OnTimeMeasure Command-line User Manual

1. Overview

The OnTimeMeasure command-line tools provide a way to control the measurement service besides using the OnTimeMeasure web-portal. This document shows how to use the command-line tools which can do:

- Manage measurement tasks
- Start/Stop measurement service
- Check measurement service status
- Query measurement results
- Query measurement archive

2. Requirements

The OnTimeMeasure command-lines tools have the following requirements:

A. A measurement slice that has been set up.

B. Python 2.5.X or 2.6.X (Currently not compatible with python 2.7 or 3.0)

3. Installation and Configuration

Download the tarball from http://ontime.oar.net/download/OnTimeControl_latest.php, extract the files to any folder.

Change the config_example.yaml to config.yaml and edit it with any text editor.

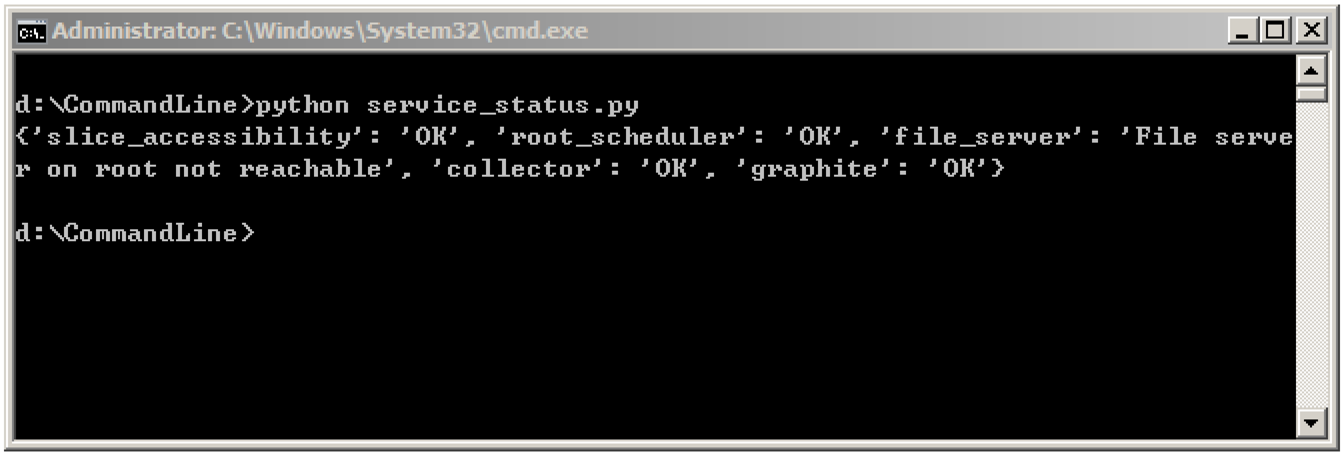
Following the in file instructions to configure the OnTimeControl. It is strongly suggested to use the "direct" connection mode. The portal model should only be used when the OnTimeControl can not connect to the RootBeacon directly.

4. Usage

4.1 Check measurement service status

Run command "service_status.py" can check both whether the command line tools are installed successfully and how the measurement service goes.

Example:



```
d:\CommandLine>python service_status.py
{'slice_accessibility': 'OK', 'root_scheduler': 'OK', 'file_server': 'File serve
r on root not reachable', 'collector': 'OK', 'graphite': 'OK'}

d:\CommandLine>
```

