
PyWind Documentation

Release 1.0.4

David Reid

Oct 03, 2017

Contents

1	Command Line App	3
1.1	Commands	3
1.2	Optional Arguments	4
1.3	Sample Usage	4
1.4	Available Commands	6
2	Sample Scripts	9
2.1	capacity_ro	9
2.2	annual_output	10
2.3	derived_unit_data	11
2.4	ofgem_certificate_search	11
2.5	ofgem_station_search	11
3	pywind Package	13
3.1	Modules	13
4	Background	33
5	Travis CI	35
6	Python 3 Support	37
7	License	39
8	ToDo List	41
9	Patches Welcome!	43
10	Contributors	45
11	Indices and tables	47
	Python Module Index	49

Contents:

Command Line App

Starting with the version 1.x releases there will be a simple command line application provided with pywind.

Commands

The app uses a command to determine the function to perform. If no command is supplied a list of available commands will be displayed.

```
$ pywind

decc_extract          DECC Monthly Planning Extract
elxon_b1320          Congestion Management Measures Countertrading
elxon_b1330          Congestion Management Measures Costs of Congestion_
↔Management Service
elxon_b1420          Installed Generation Capacity per Unit
elxon_bm_data        Derived System Prices from Elexon
elxon_bm_unit        Balancing Mechanism Unit information from Elexon
elxon_generation_inst Generation Data from the Elexon Data Portal
elxon_sbp            Derived System Prices from Elexon
ofgem_certificate_search Ofgem Certificate Search
ofgem_station_search Ofgem Station Search
roc_prices           eROC Auction Prices
```

Command	Description
decc_extract	Obtain the latest DECC monthly planning extract.
elexon_bm1320	Extract data from the Elexon API
elexon_bm1330	Extract data from the Elexon API
elexon_bm1420	Extract data from the Elexon API
elexon_bm_data	Get derived data for the Balancing Mechanism using the Elexon API
elexon_bm_unit	Balancing Mechanism unit data from Elexon
elexon_generation	Generation volume by fuel type from Elexon
elexon_sbp	System Buy & Sell price data from Elexon
ofgem_certificate	Perform a search of Ofgem Certificate records. The number of criteria is limited compared with direct use of the module <code>CertificateSearch</code> .
ofgem_station	Perform a search of Ofgem Station information. The number of criteria is limited compared with direct use of the <code>StationSearch</code> class.
roc_prices	Obtain the latest eROC auction information.

Optional Arguments

The following optional arguments are also available.

Argument	Description
-debug	Enable debugging mode.
-request-debug	Enable logging of requests made to a remote server
-log-filename	Filename to use for the logfile
-date	Specify a date for commands that use it. Format is YYYY-MM-DD
-apikey	The elexon API key. If a filename is specified the contents will be read and used.
-fromdate	First date to filter
-todate	Last date to filter
-year	Filter for a year. Only used by elexon commands
-month	Filter for a month. Only used for elexon commands.
-unit-type	Letter specifying the type of unit. Only used for <code>elexon_bm_unit</code> .
-period	Specify a period for commands that need it. Format is YYYYMM
-all-periods	Try and get data for all available periods
-scheme	The Ofgem Scheme used to filter searches. This is only used by the Ofgem commands. Options are REGO or RO.
-export filename	Export the results. Available formats are CSV, XML and XSLX.
-output filename	Filename to export data to. If not supplied the export will be saved to a generated filename.
-input filename	Process the saved file
-save	Save the downloaded data to a local file. Filename to use should be specified using the -original parameter.
-original filename	The filename to save downloaded into.

Sample Usage

To download the monthly DECC Planning extract and save it as a CSV file,


```
$ pywind decc_extract --export csv
DECC Monthly Planning Extract
```

```
Total of 4896 planning records received for July 2016
Output will be saved in monthlyextract.csv
Total of 4896 data rows written
CSV export to monthlyextract.csv completed
```

To obtain the latest eROC auction data as an Excel spreadsheet,

```
$ pywind roc_prices --export xlsx
eROC Auction Prices
```

```
/usr/lib/python2.7/dist-packages/html5lib/ihatexml.py:262: DataLossWarning: Coercing
↳non-XML name
```

```
warnings.warn("Coercing non-XML name", DataLossWarning)
```

```
Period          Average Price
```

```
-----
200210          47.12
200301          47.46
200304          46.76
200307          48.21
200310          45.93
200401          47.46
200404          49.11
200407          52.07
200410          46.12
200501          47.18
200504          46.07
```

```
...
```

```
201606          41.35
201607          41.65
```

```
Output will be saved in erocprices.xlsx
XLSX export to erocprices.xlsx completed
```

```
$pywind elxon_sbp --apikey elxon.api.key
```

```
Reading API Key from elxon.api.key
```

```
Derived System Prices from Elxon
```

```
=====
```

System adjustments are included in the figures shown below where '*' is shown.

Date	Settlement Period	Sell Price	Buy Price	Adj?
-----	-----	-----	-----	----
2017 Mar 28	1	29.0020	29.0020	
2017 Mar 28	2	29.0000	29.0000	
2017 Mar 28	3	29.1100	29.1100	
2017 Mar 28	4	29.1119	29.1119	
2017 Mar 28	5	29.1124	29.1124	
2017 Mar 28	6	28.4936	28.4936	
2017 Mar 28	7	29.4324	29.4324	
2017 Mar 28	8	29.1132	29.1132	
2017 Mar 28	9	29.1100	29.1100	
2017 Mar 28	10	29.0874	29.0874	
2017 Mar 28	11	29.0534	29.0534	
2017 Mar 28	12	29.3217	29.3217	

2017 Mar 28	13	30.1473	30.1473
2017 Mar 28	14	29.9129	29.9129
2017 Mar 28	15	30.3665	30.3665
2017 Mar 28	16	158.9650	158.9650
2017 Mar 28	17	160.0000	160.0000
2017 Mar 28	18	151.1268	151.1268
2017 Mar 28	19	33.0000	33.0000
2017 Mar 28	20	110.0000	110.0000
2017 Mar 28	21	31.3086	31.3086
2017 Mar 28	22	30.5500	30.5500
2017 Mar 28	23	30.5593	30.5593
2017 Mar 28	24	30.3133	30.3133
2017 Mar 28	25	30.2979	30.2979
2017 Mar 28	26	30.1066	30.1066
2017 Mar 28	27	30.0175	30.0175
2017 Mar 28	28	29.7841	29.7841
2017 Mar 28	29	30.3187	30.3187
2017 Mar 28	30	30.5830	30.5830
2017 Mar 28	31	30.6778	30.6778
2017 Mar 28	32	31.0000	31.0000
2017 Mar 28	33	65.5367	65.5367
2017 Mar 28	34	114.4993	114.4993
2017 Mar 28	35	113.1707	113.1707
2017 Mar 28	36	76.6555	76.6555
2017 Mar 28	37	30.7676	30.7676
2017 Mar 28	38	30.9852	30.9852
2017 Mar 28	39	31.1493	31.1493
2017 Mar 28	40	31.1500	31.1500
2017 Mar 28	41	31.1500	31.1500
2017 Mar 28	42	31.0000	31.0000
2017 Mar 28	43	31.0000	31.0000
2017 Mar 28	44	29.5678	29.5678
2017 Mar 28	45	30.1783	30.1783
2017 Mar 28	46	30.7737	30.7737
2017 Mar 28	47	51.9000	51.9000
2017 Mar 28	48	30.5000	30.5000

Available Commands

Some commands allow additional filtering:

```
$ pywind elxon_sbp --apikey elxon.api.key --period 5
Reading API Key from elxon.api.key

Derived System Prices from Elxon
=====

System adjustments are included in the figures shown below where '*' is shown.
```

