
SatNOGS Client

Release 1.8.1+47.gf2e0c2f.dirty

SatNOGS

Jul 24, 2023

CONTENTS

1 Table of Contents	3
1.1 User guide	3
1.1.1 Requirements	3
1.1.2 Installation	3
1.1.2.1 Debian	3
1.1.2.2 SatNOGS Client	3
1.1.3 Configuration	4
1.1.3.1 Environment variables	4
1.1.4 Usage	21
1.1.5 Doppler-corrected I/Q Streaming via UDP	21
1.2 Developer guide	21
1.2.1 Installation	21
1.2.2 Configuration	22
1.2.3 Code Quality Assurance	22
1.2.4 Testing	22
1.2.4.1 System testing	22
1.2.5 Automation	22
1.2.5.1 Environments	23
1.3 Working with Docker	23
1.3.1 Requirements	23
1.3.2 Building	23
1.3.3 Configuration	24
1.3.4 Usage	24
1.4 Modules reference	24
1.4.1 satnogsclient	24
1.4.2 satnogsclient.locator.locator	24
1.4.3 satnogsclient.observer.observer	24
1.4.4 satnogsclient.observer.orbital	25
1.4.5 satnogsclient.artifacts	25
1.4.6 satnogsclient.waterfall	25
1.4.7 satnogsclient.observer.worker	26
1.4.8 satnogsclient.rig	28
1.4.9 satnogsclient.rotator	29
1.4.10 satnogsclient.scheduler.tasks	30
1.4.11 satnogsclient.settings	30
1.4.12 satnogsclient.radio.flowgraphs	31
2 Indices and tables	33
Python Module Index	35

SatNOGS Client is the software responsible for automating satellite signal reception for SatNOGS. The main functions of SatNOGS Client is to pull observation jobs from SatNOGS Network and execute them on the a station host (usually a Raspberry Pi). These observation jobs create locally scheduled task which spawn SatNOGS Radio scripts for decoding and demodulating the signals. In addition to that, SatNOGS Client can optionally control a Hamlib compatible rotator to track the satellite. During the execution of an observation task, orbital calculations are performed to compensate for doppler shift and point the rotator to the correct direction. After the completion of an observation, the client gathers the observation artifacts and uploads them to SatNOGS Network.

CHAPTER
ONE

TABLE OF CONTENTS

1.1 User guide

1.1.1 Requirements

- Python 3.6+
- Hamlib 4.0+ Python bindings
- GPSD 3.22+ Python bindings
- ATLAS 3.10+ library
- HDF5 1.10+ library

1.1.2 Installation

1.1.2.1 Debian

To install the required dependencies in Debian Bullseye run:

```
$ xargs -r -a packages.debian apt-get install
```

1.1.2.2 SatNOGS Client

To install SatNOGS Client run:

```
$ pip install satnogs-client
```

This will install a console script called `satnogs-client`.

1.1.3 Configuration

Configuration of SatNOGS Client is done through environment variables. The environment variables can also be defined in a file called `.env`, place on the project root directory. The format of each line in `.env` file is `VARIABLE=VALUE`.

1.1.3.1 Environment variables

SATNOGS_API_TOKEN

Type
string

Default
None

Required
Yes

Description

SatNOGS Network API token associated with an account in SatNOGS Network. This token is secret.
It can be found in SatNOGS Network user page.

SATNOGS_PRE_OBSERVATION_SCRIPT

Type
path

Default
None

Required
No

Description

A path to an executable to be executed before an observation job is started. Execution of this script blocks the observation job.

SATNOGS_POST_OBSERVATION_SCRIPT

Type
path

Default
None

Required
No

Description

A path to an executable to be executed after an observation job has finished. Execution of this script blocks the completion of an observation job.

SATNOGS_STATION_ID

Type
integer

Default
None

Required
Yes

Description

The ID of the SatNOGS Network station this client will act as. The station must be owned by the user with the defined API token.

SATNOGS_STATION_LAT

Type
float

Default
None

Required
Yes

Description

Latitude of the station location. Higher precision of this value increases accuracy of Doppler correction while lower precision increases station location privacy.

SATNOGS_STATION_LON

Type
float

Default
None

Required
Yes

Description

Longitude of the station location. Higher precision of this value increases accuracy of Doppler correction while lower precision increases station location privacy.

SATNOGS_STATION_ELEV

Type
integer

Default
None

Required
Yes

Description

Elevation of the station location. Higher precision of this value increases accuracy of Doppler correction while lower precision increases station location privacy.

