numeter Documentation

Release latest

eNovance

June 20, 2015

Contents

Contents:

CHAPTER 1

Overview

The Numeter infrastructure : instances of every component may be added to handle the load

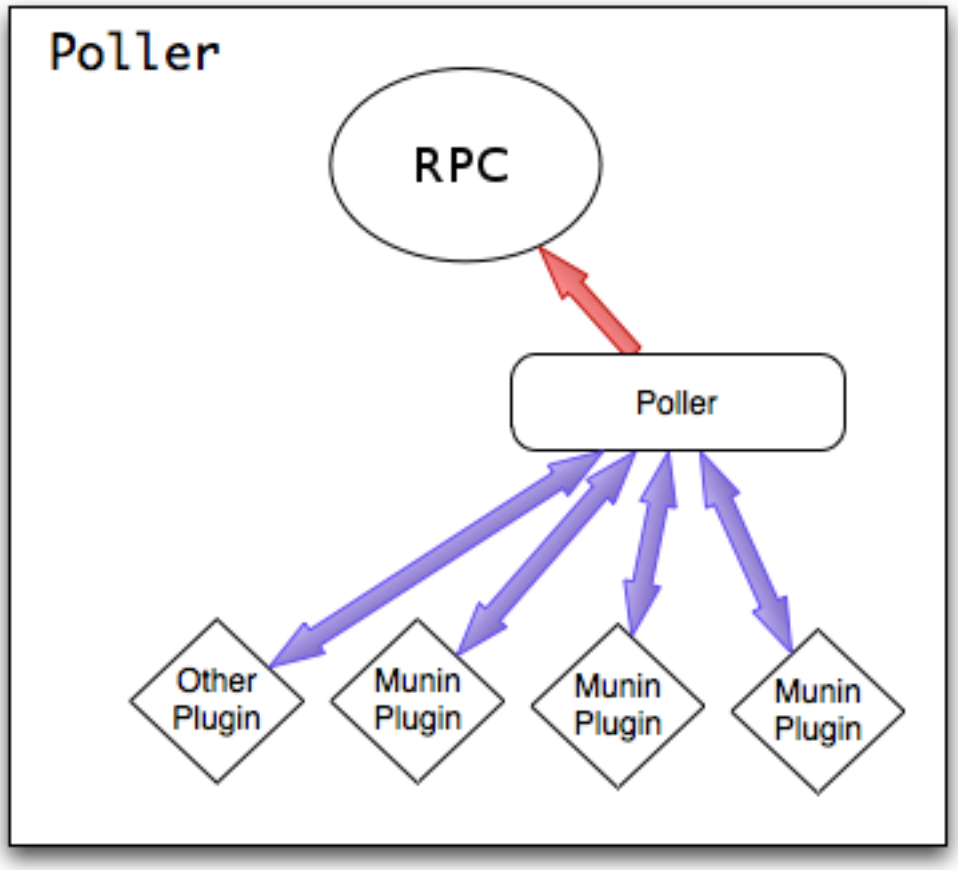
All components can be installed on the same server or on multiple servers depending on their specifications.

Numeter is written in Python and uses Redis. We are considering going from a pull mode to a push mode using rabbitmq.

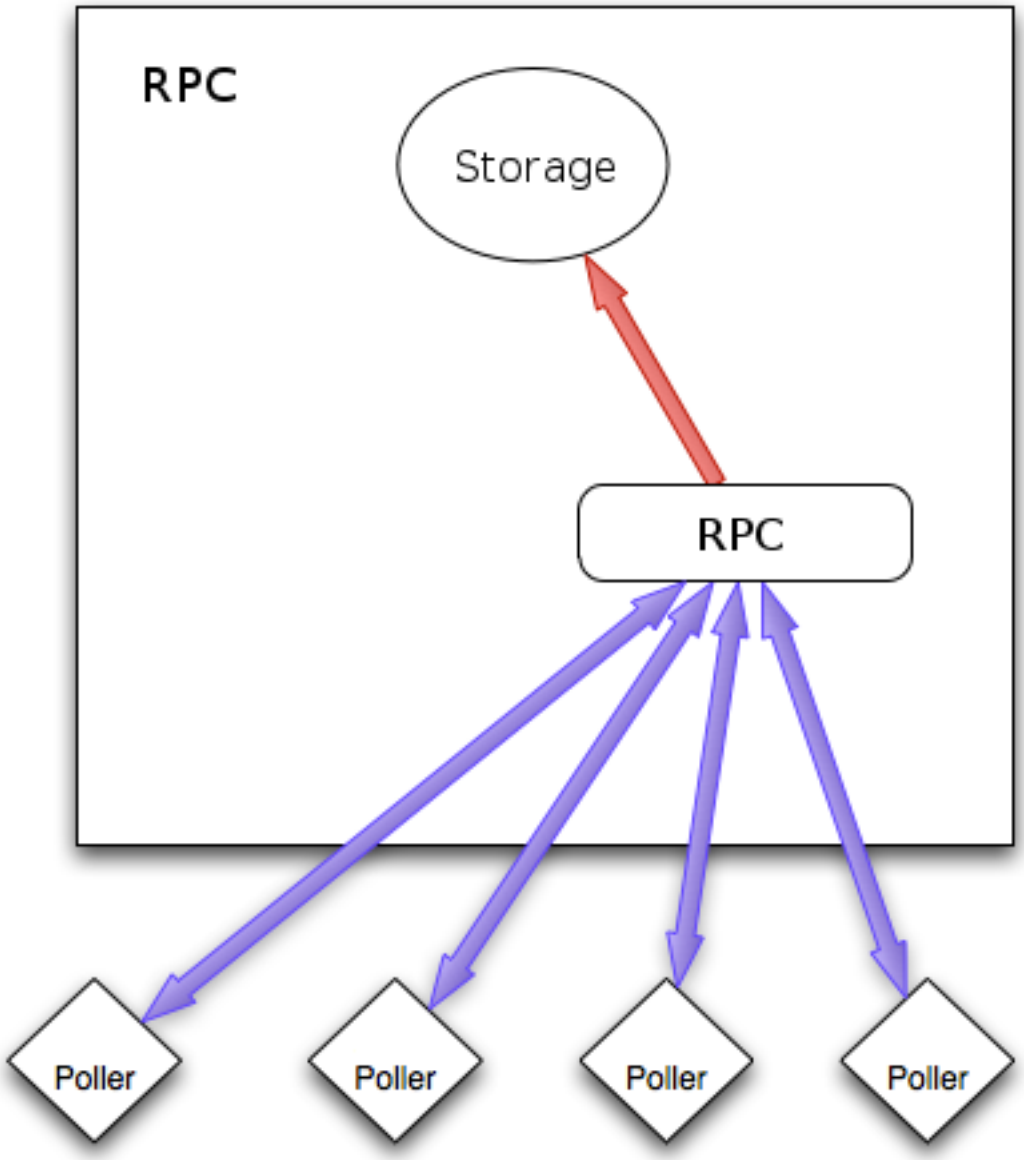
1.1 Components

Current Numeter architecture :