Date	Settlement Period	Sell Price	Buy Price	Adj?
2017 Mar 28	5	29.1124	29.1124	

```
$ pywind elxon_bm_unit --apikey elxon.api.key --unit-type I
```

Total of 373 units

NGC ID	BM ID	Active ?	BM Type	Lead Party
↔Name				
-----	-----	-----	-----	-----
↔-----				
EAD-BRTN1	I_EAD-BRTN1	Y	I, Interconnector	NGET plc
EAD-EWIC1	I_EAD-EWIC1	Y	I, Interconnector	NGET plc
EAD-FRAN1	I_EAD-FRAN1	Y	I, Interconnector	NGET plc
EAD-MOYL1	I_EAD-MOYL1	Y	I, Interconnector	NGET plc
EAD-SCOT1	I_EAD-SCOT1	Y	I, Interconnector	NGET plc
...				
iMD-SSIR1	I_IMD_SSIR1	Y	I, Interconnector	SSE (IRELAND)
↔LIMITED				

CHAPTER 2

Sample Scripts

There are a small number of sample scripts that have been written over the years and provide additional information about using pywind.

Note: The names used are probably in need of changing!

Some of the scripts date back several years in origin, so the names may no longer be appropriate!

capacity_ro

This script was written to satisfy the following request by a researcher

For a given list of windfarms, get the capacity and RO certificates issued for a given period (or periods), as an Excel spreadsheet

Ofgem were unable to limit their data to the criteria but did supply an Excel spreadsheet of all their data - which was too large to be opened on most computers!

To use this script to get the information for the station Braes of Doune

```
$ ./convert_ro.py 201601 201603
Enter a station name (or blank to finish)Braes Of Doune
Enter a station name (or blank to finish)

Total of 1 stations to process

    Braes Of Doune

Complete. Generating Excel spreadsheet certificates.xls
```

When entering station names, you can specify more than one on a line by separating them with commas, so rerun the above query for the stations Braes of Doune and Boulfruch you could do this

```
$ ./convert_ro.py --start Jan-2010 --end Dec-2010 --filename two_stations
```

You can also provide a file with one station name per line, using the **-input** parameter.

```
$ cat station.list
Braes Of Doune
Boulfruich
$ ./convert.py --input station.list 201601 201603
```

An Ofgem search is conducted for each station name supplied and **all** matching stations are added to the list of stations to have their certificate information queried and recorded.

The output is minimal but intended to keep you up to date with progress as searching for stations takes a while.

```
$ capacity_ro.py 201601 201603 --stations Griffin
Period covered will be Jan-2016 to Mar-2016. A total of 3 periods
Station names to be searched for:
  - Griffin
Enter a station name (or blank to finish)

Searching for stations...
  - Griffin
    found
A total of 4 stations will be recorded

Getting certificate data (this is quicker)...
  - Griffin Wind Farm
    added to spreadsheet
  - William Griffin 6.0kwp
    nothing to add
  - Griffin PV System
    nothing to add
  - Ronald Griffin Solar Hub
    nothing to add

Data saved to certificates.xls
```

Note: It would be nice to have better formatting for the output, but as this is just a sample script I haven't spent any time adding them, e.g. the date format on the exported spreadsheet needs setting.

annual_output

This sample script was written on behalf of Graham and an enquiry he had for a years worth of output data for Hydro & Biogas stations.

As usual REGO output will be used a proxy for output.

The library appears unable to cope with setting the technology field directly and so we restrict the options setting to the year, month and scheme and then simply filter the returned data.

derived_unit_data

This script demonstrates how to use the `bmreports.UnitData` class to get information about Constraint Payments. It does not attempt to cater for long or short days and simply assumes that there will be 48 settlement periods.

ofgem_certificate_search

One of the most common requests is to search the Ofgem database for certificate issuance. This script provides an example of using `pywind` to do this.

```
$ ofgem_certificate_search --period 201601 --generator R00160SQSC
Contacting Ofgem and preparing to search.

Filtering search:
  - generator id R00160SQSC
  - period should be 201601

Total of 1 records returned
  Issue Date  Period  Station Name  Scheme  Status
↳Certificates
-----
↳-----
  2016-04-08  Jan-2016  Griffin Wind Farm  RO  Issued
↳ 37491
```

ofgem_station_search

Sample script to demonstrate using `pywind` to search for Ofgem Stations.

```
$ ofgem_station_search.py --station Griffin
Connecting with Ofgem website and preparing the search...
Setting up filters:
  - station name contains Griffin

Getting results from Ofgem...

Query returned 4 results
  Station Name  Commission Dt  Capacity  Technology
↳ Country  Generator ID
-----
↳-----
  Griffin Wind Farm  2011-07-05  186170.00  On-shore wind (RO.
↳.. Scotland  R00160SQSC
  William Griffin 6.0kwp  2013-11-21  6.00  PV with a DNC of .
↳.. Northern Ireland  R03770NGNI
  Griffin PV System  2014-11-20  3.50  PV with a DNC of .
↳.. Northern Ireland  R09926NGNI
  Ronald Griffin Solar Hub  2014-09-19  4.00  PV with a DNC of .
↳.. Northern Ireland  R09542NGNI
```

```
$ ofgem_station_search.py --organisation Speedwell
Connecting with Ofgem website and preparing the search...
```

```
Setting up filters:
- organisation contains Speedwell

Getting results from Ofgem...

Query returned 1 results
Station Name          Commission Dt  Capacity  Technology
↔ Country            Generator ID
-----
↔-----
Speedwell             2013-10-01   6.50     PV with a DNC of .
↔.. Northern Ireland R03360NGNI
```


The *pywind Package* is split up into a series of modules, each of which deals with a different type of data.

Modules

`pywind`

`pywind.export`

Export functions for command line app.

`pywind.export.export_to_file` (*args*, *obj*)

Export a pywind object to a file based on command line arguments. The object supplied can be exported as CSV, XLSX or XML depending on the `args.format` option supplied.

Parameters

- **args** – Command line args from `argparse.parse_args()`
- **obj** – The pywind object to export.

`pywind.log`

PyWind now uses standard python logging :-)

`pywind.log.setup_logging` (*debug=False*, *stdout=True*, *request_logging=False*, *filename=None*)

Setup the logging for pywind.

Parameters

- **debug** – Enable debug level messages (default False)
- **stdout** – Enable logging to stdout (default True)
- **request_logging** – Enable full logging of network requests (default False)

- **filename** – Filename to use for log.

pywind.utils

Utility functions used by more than one module within pywind.

class pywind.utils.**StdoutFormatter** (*args, **kwargs)
Bases: object

Small class to provide easier printing to stdout.

```
>>> from pywind.utils import StdoutFormatter
>>> sof = StdoutFormatter("5s", "6s", ">10s")
>>> sof.titles("Hello", "World", "right")
Hello World      right
-----
>>> sof.row("first", "row", "right")
first row        right
```

Note: The format detection doesn't allow all options :-("10,d" is not yet supported).

formatter (titles=False)

Return the format string for the columns configured.

Parameters **titles** – True if the format will be used for titles (all strings)

Returns Format string

Return type str

row (*args)

Use the column format to generate a string. This tries to truncate long strings.

Parameters **args** – The column values.

Returns Formatted string using args

Return type str

titles (*args)

Generate the title string block.

Parameters **args** – List of titles to use. Should be at least as long as the number of columns

Returns Formatted title string

Return type str

pywind.utils.**args_get_datetime** (args)

pywind.utils.**commandline_parser** (help_text, epilog=None)

Simple function to create a command line parser with some generic options.