SATNOGS_GPSD_CLIENT_ENABLED

Type

boolean

Default

False

Required

No

Description

Enable SatNOGS Client to connect to a GPSd daemon to pull positional information. The position is queried once, during SatNOGS Client startup.

SATNOGS_GPSD_HOST

Type

host

Default

127.0.0.1

Required

No

Description

Hostname or IP address of GPSd to connect to for pulling positional information.

SATNOGS_GPSD_PORT

Type

port

Default

2947

Required

No

Description

Port of GPSd to connect to for pulling positional information.

SATNOGS_GPSD_TIMEOUT

Type
integer

Default
0

Required
No

Description

Time to wait until GPSd returns positional information. A value of 0 means to wait indefinitely.

SATNOGS_APP_PATH

Type
path

Default
/tmp/.satnogs

Required
No

Description

Base path for storing output files.

SATNOGS_OUTPUT_PATH

Type
path

Default
/tmp/.satnogs/data

Required
No

Description

Path for storing “output files”. Output files are:

- Audio recordings (`satnogs_{obs_id}_{timestamp}.ogg`)
- Waterfall Images (`waterfall_{obs_id}_{timestamp}.png`)
- Decoded frames (`data_{obs_id}`), either raw, json-encoded or png image data from the APT demodulator

Temporary (potentially incomplete) files are:

- incomplete audio recordings (`receiving_satnogs_{obs_id}_{timestamp}.ogg`)
- incomplete APT images (`receiving_data_{obs_id}_{timestamp}.png`)
- Waterfall data files (`receiving_waterfall_{obs_id}_{timestamp}.dat`)

SatNOGS Artifacts are handled separately.

SATNOGS_COMPLETE_OUTPUT_PATH

Type

path

Default**Required**

No

Description

Path to move output files once they are completed. If set to empty, output files are deleted (default). Make sure to have enough storage available as the selected folder will continuously grow (about 30MB for a 5 minute observation).

SATNOGS_INCOMPLETE_OUTPUT_PATH

Type

path

Default

/tmp/.satnogs/data/incomplete

Required

No

Description

Path for moving incomplete output files.

SATNOGS_REMOVE_RAW_FILES

Type

boolean

Default

True

Required

No

Description

Remove raw data files used for generating waterfalls.

SATNOGS_KEEP_ARTIFACTS

Type

boolean

Default

False

Required

No

Description

Do not remove artifact files after successfully uploading them.

SATNOGS_ARTIFACTS_OUTPUT_PATH

Type
path

Default
/tmp/.satnogs/artifacts

Required
No

Description
Path for storing artifact files.

SATNOGS_VERIFY_SSL

Type
boolean

Default
True

Required
No

Description
Verify SSL certificates for HTTPS requests.

SATNOGS_NETWORK_API_URL

Type
url

Default
https://network.satnogs.org/api/

Required
No

Description
URL pointing to API of SatNOGS Network.

SATNOGS_NETWORK_API_QUERY_INTERVAL

Type
integer

Default
60

Required
No

Description
Interval (in seconds) for pulling jobs from SatNOGS Network API.

SATNOGS_NETWORK_API_POST_INTERVAL

Type
integer

Default
180

Required
No

Description
Interval (in seconds) for posting observation data to SatNOGS Network API.

SATNOGS_ROT_ENABLED

Type
boolean

Default
True

Required
No

Description
Enable SatNOGS Client to connect to an antenna rotator using the Hamlib library.

SATNOGS_ROT_MODEL

Type
string

Default
ROT_MODEL_DUMMY

Required
No

Description
Rotator model to control. This value must be the model string of a Hamlib rotator.

SATNOGS_ROT_BAUD

Type
integer

Default
19200

Required
No

Description
Hamlib rotator serial interface baud rate.

SATNOGS_ROT_PORT

Type

path

Default

/dev/ttyUSB0

Required

No

Description

Path to Hamlib rotator serial port device. The device must be accessible to the user which SatNOGS Client is running.

SATNOGS_ROT_THRESHOLD

Type

integer

Default

4

Required

No

Description

Azimuth/elevation threshold for moving the rotator. Position changes below this threshold will not cause the rotator to move.

SATNOGS_ROT_FLIP

Type

boolean

Default

False

Required

No

Description

Enable rotator flipping during high elevation passes.

SATNOGS_ROT_FLIP_ANGLE

Type

integer

Default

75

Required

No

Description

Elevation angle above which the rotator will flip.

SATNOGS_RIG_IP

Type

host

Default

127.0.0.1

Required

No

Description

Hostname or IP address of Hamlib rotctld.

SATNOGS_RIG_PORT

Type

integer

Default

4532

Required

No

Description

Hamlib rigctld TCP port.

