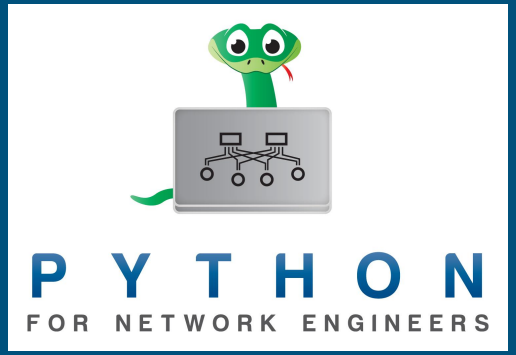
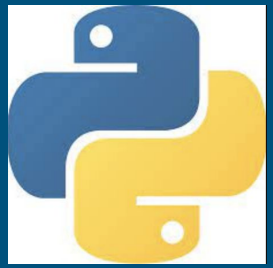




Training Session
November 2022 (Day4)

Day4 Presentation:

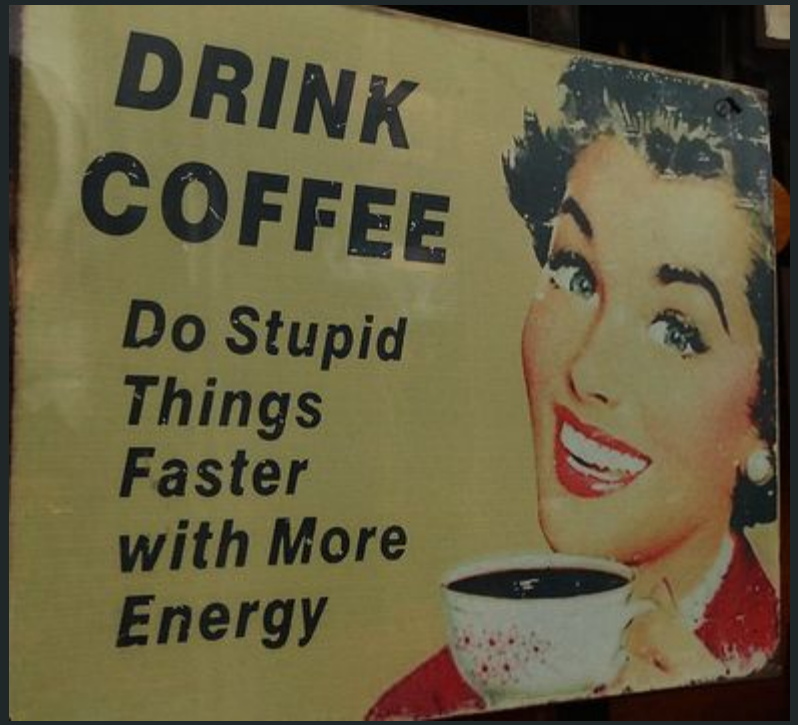
https://github.com/twin-bridges/pynet-ons-oct22/blob/main/python-training-day4.pdf

General:

Training Day
Nov 15th, Day4 (Tue) / 9AM - 5:00PM Central

Focused/Minimize Distractions

The exercises are important.

