetch the URLs from GWOSC regarding a given event
get_event_urls(event[, catalog, version, ...])	Fetch the URLs from GWOSC regarding a given event

2.2.1 get_urls

gwosc.locate.**get_urls**(*detector*, *start*, *end*, *dataset=None*, *version=None*, *sample_rate=4096*, *format='hdf5'*, *host='https://gwosc.org'*)

Fetch the URLs from GWOSC regarding a given GPS interval

Parameters

- **detector** (str) – the prefix of the relevant GW detector
- **start** (int) – the GPS start time of your query
- **end** (int) – the GPS end time of your query
- **dataset** (str, None, optional) – the name of the dataset to query, e.g. `'GW150914'`
- **version** (int, None, optional) – the data-release version for the selected datasets
- **sample_rate** (int, optional, default : `4096`) – the sampling rate (Hz) of files you want to find
- **format** (str, optional, default: `'hdf5'`) – the file format (extension) you want to find
- **host** (str, optional) – the URL of the remote GWOSC server

Returns

urls – the list of remote file URLs that contain data matching the relevant parameters

Return type

list of str

2.2.2 get_run_urls

gwosc.locate.**get_run_urls**(*run*, *detector*, *start*, *end*, *format='hdf5'*, *sample_rate=4096*, ***match*)

Fetch the URLs from GWOSC regarding a given event

Parameters

- **event** (str) – the ID of the event
- **detector** (str, optional) – the detector for files you want to find
- **start** (int) – the GPS start time of your query
- **end** (int) – the GPS end time of your query
- **format** (str, optional, default: `'hdf5'`) – the file format (extension) you want to find
- **sample_rate** (int, optional, default : `4096`) – the sampling rate (Hz) of files you want to find
- **host** (str, optional) – the URL of the remote GWOSC server

- **version** (int, None, optional) – the data-release version for the selected datasets

Returns
urls – the list of remote file URLs that contain data matching the relevant parameters

Return type
list of str

2.2.3 get_event_urls

gwosc.locate.**get_event_urls**(*event*, *catalog=None*, *version=None*, *detector=None*, *start=None*, *end=None*, *format='hdf5'*, *sample_rate=4096*, *host='https://gwosc.org'*, ***match*)

Fetch the URLs from GWOSC regarding a given event

Parameters

- **event** (str) – the ID of the event
- **detector** (str, optional) – the detector for files you want to find
- **format** (str, optional, default: `'hdf5'`) – the file format (extension) you want to find
- **sample_rate** (int, optional, default : `4096`) – the sampling rate (Hz) of files you want to find
- **host** (str, optional) – the URL of the remote GWOSC server
- **start** (int) – the GPS start time of your query
- **end** (int) – the GPS end time of your query
- **version** (int, None, optional) – the data-release version for the selected datasets

Returns
urls – the list of remote file URLs that contain data matching the relevant parameters

Return type
list of str

2.3 Querying for Timeline segments

gwosc.timeline provides functions to find segments for a given dataset.

You can search for Timeline segments, based on a flag name, and a GPS time interval as follows:

```
>>> from gwosc.timeline import get_segments
>>> get_segments('H1_DATA', 1126051217, 1126151217)
[(1126073529, 1126114861), (1126121462, 1126123267), (1126123553, 1126126832),␣
↪(1126139205, 1126139266), (1126149058, 1126151217)]
```

The output is a list of `[start, end)` 2-tuples which each represent a semi-open time interval.

For documentation on what flags are available, for example for the O1 science run, see the O1 data release page (*Data Quality*).

Functions

get_segments(flag, start, end[, host])	Return the [start, end) GPS segments for this flag
timeline_url(flag, start, end[, host])	Returns the Timeline JSON URL for a flag name and GPS interval

2.3.1 get_segments

gwosc.timeline.**get_segments**(*flag*, *start*, *end*, *host='https://gwosc.org'*)

Return the [start, end) GPS segments for this flag

Parameters

- **flag** (str) – name of flag, e.g. 'H1_DATA'
- **start** (int) – the GPS start time of your query
- **end** (int) – the GPS end time of your query
- **host** (str, optional) – the URL of the remote GWOSC server

Returns

segments – a list of [a, b) GPS segments

Return type

list of (int, int) tuples

2.3.2 timeline_url

gwosc.timeline.**timeline_url**(*flag*, *start*, *end*, *host='https://gwosc.org'*)

Returns the Timeline JSON URL for a flag name and GPS interval

2.4 Low-level API

gwosc.api provides the low-level interface functions that handle direct requests to the GWOSC host.

Module contents:

DEFAULT_URL	The default GWOSC host URL
JSON_CACHE	Cache of downloaded blobs
fetch_allevents_json([full, host])	"Returns the JSON metadata for the allevents API
fetch_allowed_params_json([host])	
fetch_catalog_json(catalog[, host])	"Returns the JSON metadata for the given catalogue
fetch_cataloglist_json([host])	Returns the JSON metadata for the catalogue list.
fetch_dataset_json(gpsstart, gpsend[, host])	Returns the JSON metadata for all datasets matching the GPS interval
fetch_event_json(event[, catalog, version, host])	Returns the JSON metadata for the given event.
fetch_filtered_events_json(select[, host])	"Return the JSON metadata for the events constrained by select
fetch_json(url, **kwargs)	Fetch JSON data from a remote URL
fetch_run_json(run, detector[, gpsstart, ...])	Returns the JSON metadata for the given science run parameters
logger	Instances of the Logger class represent a single logging channel.
urlencode(query[, doseq, safe, encoding, ...])	Encode a dict or sequence of two-element tuples into a URL query string.