Parameters

- **help_text** – The script description
- **epilog** – Epilog text

Returns Argument parser

Return type argparse.ArgumentParser

`pywind.utils.get_or_post_a_url(url, post=False, **kwargs)`

Use the requests library to either get or post to a specified URL. The return code is checked and exceptions raised if there has been a redirect or the status code is not 200.

Parameters

- **url** – The URL to be used.
- **post** – True if the request should be a POST. Default is False which results in a GET request.
- **kwargs** – Optional keyword arguments that are passed directly to the requests call.

Returns The requests object is returned if all checks pass.

Return type `requests.Response`

Raises Raises `Exception` for various errors.

Example

```
>>> from pywind.utils import get_or_post_a_url
>>> qry = {'q': 'rst document formatting'}
>>> response = get_or_post_a_url('http://www.google.com/search', params=qry)
>>> response.content
...

```

`pywind.utils.map_xml_to_dict(xml_node, mapping=None)`

Given an XML node, create a dict using the mapping of attributes/elements supplied.

The format of each mapping item is a tuple of up to 3 components,

- xml attribute
- key for dict (optional)
- type of data expected (optional)
- default value for the mapping

If the key name is not supplied, the lower cased xml attribute will be used. If the type is not given it will be assumed to be a string.

Parameters

- **xml_node** – The XML node to parse
- **mapping** – Iterable of xml element

Returns Dict of successfully extracted data

Return type `dict`

`pywind.utils.multi_level_get(the_dict, key, default=None)`

Given the level of nested data contained in some of the results, this function performs an iterative get.

Parameters

- **the_dict** – The multi-level dict to get the key from.
- **key** – The key to look for, with each level separated by ‘.’
- **default** – The default to return if the key is not found. (None if not supplied)

Returns The value of the key or default if key does not exist in the dict.

`pywind.utils.parse_response_as_xml(request)`

Given a the response object from requests, attempt to parse it’s contents as XML.

Parameters `request` – The requests object

Returns The root XML node or None if there is a parser error

Note:

•Nov 2014 Using parser with `recover=True` was the suggestion of energynumbers

`pywind.utils.valid_date(dtstr)`

Parse a string into a date using the YYYY-MM-DD format. Used by the `commandline_parser()` function.

Parameters `dtstr` – Date string to be parsed

Returns Valid date

Return type `datetime.date`

Raises `argparse.ArgumentTypeError` if the date string is not formatted as YYYY-MM-DD

`pywind.utils.valid_time(dtstr)`

Parse a string into a date using the YYYY-MM-DD format. Used by the `commandline_parser()` function.

Parameters `dtstr` – Date string to be parsed

Returns Valid date

Return type `datetime.date`

Raises `argparse.ArgumentTypeError` if the date string is not formatted as YYYY-MM-DD

`pywind.utils.xml_attr_or_element(xml_node, name)`

Attempt to get the value of name from the `xml_node`. This could be an attribute or a child element.

`pywind.bmreports`

Warning: The BM Reports website has been removed and replaced by a much more complex API interface. This page is no longer relevant and should be ignored in favour of functionality offered by the `elexon` module.

Warning: The data provided by the BMReports website is owned by Elexon UK and permission needs to be sought before reproducing it. The following functions should only be used with this restriction in mind as they access the site and download data.

The exact restrictions on the data usage are unclear at this time.

`pywind.bmreports.generation_type`

BMReports make available a number of reports, but this module provides access to their report on output by generation type for the 3 periods,

- instant
- last hour
- last 24 hours

```
class pywind.bmreports.generation_type.GenerationData
```

```
Bases: object
```

Class to allow access to the report and parse the response into usable structures.

```
PARAMS = {'element': 'generationbyfueltypetable'}
```

```
URL = 'http://www.bmreports.com/bsp/additional/soapfunctions.php'
```

```
as_dict ()
```

Return the data as a dict object.

```
get_data ()
```

Get data from the BM Reports website. Try 3 times.

```
rows ()
```

Return export data as a series of rows.

Return type dict

```
save_original (filename)
```

Save the downloaded certificate data into the filename provided.

Parameters filename – Filename to save the file to.

Return type bool

```
class pywind.bmreports.generation_type.GenerationPeriod (elm)
```

```
Bases: object
```

The basic report contains information on 3 different periods. Each will be represented by an instance of this class.

```
DT1 = '%Y-%m-%d %H:%M:%S'
```

```
DT2 = '%Y-%m-%d %H:%M'
```

```
NAMES = {'LAST24H': '24hours', 'HH': 'halfhour', 'INST': 'instant'}
```

```
as_dict ()
```

Return the data as a dict.

```
hh (elm)
```

Store the start and finish times for a half hour record. :param elm: the element to parse

```
inst (elm)
```

Store the time for an instant record. :param elm: the element to parse

```
keyname ()
```

Return the full name of the tag.

```
last24h (elm)
```

Store the start and finish date/time records for a 24 hour record. :param el: the element to parse

```
class pywind.bmreports.generation_type.GenerationRecord (el)
```

```
Bases: object
```

Class to record details of a single generation type record.

```
FUELS = {'INTFR': 'Import from France', 'NUCLEAR': 'Nuclear', 'OTHER': 'Other', 'INTIRL': 'Import from Ireland'}
```

```
as_dict ()
```

Return data as a dict.

`pywind.bmreports.prices`

BMReports make the system electricity prices available. This module contains classes to access those reports.

class `pywind.bmreports.prices.SystemPrices` (*dt=None*)

Bases: `object`

Class to get the electricity prices from BMreports.

URL = `'http://www.bmreports.com/bsp/additional/soapfunctions.php'`

as_dict ()

Return the data as a dict.

get_data ()

Get the data from the remote server.

rows ()

Generator to return rows for export.

Returns Dict containing information for a single price period.

Return type `dict`

save_original (*filename*)

Save the downloaded certificate data into the filename provided.

Parameters **filename** – Filename to save the file to.

Returns True or False

Return type `bool`

`pywind.bmreports.unit`

Unit data from BM Reports

class `pywind.bmreports.unit.BalancingUnitData` (*xml_node*)

Bases: `object`

Class to store balancing payment information for a single unit during a single period.

bid_cashflow

Return the bid cashflow.

Return type `float`

Returns Bid cashflow or 0.0

bid_volume

Get the bid volume.

Return type `float`

Returns Bid volume or 0.0

offer_cashflow

Return the bid cashflow.

Return type `float`

Returns Bid cashflow or 0.0

offer_volume

Get the bid volume.

Return type float

Returns Bid volume or 0.0

rate (*which*)

Extract the rate paid for either “bid” or “offer” from the data.

Parameters *which* – “bid” or “offer”

Returns The calculated rate

Return type float

class `pywind.bmreports.unit.BaseUnitClass`

Bases: `object`

Base class

SHEET_NAME = ‘

XLS_URL = ‘

get_list ()

Download and update the unit list.

Return type bool

rows ()

Generator to return row data.

Returns Dict of unit data

Return type dict

save_original (*filename*)

Save the downloaded certificate data into the filename provided.

Parameters *filename* – Filename to save the file to.

Returns True or False

Return type bool

class `pywind.bmreports.unit.PowerPackUnits`

Bases: `pywind.bmreports.unit.BaseUnitClass`

Download the latest Power Pack modules spreadsheet and make the list of stations available as a list.

SHEET_NAME = ‘Sheet1’

XLS_URL = ‘<http://www.bmreports.com/bsp/staticdata/PowerPackModules.xls>’

class `pywind.bmreports.unit.UnitData` (**args, **kwargs*)

Bases: `object`

Class that gets data about Balancing Mechanism Units from the Balancing Mechanism website.