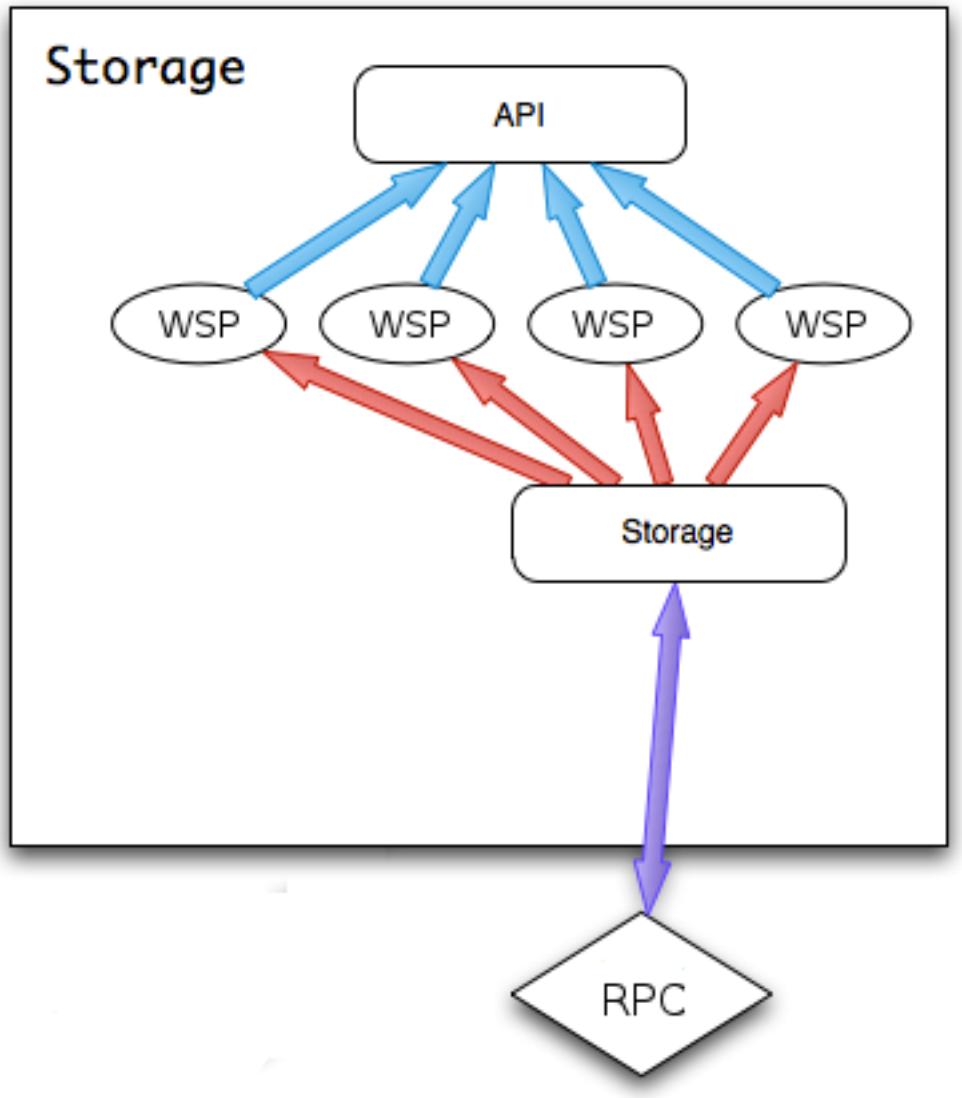
Poller : An agent installed on the servers for which graphs are desired. It gathers data and send them to an rpc. In case of network failure all datas are preserved and sent when network is back.



Rpc : Receive data from poller and provide them to storage.

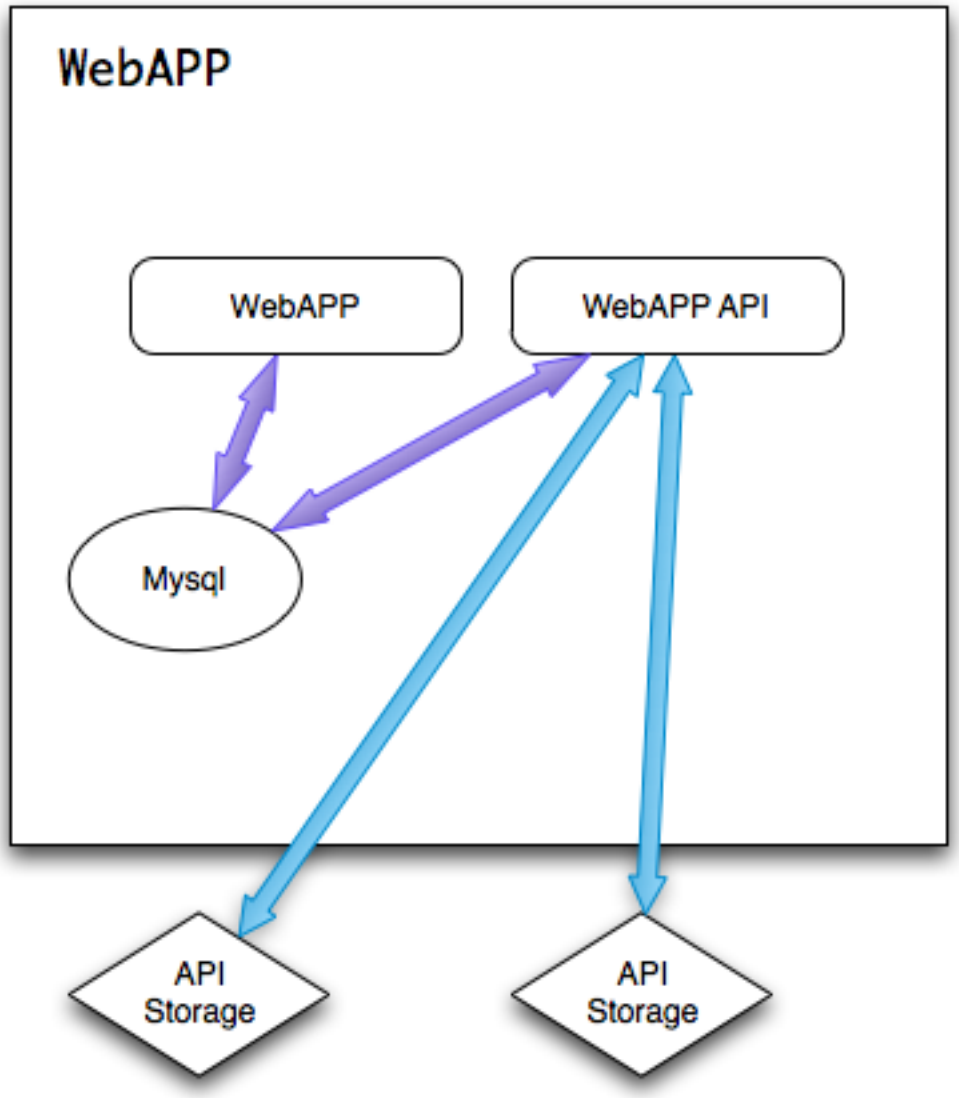


Storage : Data is fetched from the rpc and then stored in WSP files. An HTTP API allows access to the data.



