1 Python REST API with ACI (part 1)

I have already produced a general article about automation in the networking world:
https://www.linkedin.com/pulse/automation-myths-riccardo-andreetta/

A **REST**ful API is an architectural style for an Application Program Interface (API) that uses HTTP/HTTPS requests to access and use data from a device. We will approach such a topic by providing a practical and detailed example regarding Cisco ACI technology. The Python script will use some pieces of code that can be found here:
https://github.com/carlmontanari/acipdt

Most of the approaches can be generalized and used with ANY networking or security device, considering the peculiar interaction that needs to be rearranged. Despite of all the multi-vendor automation and orchestration products sold in the world, usually the 'abstraction layer' to become vendor independent is not managed by them, is managed by the customer (as we will probably see in a future post regarding Cisco NSO).
The adoption of REST API to interact with central brained systems and/or network/security devices, has been the first step toward automation, since before that every vendor had its own approach to configure their devices (Command Line Interface for Cisco or Juniper, only a GUI for Checkpoint).

1.1 Data format

Data can be represented in many different ways, there are a few standards used as these methods of interactions got more popular:
- Json
- Xml
- Yaml (used for example by Ansible)

Cisco ACI supports json/xml, the first being preferred by me because it is more 'compact'. You can represent the information in a specific and standard syntax, using **dictionaries** and **arrays** and nesting them in the way you need:

```
{
    "aaaUser": {                    ← 'aaaUser' is the key to access the value
       "attributes": {              ← 'attributes' is the key to access the value
       "name": "Riccardo",          ← 'name' is the key to access the value
       "pwd": ["list first value", "list second value"]
       }
    }
}
```

An array is a simple list of items that can be accessed through an integer index, a dictionary is based on keys associated to their values (in this case of course every key needs to be unique in the same dictionary).

1.2 Logging in

This is not a Python course nor a programming one, but things should be understandable even by people who do not have a strong programming background. As you can see, you log in by using APIC username and password, and using a couple of Python built in libraries. **Requests** manages http/https requests (in a similar way your browser does), sends a 'json' payload to the Apic with the 'aaaUser' that declares that the content is about authenticating to the device. Of course the syntax can vary for other vendors and even for other Cisco products. As it happens on the web, to avoid re-authenticating every time (which would slow down the overall process) **cookies** are passed back from ACI and are stored locally on the created object, to be used on subsequent queries.

```
r = s.post('https://{}/api/mo/aaaLogin.json'.format(self.apic),
 data=json.dumps(payload),
 verify = False)
```

The above syntax performs a ‘post’ request (i.e. send something, the authentication data), the “{}” are replaced by APIC’s ip address through the ‘format’ command. The other parameters passed to this function are the json payload (produced using a library, from the ‘payload’ text previously defined), while ‘verify=False’ avoids checking the server’s certificate.

```
import requests
import json

# Class must be instantiated with APIC IP address, username, and password
# the login method returns the APIC cookies.
class FabLogin(object):
    def __init__(self, apic, user, pword):
        self.apic = apic
        self.user = user
        self.pword = pword

    def login(self):
        # Load login json payload
        payload = '''
        {{
            "aaaUser": {{
                "attributes": {{
                    "name": "{user}",
                    "pwd": "{pword}"
                }}
            }}
        }}
        '''.format(user=self.user, pword=self.pword)
        payload = json.loads(payload, object_pairs_hook = collections.OrderedDict)
        s = requests.Session()
        # Try the request, if exception, exit program w/ error
        try:
            # Verify is disabled as there are issues if it is enabled
            r = s.post('https://{}/api/mo/aaaLogin.json'.format(self.apic),
                       data=json.dumps(payload), verify=False)
            # Capture HTTP status code from the request
            status = r.status_code
            # Capture the APIC cookie for all other future calls
            cookies = r.cookies
            # Log login status/time(?) somewhere
            if status == 400:
                print("Error 400 - Bad Request - ABORT!")
                print("Probably have a bad URL")
                sys.exit()
            if status == 401:
                print("Error 401 - Unauthorized - ABORT!")
                print("Probably have incorrect credentials")
                sys.exit()
            if status == 403:
                print("Error 403 - Forbidden - ABORT!")
                print("Server refuses to handle your request")
                sys.exit()
            if status == 404:
                print("Error 404 - Not Found - ABORT!")
                print("Seems like you're trying to POST to a page that doesn't"
                      " exist.")
                sys.exit()
        except Exception as e:
            print("Something went wrong logging into the APIC - ABORT!")
            # Log exit reason somewhere
            raise LoginFailed(e)
        self.cookies = cookies
        return cookies
```