CX_TYPE = {‘E’: ‘Embedded in Distribution System’, ‘G’: ‘Supplier (base)’, ‘I’: ‘Interconnector User’, ‘M’: ‘Other’, ‘S’:

DURATION = {‘S’: ‘Short’, ‘L’: ‘Long’}

HOST = ‘<http://www.bmreports.com>’

TYPES = {‘Derived’: ‘DerivedBMUnit’, ‘BSV’: ‘/servlet/com.logica.neta.bwp_PanBsvServlet’, ‘Dynamic’: ‘/servlet/com.l

as_dict ()

get_data ()

Get the report data and update.

Returns True or False

Return type bool

rows ()

Generator to provide export data.

Return type dict

Returns Dict formatted for internal export functions.

save_original (*filename*)

Save the downloaded certificate data into the filename provided.

Parameters **filename** – Filename to save the file to.

Returns True or False

Return type bool

class `pywind.bmreports.unit.UnitList`

Bases: `pywind.bmreports.unit.BaseUnitClass`

Get a list of the Balancing Mechanism Units with their Fuel Type and dates.

SHEET_NAME = 'BMU Fuel Types'

XLS_URL = 'http://www.bmreports.com/bsp/staticdata/BMUFuelType.xls'

by_fuel_type (*fuel*)

Return data filtered by fuel type.

Parameters **fuel** – The fuel type to return details for.

Return type list

`pywind.decc`

Note: Following the July 2016 announcements of changes in the organisation of government departments, the future of the DECC website is uncertain. While the data provided is likely still be available the module may need adjusting to continue functioning.

`pywind.decc.Report`

The DECC publish monthly extracts of planning applications for renewable projects. This module aims to make accessing this report simpler.

<https://www.gov.uk/government/publications/renewable-energy-planning-database-monthly-extract>

class `pywind.decc.extract.DeccRecord` (*app_info*)

Bases: `object`

Simple class to hold details of one DECC station.

BOOLEAN_FIELDS = ('chp_enabled', 'green_belt', 'national_park', 'aonb', 'heritage_coast', 'special_landscape_area', 'e

DATE_FIELDS = ('record_last_updated_dd_mm_yyyy', 'planning_application_submitted', 'planning_application_withd

FLOAT_FIELDS = ('installed_capacity_mwelec', 'ro_banding_roc_mwh', 'fit_tariff_p_kwh', 'cfd_capacity_mw', 'turbine

INT_FIELDS = ('ref_id', 'no_of_turbines', 'x-coordinate', 'y-coordinate')

fit_rate_mwh()
Convert the FIT Tariff rate into GBP per MWh.

Return type float

class `pywind.decc.extract.MonthlyExtract` (*filename=None*)
Bases: `object`

The MonthlyExtract class allows the current monthly data to be easily retrieved and parsed.

Note: The CSV data returned does not declare an encoding, so latin1 is presently assumed.

BASE_URL = 'https://www.gov.uk'

URL = 'https://www.gov.uk/government/publications/renewable-energy-planning-database-monthly-extract'

get_data()
Get the data from the DECC server and parse it into DECC records.

Returns True or False

Return type bool

rows()
Generator that returns records

Returns Dict of planning application information

Return type dict

save_original (*filename*)
Save the downloaded certificate data into the filename provided.

Parameters **filename** – Filename to save the file to.

Returns True or False

Return type bool

`pywind.decc.geo`

The Ordnance Survey publishes a comprehensive document on their co-ordinate systems which is available at <https://www.ordnancesurvey.co.uk/docs/support/guide-coordinate-systems-great-britain.pdf>

The conversion routines used in the Coord class are from blog posts by Hannah Fry and remain her copyright.

- <http://www.hannahfry.co.uk/blog/2012/02/01/converting-latitude-and-longitude-to-british-national-grid>
- <http://www.hannahfry.co.uk/blog/2012/02/01/converting-british-national-grid-to-latitude-and-longitude-ii>

class `pywind.decc.geo.Coord` (*e_or_lat, n_or_lon*)
Bases: `object`

Each instance of the Coord class represents a single co-ordinate, which can be created using either WGS84 or OSGB36. Conversions between the 2 systems are done as needed on demand.

__init__ (*e_or_lat, n_or_lon*)

Create the object with the initial point. Values should be supplied as float values, but integers can also be used. Providing anything else will result in a ValueError exception.

Parameters

- **e_or_lat** – Either the Easting or Latitude.

- `n_or_lon` – Either the Northing or Longitude

Raises ValueError

`as_osgb36` (*precision=None*)

Return the co-ordinate as OSGB36 Easting, Northing pair.

Parameters `precision` – The number of decimal places to return. Defaults to 3.

Return type float, float

`as_wgs84` (*precision=None*)

Return the co-ordinate as WGS84 latitude, longitude pair.

Parameters `precision` – The number of decimal places to return. Defaults to 4.

Return type float, float

`pywind.decc.utils`

`pywind.decc.utils.decimal_to_degrees` (*num, seps=None*)

Given a float value and an optional set of separators, return a formatted string.

If the separator has a precision value required that differs from the default of 2, then it should be prefixed with a colon, e.g. 3:\$ would give a separator of a \$ character after a 3 digit number.

Parameters

- `num` – The float value to parse
- `seps` – An optional list of separators to use.

`pywind.decc.utils.latlon_as_string` (*lat, lon*)

Given a numeric lat/lon, convert into “pretty strings”.

Parameters

- `lat` – Latitude
- `lon` – Longitude

Return type str

`pywind.elexon`

Warning: Use of the Elexon API requires registration and use of an API key. Registration is free.

Details of the Elexon API can be found in <https://www.elexon.co.uk/wp-content/uploads/2016/10/Application-Programming-Interfaces-API-and-Data-Push-user-guide.pdf>

`pywind.elexon.api`

Information taken from the Elexon API User Guide.

<https://www.elexon.co.uk/wp-content/uploads/2016/10/Application-Programming-Interfaces-API-and-Data-Push-user-guide.pdf>

class `pywind.elexon.api.B1320` (*apikey*)

Bases: `pywind.elexon.api.ElexonAPI`

```

XML_MAPPING = ['timeSeriesID', 'settlementDate', 'settlementPeriod', 'quantity', 'flowDirection', 'reasonCode', 'documentType']
post_item_cleanup (item)
rows ()
class pywind.elexon.api.B1330 (apikey)
    Bases: pywind.elexon.api.ElexonAPI
    XML_MAPPING = ['timeSeriesID', 'year', 'month', 'congestionAmount', 'documentType', 'processType', 'businessType', 'reasonCode']
    post_item_cleanup (item)
    rows ()
class pywind.elexon.api.B1420 (apikey)
    Bases: pywind.elexon.api.ElexonAPI
    XML_MAPPING = ['documentType', 'businessType', 'processType', 'timeSeriesID', 'powerSystemResourceType', 'year', 'reasonCode']
    post_item_cleanup (item)
    rows ()
class pywind.elexon.api.B1610 (apikey)
    Bases: pywind.elexon.api.ElexonAPI
    post_item_cleanup (item)
    rows ()
class pywind.elexon.api.B1630 (apikey)
    Bases: pywind.elexon.api.ElexonAPI
    XML_MAPPING = ['documentType', 'businessType', 'processType', 'timeSeriesID', 'quantity', 'curveType', 'resolution', 'reasonCode']
    rows ()
class pywind.elexon.api.BMUNITSEARCH (apikey=None)
    Bases: pywind.elexon.api.ElexonAPI
    Balancing Mechanism Unit Search
    CATEGORIES = {'C': 'Additional Supplier', '2': 'Supplier', 'E': 'Embedded', 'T': 'Directly connected', 'G': 'Unknown', 'U': 'Unconnected'}
    XML_MAPPING = ['recordType', 'bmUnitID', 'bmUnitType', 'leadPartyName', 'ngcBMUnitName', 'activeFlag']
    post_item_cleanup (item)
class pywind.elexon.api.DERBMDATA (apikey=None)
    Bases: pywind.elexon.api.ElexonAPI
    Derived Balancing Mechanism Data
    MULTI_RESULTS = (('bav', '/response/responseBody/bav/responseList/item'), ('oav', '/response/responseBody/oav/responseList/item'))
    post_item_cleanup (item)
class pywind.elexon.api.DERSYSDATA (apikey=None)
    Bases: pywind.elexon.api.ElexonAPI
    Derived System Data
    post_item_cleanup (item)
class pywind.elexon.api.ElexonAPI (apikey=None, report=None)
    Bases: object
    MULTI_RESULTS = None