SATNOGS_SOAPY_RX_DEVICE

Type

string

Default

None

Required

Yes

Description

SoapySDR device driver to use for RX. This setting must be defined in the form `driver=<name>` where `<name>` is the name of the SoapySDR device driver to use.

It also allows setting different devices for specific frequency ranges.

The format for multiple devices is a space delimited list of frequency range – device pairs:

```
<min. freq.>-<max. freq>:driver=<name> <min. freq.>-<max. freq.
>:driver=<name> ...
```

Where frequencies are in MHz and `<name>` is the name of the SoapySDR device driver to use for the specified frequency range.

Example: two RTL-SDR receivers: one for VHF (135..148 MHz), another for UHF (430..470 MHz).

```
135-148:driver=rtlsdr,serial=1 430-470:driver=rtlsdr,serial=2
```

SATNOGS_RX_SAMP_RATE

Type
integer

Default
None

Required
Yes

Description

SoapySDR device sample rate. Valid sample rates for attached devices can be queried using SoapySDRUtil --probe.

SATNOGS_RX_BANDWIDTH

Type
integer

Default
Flowgraph-defined

Required
No

Description

SoapySDR device RF bandwidth. This setting configures the RF filter on devices that support it.

SATNOGS_DOPPLER_CORR_PER_SEC

Type
integer

Default
Flowgraph-defined

Required
No

Description

Number of Doppler corrections per second requested by SatNOGS Radio.

SATNOGS_LO_OFFSET

Type
integer

Default
Flowgraph-defined

Required
No

Description

SoapySDR device local oscillator offset to apply. This setting is used to shift the carrier away from the DC spike.

SATNOGS_PPM_ERROR

Type

float

Default

Flowgraph-defined

Required

No

Description

SoapySDR device oscillator frequency error correction to apply. This setting is defined in parts per million.

SATNOGS_GAIN_MODE

Type

string

Default

Overall

Required

No

Description

SoapySDR device gain mode. Valid values are: Overall, Specific, Settings Field.

SATNOGS_RF_GAIN

Type

float

Default

Flowgraph-defined

Required

No

Description

SoapySDR device overall gain, in dB. Device drivers set individual, device specific gains to approximate linearity on the overall gain.

SATNOGS_ANTENNA

Type

string

Default

None

Required

Yes

Description

SoapySDR device antenna to use for RX. Valid antennas for attached devices can be queried using SoapySDRUtil --probe.

SATNOGS_DEV_ARGS

Type

string

Default

Flowgraph-defined

Required

No

Description

SoapySDR device arguments. Valid device arguments for attached devices can be queried using SoapySDRUtil --probe.

SATNOGS_STREAM_ARGS

Type

string

Default

Flowgraph-defined

Required

No

Description

SoapySDR stream arguments. Valid stream arguments for attached devices can be queried using SoapySDRUtil --probe.

SATNOGS_TUNE_ARGS

Type

string

Default

Flowgraph-defined

Required

No

Description

SoapySDR channel tune arguments.

SATNOGS_OTHER_SETTINGS

Type
string

Default
Flowgraph-defined

Required
No

Description
SoapySDR channel other settings.

SATNOGS_DC_REMOVAL

Type
boolean

Default
Flowgraph-defined

Required
No

Description
SoapySDR device automatic DC offset suppression.

SATNOGS_BB_FREQ

Type
string

Default
Flowgraph-defined

Required
No

Description
SoapySDR device baseband CORDIC frequency for devices that support it.

ENABLE_IQ_DUMP

Type
boolean

Default
False

Required
No

Description
Create I/Q data dumps for every observation. Use this feature with caution. Enabling this setting will store large amount of data on the filesystem.

IQ_DUMP_FILENAME

Type
path

Default
None

Required
No

Description
Path to file for storing I/Q data dumps.

DISABLE_DECODED_DATA

Type
boolean

Default
False

Required
No

Description
Disable output of decoded data.

UDP_DUMP_HOST

Type
host

Default
Flowgraph-defined

Required
No

Description
IP destination of UDP data with Doppler corrected I/Q.

UDP_DUMP_PORT

Type
port

Default
57356

Required
No

Description
Port for UDP data with Doppler corrected I/Q.

SATNOGS_UPLOAD_AUDIO_FILES

Type
boolean

Default
True

Required
No

Description
Enable/Disable uploading audio files to SatNOGS network.

SATNOGS_UPLOAD_WATERFALL_FILES

Type
boolean

Default
True

Required
No

Description
Enable/Disable uploading waterfalls to SatNOGS network.

