Table of Contents

Introduction **1**

NEET (Nagios External Experiment Tests) **2**

Usage "exp_test_result_parser.py" 2

Usage "exp_test_result_publisher.py" 3

Usage "neet.cfg" 3

Example of usage 3

How to configure NAGIOS box to accept external tests **4**

Client side 4

Server side 4

Introduction

NEET (**N** agios **E** xternal **E** xperiment **T** ests), is a tool to send experiment tests via "MSG" (Message BUS) to a specific QUEUE. Nagios reads from the queue and publishes the information.

NEET (Nagios External Experiment Tests)

Neet is written in python. It contains 3 files:

- **exp_test_result_parser.py**
- **exp_test_result_publisher.py**
- **neet.cfg**

Usage "exp_test_result_parser.py"

In order to use NEET, you have to follow these steps:

1. Run **exp_test_result_parser.py** . The script reads the information regarding the experiment tests from file. The path to the input file must be passed as first argument to the script. The input file must contain a section with the metrics used in the input file itself. The metrics must be described in a dictionary structure and for each field the user has to specify the position of the field in the input file, i.e.::

show hide

```
_metrics  = {
             'hostName' : {'metric_description': " Host Name",
                           'field_position'    : 0,
                          },
             'QueueName' : {'metric_description': "Queue Name",
                           'field_position'    : 1,
                          },
             'spaceTokenName' : {'metric_description': "Space Token Name",
                           'field_position'    : 2,
                          },
             'status' : {'status': "Status in string format",
                           'field_position'    : 3,
                          },
             'metricStatus' : {'metric_description': "Status in number format",
                           'field_position'    : 4,
                          },
             'jobId' :  {'metric_description': "Job ID",
                           'field_position'    : 5,
                          },
             'AthenaExitCode' : {'metric_description': "Athena Exit Code",
                           'field_position'    : 6,
                          },
             'metricName' : {'metric_description': "......",
                           'field_position'    : 7,
                          },
             'dataSetName' : {'metric_description': "......",
                           'field_position'    : 8,
                          },
             'startTime' : {'metric_description': "start time of the Job",
                           'field_position'    : 9,
                          },
             'endTime' : {'metric_description': "end time of the Job",
                           'field_position'    : 10,
                          },
             }
```

2. Create a new dictionary with the specific metric for Nagios, for example:

```
hostName: ce5.pic.es
metricName: org.atlas.WN-gangarobot_wms-atlas
metricStatus: OK
```

```
timestamp:2010-08-16T08:18:44Z
summaryData: mc09_7TeV.105193
detailsData: mc09_7TeV.105193.CompHepPythia....
EOT
```

3. call the method **send** of **exp_test_result_publisher** to dispatch the information into MSG

Usage "exp_test_result_publisher.py"

This script, is only used to publish information to Nagios,

The method format_Message takes the metrics dictionary as first parameter and uses it to create a message in "WLCG format" which can be read by Nagios. The message is sent via MSG to a dedicated QUEUE where Nagios is listening.

```
...
  send ('hostName: '+_metrics ['hostName'] +'\n'+
        'metricName: '+_metrics ['metricName']+'\n'+
        'metricStatus: '+_metrics ['metricStatus']+'\n'+
        'timestamp: '+_metrics ['timeStamp']+'\n'+
        'summaryData: '+_metrics ['summaryData']+'\n'+
        'detailsData: '+_metrics ['detailsData']+'EOT\n')

..
  conn.send(message,destination=QUEUE, ack='auto')
...
```

Usage "neet.cfg"

The user must use this file to define the parameters for the connection to MSG using the stomp protocol.

```
[Connection]
HOST: gridmsg002.cern.ch
PORT: 6163
QUEUE: /queue/grid.probe.metricOutput.EGEE.bece5c857f68263c125890256fa8833c
```

....

