SensLab command-line client documentation

Release 1

Frederic Saint-Marcel

November 22, 2012

CONTENTS

CHAPTER

ONE

ABOUT

Senslabcli is a Python (2.5 or higher) module and command-line tool client for SensLab Rest API. It provides a basic suite of operations for managing your experimentation on testbed.

You have four command-lines :

- auth-cli : store account credentials
- experiment-cli : managing experiments
- node-cli : commands on sensor nodes
- profile-cli : managing profiles experimentation.

CHAPTER

TWO

INSTALLATION

In order to achieve module installation, two primary pieces of software are needed :

- the python module argparse;
- the python module requests;

Ubuntu desktop :

```
$ sudo apt-get install python-argparse python-pip
$ sudo pip install requests
```

Fedora desktop :

```
$ yum install python-argparse python-pip
$ pip-python install requests
```

Download and unpack the client archive and run this command :

```
tar -xzvf senslabcli-1.x.tar.gz & cd senlabcli-1.x
```

Ubuntu desktop :

```
$ sudo python setup.py install
```

Fedora desktop :

```
$ python setup.py install
```

CHAPTER

THREE

GENERAL NOTE

3.1 Error handling

API Rest errors are returned using standard HTTP error code syntax. Each error message is included into response body.

3.2 Rest resource representation

The supported format for API requests and responses is JSON representation.

3.3 Authentication

We use basic http authentication (login and password) over SSL communication with your account credentials.

CHAPTER

FOUR

COMMAND-LINE

4.1 Auth client

It creates a file .senslabrc in your home directory with your credentials. When you use other command-line clients without authentication options they try to read this file and use your credentials.

```
$ auth-cli -u <login> -p <password>
```

with password prompt :

```
$ auth-cli -u <login>
```

4.2 Experiment client

The client provides the following five commands : submit, stop, get, load, info

```
$ experiment-cli -h
```

4.2.1 Submit experiment

View submit command options

```
$ experiment-cli submit -h
```

The submission is a request to the scheduler of the testbed, which allows resources (sensor nodes) reservation for experimentation with a duration. If your submission is accepted by the scheduler, the response is **id submission experiment**.

There is two types of submission :

- **physical** : based on physical sensor nodes reservation with id.
- **alias** : based on sensor nodes properties (**architecture, site, mobile**) and the number of nodes expected.

Submit a physical experiment type on Grenoble site with name test_cc1100, duration of 30 minutes and five nodes :

```
$ experiment-cli submit -n test_cc1100 -d 30 -a grenoble,1-4+5
```

Submit the same experiment with alias type :

```
$ experiment-cli submit -n test_cc1100 -d 30 -a 5,archi=wsn430:cc1100+site=grenoble
```

Submit a physical experiment type with two sites Grenoble and Rennes :

```
$ experiment-cli submit -d 30 -a grenoble,1-6+12+20 -a rennes,1-50
```

Submit the same experiment with alias type :

```
$ experiment-cli submit -d 30 -a 8,archi=wsn430:cc1100+site=grenoble
-a 50,archi=wsn430:cc2420+site=rennes
```

You can specify the associations between profiles, firmwares and sensor nodes for experiment deployment configuration.

Submit experiment with associations :

- Profile name profile_cc2420 (already saved into the user profile store, See *Add profile*) and firmware absolute path file

```
$ experiment-cli submit -d 30 -a grenoble,1-3+4,profile_cc2420,/home/user/tp.hex
```

- Only firmware relative path file

```
$ experiment-cli submit -d 30 -a grenoble,1-3+4„tp.ihex
```

- Only profile name profile_cc2420

```
$ experiment-cli submit -d 30 -a grenoble,1-3+4,profile_cc2420
```

Note: By default if you don't specify associations on sensor nodes :

- without profile nodes are configured with a power supply method dc and no polling measure configuration
- without firmware the sensor nodes are programmed with the firmware "idle".

You can view experiment JSON representation with option -jl–json without saving it on the server.

You can also schedule your experiment with -r option. For this we use Linux epoch timestamp.

Submit experiment with a schedule date (e.g. 20 Sep 2012 14:00:00) :

```
$ experiment-cli submit -n schedule_experiment -d 20 -a
grenoble,1-100 -r $(date +%s -d "20 Sep 2012 14:00:00 UTC")
```

Note: The linux command `date +%s` gives you the current Linux epoch timestamp (seconds since 1970-01-01 00:00:00 UTC)

4.2.2 Stop experiment

Stop an experiment :

```
$ experiment-cli stop
```

Note: If you have only one experiment with state Running you don't need to use id experiment option.