1.3 Getting data

This is the class to be imported to perform queries toward the APIC (ACI's brain) and get data for further processing. Again, everything is really simple and straightforward: apic is an ip address, cookies are those we have saved from the previous login (see more on that later). This time a 'get' operation is performed, since we are interested on gathering data. The following syntax can vary depending on what we want to obtain from the APIC:

`'https://172.10.2.3/api/node/class/fvBD.json'`

… in this case we are interested in obtaining ALL the objects belonging to a specific class, it can be thought of as an 'snmpwalk' operation: give me all the rows of the table containing all the objects belonging to this specific class (**fvBD** in the example, which stands for Bridge Domains). Beware that this is not a standard: there could be no classes if you're dealing with Checkpoint or another Cisco product. What you can do and how easily and (sometimes) safely you can interact with these devices, **depends on how these devices' API have been designed**.

```
# Class must be instantiated with APIC IP address and cookies
class Query(object):
    def __init__(self, apic, cookies):
        self.apic = apic
        self.cookies = cookies

    def query_class(self, query_class, query_filter=''):
        s = requests.Session()
        try:
            r = s.get('https://{}/api/node/class/{}.json{}'.format(self.apic,
                      query_class, query_filter), cookies=self.cookies,
                      verify=False)
            status = r.status_code
            payload = json.loads(r.text)
        except Exception as e:
            print("Failed to query Class. Exception: {}".format(e))
            status = 666
        return (status, payload)
```

1.4 Processing data and writing to excel

In the following code we are merging together what we have previously explained, to obtain what we are interested in. We want to retrieve a certain number of ACI classes, and write them in an excel file. Every **class** could be thought of as a table, where rows are the items belonging to that class, and columns are all the 'attributes' or parameters defining each object.
Just as an example:
fvBD

dn	uni/tn-TEST/BD-TEST_BD
OptimizeWanBandwidth	no
annotation	
arpFlood	yes
bcastP	225.1.83.192
childAction	
configIssues	
descr	VLAN 1231
epClear	no
epMoveDetectMode	
extMngdBy	
hostBasedRouting	no
intersiteBumTrafficAllow	no
…	

Parameters haven't been ALL listed here, imagine hundreds of tables like this, where each table represents a single object belonging to the class in which we are interested. We will save the output in an excel file, thus we need a few libraries to manage excel files:

```
# This python script retrives data from ACI and saves them into an excel
# for an easier processing and filtering capabilities for everyone.
from openpyxl import Workbook, load_workbook
from openpyxl.utils.cell import get_column_letter
```

'time' is a Python library to get data, 'os.path' to check (but obviously not limited to) for files existence.

```
import time
import os.path
```

Here comes the import of the previously explained classes and APIC credentials:

```
from Aci_Cal_Toolkit import FabLogin, Query
from credentials import apic_ip, apic_pwd, apic_user
```

We open the excel file, a new or already existing one:

```
day_time = time.strftime("a%Y m%m g%d")
if os.path.exists('C:/<YOUR_DIRECTORY>/ACI_class_' + day_time + '.xlsx'):
    target = load_workbook('C:/<YOUR_DIRECTORY>/ACI_class_' + day_time + ".xlsx")
else:
    target = Workbook()
```

Here we connect to the apic, saving cookies and we create a 'Query' object with these cookies.

```
apic = FabLogin (apic_ip, apic_user, apic_pwd)
cookies = apic.login()
print('Logged into apic ...')
req = Query(apic_ip, cookies)
```

We first define a list of the classes we want to query:

```
fvClass = ['mgmtRsInBStNode', 'fvBD']
```

For every class in the above array:

```
for aci_class in fvClass:
```

... we create a sheet with the class name and query for the class itself:

```
    sheet = target.create_sheet(aci_class, 0)
    print('Retrieving class '+aci_class+' ...')
    [status, payload] = req.query_class(aci_class)
    if status != 200 or not len(payload['imdata']):
        continue
    json_data = payload['imdata']
```