2.4.1 DEFAULT_URL

gwosc.api.**DEFAULT_URL = 'https://gwosc.org'**

The default GWOSC host URL

2.4.2 JSON_CACHE

gwosc.api.**JSON_CACHE = {}**

Cache of downloaded blobs

2.4.3 fetch_allevents_json

gwosc.api.**fetch_allevents_json**(*full=False*, *host='https://gwosc.org'*)

"Returns the JSON metadata for the allevents API

Parameters

host (str, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns

data – the JSON data retrieved from GWOSC and returned by `requests.Response.json()`

Return type

dict or list

2.4.4 fetch_allowed_params_json

gwosc.api.**fetch_allowed_params_json**(*host='https://gwosc.org'*)

2.4.5 fetch_catalog_json

gwosc.api.**fetch_catalog_json**(*catalog*, *host='https://gwosc.org'*)

"Returns the JSON metadata for the given catalogue

Parameters

- **catalog** (str) – the name of the event catalog, e.g. GWTC-1-confident
- **host** (str, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns
data – the JSON data retrieved from GWOSC and returnend by requests.Response.json()

Return type
dict or list

2.4.6 fetch_cataloglist_json

gwosc.api.**fetch_cataloglist_json**(*host='https://gwosc.org'*)

Returns the JSON metadata for the catalogue list.

Parameters
host (str, optional) – the URL of the GWOSC host to query

Returns
data – the JSON data retrieved from GWOSC and returned by requests.Response.json()

Return type
dict or list

2.4.7 fetch_dataset_json

gwosc.api.**fetch_dataset_json**(*gpsstart*, *gpsend*, *host='https://gwosc.org'*)

Returns the JSON metadata for all datasets matching the GPS interval

Parameters

- **gpsstart** (int) – the GPS start of the desired interval
- **gpsend** (int) – the GPS end of the desired interval
- **host** (str, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns
data – the JSON data retrieved from GWOSC and returned by json.loads

Return type
dict or list

2.4.8 fetch_event_json

gwosc.api.**fetch_event_json**(*event*, *catalog=None*, *version=None*, *host='https://gwosc.org'*)

Returns the JSON metadata for the given event.

By default, this function queries across all catalogs and all data-release versions, returning the highest available version, unless the `version` and/or `catalog` keywords are specified.

Parameters

- **event** (`str`) – the name of the event to query
- **catalog** (`str`, optional) – name of catalogue that hosts this event
- **version** (`int`, `None`, optional) – restrict query to a given data-release version
- **host** (`str`, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns

data – the JSON data retrieved from GWOSC and returned by `json.loads`

Return type

`dict` or `list`

2.4.9 fetch_filtered_events_json

gwosc.api.**fetch_filtered_events_json**(*select*, *host='https://gwosc.org'*)

"Return the JSON metadata for the events constrained by select

Parameters

- **select** (`list-like`) – a list of range constrains for the events. All ranges should have inclusive ends (<= and >= operators).
- **host** (`str`, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns

data – the JSON data retrieved from GWOSC and returnend by `requests.Response.json()`

Return type

`dict` or `list`

Example

```
>>> fetch_filtered_events_json(
...     select=[
...         "mass-1-source <= 5",
...         "mass-2-source =< 10",
...         "10 <= luminosity-distance <= 100",
...     ]
... )
```

2.4.10 fetch_json

gwosc.api.**fetch_json**(*url*, ***kwargs*)

Fetch JSON data from a remote URL

Parameters

- **url** (str) – the remote URL to fetch
- ****kwargs** – other keyword arguments are passed directly to `requests.get()`

Returns

data – the data fetched from `url` as parsed by `requests.Response.json()`

Return type

`dict` or `list`

See also:

`json.loads`

for details of the JSON parsing

2.4.11 fetch_run_json

gwosc.api.**fetch_run_json**(*run*, *detector*, *gpsstart=0*, *gpsend=99999999999*, *host='https://gwosc.org'*)

Returns the JSON metadata for the given science run parameters

Parameters

- **run** (str) – the name of the science run, e.g. `'O1'`
- **detector** (str) – the prefix of the GW detector, e.g. `'L1'`
- **gpsstart** (int) – the GPS start of the desired interval
- **gpsend** (int) – the GPS end of the desired interval
- **host** (str, optional) – the URL of the GWOSC host to query, defaults to https://gwosc.org

Returns

data – the JSON data retrieved from GWOSC and returned by `json.loads`

Return type

`dict` or `list`

2.4.12 logger

gwosc.api.**logger = <Logger gwosc.api (WARNING)>**

Instances of the Logger class represent a single logging channel. A "logging channel" indicates an area of an application. Exactly how an "area" is defined is up to the application developer. Since an application can have any number of areas, logging channels are identified by a unique string. Application areas can be nested (e.g. an area of "input processing" might include sub-areas "read CSV files", "read XLS files" and "read Gnumeric files"). To cater for this natural nesting, channel names are organized into a namespace hierarchy where levels are separated by periods, much like the Java or Python package namespace. So in the instance given above, channel names might be "input" for the upper level, and "input.csv", "input.xls" and "input.gnu" for the sub-levels. There is no arbitrary limit to the depth of nesting.

CHAPTER

THREE

INDEX

- genindex
- modindex

PYTHON MODULE INDEX

g

INDEX