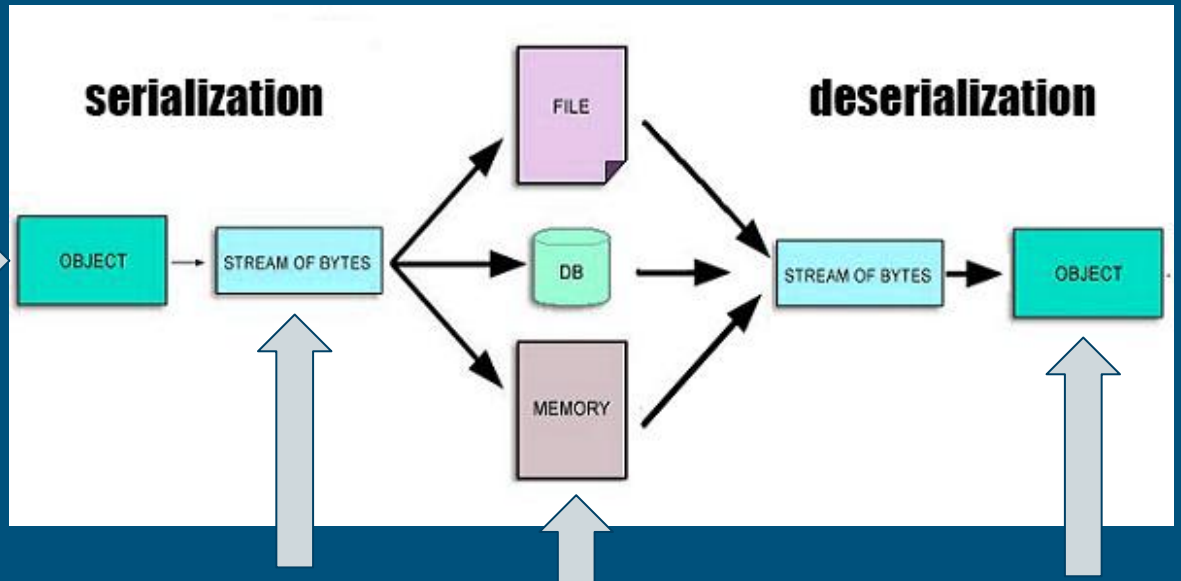


Flickr: Ben Sutherland

Data Serialization

Data Structures in Your Program

Why do we need data serialization?



Convert to a stream of bytes (serialize)

Go across the network, saved to a file, etc.

Read by Another Program / Computer

Data Serialization

Characteristics of JSON

Dictionary

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {"type": "home","number": "212 555-1234"},
    {"type": "office","number": "646 555-4567"}
  ],
  "children": ["Catherine", "Thomas", "Trevor"],
  "spouse": null
}
```

String

Boolean

Number

List

Reference Material in:

{{ github_repo }}/json_yaml

JSON Exercise1

Using Python and the "json" library, read in the file "show_apgroup.json" as structured data. This file is located in "day4/json_exercises".

Which Python data-structure is returned (as the outermost data structure)?

From this returned data-structure return a simple list of the AP-Group Names. Your output should look as follows:

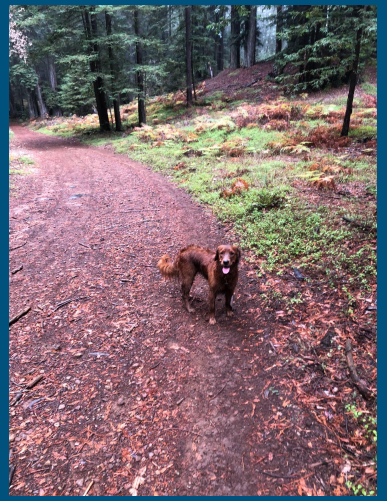
output:

['default', 'NoAuthApGroup']

GitHub: {{ repo }}/day4/json_exercises/exercise1.txt

GitHub: {{ repo }}/day4/json_exercises/exercise2.txt

JSON Exercise2



Embed the following in your Python program as a string:

```
{
   "VLAN CONFIGURATION": [
      {
         "AAA Profile": "N/A",
         "Description": "Default",
         "Option-82": "Disabled",
         "Ports": "GE0/0/1 Pc0-7",
         "VLAN": "1"
      },
      {
         "AAA Profile": "N/A",
         "Description": "VLAN0235",
         "Option-82": "Disabled",
         "Ports": "GE0/0/0",
         "VLAN": "235"
      }
   ],
   "_meta": [ "VLAN", "Description", "Ports", "AAA Profile", "Option-82"]
}
```

Use json.loads() to convert this JSON string over to Python data structures.

Print out the type of your converted data structure. It should be a dictionary after the conversion.

Remember you can use triple-quotes for a multiline string in Python.

REST API

https://198.41.81.149:4343/api

https://10.10.10.10:4343/api/

aruba
a Hewlett Packard Enterprise company

- CONTAINERS
  - -- AP Provisioning
  - -- Authentication
  - -- Controller
  - -- Crypto
  - -- External Services
  - -- Hierarchy
  - -- Interfaces
  - -- L2/L3 Protocols
  - -- Load Balancing & Redun

ArubaOS JSON API Specification

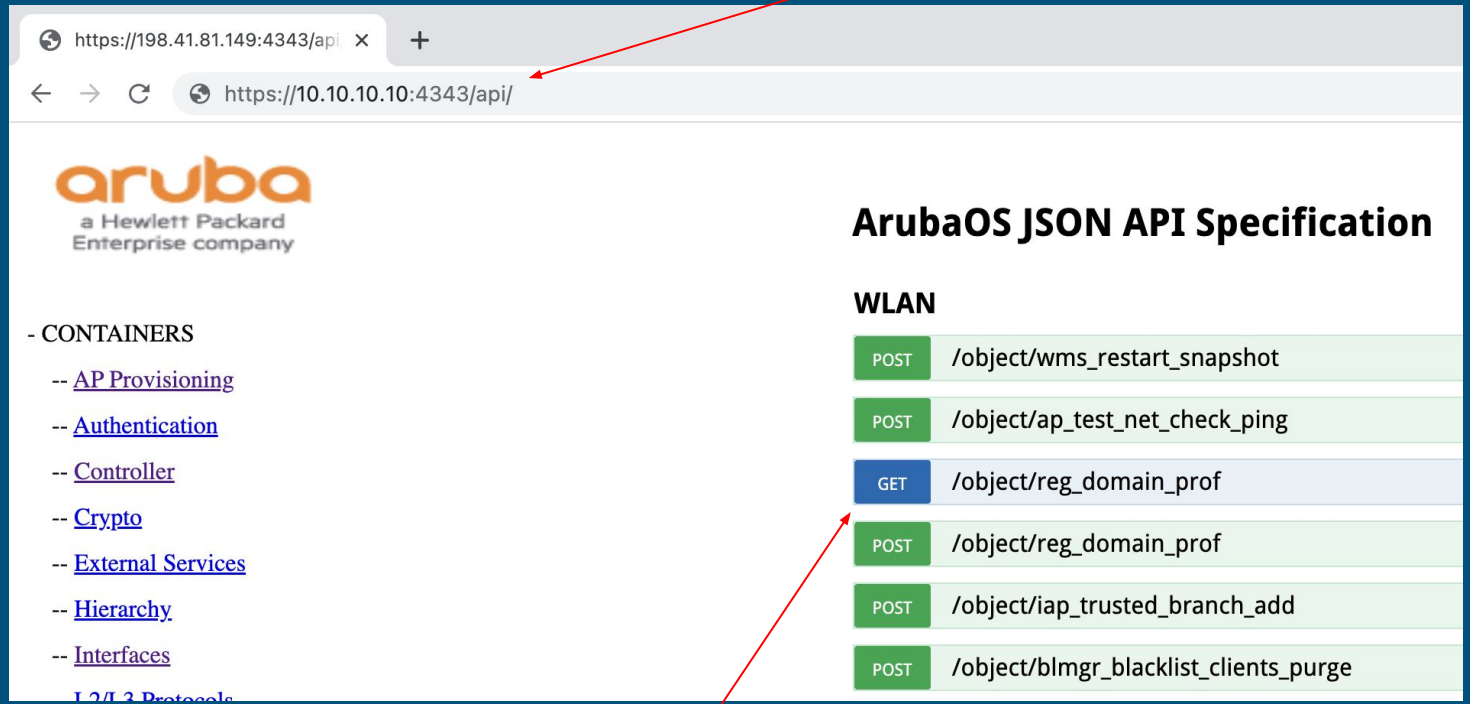
WLAN

Show/Hide | List Operations | Expand Operations

POST	/object/wms_restart_snapshot	Wms Restart Snapshot
POST	/object/ap_test_net_check_ping	AP Test Net Check Ping
GET	/object/reg_domain_prof	Reg Domain Profile
POST	/object/reg_domain_prof	Reg Domain Profile
POST	/object/iap_trusted_branch_add	Iap Trusted Branch Add
POST	/object/blmgr_blacklist_clients_purge	Blmgr Blacklist Clients Purge
POST	/object/stm_blacklist_client_add	Stm Blacklist Client Add

REST API - Characteristics

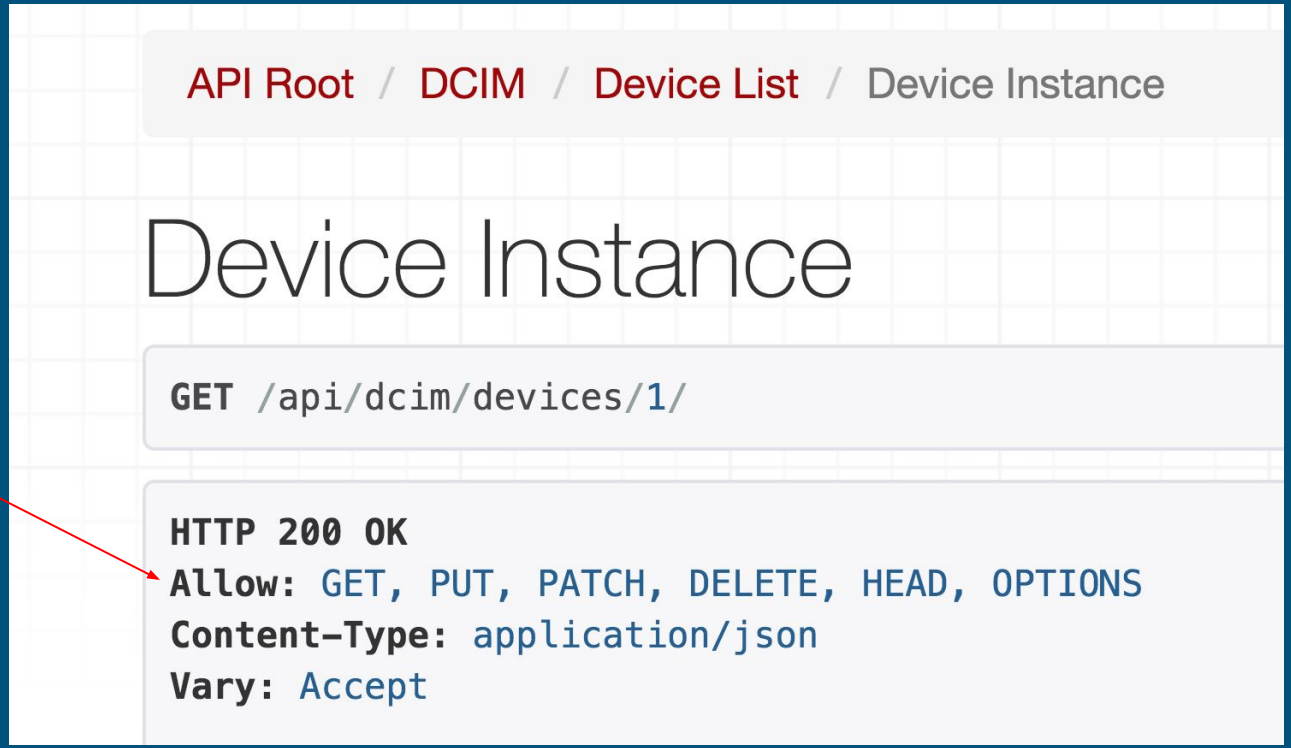
URL - the object I am accessing.



HTTP Methods

REST API - Other HTTP Methods

Available HTTP Methods



REST API - CRUD

- Create - HTTP Post
- Read - HTTP Get
- Replace - HTTP Put (Post)
- Update - HTTP Patch (Post)
- Delete - HTTP Delete (Post)

HTTP Methods Ref: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

But, Before Anything Happens We Must Authenticate


