OpenMUL – Python App and RESTAPI Dev Guide

open
MUL

Contents

After reading this document one should be able to:

- Make new NBAPI’s for Mul controller
- Use python APIs in an application
- Use REST-APIs in a web application

Prerequisites:

- MuL Controller code and other required packages/libraries. Please refer to this post for more detailed instruction.
- Average C/Python/Python-swig language knowledge

Native Python Application

The build process of OpenMUL controller creates a python library which can directly access access various apis exported by different subsystems of MUL Controller. For more info on the APIs exported kindly refer to “Openmul_MLAPI-ReferenceDoc”

After build Mul controller code, the python application enabler libraries are generated as below:

```
application/nbapi/c-swig/.libs/_mul_nbapi.so
application/nbapi/c-swig/.libs/_mul_nbapi.so.0
application/nbapi/c-swig/.libs/_mul_nbapi.so.0.0.0
application/nbapi/c-swig/mul_nbapi.py
```

One can either copy these file to his/her private python-application folder or set the python environment variable as follows:

```
$ pwd
openmul
$ topdir=`pwd`
$ export PYTHONPATH=`pwd`/application/nbapi/c-swig/:`pwd`/application/nbapi/c-
swig/.libs/:${topdir}/mul/.libs/:${topdir}/services/loadable/topo_routing/.libs/:`pwd`/common-
libs/util-linux/libuuid/.libs/
```

Now writing a native python program is as easy as follows:
<File: switch_show.py>

```
from mul_nbapi import get_switch_all

switch_list = get_switch_all()

for switch in switch_list:
    print '0x%lx' %switch.switch_id.datapath_id
```

Adding a new REST-API

Define an api

```
[GET/POST/DELETE] /1.0/api/{value}/example
```

Create api handler. Make an api handler under this directory :

```
application/nbapi/py-tornado/app/handler/
```

<application/nbapi/py-tornado/app/handler/to/your/directory/yourhandlerfile.py>

```
import logging
import json
import colander

from app.lib import mul_nbapi as mul
from app.handler.base import BaseHandler

logger = logging.getLogger("YourHandler");
logger.setLevel(logging.DEBUG)

class YourHandler(BaseHandler):

    def get(self, …..):
       ….
       self.finish({"json_param" : "value"})
    def post(self, ……):

    def delete(self, ……):
```

4) Register the api in main REST server handler

<application/nbapi/py-tornado/server.py>

```
….
from app.handler.to.your.directory.yourhandlerfile import YourHandler
….
class App(tornado.web.Application):
    def __init__(self):
        handlers = [
       …….. .
        ]
        handler += (r"/1.0/api/(python_regexp)/api",  YourHandler)
        tornado.web.Application.__init__(self, handlers, debug=True)
```

Using OpenMUL RESTful API in webapp

OpenMUL RESTful API’s returns JSON formatted data, which can be easily parsed in Python.

Example : Get all switch’s datapath-id in a Openflow domain

```
import requests, json, sys
baseurl = “http://<restapi server-ip>:<restapi server-port>”
r = requests.get(baseurl+“/1.0/topology/switch)
switch_list = json.loads(r.text)[‘switches’]
for switch in switch_list:
    print switch[‘dpid’]
```