Webapp : A Django webapp displays data using the js library like dygraphs

- http://dygraphs.com/



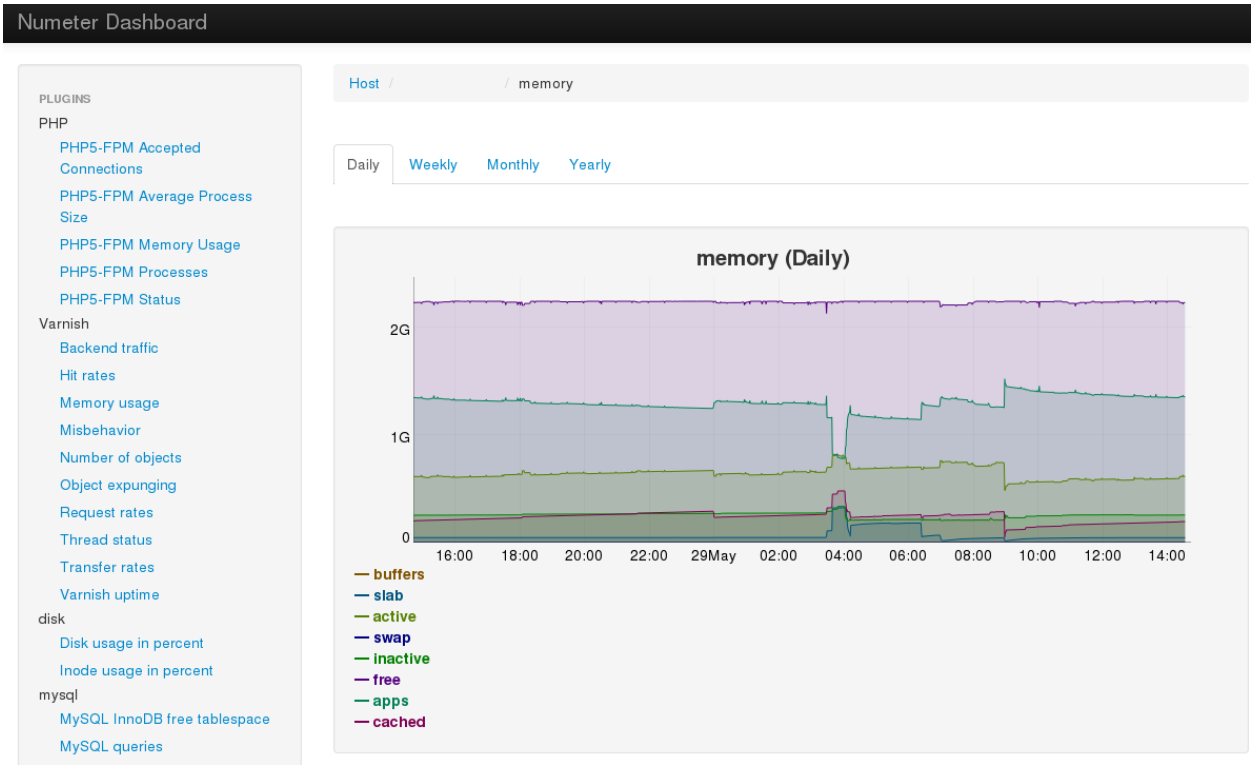
1.2 Features

- Autoconfigure display with plugin datas
- Get data from external sources like munin
- No data loss in case of network failure
- PNG graph export
- User & Group management
- Automatic dashboard creation
- Open architecture allows new features to be created easily
- Scalable

1.3 Functional architecture

Functional architecture :

1.4 Screenshot



CHAPTER 2

Poller

2.1 Documentation

2.1.1 Troubleshooting

The poller logs are located in :

```
/var/log/numeter/numeter-poller.log
```

2.2 Installation

Warning: Do not forget to install a RPC like rabbitmq-server overview.html#components

2.2.1 Debian packages

Build your own Debian packages with git-buildpackage or pbuilder for exemple or use numeter repo : http://repo.numeter.com (coming soon)

```
apt-get install numeter-poller
```

Note: numeter-common depends of python-oslo.messaging. Oslo.messaging is actually packaged only in jessie. If you build your own package for wheezy, actually you can find the others depends (oslo.config, stevedore) in this repository

```
deb http://cloud.pkgs.enovance.com/wheezy-grizzly grizzly main
apt-key adv --recv-keys --keyserver keyserver.ubuntu.com E52660B15D964F0B
```

See also:

Quick config

2.2.2 Manual

Depends:

- python-kombu
- python-oslo.messaging
- numeter-common (rpc module)
- python-daemon
- (rabbitmq-server)
- (munin-node)

Python-daemon

Quick exemple of python-daemon installation

From packages :

```
apt-get install python-daemon
```

Python-kombu

Quick exemple of python-kombu installation

From packages :

```
apt-get install python-kombu
```

Python-oslo.messaging

Quick exemple of python-oslo.messaging installation

From packages :

```
apt-get install python-oslo.messaging
```

From sources : (Get tested version)

Depends

```
apt-get install python-dev python-setuptools
easy_install pip
```

Setup:

```
pip install git+git://github.com/openstack/oslo.messaging.git@7914181398630cbcbc25543d72871ccf812df51
```

Numeter poller

Get Numeter sources :

```
git clone https://github.com/enovance/numeter
cd numeter
```

Numeter common :

```
cd common && python setup.py install
```

Numeter poller :

```
cd poller && python setup.py install
```

See also:

Quick config

2.2.3 Quick config

Numeter poller

Enable numeter poller:

```
vim /etc/numeter/numeter_poller.cfg
enable = false -> true
```

Also add a uniq id in host_id :

```
host_id = 123456789
```

Try to launch numeter-poller one time :

```
numeter-poller
```

Start numeter-poller daemon :