```

XML_MAPPING = None

get_data (**params)

Get data from the Exelon servers and attempt to parse it into a series of dicts each representing a record. Parameters are passed as a dict. Multiple sets of data are created in the multi member, single sets in items.

post_item_cleanup (item)

Holder for a subclassed function to transform the members of the basic dict into something more useful.

class `pywind.exelon.api.FUELINST` (apikey=None)

Bases: `pywind.exelon.api.ExelonAPI`

Instant Generation by Fuel Type

XML_MAPPING = ['recordType', 'startTimeOfHalfHrPeriod', 'settlementPeriod', 'publishingPeriodCommencingTime', 'c

post_item_cleanup (item)

class `pywind.exelon.api.UOU2T52W` (apikey=None)

Bases: `pywind.exelon.api.ExelonAPI`

National Output Useable by Fuel Type and BM Unit (2-52 Weeks Ahead)

`pywind.exelon.api.make_exelon_url` (report, version)

`pywind.exelon.unit`

class `pywind.exelon.unit.BalancingData` (api_key)

Bases: `object`

Parent class to hold data from the :mod:DERBMDATA API.

get_data (**params)

class `pywind.exelon.unit.BalancingPeriodData`

Bases: `object`

Class that holds the volume and cashflow totals for a single station/period.

add_data (element, item)

bid_rate

offer_rate

class `pywind.exelon.unit.BalancingUnitData` (item)

Bases: `object`

Class to hold information about a single station for multiple periods

add_data (element, item)

`pywind.ofgem`

`pywind.ofgem.objects`

class `pywind.ofgem.objects.CertificateStation` (name, g_id, capacity, scheme)

Bases: `object`

We are normally interested in knowing about certificates issued to a station, so this class attempts to simplify this process. Once issued all certificates will be accounted for, but the final owner and status may change. This

class attempts to take a bunch of Certificate objects and simplify them into a final set, with ownership and status correctly attributed.

add_cert (*cert*)

as_row ()

class `pywind.ofgem.objects.Certificates` (*node*)

Bases: `pywind.ofgem.objects.OfgemObjectBase`

Certificate Number Fact Sheet https://www.ofgem.gov.uk/sites/default/files/docs/roc_identifier_fact_sheet_dec_2015.pdf

XML_MAPPING = (('textbox4', 'generator_id'), ('textbox13', 'name'), ('textbox5', 'scheme'), ('textbox19', 'capacity', 'floa

certificates

Number of certificates covered by this object.

Return type `int`

digits

Number of digits that store the certificate number.

Return type `int`

finish

Return the numeric finish number for the certificates. Each certificate number contains the station, period and the number of the certificate, so this function extracts the numeric part.

Returns Finish number of the certificates referenced

Return type `integer`

output

Calculate the output based on the number of certs issued and factor.

Returns Numeric output or 0

Return type `float`

output_summary ()

Return a string with the output for the certificates.

Return type `str`

start

Return the numeric start number for the certificates. Each certificate number contains the station, period and the number of the certificate, so this function extracts the numeric part.

Returns Start number of the certificates referenced

Return type `int`

station_details ()

Get a dict object with the station information for these certificates.

Returns Dict with just information relevant to identifying the station

Return type `dict`

class `pywind.ofgem.objects.OfgemObjectBase` (*node*)

Bases: `object`

XML_MAPPING = `None`

as_row ()

Return the information in correct format for `rows` () usage

Returns Formatted attribute dict

Return type dict

class `pywind.ofgem.objects.Station` (*node*)

Bases: `pywind.ofgem.objects.OfgemObjectBase`

Store details of a single station using data from Ofgem.

The exposed object makes the individual pieces of data available by acting as a dict, i.e. ...code:

```
name = station['name']
```

The convenience function `as_string()` will return a full list of the data formatted for display in a terminal.

XML_MAPPING = (('GeneratorID', 'generator_id'), ('StatusName', 'status'), ('GeneratorName', 'name'), ('SchemeName', 'scheme_name'))

`pywind.ofgem.search`

class `pywind.ofgem.search.CertificateSearch` (*filename=None*)

Bases: `object`

Getting information about certificates issued by Ofgem requires accessing their webform. This class provides a simple way of doing that. Class that queries ofgem for certificate data. If it succeeds then

There are 2 generator methods that allow iterating through the returned data, - each call to `stations()` will return a list of `Certificates` objects related to a single station. - each call to `certificates()` will return a single `Certificates` object.

```
>>> from pywind.ofgem.CertificateSearch import CertificateSearch
>>> ocs = CertificateSearch()
>>> ocs.start()
True
>>> ocs.set_period(201601)
True
>>> ocs.get_data()
True
>>> len(ocs)
4898
```

NSMAP = {'a': 'CertificatesExternalPublicDataWarehouse'}

START_URL = 'ReportViewer.aspx?ReportPath=/DatawarehouseReports/CertificatesExternalPublicDataWarehouse&ReportID=1'

certificates ()

Generator that returns `Certificates` objects.

Returns `Certificates` objects

Return type `Certificates`

filter_generation_type (*what*)

Filter certificates by generation type

filter_generator_id (*acc_no*)

Filter certificates by generator id (accreditation number).

Note: Values supplied are upper cased automatically.

Parameters `acc_no` – Accreditation/Generation number

Return type `bool`

filter_scheme (*what*)

Filter certificates by scheme

Parameters **what** – Scheme abbreviation [REGO, RO]

Return type `bool`

filter_technology (*what*)

Filter certificates by technology group

get_data ()

Submit the form, get the results and parse them into `Certificate` objects

Return type `bool`

parse_filename (*filename*)

Parse an Ofgem generated file of certificates. This parses downloaded Ofgem files.

Parameters **filename** – The filename to be parsed

Returns True or False

Return type `bool`

rows ()

Generator function that returns a station each time it is called.

Returns A function that returns a dict containing information on one station.

Return type generator

save_original (*filename*)

Save the downloaded certificate data into the filename provided.

Parameters **filename** – Filename to save the file to.

Return type `bool`

set_finish_month (*month*)

Set the finish month for certificates

Parameters **month** – Numeric month number

Return type `bool`

set_finish_year (*year*)

Set the finish year for certificates

Parameters **year** – Numeric year to be set

Return type `bool`

set_period (*yearmonth*)

Set the year and month for certificates.

Parameters **yearmonth** – Numeric period in YYYYMM format

Returns True or False

Return type `bool`

set_start_month (*month*)

Set the start month for certificates

Parameters **month** – Numeric month number

Return type `bool`

set_start_year (*year*)

Set the start year for certificates

Parameters *year* – Numeric year to be set

Return type `bool`

start ()

Retrieve the form from Ofgem website so we can start updating it.

