SatNOGS Client Documentation

Release 1.0+0.gcee4be5.dirty

SatNOGS

Oct 12, 2019

Contents

CHAPTER 1

SatNOGS client architecture

1.1 Overview

SatNOGS client is the part of our software stack that:

- Fetches observation jobs from the network.
- Schedules locally when the observation starts/ends.
- Does orbital calculation for the position of the observer and the tracked object (using `PyEphem`).
- Sends `rotctl/rigctl` commands to control **SatNOGS** rotator.
- Spawns processes to run demodulation/decoding software with the signal received as input.
- Posts observation data back to the network.

1.2 Modules

Following the paradigm of **SatNOGS** being extensively modular, **SatNOGS** client is designed to have discrete modules with specific functionality.

1.2.1 `scheduler`

- Build using `apscheduler` library.
- Stores tasks in `sqlite`.

1.2.1.1 Tasks

- `get_jobs`: Queries **SatNOGS Network API** to get jobs scheduled for the ground station.
- `spawn_observation`: Initiates an `Observer` instance and runs the observation.

- `post_observation_data`: Gathers output files, parses filename and posts data back to **SatNOGS Network API**.

1.2.2 observer.observer

Given initial description of the observation (`tle`, `start`, `end`)

- Checks input for sanity.
- Initializes `WorkerTrack` and `WorkerFreq` instances that start `rigctl/rotctl`. communication using `trackstart` method.
- Starts/Stops GNU Radio script (`gr-satnogs`), which collects the data.
- Processes produced data from observation (ogg file, waterfall plotting).

1.2.3 observer.worker

- Facilitates as a worker for `rigctl/rotctl`.
- Is the lowest abstraction level on `rigctl/rotctl` communications.
- Tracks object until end of observation is reached.

1.2.4 observer.orbital

- Implements orbital calculations using `PyEphem`.
- Provides `pinpoint` method the returns alt/az position of tracked object.

1.2.5 satnogs-client

- `satnogs-client`: A console script which runs the scheduler queue in the background and fetch jobs from the network.

CHAPTER 2

satnogsclient Package

2.1 satnogsclient Package

SatNOGS Client module initialization

satnogsclient.__init__.**main**()
Main function

2.2 settings Module

SatNOGS Client settings file

2.3 Subpackages

2.3.1 observer Package

2.3.1.1 commsocket Module

class satnogsclient.observer.commsocket.**Commsocket**(*ip_address*, *port*)
Bases: object

Handles connectivity with remote ctl demons Namely: rotctl and rigctl

accept()

bind()

buffer_size

connect()

disconnect()

ip_address

is_connected

listen()

port

receive(*size*)

send(*message*)

send_not_recv(*message*)

tasks_buffer_size

2.3.1.2 observer Module

class satnogsclient.observer.observer.**Observer**
Bases: object

observe()
Starts threads for rotcrl and rigctl.

plot_waterfall()

poll_gnu_proc_status()

remove_waterfall_file()

rename_data_file()

rename_ogg_file()

run_rig()

run_rot()

setup(*observation_id*, *tle*, *observation_end*, *frequency*, *baud*, *script_name*)
Sets up required internal variables. * returns True if setup is ok * returns False if issue is encountered

2.3.1.3 orbital Module

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.

2.3.1.4 worker Module

class satnogsclient.observer.worker.**Worker**(*ip*, *port*, *time_to_stop=None*, *frequency=None*, *proc=None*, *sleep_time=None*)
Bases: object

Class to facilitate as a worker for rotctl/rigctl.

check_observation_end_reached()

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.

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*, *time_to_stop=None*, *frequency=None*, *proc=None*, *sleep_time=None*)
Bases: *satnogsclient.observer.worker.Worker*

send_to_socket(*pin*, *sock*)

class satnogsclient.observer.worker.**WorkerTrack**(*ip*, *port*, *time_to_stop=None*, *frequency=None*, *proc=None*, *sleep_time=None*)
Bases: *satnogsclient.observer.worker.Worker*

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

2.3.2 scheduler Package

2.3.2.1 scheduler Package

2.3.2.2 tasks Module

satnogsclient.scheduler.tasks.**get_jobs**()
Query SatNOGS Network API to GET jobs.

satnogsclient.scheduler.tasks.**get_observation**(*job_id*)

satnogsclient.scheduler.tasks.**get_observation_list**()

satnogsclient.scheduler.tasks.**post_data**()
PUT observation data back to Network API.

satnogsclient.scheduler.tasks.**signal_term_handler**()

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

satnogsclient.scheduler.tasks.**status_listener**()

CHAPTER 3

Installation

Note: These installation steps are intended to be used for contributing to the satnogs-client codebase. If you are interested in setting up satnogs-client for your ground station check the wiki.

Requirements: You will need python, python-virtualenvwrapper, pip and git

1. **Build the environment**

   Clone source code from the repository:

   ```
   $ git clone https://gitlab.com/librespacefoundation/satnogs/satnogs-client.git
   ```

   Set up the virtual environment. On first run you should create it and link it to your project path.:

   ```
   $ cd satnogs-client
   $ mkvirtualenv satnogs-client -a .
   ```

   Activate your python virtual environment:

   ```
   $ workon satnogs-client
   ```

   Install local development requirements:

   ```
   $ pip install .
   ```

2. **Run the client**

   Create an `.env` file on the project root and configure the client environment variables.

   Run `satnogs-client`:

   ```
   $ satnogs-client
   ```

Python Module Index

S

Index

R

S

T

W