SATNOGS_WATERFALL_AUTORANGE

Type
boolean

Default
True

Required
No

Description
Automatically set power level range of waterfall images.

SATNOGS_WATERFALL_MIN_VALUE

Type
integer

Default
-100

Required
No

Description
Minimum power level of waterfall images.

SATNOGS_WATERFALL_MAX_VALUE

Type
integer

Default
-50

Required
No

Description
Maximum power level of waterfall images.

SATNOGS_ARTIFACTS_ENABLED

Type
boolean

Default
False

Required
No

Description
Enable generation and uploading of HDF5 artifacts files to SatNOGS DB.

SATNOGS_ARTIFACTS_API_URL

Type
url

Default
<https://db.satnogs.org/api/>

Required
No

Description
URL pointing to API of SatNOGS DB for uploading artifacts.

SATNOGS_ARTIFACTS_API_POST_INTERVAL

Type
integer

Default
180

Required
No

Description
Interval (in seconds) for posting artifacts to SatNOGS DB.

SATNOGS_ARTIFACTS_API_TOKEN

Type

string

Default

None

Required

No

Description

SatNOGS DB API token associated with an account in SatNOGS DB. This token is secret. It is used to upload artifacts to SatNOGS DB. It can be found in SatNOGS DB user page.

SATNOGS_LOG_LEVEL

Type

string

Default

WARNING

Required

No

Description

SatNOGS Client logging level. Valid values are:

- CRITICAL
- ERROR
- WARNING
- INFO
- DEBUG

SATNOGS_SCHEDULER_LOG_LEVEL

Type

string

Default

WARNING

Required

No

Description

SatNOGS Client scheduler logging level. Valid values are:

- CRITICAL
- ERROR
- WARNING
- INFO

- DEBUG

SENTRY_DSN

Type
string

Default
d50342fb75aa8f3945e2f846b77a0cdb7c7d2275

Required
No

Description
Sentry Data Source Name used for sending events to application monitoring and error tracking server.

SENTRY_ENABLED

Type
boolean

Default
False

Required
No

Description
Enable sending events to Sentry application monitoring and error tracking server.

1.1.4 Usage

To execute the script, run it on the command line:

```
$ satnogs-client
```

1.1.5 Doppler-corrected I/Q Streaming via UDP

For realtime analysis during an observation each SatNOGS flowgraph offers streaming of doppler corrected I/Q data via an UDP sink. The destination of this sink can be configured by the environment variables *UDP_DUMP_HOST* and *UDP_DUMP_PORT*.

1.2 Developer guide

1.2.1 Installation

To install the required dependencies in Debian Bullseye run:

```
$ xargs -r -a packages.debian apt-get install
```

It is recommended to install the client in a virtualenv. The virtualenv needs to have access to system Python bindings. To create the virtualenv, you can use `virtualenvwrapper`. On the first time, create the virtualenv by running:

```
$ mkvirtualenv --system-site-packages -a . satnogs-client
```

To activate the virtualenv after it is created run:

```
$ workon satnogs-client
```

To install SatNOGS Client for development run in the project root directory:

```
$ pip install -e .[dev]
```

1.2.2 Configuration

This project uses `python-dotenv`. Configuration of `satnogsclient/settings.py` can be overridden by setting the respective environment variables or an `.env` file placed on the project root directory. Check [Configuration](#) for a list of all configuration variables.

1.2.3 Code Quality Assurance

The following code quality assurance tools are used in this project:

- `flake8`
- `isort`
- `yapf`
- `pylint`
- `robotframework`

1.2.4 Testing

1.2.4.1 System testing

Robot Framework is used for system testing. `robot/testsuites` contain Robot test cases and suites.

1.2.5 Automation

`tox` is used to automate development tasks. To execute the default list of tasks run:

```
$ tox
```

1.2.5.1 Environments

The following `tox` environments are available:

- `flake8` - Check code for common errors, coding style and complexity
- `isort` - Check code for correct imports order
- `isort-apply` - Sort imports
- `yapf` - Check code for correct formatting
- `yapf-apply` - Reformat source code
- `pylint` - Execute static code analysis
- `build` - Build source and binary distributions
- `upload` - Upload source and binary distributions to PyPI
- `docs` - Build documentation
- `deps` - Check for broken dependencies
- `pytest` - Execute application tests
- `robot-lint` - Lint system test cases and suites
- `robot-tidy` - Reformat system test cases and suites
- `robot` - Execute system testing

To execute a single environment run:

```
$ tox -e <environment>
```

1.3 Working with Docker

1.3.1 Requirements

- Docker Engine 20.10+
- Docker Compose 1.29+

1.3.2 Building

To build the SatNOGS Client image run:

```
$ docker-compose build
```

To build based on the unstable image of `librespace/gnuradio` run:

```
$ GNURADIO_IMAGE_TAG=satnogs-unstable docker-compose build
```

1.3.3 Configuration

Edit `docker-compose.yml` to set SatNOGS Client environment variables. Check [Configuration](#) for a list of all configuration variables.