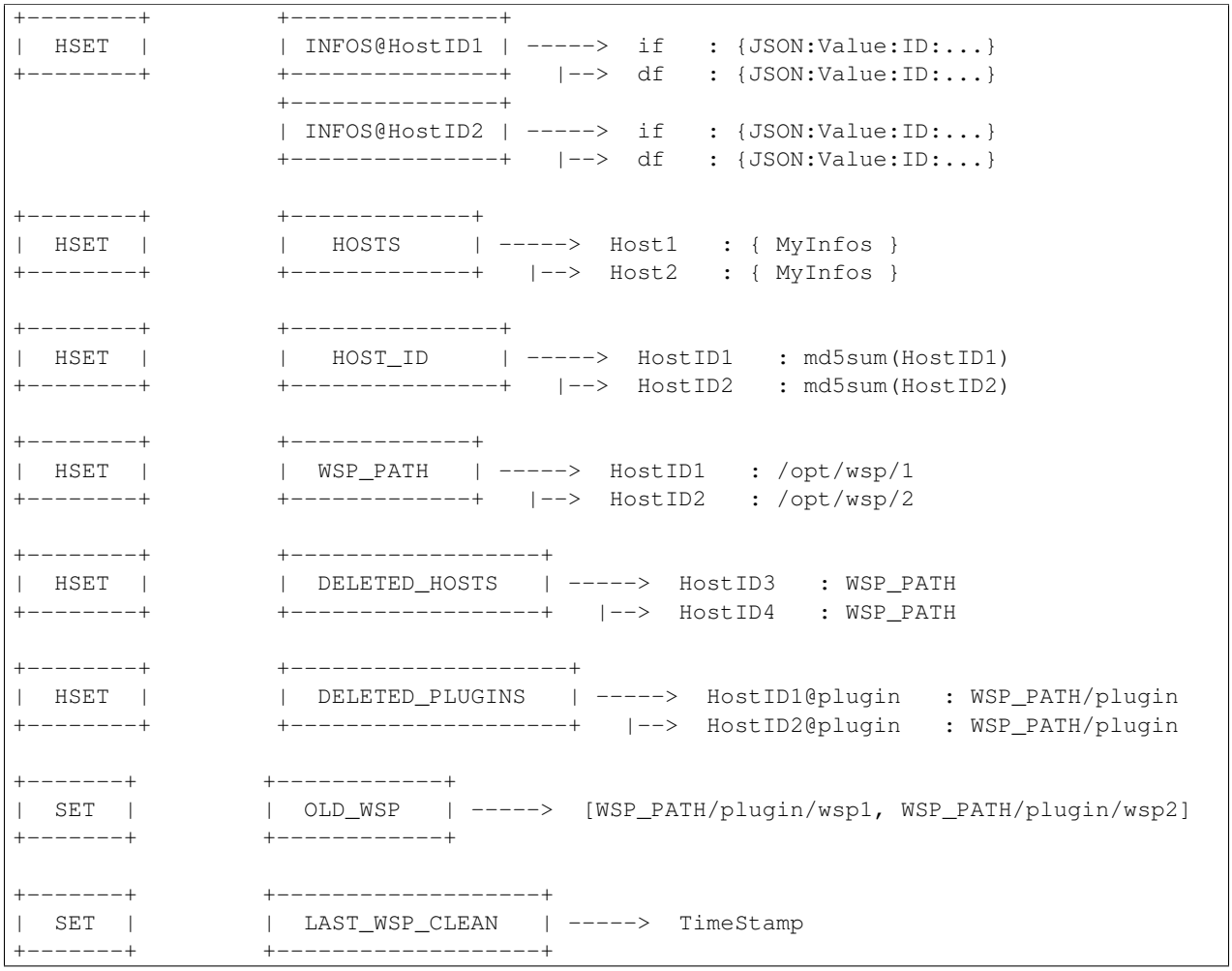
```
/etc/init.d/numeter-poller start
```

CHAPTER 3

Storage

3.1 Documentation

3.1.1 Local database format



3.1.2 Example of host-list file

Example of host list :

```
#Poller <hostID>
myhost-123456
myhost-789456
```

3.1.3 Troubleshooting

Get the host list in the storage

```
redis-cli -n 2 HGETALL HOSTS
```

Get plugin info for a host

```
redis-cli -n 2 hget "INFOS@<hostid>" df
```

Get the wsp path for a host

```
redis-cli -n 2 hget "WSP_PATH" <hostid>
```

3.1.4 Test the storage

Get hosts list from the rest api

```
curl http://127.0.0.1:8080/numeter-storage/hosts
```

Get plugin list from the rest api for a host

```
curl http://127.0.0.1:8080/numeter-storage/list?host=1349792119-d753e11726b68a08dbb5847f0d4b8f8e
```

The rest api code can be found in **/usr/share/numeter/storage/numeter_uwsgi.py**

3.2 Installation

Warning: Do not forget to install a RPC like rabbitmq-server overview.html#components

Note: Numeter storage use **nginx**, **python-flask** and **uwsgi** to provide a "rest" api.

3.2.1 Debian packages

Build your own Debian packages with git-buildpackage or pbuilder for exemple or use numeter repo : http://repo.numeter.com (coming soon)

```
apt-get install numeter-storage
```

Note: numeter-common depends of python-oslo.messaging. Oslo.messaging is actually packaged only in jessie. If you build your own package for wheezy, actually you can find the others depends (oslo.config, stevedore) in this repository

```
deb http://cloud.pkgs.enovance.com/wheezy-grizzly grizzly main
apt-key adv --recv-keys --keyserver keyserver.ubuntu.com E52660B15D964F0B
```

After that you need to configure **nginx** :

```
cp /usr/share/doc/numeter-storage/numeter-storage-web.nginx.example /etc/nginx/sites-available/numete
ln -s /etc/nginx/sites-available/numeter-storage-web /etc/nginx/sites-enabled/
/etc/init.d/nginx restart
```

Same thing for **uwsgi** :

```
cp /usr/share/doc/numeter-storage/numeter-storage-uwsgi.ini.example /etc/uwsgi/apps-available/numeter
ln -s /etc/uwsgi/apps-available/numeter-storage-uwsgi.ini /etc/uwsgi/apps-enabled/
/etc/init.d/uwsgi restart
```

See also:

Quick config

3.2.2 Manual

Depends:

- python-kombu
- python-oslo.messaging
- numeter-common (rpc module)
- python-redis >= 2.4.9
- redis >= 2:2.4.14
- python-whisper
- nginx
- uwsgi
- python-flask
- python-daemon
- (rabbitmq-server)

Nginx

From packages :

```
apt-get install nginx
```

Uwsgi

From packages :

```
apt-get install uwsgi uwsgi-plugin-python
```

Python flask

From packages :

```
apt-get install python-flask
```

Python-whisper

From packages :

```
apt-get install python-whisper
```

Python-redis

Quick exemple of python-redis installation

From packages :

```
apt-get install python-redis
```

From sources :

```
git clone https://github.com/andymccurdy/redis-py
cd redis-py
python setup.py install
```

