Leveraging opCharts API with Python Scripts

Objective

Some customers may want to leverage the opCharts API with python scripts. This article will provide one example of how this could be done.

Related pages

opCharts REST API Reference

Example

For this example we'll build a HTML Network Status web page via a CGI script. This page will be comprised of tables; the first table will summarize group status while the subsequent tables will provide node level details for each group.

Define modules and variables

This example will utilize the following modules.

- urllib.request - https://docs.python.org/3/library/urllib.request.html
- http.cookiejar - https://docs.python.org/3/library/http.cookiejar.html
- json - https://docs.python.org/3/library/json.html

We also need to define the following variables.

- Protocol (http or https)
- Server
- Username
- Password

```
#!/usr/bin/python3

import urllib.request, http.cookiejar, json

PROTOCOL = 'https'
SERVER = 'demo.opmantek.com'
USERNAME = 'nmis'

with open('/var/www/opChartsApi.conf', 'r') as confFile:
    USERPASS = confFile.read().strip()
```

Configure URL handling

```
CJ = http.cookiejar.CookieJar()
OPENER = urllib.request.build_opener(urllib.request.HTTPCookieProcessor(CJ))

loginUrl = (PROTOCOL + '://' + SERVER + '/omk/opCharts/login')
loginDict = {'username' : USERNAME , 'password' : USERPASS}
DATA = urllib.parse.urlencode(loginDict).encode("utf-8")
REQUEST = urllib.request.Request(loginUrl, DATA)
OPENER.open(REQUEST)
```

Query opCharts for Node Status

```
nodeData = OPENER.open(PROTOCOL + '://' + SERVER + '/omk/opCharts/nodes.json')
nodeDecoded = nodeData.read().decode()
nodeList = json.loads(nodeDecoded)
```

Build Group Centric Data Structure

```
groupStatus = {}

for n in nodeList:
    if n['group'] not in groupStatus:
        groupStatus[n['group']] = {'up' : 0, 'down' : 0}
        groupStatus[n['group']]['nodes'] = {}
    if n['nodedown'] == 'true':
        groupStatus[n['group']]['down'] += 1
    else:
        groupStatus[n['group']]['up'] += 1
    groupStatus[n['group']]['nodes'][n['name']] = {'name' : n['name'], 'location' : n['location'], 'type' : n
['nodeType'], 'net' : n['netType'], 'role' : n['roleType'], 'status' : n['nodestatus']}
```

Loop Through The Data Building Tables

```
print('Content-type: text/html\n\n')
print('<html>')
print('<style>table, th, td { border: 1px solid black; border-collapse: collapse;}</style>')
print('<head><title>Opmantek Network Status</title></head>')
print('<body><strong>Opmantek Network Status</strong><br><br>')
print('<table><tr><th>Group</th><th>Nodes Up</th><th>Nodes Down</th></tr>')
for k in sorted(groupStatus):
    print('<tr><td>' + k + '</td><td> ' + str(groupStatus[k]['up']) + '</td><td> ' + str(groupStatus[k]
['down']) + '</td></tr>')
print('</table><br><br>')
for k in sorted(groupStatus):
    print('<strong>Group: ' + k + '</strong><br>')
    print('<table><tr><th>Node</th><th>Location</th><th>Type</th><th>Net</th><th>Role</th><th>Status</th></tr>')
    for g in sorted(groupStatus[k]['nodes']):
        print('<tr><td>' + groupStatus[k]['nodes'][g]['name'] + '</td><td> ' + groupStatus[k]['nodes'][g]
['location'] + ' </td><td> ' + groupStatus[k]['nodes'][g]['type'] + ' </td><td> ' + groupStatus
[k]['nodes'][g]['net'] + ' </td><td> ' + groupStatus[k]['nodes'][g]['role'] + ' </td><td> ' +
groupStatus[k]['nodes'][g]['status'] + ' </td>')
    print('</table><br><br>')
print('</body>')
print('</html>')
```