1.3.4 Usage

To bring up SatNOGS Client containers, run:

```
$ docker-compose up
```

1.4 Modules reference

1.4.1 satnogsclient

SatNOGS Client module initialization

`satnogsclient.main()`

Main function

1.4.2 satnogsclient.locator.locator

```
class satnogsclient.locator.locator.Locator
```

```
    static show_location(gpsd)
```

```
    update_location()
```

1.4.3 satnogsclient.observer.observer

```
class satnogsclient.observer.observer.Observer
```

```
    observe()
```

Starts threads for rotctl and rigctl.

```
    plot_waterfall(waterfall)
```

```
    poll_gnu_proc_status(flowgraph)
```

```
    remove_waterfall_file()
```

```
    rename_data_file()
```

```
    rename_ogg_file()
```

```
    run_rig()
```

```
    run_rot()
```

```
    setup(observation_id, tle, observation_end, frequency, mode, baud)
```

Sets up required internal variables. * returns True if setup is ok * returns False if issue is encountered

```
satnogsclient.observer.observer.post_and_save_artifacts(artifacts_file, observation_id)
```

Accepts a TemporaryFile *artifacts_file* opened at the beginning of the file, saves it to disk if enabled, then tries to upload it to satnogs-db. Finally closes this file.

Parameters

- **artifacts_file** (*file*) – The Artifact file (opened at position 0).
- **observation_id** (*str*) – The observation id of the associated observation (for enhanced logging).

```
satnogsclient.observer.observer.rx_device_for_frequency(rx_device_spec, frequency)
```

```
satnogsclient.observer.observer.save_artifacts(artifacts_file, filename)
```

Accepts a TemporaryFile *artifacts_file* opened at the beginning of the file and writes this file to disk. Finally seeks back to the beginning of the file.

NOTE: Does NOT close this file.

Parameters

- **artifacts_file** (*file*) – TemporaryFile, open at position 0
- **filename** (*str*) – Filename of artifacts file

1.4.4 satnogsclient.observer.orbital

```
satnogsclient.observer.orbital.pinpoint(observer_dict, satellite_dict, timestamp=None)
```

Provides azimuth and altitude of tracked object.

args:

observer_dict: dictionary with details of observation point. *satellite_dict*: dictionary with details of satellite. *time*: timestamp we want to use for pinpointing the observed object.

returns:

Dictionary containing azimuth, altitude and “ok” for error detection.

1.4.5 satnogsclient.artifacts

```
class satnogsclient.artifacts.Artifacts(waterfall, metadata)
```

```
create()
```

1.4.6 satnogsclient.waterfall

```
exception satnogsclient.waterfall.EmptyArrayError
```

Empty data array exception

```
class satnogsclient.waterfall.Waterfall(datafile_path)
```

Parse waterfall data file

Parameters

- **datafile_path** (*str_array*) – Path to data file

```
plot(figure_path, vmin=None, vmax=None)
    Plot waterfall into a figure

    Parameters
        • figure_path (str) – Path of figure file to save
        • vmin (int) – Minimum value range
        • vmax (int) – Maximum value range
```

```
satnogsclient.waterfall._compress_waterfall(waterfall)
    Compress spectra of waterfall
```

```
    Parameters
        waterfall (dict) – Waterfall data

    Returns
        Compressed spectra

    Return type
        dict
```

```
satnogsclient.waterfall._get_waterfall(datafile_path)
    Get waterfall data
```

```
    Parameters
        datafile_path (str_array) – Path to data file

    Returns
        Waterfall data including compressed data

    Return type
        dict
```

```
satnogsclient.waterfall._read_waterfall(datafile_path)
    Read waterfall data file
```

```
    Parameters
        datafile_path (str) – Path to data file

    Raises
        EmptyArrayError – Empty waterfall data

    Returns
        Waterfall data

    Return type
        dict
```