Redis server

From packages :

```
apt-get install redis-server
```

Python-daemon

Quick exemple of python-daemon installation

From packages :

```
apt-get install python-daemon
```

Python-kombu

Quick exemple of python-kombu installation

From packages :

```
apt-get install python-kombu
```

Python-oslo.messaging

Quick exemple of python-oslo.messaging installation

From packages :

```
apt-get install python-oslo.messaging
```

From sources : (Get tested version)

Depends

```
apt-get install python-dev python-setuptools
easy_install pip
```

Setup:

```
pip install git+git://github.com/openstack/oslo.messaging.git@7914181398630cbcbc25543d72871ccf812df51
```

Numeter storage

Get Numeter sources :

```
git clone https://github.com/enovance/numeter
cd numeter
```

Numeter common :

```
cd common && python setup.py install
```

Numeter storage :

```
cd storage && python setup.py install
```

After that you need to configure **nginx** :

```
cp storage/storage-web/numeter-storage-web.nginx.example /etc/nginx/sites-available/numeter-storage-w
ln -s /etc/nginx/sites-available/numeter-storage-web /etc/nginx/sites-enabled/
/etc/init.d/nginx reload
```

Same thing for **uwsgi** :

```
cp storage/storage-web/numeter-storage-uwsgi.ini.example /etc/uwsgi/apps-available/numeter-storage-uw
ln -s /etc/uwsgi/apps-available/numeter-storage-uwsgi.ini /etc/uwsgi/apps-enabled/
/etc/init.d/uwsgi restart
```

See also:

Quick config

3.2.3 Quick config

Numeter storage

Enable numeter storage:

```
vim /etc/numeter/numeter_storage.cfg
enable = false -> true
```

Add a poller in host-list file:

```
vim /etc/numeter/host-list
#Poller <hostID>
myhost-123456
```

Try to launch numeter-storage :

```
numeter-storage
```

Start numeter-storage daemon :

```
/etc/init.d/numeter-storage start
```

CHAPTER 4

Webapp

4.1 Documentation

4.1.1 Informations

This module is a Web server created for visualize data from Storage(s). In few lines Webapp purposes followings stuff:

- Authentication and authorization made by user and group management
- Multiple storga management
- REST API interface
- Storages' API centralization

Unlike other web graph system, Numeter doesn't only purpose a graph tree. It allow users to organize Data Sources in custom level of displaying.

WebApp is completly independant from other Numeter modules. The only connection point is Storage's REST API.

4.1.2 External Links

4.1.3 Troubleshooting

4.1.4 Configuration

Webapp configuration is located in `/etc/numeter/numeter_webapp.cfg`.

`database` section

`engine`

Warning: Required

This setting defines which database backend will be used. Choose among Django engine or a custom. Default Django's engines a

- `django.db.backends.mysql`

- `django.db.backends.postgresql_psycopg2`
- `django.db.backends.sqlite3`
- `django.db.backends.oracle`

You can use a database backend that doesn't ship with Django by setting ENGINE to a fully-qualified path (i.e. `mypackage.backends.whatever`).

Example:

```
engine = django.db.backends.sqlite3
```

name

Warning: Required

This setting defines name of the database to use. For SQLite, it's the full path to the database file.

Examples:

```
name = numeter
name = /var/db/numeter.db # with Sqlite
```

user

Note: Not required with Sqlite

This setting defines username to use when connecting to the database.

Example:

```
user = numeter
```

password

Note: Not required with Sqlite

This setting defines password to use when connecting to the database.

Example:

```
password = yourpass
```

host

Note: Optionnal

This setting defines host to use when connecting to the database. An empty string means localhost.

Example:

```
host = db.my.lan
```

port

Note: Optionnal

This setting defines port to use when connecting to the database. An empty string means the default port.

Example:

```
port = 3306
```

cache section

use_cache

Note: Optionnal, defaults to `False`.

This setting defines if a cache backend will be used.

Example:

```
use_cache = True
```

location

Warning: Required if `use_cache = True`

This setting defines location of the cache to use. This might be the directory for a file system cache, a host and port for a memcache server, or simply an identifying name for a local memory cache.

Example:

```
location = unix:/var/run/memcached.sock
```

timeout

Note: Optionnal, defaults to `300`.

This setting defines number of seconds before a cache entry is considered stale.

Example:

```
timeout = 3000
```

max_entries

Note: Optionnal, defaults to `300`.

This setting defines maximum number of entries allowed in the cache before old values are deleted.

Example:

```
max_entries = 3000
```

storage section

timeout

Note: Optionnal, defaults to `5`.

This setting defines timeout for connection to storages.

Example:

```
timeout = 10
```

global section

admins

Warning: Required

This setting is a tuple that lists people who get code error notifications.

Example:

```
admins = [ ('admin', 'admin@locahost') ]
```

allowed_hosts

Note: Optionnal, defaults to `['*']`.

This setting is a list of strings representing the host/domain names that this site can serve. *String* ‘‘*’‘‘ representing all.

Example:

```
admins = ['numeter.my.lan','numeter.lan']
```

timezone

Warning: Required

This setting defines time zone for this installation.

Example:

```
timezone = Europe/Paris
```

language_code

Note: Optionnal, defaults to `en-us`.

This setting defines default language of website.

Example:

```
language_code = en-us
```

media_root

Note: Optionnal, defaults to `/var/www/numeter/media/`.

This setting defines location of media files. In facts, a folder named graphlib will be created in. And each folder inside will be a graphic JavaScript library which user will choose.

By default `media_root`'s tree is the following:

```
media_root
|- graphlib
   |- dygraph
      |- dygraph-combined.js
      |- dygraph-numeter.js
```

The default JavaScript library is Dygraph.

Example:

```
media_root = /var/www/numeter/media/
```

secret_key_file

Note: Optionnal, defaults to `/etc/numeter/secret_key.txt`.

This setting defines location of secret key file.

Example:

```
media_root = /etc/key.txt
```

logging section

use_logging

Note: Optionnal, defaults to `False`.

This setting defines if logging will be used. Be sure to have correct logging locations with writing rigth, else webapp won't launch.

Example:

```
use_logging = False
```

info_file

Note: Optionnal, defaults to `/var/log/numeter/webapp/info.log`.

This setting defines location of info log file.

Example:

```
info_file = /tmp/info.log
```

error_file

Note: Optionnal, defaults to `/var/log/numeter/webapp/error.log`.

This setting defines location of error log file.

Example:

```
error_file = /tmp/error.log
```

file_size

Note: Optionnal, defaults to `1000000`.