Returns True or False

Return type `bool`

stations ()

Generator that returns a Return a list of stations related to the certificates

class `pywind.ofgem.search.StationSearch`

Bases: `object`

Performing a station search using the Ofgem website takes a while due to the 3.5M initial file and the 2M replies that are sent. Parsing these takes time, so patience is needed.

```
>>> from pywind.ofgem.StationSearch import StationSearch
>>> oss = StationSearch()
>>> oss.start()
True
>>> oss.filter_name('griffin')
True
>>> oss.get_data()
True
>>> len(oss)
4
```

START_URL = `'ReportViewer.aspx?ReportPath=/Renewables/Accreditation/AccreditedStationsExternalPublic&ReportV`

filter_generator_id (*accno*)

Filter stations based on generator id.

Return type `bool`

filter_name (*name*)

Filter stations based on name. The search will return all stations containing the supplied name.

Parameters *name* – The name to filter for

Return type `bool`

filter_organisation (*org_name*)

Filter stations based on generator id.

Parameters *org_name* – Organisation name to filter

Return type `bool`

filter_scheme (*scheme*)

Filter stations based on scheme they are members of.

Return type `bool`

filter_technology (*what*)

Filter stations based on technology.

Return type `bool`

get_data ()

Get data from form.

Return type `bool`

rows ()

Generator to return dicts of station information.

Returns Dict of station information

Return type `dict`

save_original (*filename*)

Save the downloaded station data into the filename provided.

Parameters **filename** – Filename to save the file to.

Return type `bool`

start ()

Retrieve the form from Ofgem website so we can start updating it.

`pywind.ofgem.form`

To get data from the Ofgem website we need to use one of their web forms. These use MS javascript to work and were originally only usable in IE 8 or IE9. Modern versions are usable in more browsers but the forms themselves have also evolved and are more complex than is ideal.

class `pywind.ofgem.form.OfgemForm` (*url*)

Bases: `object`

Class to represent an instance of an Ofgem form.

get ()

Attempt to get the initial version of the form from the website.

save_original (*filename*)

Save the original, downloaded source into the filename provided.

Parameters **filename** – Filename to save the file to.

Returns True or False

Return type `boolean`

set_value (*lbl, value*)

submit ()

Submit the form data and update based on response. Given how slow the parsing of a 3M HTML page is, try and use the X-MicrosoftAjax: Delta=true header to get smaller blocks for processing.

update ()

Submit the form data and update based on response. Given how slow the parsing of a 3M HTML page is, try and use the X-MicrosoftAjax: Delta=true header to get smaller blocks for processing.

`pywind.ofgem.form_data`

Each Ofgem web form contains a lot of information. Classes in this file try to make managing the data easier.

`class pywind.ofgem.form_data.FormData (initial_data='', stored_file=None)`

Bases: `object`

Class to store and allow easy manipulation of data from an Ofgem form.

`as_post_data (quoted=True, submit=False)`

Process the form elements and return in a dict suitable for using as POST data.

Parameters

- **quoted** – If set the returned data will be fully quoted.
- **submit** – True only if this is a submission post.

Returns Dict of data to be posted as name: value pairs

Return type `dict`

`set_value_by_label (lbl, value)`

Set a value based on a label.

`update (content='')`

Given some content, update the form.

Parameters **content** – The content to update the form from

Returns True or False

Return type `bool`

`value_for_label (lbl)`

`pywind.ofgem.form_data.element_attributes (elm)`

Return a dict of the basic attributes we want from an XML element.

`pywind.ofgem.form_data.quote ('abc def') → 'abc%20def'`

Each part of a URL, e.g. the path info, the query, etc., has a different set of reserved characters that must be quoted.

RFC 2396 Uniform Resource Identifiers (URI): Generic Syntax lists the following reserved characters.

reserved = `"," | "/" | "?" | ":" | "@" | "&" | "=" | "+" | "$" | "`

Each of these characters is reserved in some component of a URL, but not necessarily in all of them.

By default, the quote function is intended for quoting the path section of a URL. Thus, it will not encode `'/`. This character is reserved, but in typical usage the quote function is being called on a path where the existing slash characters are used as reserved characters.

`pywind.ofgem.form_data.selected_list (element)`

Given an element dict, return a list of the indexes.

`pywind.roc`

`pywind.roc.eroc`

`class pywind.roc.eroc.EROCPrices`

Bases: `object`

The `EROCPrices` class provides access to the auction data from the eRIC website. Once created the `get_prices()` is called to get the latest information from their website and parse the results into period prices.

Once parsed period information can be accessed by using the class object as a dict, i.e. `eroc[200701]` would return the average price for auction(s) held in Jan 2007.

```
>>> from pywind.roc.eroc import EROCPrices
>>> eroc = EROCPrices()
>>> eroc.get_prices()
True
>>> eroc[200701]
46.17
```

URL = 'http://www.epowerauctions.co.uk/erocrecord.htm'

get_prices ()

Get the recent prices.

Returns True or False

Return type bool

prices ()

Generator to return the calculated price data.

Returns Tuple of period, price

Return type tuple

process_file (local_file)

Process a local HTML file.

Parameters `local_file` – Filename to be parsed.

Returns True or False

Return type bool

rows ()

Generator to return the full auction data as rows.

Returns Dict of price data

Return type dict

`pywind.roc.eroc.parse_date_string (dstr)`

CHAPTER 4

Background

When I started looking at using available online information for renewable projects I was surprised at just how hard it was to access. This package is the evolution of those attempts. It aims to provide a simple way to access and then use the information and as such is constantly evolving.

All of the sources referenced are freely available, though some have (*unclear*) restrictions on using the data. I have tried to include warnings in the documentation when this is the case.

My documentation skills are still weak, so there may well be errors in these documents - please let me know and I'll try to fix them!

CHAPTER 5

Travis CI

In an effort to help find errors, I've enabled pywind for TravisCI. <https://travis-ci.org/>

To view the latest results click on the badge below :-)

CHAPTER 6

Python 3 Support

I am aiming to make the package fully Python 3 compatible, but my initial focus is Python 2.7 as that is the version that it will be immediately used with.

At the present time, support is partial and many things will not work as expected.

This is free and unencumbered software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or distribute this software, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

In jurisdictions that recognize copyright laws, the author or authors of this software dedicate any and all copyright interest in the software to the public domain. We make this dedication for the benefit of the public at large and to the detriment of our heirs and successors. We intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this software under copyright law.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

For more information, please refer to <http://unlicense.org/>

CHAPTER 8

ToDo List

There are many improvements that the `pywind` package could benefit from, but these are the ones currently on my radar!

- improve test coverage
- add class to “accumulate” and “group” certificates to produce summary information

CHAPTER 9

Patches Welcome!

I'm always keen to improve the package, so if you have suggestions, comments, patches or even better pull requests - send them along!

CHAPTER 10

Contributors

The following people have helped...

