SatNOGS Documentation

Release 1

SatNOGS

Nov 12, 2018

Contents

This is the documentation for all parts of the SatNOGS project.

Contents:

CHAPTER 1

satnogs-client

Contents:

1.1 SatNOGS client architecture

1.1.1 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.

1.1.2 Modules

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

`scheduler`

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

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**.

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).

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.

observer.orbital

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

bin/scripts

- `satnogs-client`: Run the scheduler queue in the background and fetch jobs from the network.

1.2 satnogsclient Package

1.2.1 satnogsclient Package

1.2.2 settings Module

1.2.3 Subpackages

observer Package

commsocket Module

class satnogsclient.observer.commsocket.**Commsocket**(*ip*, *port*)
Handles connectivity with remote ctl demons Namely: rotctl and rigctl

accept()

bind()

buffer_size

connect()

disconnect()

ip

is_connected

listen()

port

receive(*size*)

send(*message*)

send_not_recv(*message*)

tasks_buffer_size

observer Module

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.

worker Module

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

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()

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

trackstart(*port*, *start_thread*)
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(*p*, *sock*)

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

Bases: *satnogsclient.observer.worker.Worker*

send_to_socket(*p*, *sock*)

scheduler Package

scheduler Package

tasks Module

1.3 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 .
```

CHAPTER 2

satnogs-network

2.1 Installation

2.1.1 Docker Installation

1. **Requirements**

   You will need docker and docker-compose.

2. **Build the containers**

   Clone source code from the repository:

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

   Set your environmental variables:

   ```
   $ cp env-dist .env
   ```

   Start database container (Run it in the background):

   ```
   $ docker-compose up --detach db
   ```

   Build satnogs-network container:

   ```
   $ docker-compose build web
   ```

   Run the initialize script to populate the database with scheme and demo data:

   ```
   $ docker-compose run web python manage.py initialize
   ```

   Note that the above command requires internet connection, since it fetches Satellite and Transmitter data from SatNOGS-DB

3. **Run it!**

   Run satnogs-network (in the foreground):

```
$ docker-compose up
```

   Your satnogs-network development instance is available in localhost:8000. Go hack!

2.1.2 VirtualEnv Installation

1. **Install the requirements**

   Generic requirements which you will need:

```
python, python-virtualenvwrapper, pip and git, libmysqlclient-dev (mysql_config)
```

   Debian Stretch (9) specific requirements

```
$ sudo apt-get install libmariadbclient-dev python-pip virtualenvwrapper
```

2. **Build the environment**

   Clone source code from the repository:

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

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

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

   Set your environmental variables:

```
$ cp env-dist .env
```

   Activate your python virtual environment:

```
$ workon satnogs-network
```

   Install local development requirements:

```
$ (satnogs-network)$ pip install --require-hashes --no-deps -r requirements/dev.
↪txt
```

3. **Database**

   Create, setup and populate the database with demo data:

```
(satnogs-network)$ ./manage.py initialize
```

   Note that the above command requires internet connection, since it fetches Satellite and Transmitter data from SatNOGS-DB

4. **Run it!**

   Just run it:

```
(satnogs-network)$ ./manage.py runserver
```

   Your satnogs-network development instance is available in localhost:8000. Go hack!

2.2 Contribute

Thank you for your interest in contributing to SatNOGS! There are always bugs to file; bugs to fix in code; improvements to be made to the documentation; and more.

The below instructions are for software developers who want to work on satnogs-network code.

2.2.1 Git workflow

When you want to start contributing, you should follow the installation instructions, then...

1. (Optional) Set your cloned fork to track upstream changes (changes to the main repository), then fetch and merge changes from the upstream branch:

```
$ git remote add --track master upstream git://gitlab.com/librespacefoundation/
↪satnogs/satnogs-network
$ git fetch upstream
$ git merge upstream/master
```

2. Set up a branch for a particular set of changes and switch to it:

```
$ git branch my_branch
$ git checkout my_branch
```

3. Commit changes to the code!
4. Code!
5. Lint the code:

```
$ flake8 network
```

   and fix any errors.

6. Commit changes to the code!
7. When you're done, figure out how many commits you've made:

```
$ git log
```

8. Squash all those commits into a single commit that has a good git commit message. (Example assumes you made 4 commits):

```
$ git rebase -i HEAD~4
```

9. Use the interactive editor that pops up to pick/squash your commits:

```
pick 01d1239 [fix bug 893291] Make it go to 11
squash 32as32p added the library and made some minor changes
squash 30ame3z build the template
squash 91pcla8 ugh fix a semicolon bug in that last commit
```

10. Push your changes to your fork:

```
$ git push origin my_branch
```

11. Issue a pull request on Gitlab
12. Wait to hear from one of the core developers

If you're asked to change your commit message, you can amend the message and force commit:

```
$ git commit --amend
$ git push -f origin my_branch
```

If you're asked to make changes on your code you can stage them and amend the commit:

```
$ git add my_changed_files
$ git commit --amend
$ git push -f origin my_branch
```

If you need more Git expertise, a good resource is the Git book.

2.2.2 Templates

satnogs-network uses Django's template engine templates.

2.2.3 Frontend development

To be able to manage the required javascript libraries, install the development dependencies with npm:

```
$ npm install
```

Development tasks like the download of assets, code linting and tests are managed with gulp:

```
$ gulp
```

Frontend dependencies are stored in packages.json, handled by yarn. To add a new dependency, e.g. satellite.js, call:

```
$ yarn add satellite.js
```

Manually add the new required files to the list of "assets" in packages.json, then start the download with:

```
$ gulp assets
```

The assets are stored in the repository, thus don't forget to create a commit after you add/update/remove dependencies.

2.2.4 Coding Style

1. Four space indentation (no tabs), two whitespace on html documents.
2. Use single quotes for strings. Double quotes used only for html attributes.
3. Keep lines shorter than 100 characters when possible (especially at python code)

Follow the PEP8 and PEP257 Style Guides.

Most important things:

1. Separate top-level function and class definitions with two blank lines.
2. Method definitions inside a class are separated by a single blank line.
3. Use whitespace between comma seperated values.
4. Use white space between assignments and expressions (except parameter values).
5. Don't use whitespace before or after parentheses, brackets or braces.

6. Classes should use CamelCase naming.
7. Functions should use lowercase naming.

2.2.5 What to work on

You can check opened issues. We regurarly open issues for tracking new features. You pick one and start coding.

CHAPTER 3

satnogs-db

3.1 Installation

3.1.1 Docker Installation

1. **Requirements**

   You will need docker and docker-compose.

2. **Get the source code**

   Clone source code from the repository:

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

3. **Configure settings**

   Set your environmental variables:

   ```
   $ cp env-dist .env
   ```

4. **Run it!**

   Run satnogs-db:

   ```
   $ docker-compose up -d --build
   ```

5. **Populate database**

   Create, setup and populate the database with demo data:

   ```
   $ docker-compose exec web djangoctl.sh initialize
   ```

   Your satnogs-db development instance is available in localhost:8000. Go hack!

3.1.2 VirtualEnv Installation

1. **Requirements**

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

2. **Get the source code**

   Clone source code from the repository:

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

3. **Build the environment**

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

```
$ mkvirtualenv satnogs-db -a .
```

4. **Configure settings**

   Set your environmental variables:

```
$ cp env-dist .env
```

5. **Run it!**

   Activate your python virtual environment:

```
$ workon satnogs-db
```

   Just run it:

```
(satnogs-db)$ ./bin/djangoctl.sh develop .
```

6. **Populate database**

   Create, setup and populate the database with demo data:

```
(satnogs-db)$ ./bin/djangoctl.sh initialize
```

   Your satnogs-db development instance is available in localhost:8000. Go hack!

3.2 Contribute

Thank you for your interest in contributing to SatNOGS! There are always bugs to file; bugs to fix in code; improvements to be made to the documentation; and more.

The below instructions are for software developers who want to work on satnogs-db code.

3.2.1 Git workflow

When you want to start contributing, you should follow the installation instructions, then...

1. (Optional) Set your cloned fork to track upstream changes (changes to the main repository), then fetch and merge changes from the upstream branch:

```
$ git remote add --track master upstream git://gitlab.com/librespacefoundation/
↪satnogs/satnogs-db
$ git fetch upstream
$ git merge upstream/master
```

2. Set up a branch for a particular set of changes and switch to it:

```
$ git branch my_branch
$ git checkout my_branch
```

3. Commit changes to the code!
4. Code!
5. Lint the code and fix any errors:

```
$ flake8 db
```

6. Commit changes to the code!
7. When you're done, figure out how many commits you've made:

```
$ git log
```

8. Squash all those commits into a single commit that has a good git commit message. (Example assumes you made 4 commits):

```
$ git rebase -i HEAD~4
```

9. Use the interactive editor that pops up to pick/squash your commits:

```
pick 01d1239 [fix bug 893291] Make it go to 11
squash 32as32p added the library and made some minor changes
squash 30ame3z build the template
squash 91pcla8 ugh fix a semicolon bug in that last commit
```

10. Push your changes to your fork:

```
$ git push origin my_branch
```

11. Issue a pull request on Gitlab
12. Wait to hear from one of the core developers

If you're asked to change your commit message, you can amend the message and force commit:

```
$ git commit --amend
$ git push -f origin my_branch
```

If you're asked to make changes on your code you can stage them and amend the commit:

```
$ git add my_changed_files
$ git commit --amend
$ git push -f origin my_branch
```

If you need more Git expertise, a good resource is the Git book.

3.2.2 Templates

satnogs-db uses Django's template engine templates.

3.2.3 Coding Style

1. Four space indentation (no tabs), two whitespace on html documents.
2. Use single quotes for strings. Double quotes used only for html attributes.
3. Keep lines shorter than 100 characters when possible (especially at python code)

Follow the PEP8 and PEP257 Style Guides.

Most important things:

1. Separate top-level function and class definitions with two blank lines.
2. Method definitions inside a class are separated by a single blank line.
3. Use whitespace between comma seperated values.
4. Use white space between assignments and expressions (except parameter values).
5. Don't use whitespace before or after parentheses, brackets or braces.
6. Classes should use CamelCase naming.
7. Functions should use lowercase naming.

3.2.4 What to work on

You can check opened issues. We regurarly open issues for tracking new features. You pick one and start coding.

3.3 Maintenance

3.3.1 Updating frontend dependencies

The frontend dependencies are managed with *npm* as defined in the *package.json*. The following are required to perform an update of the dependencies:

1. Bump versions in *package.json*
2. Download and install the latest version of the dependencies

   ```
   $ npm install
   ```

3. Move the installed version into to satnogs-db source tree

   ```
   $ ./node_modules/.bin/gulp
   ```

4. Stage & commit the updated files in *db/static/*.

3.4 API

SatNOGS-DB API is a REST API that provides detailed information about Satellites and Transmitters. This document explains how to use the API to retrieve data for your application.

3.4.1 Using API Data

API access is public to anyone. No form of authentication is required. All API data are freely distributed under the CC BY-SA license.

3.4.2 API Methods

Satellites

The `satellites` method of the SatNOGS DB API returns all Satellites currently used for gathering Transmitters data.

Endpoint

```
https://db.satnogs.org/api/satellites/
```

Examples

Show a specific satellite using its Norad Cat ID:

Request:

```
/api/satellites/25544/?format=json
```

Response:

```
{
    norad_cat_id": 25544,
    "name": "ISS (ZARYA)"
}
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

W