This setting defines size of log file before rotate. Size is in byte.

Example:

```
file_size = 2000000
```

debug section

debug

Note: Optionnal, defaults to `False`.

Warning: Never set as True in prodiction environment, it could be a real leak of informations.

This setting defines if debuging is enabled.

Example:

```
debug = True
```

use_mock_storage

Note: Optionnal, defaults to `False`.

This setting defines if Mock Storage is enabled. Useful for test numeter without other modules. Two mocks are available over `/mock/X/`, where `X` is `1` or `2`.

Example:

```
use_mock_storage = True
```

use_debug_toolbar

Note: Optionnal, defaults to `False`, set as `False` if `debug = False`.

This setting defines if Django Debug Toolbar is enabled.

Example:

```
use_debug_toolbar = True
```

debug_internal_ips

Note: Optionnal, defaults to `['127.0.0.1']`.

This setting is a list of IPs which use Debug Toolbar.

Example:

```
debug_internal_ips = ['192.168.0.2','192.168.0.3']
```

4.2 Installation

4.2.1 Debian packages

Build your own Debian packages with git-buildpackage or pbuilder for exemple or use numeter repo : http://repo.numeter.com (coming soon)

```
apt-get install numeter-webapp
```

4.2.2 Manual

Depends:

- python-django
- django-rest-framework

- django-filter

Django and rest framework setup

You need `Django >= 1.5` so you need at least `wheezy-backports` sources. You also need `rest-framework` and `django-filter`

From packages :

```
echo 'deb http://ftp.fr.debian.org/debian wheezy-backports main' >> /etc/apt/sources.list.d/debian-ba
apt-get update
apt-get install -t wheezy-backports python-django python-mimeparse

pip install djangorestframework==2.4.0
pip install django-filter
```

Numeter Webapp setup

Get sources

```
git clone https://github.com/enovance/numeter
```

Launch the setup

```
cd numeter/web-app
python setup.py install
```

4.2.3 Configure

Webapp's settings are by default in `/etc/numeter/numeter_webapp.cfg`. You have to define following stuff:

- A database backend
- A web frontend
- Optionnaly a cache backend

Backend database

All Django's database engine are supported, feel free to test third engine.

MySQL

Install python requirements:

```
apt-get install mysql-server mysql-client
apt-get install python-mysqldb
```

Create valid user and database:

```
CREATE DATABASE numeter;
GRANT ALL ON numeter.* TO numeter@'localhost' IDENTIFIED BY 'yourpass';
```

Edit Webaap configuration file, `/etc/numeter/numeter_webapp.cfg`

```
sed -i -re '
  s/^engine.*/engine = django.db.backends.mysql/ ;
  s/^name.*/name = numeter/ ;
  s/^user.*/user = numeter/ ;
  s/^password.*/password = yourpass/ ;
  s/^host.*/host = localhost/ ;
  s/^port.*/port = 3306/' /etc/numeter/numeter_webapp.cfg
```

Sqlite3

```
apt-get install sqlite3
```

Note: You do not need to create database. Only choose a file which writable by django user.

Numeter webapp configuration

Edit Webapp configuration file, follow comments inside for help

```
vim /etc/numeter/numeter_webapp.cfg
```

If you are sure to have a good database configuration

```
numeter-webapp syncdb
```

Note: It will ask you to create a first superuser. If you choose 'no', it will be possible to make it with `numeter-webapp add-user -S`.

Use a web front

We advise to use Apache + mod_wsgi or Nginx + uWSGI. The Django embebbed webserver is usefull for a fast launch witout configuration.

Django embebbed webserver

```
numeter-webapp runserver 0.0.0.0:8000
```

Warning: Not recommended in production environment.

Apache

Install mod wsgi :

```
apt-get install libapache2-mod-wsgi
a2enmod wsgi
```

Copy the site template in your apache configuration:

```
cp web-app/extras/numeter-apache.example /etc/apache2/sites-available/numeter-webapp
# For packaged setup /usr/share/doc/numeter/webapp/numeter-webapp
RESTFW_DIR=$(dirname $(python -c 'import rest_framework;print rest_framework.__file__'))
NUMETER_DIR=$(dirname $(python -c 'import numeter_webapp;print numeter_webapp.__file__'))
sed -i "s#@APP_DIR@#$NUMETER_DIR# ; "s#@RESTFW_DIR@#$RESTFW_DIR# ; s/ServerName.*/ServerName YoursSe
a2ensite numeter-webapp
```

Note: You can customize it for match with your security requirements.

With debian example config are located in */usr/share/doc/numeter-storage/*

Nginx

Install nginx and uwsgi :

```
apt-get install nginx uwsgi uwsgi-plugin-python
```

Copy the site template in your apache configuration:

```
NUMETER_DIR=$(dirname $(python -c 'import numeter_webapp;print numeter_webapp.__file__'))
# Nginx
cp web-app/extras/numeter-nginx.example /etc/nginx/sites-available/numeter-webapp
ln -s /etc/nginx/sites-available/numeter-webapp /etc/nginx/sites-enabled/
sed -i "s#@APP_DIR@#$NUMETER_DIR#g" /etc/nginx/sites-available/numeter-webapp
/etc/init.d/nginx restart
# uwsgi
cp web-app/extras/numeter_webapp.ini.example /etc/uwsgi/apps-available/numeter_webapp.ini
ln -s /etc/uwsgi/apps-available/numeter_webapp.ini /etc/uwsgi/apps-enabled/
sed -i "s#@APP_DIR@#$NUMETER_DIR#g" /etc/uwsgi/apps-available/numeter_webapp.ini
/etc/init.d/uwsgi restart
```

Note: You can customize it for match with your security requirements.

With debian example config are located in */usr/share/doc/numeter-storage/*

Use memcached backend

Note: Cache is not mandatory

Install Memcached

```
apt-get install memcached python-memcache
```

Set 'use_cache' to True in numeter_webapp.cfg:

```
sed -i 's/use_cache.*=.*/use_cache = True/' /etc/numeter/numeter_webapp.cfg
```

Note: By default Webapp will connect to Memcached with a Unix socket at `/var/run/memcached.sock`.

4.2.4 First launch

```
numeter-webapp syncdb
numeter-webapp storage add --name=local_storage --port=8080 --url_prefix=/numeter-storage --address=1
```

Note: `syncdb` will ask you to create a first superuser. If you choose 'no', it will be possible to make it with `numeter-webapp user add -S`.

Launch your Web frontend or simply `numeter-webapp runserver 0.0.0.0:80`. You must add hosts in database for use it, go to 'Configuration/Storage/YourStorage/' and click on button Create hosts.