4.2.3 Get experiment

Get experiment submission JSON representation :

```
$ experiment-cli get -j
```

Get experiment archive (tar.gz) containing submission JSON representation and firmware(s) :

```
$ experiment-cli get -a
```

Get experiment resources list JSON representation (e.g. sensor nodes) :

```
$ experiment-cli get -r
```

Get experiment state JSON representation :

```
$ experiment-cli get -s
```

Note: If you have only one experiment with state Running you don't need to use id experiment option.

4.2.4 Load an experiment

Load an experiment JSON representation with absolute path and submit on server.

```
$ experiment-cli load -f /home/user/experiment.json
```

Load an experiment JSON representation with relative path and submit on server.

```
$ experiment-cli load -f experiment.json
```

If you have a firmware association in your experiment JSON representation, we open by default firmware file with relative path. If you want to use firmware file with absolute path you can use -l option to refer to the firmware(s) list in the JSON representation.

```
$ experiment-cli load -f experiment.json -l /home/user/tp.hex,/home/user/firm.ihex
```

4.2.5 Testbed information for experiment submission

Get testbed sites list JSON representation

```
$ experiment-cli info -s
```

Get testbed resources list JSON representation

```
$ experiment-cli info -r
```

Get testbed resources state list JSON representation (e.g. command-line NODE_LIST format : 1-5+7)

```
$ experiment-cli info -rs
```

Get total number of upcoming, running and past user's experiments JSON representation

```
$ experiment-cli info -t
```

Get user's experiment list JSON representation

```
$ experiment-cli info -e
```

You can filter your search with attribute state, limit (number of experiments) and offset (start index of number experiments).

```
$ experiment-cli info -e --state Running,Terminated --offset 10
--limit 20
```

4.3 Node client

Four commands are available to interact with sensor nodes experiment. If you have only one experiment with state Running you don't need to use id experiment option. Also if you want to launch commands for **all sensor nodes** experiment you don't use option -a in the command-line.

```
$ node-cli -h
```

- **update** : flash firmware on sensor node(s)

  All sensor nodes of your current experiment with state Running :

  ```
  $ node-cli --update /home/cc1100.hex
  ```

  A set of experiment nodes :

  ```
  $ node-cli -a grenoble,1-34 --update /home/cc1100.hex
  ```

  If you have several experiments with state Running you must use experiment id option :

  ```
  $ node-cli -i 32 --update cc1100.hex
  ```

- **stop** : stop sensor node(s)

  ‘‘$ nodde-cli –stop ‘‘

- **reset** : stop and start sensor node(s)

  ```
  $ node-cli --reset
  ```

- **start** : start sensor node(s)

  ‘‘$ node-cli –start‘

  You can also specify the powered method with battery (dc by default) :

  ```
  $ node-cli --start -b
  ```

4.4 Profile client

You can create profiles to store you favourite sensor nodes configuration, for reusing it into your future experiments. A profile is the combination of a power supply mode and an automatic measure configuration (e.g. polling).

For each user there is a **profile store** on Senslab server and the command-line has four commands (add, del, get, load) for managing it.

```
$ profile-cli -h
```

4.4.1 Add profile

View add command options

```
$ profile-cli add -h
```

Add a profile on server with name test_consemptium, power supply method dc and power consemptium measure (current, voltage & power) with frequency 5000 milliseconds

```
$ profile-cli add -n test_consemptium -p dc -current -voltage -power
-cfreq 5000
```

Add a profile on server with name test_sensor, power supply method battery and temperature sensor measure

```
$ profile-cli add -n test_sensor -p battery -temperature
```

Note: You can view profile JSON representation with option -jl–json without saving it on server.

If you don’t specify frequency for measure we use by default the frequency max in option choices.

4.4.2 Delete profile

Delete profile on server with name test_profile

```
$ profile-cli del -n test_profile
```

4.4.3 Get profile

Get profile JSON representation with name test_profile

```
$ profile-cli get -n test_profile
```

Get profiles list JSON representation

```
$ profile-cli get -l
```

4.4.4 Load profile

Load a profile JSON representation with absolute path and save it on server

```
$ profile-cli load -f /home/user/profile.json
```

Load a profile JSON representation with relative path and save it on server

```
$ profile-cli load -f profile.json
```