1.4.7 satnogsclient.observer.worker

```
class satnogsclient.observer.Worker(time_to_stop=None, sleep_time=None)
    Class to facilitate as a worker for rotctl/rigctl.

    _communicate_tracking_info()

    _observation_end = None

    _sleep_time = 0.1
```

```

_stay_alive = False

property is_alive
    Returns if tracking loop is alive or not.

observer_dict = {}

satellite_dict = {}

send_to_socket(pin, sock)

trackobject(observer_dict, satellite_dict)
    Sets tracking object. Can also be called while tracking to manipulate observation.

Parameters
    • observer_dict (dict) – Location of the Observer, example: {‘lon’: 0.0, ‘lat’: 0.0, ‘elev’:0.0}
    • satellite_dict (dict) – TLE of the satellite, example: {‘tle0’: ‘’, ‘tle1’: ‘’, ‘tle2’: ‘’}

trackstart()
    Starts the thread that communicates tracking info to remote socket. Stops by calling trackstop()

trackstop()
    Sets object flag to false and stops the tracking thread.

class satnogsclient.observer.worker.WorkerFreq(ip, port, **kwargs)

    _communicate_tracking_info()
        Runs as a daemon thread, communicating tracking info to remote socket. Uses observer and satellite objects set by trackobject(). Will exit when observation_end timestamp is reached.

    _frequency = None

    send_to_socket(pin, sock)

    trackobject(frequency, observer_dict, satellite_dict)
        Track object

Parameters
    • frequency (int) – Frequency of original signal in Hz
    • observer_dict (dict) – Location of the Observer example: {‘lon’: 0.0, ‘lat’: 0.0, ‘elev:0.0}
    • satellite_dict (dict) – TLE of the satellite example: {‘tle0’: ‘’, ‘tle1’: ‘’, ‘tle2’: ‘’}

class satnogsclient.observer.worker.WorkerTrack(port, **kwargs)

    _altitude = None

    _azimuth = None

    _communicate_tracking_info()
        Runs as a daemon thread, communicating tracking info to remote socket. Uses observer and satellite objects set by trackobject(). Will exit when observation_end timestamp is reached.

    _flip = False

```

```
_midpoint = None  
  
static find_midpoint(observer_dict, satellite_dict, start)  
  
static flip_coordinates(azi, alt, timestamp, midpoint)  
  
static normalize_angle(num, lower=0, upper=360)  
  
send_to_socket(pin, sock)  
  
trackobject(*args)
```

Track object

Parameters

args – Positional Arguments of parent class Worker

1.4.8 satnogsclient.rig

```
class satnogsclient.rig.Rig(model=Hamlib.RIG_MODEL_DUMMY, path='',  
                           debug=Hamlib.RIG_DEBUG_WARN)
```

Communicate and interface with rigs

Parameters

- **model** – Model of Hamlib rig
- **path** (*str, optional*) – Path or address to Hamlib rig device
- **debug** (*int, optional*) – Hamlib rig debug level

close()

Close Hamlib rig device

property frequency

Get rig frequency

Returns

Rig frequency

Return type

float

open()

Open Hamlib rig device

property vfo

Get active VFO

Returns

Active VFO

Return type

int

1.4.9 satnogsclient.rotator

class `satnogsclient.rotator.Rotator(model, baud, port)`

Communicate and interface with rotators

Parameters

- **model** (*str*) – Model of rotator e.g. “ROT_MODEL_EASYCOMM3” or “ROT_MODEL_DUMMY”
- **baud** (*int, optional*) – The baud rate of serial communication, e.g. 19200
- **port** (*str*) – The port of the rotator, e.g. “/dev/ttyUSB0”

close()

End the communication with rotator

get_conf(cmd)

Return the configuration of a register

Parameters

cmd (*int*) – Number of the register

Returns

Value of register

Return type

str

get_info()

Return information about the rotator

Returns

Information about the rotator

Return type

str

move(direction, speed)

Move the rotator with speed (mdeg/s) to specific direction

Parameters

- **direction** (*str*) – The direction of movement, e.g. ROT_MOVE_UP, ROT_MOVE_DOWN, ROT_MOVE_LEFT, ROT_MOVE_RIGHT
- **speed** (*int*) – The velocity set point in mdeg/s

open()

Start the communication with rotator

park()

Move the rotator to park position and return the current position

Returns

Current position in degrees

Return type

None or list[float, float]

property position

Return the position in degrees of azimuth and elevation

Returns

Position in degrees

Return type

list[float, float]

reset()

Move the rotator to home position and return the current position

Returns

Current position in degrees

Return type

None or list[float, float]

stop()

Stop the rotator and return the current position

Returns

Current position in degrees

Return type

None or list[float, float]

1.4.10 satnogsclient.scheduler.tasks

satnogsclient.scheduler.tasks.get_jobs()

Query SatNOGS Network API to GET jobs.

satnogsclient.scheduler.tasks.keep_or_remove_file(filename)

satnogsclient.scheduler.tasks.post_data()

PUT observation data back to Network API.

satnogsclient.scheduler.tasks.spawn_observer(kwargs)**

satnogsclient.scheduler.tasks.status_listener()

satnogsclient.scheduler.tasks.upload_observation_data(observation_id, observation, fil)

Upload observation data to SatNOGS Network API.