```python
from getpass import getpass
import requests

PASSWORD = getpass("Enter Aruba Controller password: ")
HTTP_HEADERS = {"Content-Type": "application/json"}

# URL
host = "aruba.lasthop.io"
api_port = "4343"
login_url = f"https://{host}:{api_port}/v1/api/login"

# Creds
username = "kbyers"
creds = f"username={username}&password={PASSWORD}"

session = requests.Session()
response = session.post(login_url, data=creds, headers=HTTP_HEADERS, verify=False)
```

HTTP Headers

URL

Credentials

But, Before Anything Happens We Must Authenticate

Use Python requests

Session will automatically store cookies

```
session = requests.Session()
response = session.post(login_url, data=creds, headers=HTTP_HEADERS, verify=False)
```

Perform an HTTP Post

Login URL

Pass in credentials

Failure!

```
(Pdbr) response
<Response [401]>
(Pdbr) response.status_code
401
```

Authentication Failed

```
(Pdbr) response.json()
{'_global_result': {'status': '1', 'status_str': 'Unauthorized request, authentication failed'}}
```

Success!


```
(Pdbr) response
<Response [200]>
(Pdbr) response.status_code
200
```

Authentication Worked

```
(Pdbr) response
<Response [200]>
(Pdbr) response.json()
{'_global_result': {'status': '0', 'status_str': "You've logged in successfully.", 'UIDARUBA':
'MTI4ZDhhZDMtZTE1NC00M2FmLTgzNGEtNWRi', 'X-CSRF-Token': 'NDM1N2M5MWEtZGZjNy00NWNhLTg5MDUtMzM1'}}
```

And one additional thing we need to (for newer Aruba Controllers)

"Starting from ArubaOS 8.7.0.0, UIDARUBA is deprecated. The X-CSRF-Token will be used in the header. The UIDARUBA is still available for backward compatibility with older ArubaOS firmware versions."

```
if response.status_code == 200:
    auth_response = response.json().get("_global_result")
    if auth_response.get("X-CSRF-Token"):
        session.headers["X-CSRF-Token"] = auth_response["X-CSRF-Token"]
```

The newer Aruba controllers require that you set this X-CSRF-Token in the HTTP_HEADERS

Exercise

Access the Aruba Controller API Documentation at:

https://aruba.lasthop.io:4343/api/

Username/Password: <Will Provide>

Explore a bit on one of the "container" categories.

Look into some particular "GET" operation in the documentation. Test the "try it out" option (do GET operations only, no POST operations).

Exercise

Get Python + API Authentication Working to Aruba Lab Controller.

Controller: aruba.lasthop.io

Username/Password: <Will Provide>

Use reference code here:

https://github.com/twin-bridges/pynet-ons-oct22/blob/main/day4/aruba_auth/aruba_auth_simple.py

Try to understand what the reference code is doing. Ask questions here for steps that don't make sense.

We have authenticated. Can we retrieve data!