Data is organized into a dictionary of dictionaries, you can potentially print it out to better learn how to deal with it. We go through the 'attributes' values (in the above example the 'dn', 'OptimizeWanBandwidth', 'annotation', 'arpFlood', …) and write them in the first row of the sheet (they are the columns values). To write a value into the excel, it's simply "sheet['B1'] = value". The useful function get_column_letter return the right letter value based on an integer number (1→ 'A', 2→'B').

```
# returned payload data, store into json_data variable
{
    "fvBD": {
       "attributes": {
       "dn": "uni/tn-TEST/BD-TEST_BD",
       "OptimizeWanBandwidth": "no",
       "annotation": "",
       "arpFlood": "yes",
       "bcastP": "225.1.83.192",
       …
       }
    },
    "class_2": {
       "attributes": {
       "<attr_1>": "<value_1>",
       "<attr_2>": "<value_2>",
       "<attr_3>": "<value_3>",
       "<attr_4>": "<value_4>",
       "<attr_5>": "<value_5>",
       }
    },
    …
}
```

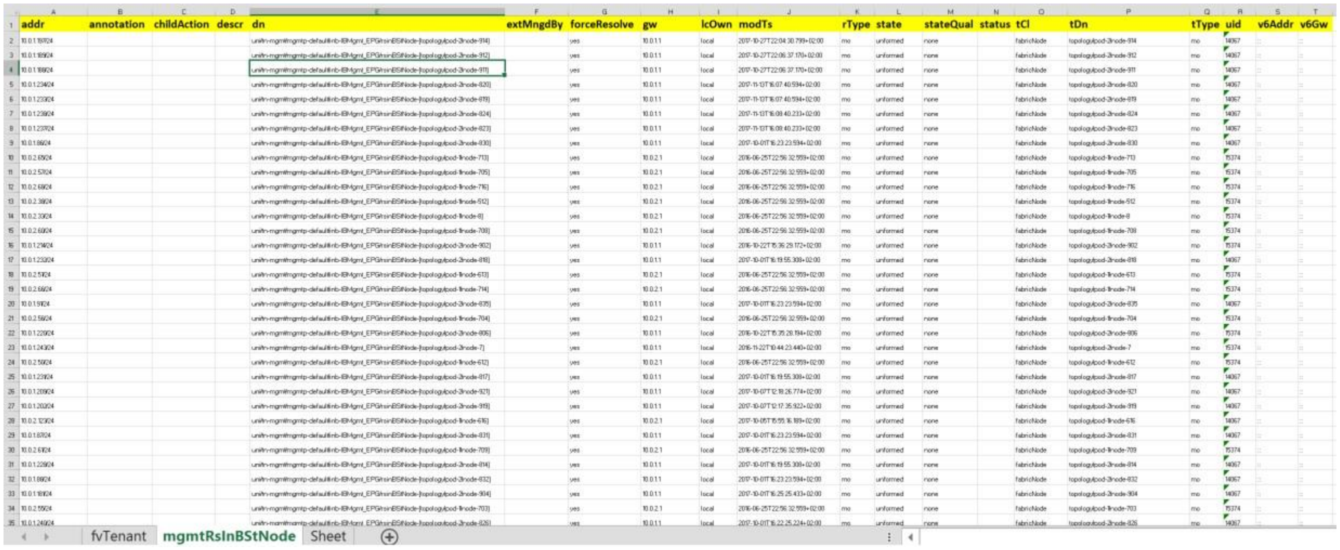
```
    cols = []
    for attr in sorted(json_data[0][aci_class]['attributes']):
        cols.append(attr)
        sheet[get_column_letter(len(cols))+'1'] = attr
```

Now it's time to cycle over the retrieved objects, and store them into the excel file:

```
    row_counter = 2
    for obj in json_data:
        for i in range(len(cols)):
           sheet[get_column_letter(i+1)+str(row_counter)] = obj[aci_class]['attributes'][cols[i]]
        row_counter += 1

target.save('C:/<YOUR_DIRECTORY>/ACI_class_'+day_time+".xlsx")
```

That’s it. How powerful is it ? Such few lines for such a powerful output: now you can compare all the parameters configured for the objects belonging to the same class, using the embedded powerful filtering tools of excel.