1.4.11 satnogsclient.settings

SatNOGS Client settings file

satnogsclient.settings._cast_or_none(func, value)

satnogsclient.settings.validate(logger)

Validate the provided settings: - Check for the existence of all required variables - Validate format of the provided value for some required variables

Since this module has to be loaded before the logger has been initialized, this method requires a configured logger to be passed.

Arguments: logger – the output logger

1.4.12 satnogsclient.radio.flowgraphs

```
class satnogsclient.radio.flowgraphs.Flowgraph(device, sampling_rate, frequency, mode, baud,  
                                              output_data)
```

Execute SatNOGS Flowgraphs

Parameters

- **device** (*str*) – SoapySDR device
- **sampling_rate** (*int*) – Sampling rate
- **frequency** (*int*) – RX frequency
- **mode** (*str*) – Mode of operation
- **baud** (*int*) – Baud rate or WPM
- **output_data** (*dict*) – Dictionary of output data

property enabled

Get flowgraph running status

Returns

Flowgraph running status

Return type

bool

property info

Get information and parameters of flowgraph and radio

Returns

Information about flowgraph and radio

Return type

dict

**CHAPTER
TWO**

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

`satnogsclient`, 24
`satnogsclient.artifacts`, 25
`satnogsclient.locator.locator`, 24
`satnogsclient.observer.observer`, 24
`satnogsclient.observer.orbital`, 25
`satnogsclient.observer.worker`, 26
`satnogsclient.radio.flowgraphs`, 31
`satnogsclient.rig`, 28
`satnogsclient.rotator`, 29
`satnogsclient.scheduler.tasks`, 30
`satnogsclient.settings`, 30
`satnogsclient.waterfall`, 25

INDEX

Symbols

_altitude (*satnogsclient.observer.worker.WorkerTrack attribute*), 27
_azimuth (*satnogsclient.observer.worker.WorkerTrack attribute*), 27
_cast_or_none() (*in module satnogsclient.settings*), 30
_communicate_tracking_info()
 (*satnogsclient.observer.worker.Worker method*), 26
_communicate_tracking_info()
 (*satnogsclient.observer.worker.WorkerFreq method*), 27
_communicate_tracking_info()
 (*satnogsclient.observer.worker.WorkerTrack method*), 27
_compress_waterfall() (*in module satnogsclient.waterfall*), 26
_flip (*satnogsclient.observer.worker.WorkerTrack attribute*), 27
_frequency (*satnogsclient.observer.worker.WorkerFreq attribute*), 27
_get_waterfall() (*in module satnogsclient.waterfall*), 26
_midpoint (*satnogsclient.observer.worker.WorkerTrack attribute*), 27
_observation_end (*satnogsclient.observer.worker.Worker attribute*), 26
_read_waterfall() (*in module satnogsclient.waterfall*), 26
_sleep_time (*satnogsclient.observer.worker.Worker attribute*), 26
_stay_alive (*satnogsclient.observer.worker.Worker attribute*), 26

A

Artifacts (*class in satnogsclient.artifacts*), 25

C

close() (*satnogsclient.rig.Rig method*), 28
close() (*satnogsclient.rotator.Rotator method*), 29
create() (*satnogsclient.artifacts.Artifacts method*), 25

E

EmptyArrayError, 25
enabled (*satnogsclient.radio.flowgraphs.Flowgraph property*), 31

F

find_midpoint() (*satnogsclient.observer.worker.WorkerTrack static method*), 28
flip_coordinates() (*satnogsclient.observer.worker.WorkerTrack static method*), 28
Flowgraph (*class in satnogsclient.radio.flowgraphs*), 31
frequency (*satnogsclient.rig.Rig property*), 28

G

get_conf() (*satnogsclient.rotator.Rotator method*), 29
get_info() (*satnogsclient.rotator.Rotator method*), 29
get_jobs() (*in module satnogsclient.scheduler.tasks*), 30

I

info (*satnogsclient.radio.flowgraphs.Flowgraph property*), 31
is_alive (*satnogsclient.observer.worker.Worker property*), 27