Example of usage

Now, Neet works to publish **GangaRobot Experiment Test result** , it's running with the user **ddmusr03** in:

```
 /afs/cern.ch/user/d/ddmusr03/public/nagios_publisher
```

There is a script file, **gangarobot_publisher.sh**, running hourly, that downloads a GangaRobot Experiment files (type LatestResult*) and gives one file at a time to **exp_test_result_parser.py**

```
$HOME/public/nagios_publisher/hammercloud.cern.ch/sam/ -name 'LatestResults*'  -exec  $HOME/publi
```

How to configure NAGIOS box to accept external tests

Client side

The lightweight library used to send messages via stomp protocol to a Message Bus requires a small configuration client side (referring to as *neet.conf*). Once the formatted input file for the parser, one has to insure that the publisher script points to the right configuration file:

this is the relevant part of the file exp_test_result_publisher.py to be modified

```
config.read("/afs/cern.ch/user/s/santinel/scratch0/grid/conditionDB/publisher/neet.cfg")
```

and then editing this file cfg file one sets the destination message bus endpoints.

```
[lxplus406] ~/scratch0/grid/conditionDB/publisher $ cat /afs/cern.ch/user/s/santinel/scratch0/gri
# Neet config file

[Connection]
HOST: gridmsg102.cern.ch
PORT: 6163
#QUEUE: /queue/grid.probe.LHCb-externals
QUEUE: /queue/grid.probe.metricOutput.EGEE.sam-lhcb_cern_ch
```

Server side

The publication of well formatted test results in the Message Bus via NEET is a trivial operations; the configuration of the Nagios box to consume these results is also very simple but a bit of care must be payed. These tests are passive checks but they do not have parents active checks in Nagios. A clean solution proposed in this document is to define in the ncg.conf file a profile ad hoc where generic VO test can be later defined in the Hash.pm. We show in this example the ncg.conf file used by LHCb:

```
$ cat /etc/ncg/ncg.con
...
...
<NCG::LocalMetrics>
  <File>
    DB_FILE=/etc/ncg/ncg.localdb
    DB_DIRECTORY=/etc/ncg/ncg-localdb.d
  </File>
  <Hash>
      PROFILE = VO_PROD
      VO_FQAN = /lhcb/Role=production
  </Hash>
  <Hash>
      PROFILE = VO_EXTERNAL
      VO_FQAN = lhcb
  </Hash>
  <Hash>
      PROFILE = VO_PILOT
      VO_FQAN = /lhcb/Role=pilot
  </Hash>
</NCG::LocalMetrics>
```

We have defined two profiles:

- VO_PROD: used to define all metrics running within Nagios with Role=production
- VO_PILOT: used to define all metrics running within Nagios with Role=pilot
- VO_EXTERNALS: used to define all metrics running **outside** Nagios with undefined FQAN used to be submitted.

Please note that the metricName of your external tests (as will be known to Nagios) will be the metric name + "-lhcb" appended at the end. In your clients please insure that metricName field reflects exactly this convention otherwise Nagios will not publish results. Now we have to inform Nagios about these new external metrics and this is almost identical of the definition of any other test. Add the following lines in your Hash.pm for each of the metrics that you want to publish. Please note here the parent is *org.sam.CE-JobState* just because we wanted to publish this WN specific tests with the rest of the WN tests submitted via Nagios. But this is not compulsory. Took exactly this example and change the name of your test from org.lhcb.WN-CondDB to whatever you want.

```
#### external submission via NEET
#org.lhcb.WN-CondDB
$WLCG_SERVICE->{'org.lhcb.WN-CondDB'}->{flags}->{PASSIVE} = 1;
$WLCG_SERVICE->{'org.lhcb.WN-CondDB'}->{parent} = "org.sam.CE-JobState";
$WLCG_SERVICE->{'org.lhcb.WN-CondDB'}->{flags}->{VO} = 1;
$WLCG_SERVICE->{'org.lhcb.WN-CondDB'}->{flags}->{OBSESS} = 1;
$WLCG_SERVICE->{'org.lhcb.WN-CondDB'}->{metricset} = 'org.lhcb.WN';
```

At this point you have to inform Nagios that this metrics belongs to the VO_EXTERNAL profile for the service CE. This is trivially done as described in the relevant snipped of the Hasp.pm:

```
..
..
..
### external publisher for VO_external
$WLCG_NODETYPE->{VO_EXTERNAL}->{CE} = [
'org.lhcb.WN-CondDB'
];

$WLCG_NODETYPE->{VO_EXTERNAL}->{SRMv2} = [
'org.lhcb.SRM-Dirac-Unit'
];
..
..
```

At this point just re-configure your nagios box and restart all services as reported in the following procedure.

-- AleDiGGi - 16-Aug-2010

This topic: LCG > ExperimentTestsMSGPublisher
Topic revision: r11 - 2010-11-29 - unknown