CHAPTER 5

Roadmap

roadmap :

- Change poller pull to push with rabbitmq
- Storage bench and try differents storage type like carbon or mongodb
- Improve unittest and make numeter debian reposiroty
- poller : Make poller module or break the poller to allow Split datas from configuration (which able to push RAW datas without configuration like carbon)
- Provide new webapp in Django

CHAPTER 6

About

Numeter is a new and dynamic graphing solution made by some of the folks at eNovance. We use it as part of our cloud solutions. It is based on Python, sexy and highly scalable.

Like many, we started with Cacti and Munin a long time ago. Then, we joined these two software packages into one and we called it Mucti.

Over time and with an increasing number of hosts, we have seen the limitations of this approach. We launched the Numeter project, aiming for more modularity and scalability.

Each component of the Numeter architecture can be installed separately. Our policy for the project is to rely on standard technologies and therefore JSON is used for communication between the components and to access data using the HTTP API

You are free to use only the components you need. For example, the poller with your modules or develop your own application to process the data contained in the storage component.

CHAPTER 7

FAQ

How to build the sphinx doc ? : The api.rst use doc strings and need to import Numeter modules. So the doc is builded by tox.

```
tox -e docs
```

How to launch unit tests ? : Launch unit tests with tox

```
tox -epy27 tests/units
```

CHAPTER 8

api

8.1 numeter

8.1.1 numeter.queue.client

class numeter.queue.client.**BaseAPIClient**(*transport*)

Bases: oslo_messaging.rpc.client.RPCClient

__init__(*transport*)

__module__ = 'numeter.queue.client'

ping(*context*, *topic*, *args=None*)

poller_msg(*context*, *topic*, *args=None*)

numeter.queue.client.**get_rpc_client**(*hosts=[]*, *password='guest'*)

8.1.2 numeter.queue

8.1.3 numeter.queue.server

class numeter.queue.server.**OverrideMessageHandlingServer**(*transport*, *dispatcher*, *executor='blocking'*)

Bases: oslo_messaging.server.MessageHandlingServer

override MessageHandlingServer to bind multiple topics

__module__ = 'numeter.queue.server'

start()

class numeter.queue.server.**Targets**(*targets*)

Bases: oslo_messaging.target.Target

Target objects with additionnal targets

__init__(*targets*)

__module__ = 'numeter.queue.server'

numeter.queue.server.**_get_rpc_server**(*transport*, *target*, *endpoints*, *executor='blocking'*, *serializer=None*)

numeter.queue.server.**get_rpc_server**(*topics*, *server*, *hosts*, *endpoints*, *password='guest'*)

8.1.4 numeter.redis

class numeter.redis.**RedisConnect**(*host='127.0.0.1'*, *port=6379*, *socket_timeout=2*, *password=None*, *db=0*)

Bases: object

Frontend redis avec try sur les get

__dict__ = dict_proxy({'redis_hmset': <function redis_hmset at 0x7f15d222bd70>, '__module__': 'numeter.redis', 'red

__init__(*host='127.0.0.1'*, *port=6379*, *socket_timeout=2*, *password=None*, *db=0*)

__module__ = 'numeter.redis'

__weakref__
list of weak references to the object (if defined)

redis_connect()

redis_get(*key*)

redis_hdel(*name*, *key*)

redis_hexists(*name*, *key*)

redis_hget(*name*, *key*)

redis_hgetall(*name*)

redis_hkeys(*name*)

redis_hlen(*name*)

redis_hmget(*name*, *key*)

redis_hmset(*name*, *mapping*)

redis_hset(*name*, *key*, *value*)

redis_hvals(*name*)

redis_info()

redis_keys(*key*)

redis_set(*key*, *value*)

redis_zadd(*name*, *value*, *score*)

redis_zcount(*name*, *valmin*, *valmax*)

redis_zrangebyscore(*name*, *valmin*, *valmax*, *start=None*, *num=None*)

redis_zremrangebyscore(*name*, *valmin*, *valmax*)

8.1.5 numeter.storage.numeter_storage_endpoints

class numeter.storage.numeter_storage_endpoints.**StorageEndpoint**(*storage*)

Bases: object

__dict__ = dict_proxy({'__module__': 'numeter.storage.numeter_storage_endpoints', 'poller_msg': <function poller_n

__init__(*storage*)

__module__ = 'numeter.storage.numeter_storage_endpoints'

__weakref__
list of weak references to the object (if defined)

_call_write_data (*ctxt*, *args*)

_call_write_info (*ctxt*, *args*)

ping (*ctxt*, *args*)

poller_msg (*ctxt*, *args*)

8.2 numeter.storage

8.2.1 numeter.storage.numeter_storage

class numeter.storage.numeter_storage.**Storage** (*configFile='/etc/numeter_storage.cfg'*)
Bases: object

__dict__ = dict_proxy({'__module__': 'numeter.storage.numeter_storage', '_get_host_list': <function _get_host_list at

__init__ (*configFile='/etc/numeter_storage.cfg'*)

__module__ = 'numeter.storage.numeter_storage'

__weakref__
list of weak references to the object (if defined)

_get_hostIDHash (*hostID*)
Get cached hostID hash or add in cache

_get_host_list ()
Get host list

_write_data (*hostID*, *plugin*, *data_json*)

_write_info (*hostID*, *info_json*)

cleanHosts (*writedHosts*)
Clean info in redis and wsp

cleanInfo (*writedInfos*, *hostID*)
Clean info in redis and wsp

cleanOldWSP ()
Clean old wsps

getgloballog ()
Init du logger (fichier et stdr)

jsonToPython (*data*)
Convert json to python

paramsVerification ()
Args verification

pythonToJson (*data*)
Convert python to json

readConf ()
Read configuration file

redisStartConnexion ()

startStorage()
Start storage

8.3 numeter.poller

8.3.1 numeter.poller.munin_connect

class numeter.poller.munin_connect.**MuninConnection**(*munin_host='127.0.0.1'*, *munin_port=4949*)
Bases: object

Read lines from MuninSock. Provide also basic munin-node commandes :

- Config
- Fetch
- List
- Nodes

__dict__ = dict_proxy({'__module__': 'numeter.poller.munin_connect', '_iterline': <function _iterline at 0x7f15d1c9b0

__init__(*munin_host='127.0.0.1'*, *munin_port=4949*)

__module__ = 'numeter.poller.munin_connect'

__weakref__
list of weak references to the object (if defined)

_iterline()

_readline()

munin_config(*key*)

munin_fetch(*key*)

munin_list()

munin_nodes()

class numeter.poller.munin_connect.**MuninSock**(*host*, *port*)
Bases: object

Open and close a socket with munin-node socket. This class allow you to use with statement (for python < 3).

```
with MuninSock(self.munin_host, self.munin_port) as sock:
  ...
```