- energynumbers

CHAPTER 11

Indices and tables

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

p

pywind.bmreports.generation_type, 16
pywind.bmreports.prices, 18
pywind.bmreports.unit, 18
pywind.decc.extract, 20
pywind.decc.geo, 21
pywind.decc.utils, 22
pywind.elexon.api, 22
pywind.elexon.unit, 24
pywind.export, 13
pywind.log, 13
pywind.ofgem.form, 29
pywind.ofgem.form_data, 29
pywind.ofgem.objects, 24
pywind.ofgem.search, 26
pywind.roc.eroc, 30
pywind.utils, 14

s

sample_scripts.annual_output, 10
sample_scripts.capacity_ro, 9
sample_scripts.derived_unit_data, 11
sample_scripts.ofgem_certificate_search,
11
sample_scripts.ofgem_station_search, 11

Symbols

`__init__()` (pywind.decc.geo.Coord method), 21

A

`add_cert()` (pywind.ofgem.objects.CertificateStation method), 25

`add_data()` (pywind.elexon.unit.BalancingPeriodData method), 24

`add_data()` (pywind.elexon.unit.BalancingUnitData method), 24

`args_get_datetime()` (in module pywind.utils), 14

`as_dict()` (pywind.bmreports.generation_type.GenerationData method), 17

`as_dict()` (pywind.bmreports.generation_type.GenerationPeriod method), 17

`as_dict()` (pywind.bmreports.generation_type.GenerationRecord method), 17

`as_dict()` (pywind.bmreports.prices.SystemPrices method), 18

`as_dict()` (pywind.bmreports.unit.UnitData method), 19

`as_osgb36()` (pywind.decc.geo.Coord method), 22

`as_post_data()` (pywind.ofgem.form_data.FormData method), 30

`as_row()` (pywind.ofgem.objects.CertificateStation method), 25

`as_row()` (pywind.ofgem.objects.OfgemObjectBase method), 25

`as_wgs84()` (pywind.decc.geo.Coord method), 22

B

B1320 (class in pywind.elexon.api), 22

B1330 (class in pywind.elexon.api), 23

B1420 (class in pywind.elexon.api), 23

B1610 (class in pywind.elexon.api), 23

B1630 (class in pywind.elexon.api), 23

BalancingData (class in pywind.elexon.unit), 24

BalancingPeriodData (class in pywind.elexon.unit), 24

BalancingUnitData (class in pywind.bmreports.unit), 18

BalancingUnitData (class in pywind.elexon.unit), 24

BASE_URL (pywind.decc.extract.MonthlyExtract attribute), 21

BaseUnitClass (class in pywind.bmreports.unit), 19

bid_cashflow (pywind.bmreports.unit.BalancingUnitData attribute), 18

bid_rate (pywind.elexon.unit.BalancingPeriodData attribute), 24

bid_volume (pywind.bmreports.unit.BalancingUnitData attribute), 18

BMUNITSEARCH (class in pywind.elexon.api), 23

BOOLEAN_FIELDS (pywind.decc.extract.DeccRecord attribute), 20

by_fuel_type() (pywind.bmreports.unit.UnitList method), 20

C

CATEGORIES (pywind.elexon.api.BMUNITSEARCH attribute), 23

Certificates (class in pywind.ofgem.objects), 25

certificates (pywind.ofgem.objects.Certificates attribute), 25

certificates() (pywind.ofgem.search.CertificateSearch method), 26

CertificateSearch (class in pywind.ofgem.search), 26

CertificateStation (class in pywind.ofgem.objects), 24

commandline_parser() (in module pywind.utils), 14

Coord (class in pywind.decc.geo), 21

CX_TYPE (pywind.bmreports.unit.UnitData attribute), 19

D

DATE_FIELDS (pywind.decc.extract.DeccRecord attribute), 20

DeccRecord (class in pywind.decc.extract), 20

decimal_to_degrees() (in module pywind.decc.utils), 22

DERBMDATA (class in pywind.elexon.api), 23

DERSYSDATA (class in pywind.elexon.api), 23

digits (pywind.ofgem.objects.Certificates attribute), 25

DT1 (pywind.bmreports.generation_type.GenerationPeriod attribute), 17

- DT2 (pywind.bmreports.generation_type.GenerationPeriod attribute), 17
- DURATION (pywind.bmreports.unit.UnitData attribute), 19
- ## E
- element_attributes() (in module pywind.ofgem.form_data), 30
- ElexonAPI (class in pywind.elexon.api), 23
- EROCPrices (class in pywind.roc.eroc), 30
- export_to_file() (in module pywind.export), 13
- ## F
- filter_generation_type() (pywind.ofgem.search.CertificateSearch method), 26
- filter_generator_id() (pywind.ofgem.search.CertificateSearch method), 26
- filter_generator_id() (pywind.ofgem.search.StationSearch method), 28
- filter_name() (pywind.ofgem.search.StationSearch method), 28
- filter_organisation() (pywind.ofgem.search.StationSearch method), 28
- filter_scheme() (pywind.ofgem.search.CertificateSearch method), 27
- filter_scheme() (pywind.ofgem.search.StationSearch method), 28
- filter_technology() (pywind.ofgem.search.CertificateSearch method), 27
- filter_technology() (pywind.ofgem.search.StationSearch method), 28
- finish (pywind.ofgem.objects.Certificates attribute), 25
- fit_rate_mwh() (pywind.decc.extract.DeccRecord method), 20
- FLOAT_FIELDS (pywind.decc.extract.DeccRecord attribute), 20
- formatter() (pywind.utils.StdoutFormatter method), 14
- FormData (class in pywind.ofgem.form_data), 29
- FUELINST (class in pywind.elexon.api), 24
- FUELS (pywind.bmreports.generation_type.GenerationRecord attribute), 17
- ## G
- GenerationData (class in pywind.bmreports.generation_type), 16
- GenerationPeriod (class in pywind.bmreports.generation_type), 17
- GenerationRecord (class in pywind.bmreports.generation_type), 17
- get() (pywind.ofgem.form.OfgemForm method), 29
- get_data() (pywind.bmreports.generation_type.GenerationData method), 17
- get_data() (pywind.bmreports.prices.SystemPrices method), 18
- get_data() (pywind.bmreports.unit.UnitData method), 19
- get_data() (pywind.decc.extract.MonthlyExtract method), 21
- get_data() (pywind.elexon.api.ElexonAPI method), 24
- get_data() (pywind.elexon.unit.BalancingData method), 24
- get_data() (pywind.ofgem.search.CertificateSearch method), 27
- get_data() (pywind.ofgem.search.StationSearch method), 29
- get_list() (pywind.bmreports.unit.BaseUnitClass method), 19
- get_or_post_a_url() (in module pywind.utils), 14
- get_prices() (pywind.roc.eroc.EROCPrices method), 31
- ## H
- hh() (pywind.bmreports.generation_type.GenerationPeriod method), 17
- HOST (pywind.bmreports.unit.UnitData attribute), 19
- ## I
- inst() (pywind.bmreports.generation_type.GenerationPeriod method), 17
- INT_FIELDS (pywind.decc.extract.DeccRecord attribute), 20
- ## K
- keyname() (pywind.bmreports.generation_type.GenerationPeriod method), 17
- ## L
- last24h() (pywind.bmreports.generation_type.GenerationPeriod method), 17
- latlon_as_string() (in module pywind.decc.utils), 22
- ## M
- make_elexon_url() (in module pywind.elexon.api), 24
- map_xml_to_dict() (in module pywind.utils), 15
- MonthlyExtract (class in pywind.decc.extract), 21
- multi_level_get() (in module pywind.utils), 15
- MULTI_RESULTS (pywind.elexon.api.DERBMDATA attribute), 23
- MULTI_RESULTS (pywind.elexon.api.ElexonAPI attribute), 23
- ## N
- NAMES (pywind.bmreports.generation_type.GenerationPeriod attribute), 17