4.2 Manage measurement tasks

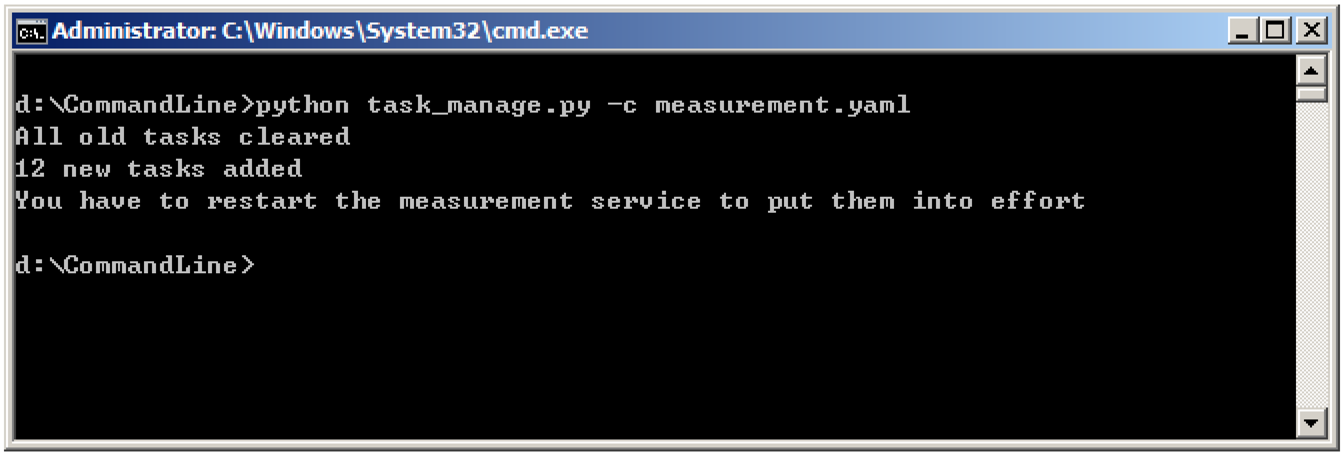
"task_manage.py" manages the measurement tasks, which are configured in a YAML file. An example "measurement.yaml" is included in the package with contents:

```
# Measurement task settings
# Hint: A space is required after ':'

##################################################
# Supported patters:
# - Periodic
# - RandomPoisson
# - RandomExponential
# - RandomGaussian
# - StratifiedRandom
# - Adaptive
##################################################
pattern: Periodic

##################################################
# Measurement links
# Please use node names as source and destination
# Supported metrics:
# - RoundtripDelay
# - Throughput
# - Loss
# - Jitter
# - RouteChanges
##################################################
links:
- source: WASH
  destination: SALT
  metric: [Throughput, RoundtripDelay, Jitter, RouteChanges]
```

```
- source: SALT
  destination: WASH
  metric: [Jitter, RoundtripDelay, Throughput, RouteChanges]
```

Create your measurement configuration file by following the example and comments of the file. Then run the "task_manage.ph" with parameter "-c <measurement_config_file.yaml>".

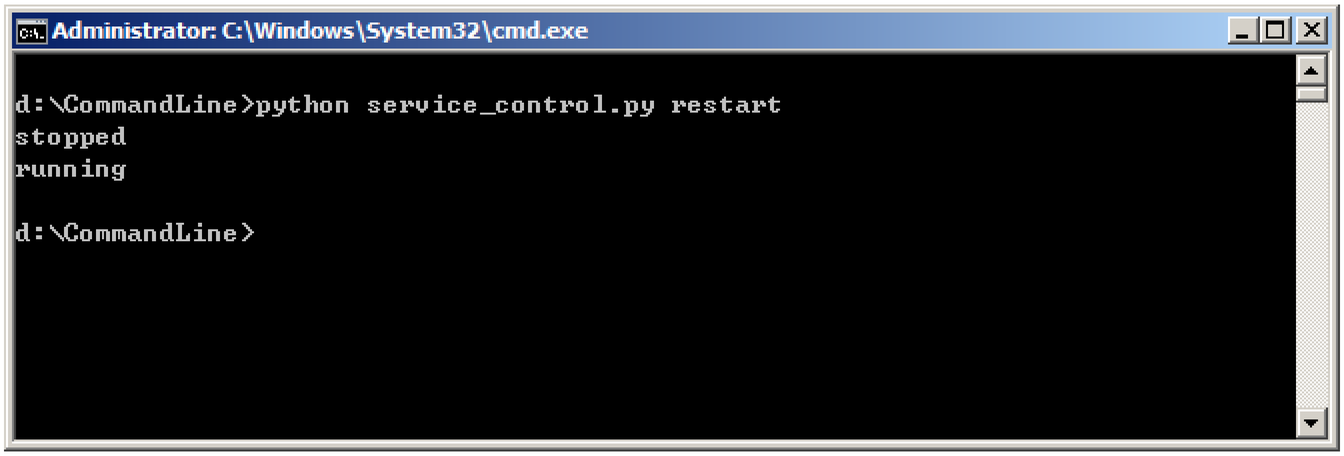
An example:


Administrator: C:\Windows\System32\cmd.exe

```
d:\CommandLine>python task_manage.py -c measurement.yaml
All old tasks cleared
12 new tasks added
You have to restart the measurement service to put them into effort

d:\CommandLine>
```

4.3 Measurement service control

"service_control.py" controls the measurement service. Supported parameters are "start", "stop", "restart".


Administrator: C:\Windows\System32\cmd.exe

```
d:\CommandLine>python service_control.py restart
stopped
running

d:\CommandLine>
```

4.4 Query measurement results

"query.py" queries the measurement results. Parameters for the query are:

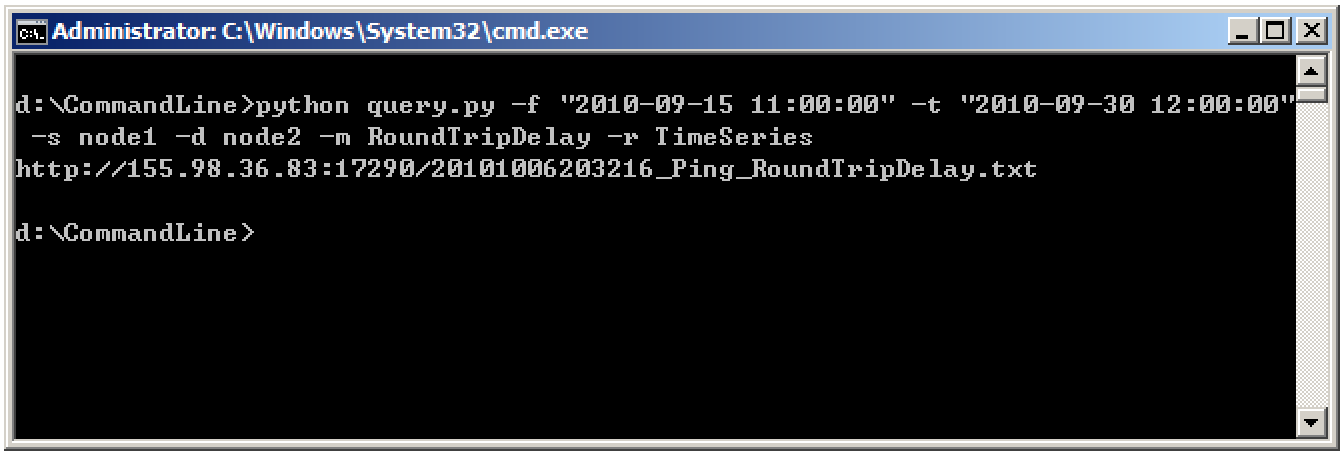
```
-f from time "YYYY-MM-DD HH:MM:SS" in UTC
```

```
-t to time "YYYY-MM-DD HH:MM:SS" in UTC
-s source node name
-d destination node name
-m metric {RountTripDelay|Throughput|Loss|Jitter}
-r result type
{RawFiles|TimeSeries|TimeSeriesPlusAnomalies|TimeSeriesPlusForecasts}
```

Measurement results are automatically downloaded into the local folder.

An example query:

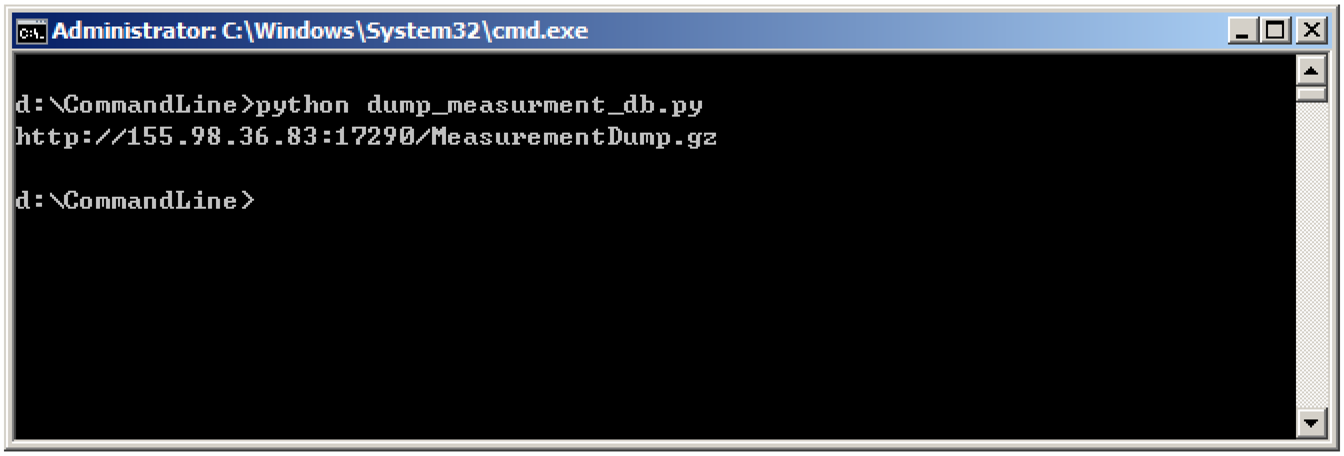
```
python query.py -f "2010-09-15 11:00:00" -t "2010-09-30 12:00:00" -s
node1 -d node2 -m RoundTripDelay -r TimeSeries
```



```
d:\CommandLine>python query.py -f "2010-09-15 11:00:00" -t "2010-09-30 12:00:00"
 -s node1 -d node2 -m RoundTripDelay -r TimeSeries
http://155.98.36.83:17290/20101006203216_Ping_RoundTripDelay.txt

d:\CommandLine>
```

5. Query measurement archive

"dump_measurement_db.py" queries the measurement archive in the root beacon and automatically downloaded into the local folder.



```
d:\CommandLine>python dump_measurment_db.py
http://155.98.36.83:17290/MeasurementDump.gz

d:\CommandLine>
```