__dict__ = dict_proxy({'__module__': 'numeter.poller.munin_connect', '__exit__': <function __exit__ at 0x7f15d1c96e

__enter__()
Open socket with munin-node

__exit__(*type*, *value*, *traceback*)
Close socket with munin-node

__init__(*host*, *port*)

- host : munin-node host
- port : munin-node port

__module__ = 'numeter.poller.munin_connect'

__weakref__
list of weak references to the object (if defined)

8.3.2 numeter.poller.storeandforward

class numeter.poller.storeandforward.**StoreAndForward**(*cache_file='/dev/shm/store_and_forward.json'*, *logger='numeter.poller.storeandforward'*)

Bases: object

Use case exemple :

Init logging level with debug stream handler:

```
logging.getLogger('StoreAndForward').setLevel(logging.INFO)
logging.getLogger('StoreAndForward').addHandler(logging.StreamHandler())

from time import time

with StoreAndForward(cache_file='./sandbox/cache_storeandforward.json') as cache :

    # Add stored message
    cache.add_message('DATA', 'munin.if_eth0.up', '{%s content}' % time())

    Read stored message
    for message in cache.consume():
        print message
```

__dict__ = dict_proxy({'__module__': 'numeter.poller.storeandforward', '_load_cache': <function _load_cache at 0x7f

__enter__()

__exit__(*type*, *value*, *traceback*)

__init__(*cache_file='/dev/shm/store_and_forward.json'*, *logger='numeter.poller.storeandforward'*)

__module__ = 'numeter.poller.storeandforward'

__weakref__
list of weak references to the object (if defined)

_dump_cache()

_load_cache()

add_message(*msgType*, *plugin*, *msgContent*)

consume()

8.3.3 numeter.poller.muninModule

class numeter.poller.muninModule.**MuninModule**(*configParser=None*)

Bases: *numeter.poller.modulesGeneric.ModulesGeneric*

This poller module fetch configs and datas from a munin-node daemon.

Configuration section for this module :

```
[MuninModule]
plugins_enable = ^.*$    ; Regexp matching plugins to fetch
munin_host = 127.0.0.1   ; Munin Host
munin_port = 4949        ; Munin Port
```

__init__(*configParser=None*)

When numeter load a module, the only one parameter is the config parser. It's allow you to add section in numeter config file for your own module.

__module__ = 'numeter.poller.muninModule'

formatFetchData(*plugin*)

Execute a `munin fetch` for a given plugin name. Get the result and format the raw result for `getData` :

```
data = {  'TimeStamp': nowTimestamp,
          'Plugin': plugin,
          'Values': pluginData
}
```

formatFetchInfo(*plugin*)

Execute a `munin config` for a given plugin name. Get the result and format the raw result for `getInfo` :

```
infos =  {    'Plugin': plugin,
              'Base': '1000',
              'Describ': '',
              'Title': plugin,
              'Vlabel': '',
              'Order': '',
              'Infos': {}
         }
```

getData()

get and return all datas collected from munin-node. See modulesGeneric return format

For each plugin matching `plugins_enable` regex in the config file, launch `formatFetchData` and append result to `getData` output

getInfo()

get and return all infos collected from munin-node. See modulesGeneric return format

For each plugin matching `plugins_enable` regex in the config file, launch `formatFetchInfo` and append result to `getInfo` output

getParserConfig()

Read MuninModule from configParser

8.3.4 numeter.poller.cachelastvalue

class numeter.poller.cachelastvalue.**CacheLastValue**(*cache_file='/dev/shm/cache_last_value.json'*, *logger='numeter.poller.cachelastvalue'*)

Bases: `object`

__dict__ = dict_proxy({'__module__': 'numeter.poller.cachelastvalue', '_load_cache': <function _load_cache at 0x7f15

__enter__()

__exit__(*type*, *value*, *traceback*)

__init__(*cache_file='/dev/shm/cache_last_value.json'*, *logger='numeter.poller.cachelastvalue'*)

__module__ = 'numeter.poller.cachelastvalue'

__weakref__
list of weak references to the object (if defined)

_dump_cache()

_load_cache()

get_value(*key*)

save_value(*key*, *timestamp*, *value*)

8.3.5 numeter.poller.numeter_poller

class numeter.poller.numeter_poller.**Poller**(*configFile='/etc/numeter_poller.cfg'*)
Bases: object

__dict__ = dict_proxy({'__module__': 'numeter.poller.numeter_poller', 'getMyInfo': <function getMyInfo at 0x7f15d1

__init__(*configFile='/etc/numeter_poller.cfg'*)

__module__ = 'numeter.poller.numeter_poller'

__weakref__
list of weak references to the object (if defined)

_sendData(*allDatas*)
Send data in rpc

_sendInfo(*allInfos*)
Send info in rpc

_sendMsg(*msgType*, *plugin*, *msgContent*)

_store_and_forward_sendMsg(*msgType*, *plugin*, *msgContent*)

convertToJson(*data*)
Convert data to json

getMyInfo()
Read my infos in conf file

getgloballog()
Init logger (file and stdr)

loadModules()
Get and write data / infos of all modules

pollerTimeToGo()
LAST + poller_time <= NOW calcule aussi le refresh time

readConf()
Read configuration file

startPoller()
Start the poller

writeInSimulateFile(*message*)
Write in simulate file

8.3.6 numeter.poller.modulesGeneric

class numeter.poller.modulesGeneric.**ModulesGeneric**(*configParser=None*)
Bases: object

Module generic is the skeleton to build your own numeter module. If you want to write your own numeter module, you can take this class and build your own with it.

__dict__ = dict_proxy({'__module__': 'numeter.poller.modulesGeneric', 'getInfo': <function getInfo at 0x7f15d1c96b1

__init__(*configParser=None*)
When numeter load a module, the only one parameter is the config parser. It's allow you to add section in numeter config file for your own module.

__module__ = 'numeter.poller.modulesGeneric'

__weakref__
list of weak references to the object (if defined)

getData()
Get and return all collected datas. The output must be formated like that :

```
data=   [{      'TimeStamp': 1410907894,
                'Plugin': 'if',
                'Values': {
                            'up' : 40,
                            'down' : 15,
                          }
         }]
```

getInfo()
Return for all datas you have collected. The output must be formated like that :

```
infos=   [{   'Plugin': 'if',
              'Base': '1000',
              'Describ': '',
              'Title': 'if eth0',
              'Vlabel': '',
              'Order': '',
              'Infos': {
                    "down":{"type": "COUNTER", "id": "down", "label": "received"},
                    "up":{"type": "COUNTER", "id": "up", "label": "upload"},
              }
          }]
```

> **Warning:** Each DataSource (each datas) you define in getData must have at least an entry in getInfo return like `"id":{"id": "up"}`

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

n

Symbols

A

B

C

F

G

J

L

M

N

O

P

R

S

T

W