- NSMAP (pywind.ofgem.search.CertificateSearch attribute), 26
- ## O
- offer_cashflow (pywind.bmreports.unit.BalancingUnitData attribute), 18
- offer_rate (pywind.elexon.unit.BalancingPeriodData attribute), 24
- offer_volume (pywind.bmreports.unit.BalancingUnitData attribute), 18
- OfgemForm (class in pywind.ofgem.form), 29
- OfgemObjectBase (class in pywind.ofgem.objects), 25
- output (pywind.ofgem.objects.Certificates attribute), 25
- output_summary() (pywind.ofgem.objects.Certificates method), 25
- ## P
- PARAMS (pywind.bmreports.generation_type.GenerationData attribute), 17
- parse_date_string() (in module pywind.roc.eroc), 31
- parse_filename() (pywind.ofgem.search.CertificateSearch method), 27
- parse_response_as_xml() (in module pywind.utils), 15
- post_item_cleanup() (pywind.elexon.api.B1320 method), 23
- post_item_cleanup() (pywind.elexon.api.B1330 method), 23
- post_item_cleanup() (pywind.elexon.api.B1420 method), 23
- post_item_cleanup() (pywind.elexon.api.B1610 method), 23
- post_item_cleanup() (pywind.elexon.api.BMUNITSEARCH method), 23
- post_item_cleanup() (pywind.elexon.api.DERBMDATA method), 23
- post_item_cleanup() (pywind.elexon.api.DERSYSDATA method), 23
- post_item_cleanup() (pywind.elexon.api.ElexonAPI method), 24
- post_item_cleanup() (pywind.elexon.api.FUELINST method), 24
- PowerPackUnits (class in pywind.bmreports.unit), 19
- prices() (pywind.roc.eroc.EROCPrices method), 31
- process_file() (pywind.roc.eroc.EROCPrices method), 31
- pywind.bmreports.generation_type (module), 16
- pywind.bmreports.prices (module), 18
- pywind.bmreports.unit (module), 18
- pywind.decc.extract (module), 20
- pywind.decc.geo (module), 21
- pywind.decc.utils (module), 22
- pywind.elexon.api (module), 22
- pywind.elexon.unit (module), 24
- pywind.export (module), 13
- pywind.log (module), 13
- pywind.ofgem.form (module), 29
- pywind.ofgem.form_data (module), 29
- pywind.ofgem.objects (module), 24
- pywind.ofgem.search (module), 26
- pywind.roc.eroc (module), 30
- pywind.utils (module), 14
- ## Q
- quote() (in module pywind.ofgem.form_data), 30
- ## R
- rate() (pywind.bmreports.unit.BalancingUnitData method), 19
- row() (pywind.utils.StdoutFormatter method), 14
- rows() (pywind.bmreports.generation_type.GenerationData method), 17
- rows() (pywind.bmreports.prices.SystemPrices method), 18
- rows() (pywind.bmreports.unit.BaseUnitClass method), 19
- rows() (pywind.bmreports.unit.UnitData method), 20
- rows() (pywind.decc.extract.MonthlyExtract method), 21
- rows() (pywind.elexon.api.B1320 method), 23
- rows() (pywind.elexon.api.B1330 method), 23
- rows() (pywind.elexon.api.B1420 method), 23
- rows() (pywind.elexon.api.B1610 method), 23
- rows() (pywind.elexon.api.B1630 method), 23
- rows() (pywind.ofgem.search.CertificateSearch method), 27
- rows() (pywind.ofgem.search.StationSearch method), 29
- rows() (pywind.roc.eroc.EROCPrices method), 31
- ## S
- sample_scripts.annual_output (module), 10
- sample_scripts.capacity_ro (module), 9
- sample_scripts.derived_unit_data (module), 11
- sample_scripts.ofgem_certificate_search (module), 11
- sample_scripts.ofgem_station_search (module), 11
- save_original() (pywind.bmreports.generation_type.GenerationData method), 17
- save_original() (pywind.bmreports.prices.SystemPrices method), 18
- save_original() (pywind.bmreports.unit.BaseUnitClass method), 19
- save_original() (pywind.bmreports.unit.UnitData method), 20
- save_original() (pywind.decc.extract.MonthlyExtract method), 21
- save_original() (pywind.ofgem.form.OfgemForm method), 29
- save_original() (pywind.ofgem.search.CertificateSearch method), 27

save_original() (pywind.ofgem.search.StationSearch method), 29

selected_list() (in module pywind.ofgem.form_data), 30

set_finish_month() (pywind.ofgem.search.CertificateSearch method), 27

set_finish_year() (pywind.ofgem.search.CertificateSearch method), 27

set_period() (pywind.ofgem.search.CertificateSearch method), 27

set_start_month() (pywind.ofgem.search.CertificateSearch method), 27

set_start_year() (pywind.ofgem.search.CertificateSearch method), 28

set_value() (pywind.ofgem.form.OfgemForm method), 29

set_value_by_label() (pywind.ofgem.form_data.FormData method), 30

setup_logging() (in module pywind.log), 13

SHEET_NAME (pywind.bmreports.unit.BaseUnitClass attribute), 19

SHEET_NAME (pywind.bmreports.unit.PowerPackUnits attribute), 19

SHEET_NAME (pywind.bmreports.unit.UnitList attribute), 20

start (pywind.ofgem.objects.Certificates attribute), 25

start() (pywind.ofgem.search.CertificateSearch method), 28

start() (pywind.ofgem.search.StationSearch method), 29

START_URL (pywind.ofgem.search.CertificateSearch attribute), 26

START_URL (pywind.ofgem.search.StationSearch attribute), 28

Station (class in pywind.ofgem.objects), 26

station_details() (pywind.ofgem.objects.Certificates method), 25

stations() (pywind.ofgem.search.CertificateSearch method), 28

StationSearch (class in pywind.ofgem.search), 28

StdoutFormatter (class in pywind.utils), 14

submit() (pywind.ofgem.form.OfgemForm method), 29

SystemPrices (class in pywind.bmreports.prices), 18

T

titles() (pywind.utils.StdoutFormatter method), 14

TYPES (pywind.bmreports.unit.UnitData attribute), 19

U

UnitData (class in pywind.bmreports.unit), 19

UnitList (class in pywind.bmreports.unit), 20

UOU2T52W (class in pywind.elexon.api), 24

update() (pywind.ofgem.form.OfgemForm method), 29

update() (pywind.ofgem.form_data.FormData method), 30

URL (pywind.bmreports.generation_type.GenerationData attribute), 17

URL (pywind.bmreports.prices.SystemPrices attribute), 18

URL (pywind.decc.extract.MonthlyExtract attribute), 21

URL (pywind.roc.eroe.EROEPrices attribute), 31

V

valid_date() (in module pywind.utils), 16

valid_time() (in module pywind.utils), 16

value_for_label() (pywind.ofgem.form_data.FormData method), 30

X

XLS_URL (pywind.bmreports.unit.BaseUnitClass attribute), 19

XLS_URL (pywind.bmreports.unit.PowerPackUnits attribute), 19

XLS_URL (pywind.bmreports.unit.UnitList attribute), 20

xml_attr_or_element() (in module pywind.utils), 16

XML_MAPPING (pywind.elexon.api.B1320 attribute), 22

XML_MAPPING (pywind.elexon.api.B1330 attribute), 23

XML_MAPPING (pywind.elexon.api.B1420 attribute), 23

XML_MAPPING (pywind.elexon.api.B1630 attribute), 23

XML_MAPPING (pywind.elexon.api.BMUNITSEARCH attribute), 23

XML_MAPPING (pywind.elexon.api.ElexonAPI attribute), 23

XML_MAPPING (pywind.elexon.api.FUELINST attribute), 24

XML_MAPPING (pywind.ofgem.objects.Certificates attribute), 25

XML_MAPPING (pywind.ofgem.objects.OfgemObjectBase attribute), 25

XML_MAPPING (pywind.ofgem.objects.Station attribute), 26