addr	annotation	childAction	descr	dn	extMngdBy	forceResolve	gw	lcOwn	modTs	rType	state	stateQual	status	tCl	tDn	tType	uid	v6Addr	v6Gw
10.0.1.197/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-914]		yes	10.0.1.1	local	2017-10-27T22:04:30.799+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-914	mo	14067	::	::
10.0.1.189/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-912]		yes	10.0.1.1	local	2017-10-27T22:06:37.170+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-912	mo	14067	::	::
10.0.1.188/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-911]		yes	10.0.1.1	local	2017-10-27T22:06:37.170+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-911	mo	14067	::	::
10.0.1.234/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-820]		yes	10.0.1.1	local	2017-11-13T16:07:40.594+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-820	mo	14067	::	::
10.0.1.233/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-819]		yes	10.0.1.1	local	2017-11-13T16:07:40.594+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-819	mo	14067	::	::
10.0.1.238/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-824]		yes	10.0.1.1	local	2017-11-13T16:08:40.233+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-824	mo	14067	::	::
10.0.1.237/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-823]		yes	10.0.1.1	local	2017-11-13T16:08:40.233+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-823	mo	14067	::	::
10.0.1.86/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-830]		yes	10.0.1.1	local	2017-10-01T16:23:23.594+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-830	mo	14067	::	::
10.0.2.65/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-713]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-713	mo	15374	::	::
10.0.2.57/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-705]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-705	mo	15374	::	::
10.0.2.68/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-716]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-716	mo	15374	::	::
10.0.2.38/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-512]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-512	mo	15374	::	::
10.0.2.33/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-8]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-8	mo	15374	::	::
10.0.2.60/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-708]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-708	mo	15374	::	::
10.0.1.214/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-902]		yes	10.0.1.1	local	2016-10-22T15:36:29.172+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-902	mo	15374	::	::
10.0.1.232/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-818]		yes	10.0.1.1	local	2017-10-01T16:19:55.308+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-818	mo	14067	::	::
10.0.2.57/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-613]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-613	mo	15374	::	::
10.0.2.66/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-714]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-714	mo	15374	::	::
10.0.1.97/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-835]		yes	10.0.1.1	local	2017-10-01T16:23:23.594+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-835	mo	14067	::	::
10.0.2.56/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-704]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-704	mo	15374	::	::
10.0.1.220/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-806]		yes	10.0.1.1	local	2016-10-22T15:35:28.194+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-806	mo	15374	::	::
10.0.1.243/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-7]		yes	10.0.1.1	local	2016-11-22T10:44:23.440+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-7	mo	15374	::	::
10.0.2.50/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-612]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-612	mo	15374	::	::
10.0.1.231/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-817]		yes	10.0.1.1	local	2017-10-01T16:19:55.308+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-817	mo	14067	::	::
10.0.1.209/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-921]		yes	10.0.1.1	local	2017-10-07T12:18:26.774+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-921	mo	14067	::	::
10.0.1.203/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-919]		yes	10.0.1.1	local	2017-10-07T12:17:35.922+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-919	mo	14067	::	::
10.0.2.123/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-616]		yes	10.0.2.1	local	2017-10-05T15:55:16.189+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-616	mo	14067	::	::
10.0.1.87/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-831]		yes	10.0.1.1	local	2017-10-01T16:23:23.594+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-831	mo	14067	::	::
10.0.2.67/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-709]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-709	mo	15374	::	::
10.0.1.228/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-814]		yes	10.0.1.1	local	2017-10-01T16:19:55.308+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-814	mo	14067	::	::
10.0.1.88/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-832]		yes	10.0.1.1	local	2017-10-01T16:23:23.594+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-832	mo	14067	::	::
10.0.1.181/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-904]		yes	10.0.1.1	local	2017-10-01T16:25:25.433+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-904	mo	14067	::	::
10.0.2.55/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-1/node-703]		yes	10.0.2.1	local	2016-06-25T22:56:32.559+02:00	mo	unformed	none		fabricNode	topology/pod-1/node-703	mo	15374	::	::
10.0.1.240/24				uni/tn-mgmt/mgmtp-default/inb-IBMgmt_EPG/rsInBStNode-[topology/pod-2/node-826]		yes	10.0.1.1	local	2017-10-01T16:22:25.224+02:00	mo	unformed	none		fabricNode	topology/pod-2/node-826	mo	14067	::	::

Riccardo Andreetta