K

keep_or_remove_file() (*in module satnogsclient.scheduler.tasks*), 30

L

Locator (*class in satnogsclient.locator.locator*), 24

M

main() (*in module satnogsclient*), 24

module
 satnogsclient, 24
 satnogsclient.artifacts, 25
 satnogsclient.locator.locator, 24
 satnogsclient.observer.observer, 24
 satnogsclient.observer.orbital, 25
 satnogsclient.observer.worker, 26

satnogsclient.radio.flowgraphs, 31
satnogsclient.rig, 28
satnogsclient.rotator, 29
satnogsclient.scheduler.tasks, 30
satnogsclient.settings, 30
satnogsclient.waterfall, 25
move() (*satnogsclient.rotator.Rotator method*), 29

N

normalize_angle() (*satnogsclient.observer.worker.Worker static method*), 28

O

observe() (*satnogsclient.observer.observer.Observer method*), 24
Observer (*class in satnogsclient.observer.observer*), 24
observer_dict (*satnogsclient.observer.worker.Worker attribute*), 27
open() (*satnogsclient.rig.Rig method*), 28
open() (*satnogsclient.rotator.Rotator method*), 29

P

park() (*satnogsclient.rotator.Rotator method*), 29
pinpoint() (*in module satnogsclient.observer.orbital*), 25
plot() (*satnogsclient.waterfall.Waterfall method*), 25
plot_waterfall() (*satnogsclient.observer.observer.Observer method*), 24
poll_gnu_proc_status() (*satnogsclient.observer.observer.Observer method*), 24
position (*satnogsclient.rotator.Rotator property*), 29
post_and_save_artifacts() (*in module satnogsclient.observer.observer*), 24
post_data() (*in module satnogsclient.scheduler.tasks*), 30

R

remove_waterfall_file() (*satnogsclient.observer.observer.Observer method*), 24
rename_data_file() (*satnogsclient.observer.observer.Observer method*), 24
rename_ogg_file() (*satnogsclient.observer.observer.Observer method*), 24
reset() (*satnogsclient.rotator.Rotator method*), 30
Rig (*class in satnogsclient.rig*), 28
Rotator (*class in satnogsclient.rotator*), 29
run_rig() (*satnogsclient.observer.observer.Observer method*), 24
run_rot() (*satnogsclient.observer.observer.Observer method*), 24
rx_device_for_frequency() (*in module satnogsclient.observer.observer*), 25

S

satellite_dict (*satnogsclient.observer.worker.Worker attribute*), 27
satnogsclient module, 24
satnogsclient.artifacts module, 25
satnogsclient.locator locator module, 24
satnogsclient.observer.observer module, 24
satnogsclient.observer.orbital module, 25
satnogsclient.observer.worker module, 26
satnogsclient.radio.flowgraphs module, 31
satnogsclient.rig module, 28
satnogsclient.rotator module, 29
satnogsclient.scheduler.tasks module, 30
satnogsclient.settings module, 30
satnogsclient.waterfall module, 25
save_artifacts() (*in module satnogsclient.observer.observer*), 25
send_to_socket() (*satnogsclient.observer.worker.Worker method*), 27
send_to_socket() (*satnogsclient.observer.worker.WorkerFreq method*), 27
send_to_socket() (*satnogsclient.observer.worker.WorkerTrack method*), 28
setup() (*satnogsclient.observer.observer.Observer method*), 24
show_location() (*satnogsclient.locator.locator.Locator static method*), 24
spawn_observer() (*in module satnogsclient.scheduler.tasks*), 30
status_listener() (*in module satnogsclient.scheduler.tasks*), 30
stop() (*satnogsclient.rotator.Rotator method*), 30

T

trackobject() (*satnogsclient.observer.worker.Worker method*), 27
trackobject() (*satnogsclient.observer.worker.WorkerFreq method*), 27
trackobject() (*satnogsclient.observer.worker.WorkerTrack method*), 28
trackstart() (*satnogsclient.observer.worker.Worker method*), 27

`trackstop()` (*satnogsclient.observer.worker.Worker method*), 27

U

`update_location()` (*satnogsclient.locator.locator.Locator method*), 24

`upload_observation_data()` (*in module satnogsclient.scheduler.tasks*), 30

V

`validate()` (*in module satnogsclient.settings*), 30

`vfo` (*satnogsclient.rig.Rig property*), 28

W

`Waterfall` (*class in satnogsclient.waterfall*), 25

`Worker` (*class in satnogsclient.observer.worker*), 26

`WorkerFreq` (*class in satnogsclient.observer.worker*), 27

`WorkerTrack` (*class in satnogsclient.observer.worker*), 27