```
# Test a GET operation
relative_url = "object/int_vlan"
base_url = f"https://{host}:{api_port}/v1/configuration/"

full_url = f"{base_url}{relative_url}"
print(full_url)

response = session.get(full_url, verify=SSL_VERIFY)
rich.print(response.json())
```

The URL we are querying

Use "requests" session which contains "SESSION" cookie (i.e. UIDARUBA value)

X-CSRF-Token is contained in HTTP Headers (which are bound to "session")

It's Alive!

```
{
    '_data': {
        'int_vlan': [
            {
                'id': 235,
                'int_vlan_ip': {'ipaddr': '10.5.235.12', 'ipparams': 'ipaddrmask', 'ipmask': '255.255.255.0'},
                'int_vlan_routing': {'_present': True, '_flags': {'default': True}},
                'int_vlan_ndra_hlimit': {'_flags': {'default': True}, 'value': 64},
                'int_vlan_ndra_interval': {'_flags': {'default': True}, 'value': 600},
                'int_vlan_ndra_ltime': {'_flags': {'default': True}, 'value': 1800},
                'int_vlan_ndra_mtu': {'_flags': {'default': True}, 'value': 1500},
                'int_vlan_nd_reachtime': {'_flags': {'default': True}, 'value': 0},
                'int_vlan_nd_rtrans_time': {'_flags': {'default': True}, 'value': 0},
                'int_vlan_mtu': {'_flags': {'default': True}, 'value': 1500},
                'int_vlan_suppress_arp': {'_present': True, '_flags': {'default': True}},
                'int_vlan_ip_ospf_cost': {'_flags': {'default': True}, 'value': 1},
                'int_vlan_ip_ospf_dead_interval': {'_flags': {'default': True}, 'value': 40},
                'int_vlan_ip_ospf_hello_interval': {'_flags': {'default': True}, 'value': 10},
                'int_vlan_ip_ospf_prior': {'_flags': {'default': True}, 'value': 1},
                'int_vlan_ip_ospf_retransmit_int': {'_flags': {'default': True}, 'value': 5},
                'int_vlan_ip_ospf_transmit_delay': {'_flags': {'default': True}, 'value': 1}
            }
        ]
    }
}
```

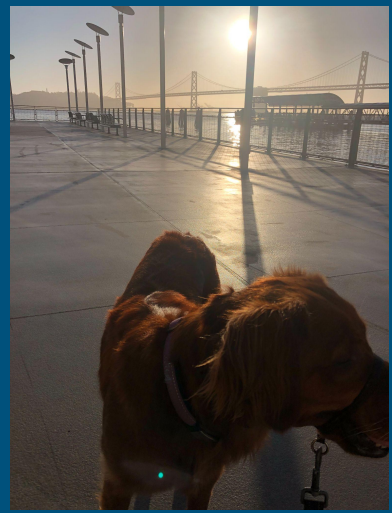
Another Word About Auth and Controllers Using AOS <= 8.6.X.X Software

Require UIDARUBA=<session-id> to be part of the URL query string

UIDARUBA=AZhkZjM3ZmQtZDUwYi00ZjhmLWFhZmEtODE0

For example:

https://aruba.lasthop.io:4343/v1/configuration/object/int_vlan?UIDARUBA=AZhkZjM3ZmQtZDUwYi00ZjhmLWFhZmEtODE0



Exercise: Make the Aruba Authentication Code More Easily Reusable.

Convert Authentication to a Function.

Function should take a Requests' Session object, a hostname, and an optional API port (default the API port to 4343).

The function should set the session.headers["X-CSRF-Token"] to the corresponding "X-CSRF-Token" returned from the controller (if it exists).

Additionally, the function should return the UIDARUBA token (if it exists). Otherwise the function should return None.

GitHub: {{ repo }}/day4/aruba_auth_exercises/exercise1.txt

Now that we have a reusable auth function, make sure it is on $PYTHONPATH

```python
def auth(session, host, api_port=4343):
    http_headers = {"Content-Type": "application/json"}

    # Creds
    USERNAME = "kbyers"
    PASSWORD = getpass("Enter Aruba Controller password: ")
    creds = f"username={USERNAME}&password={PASSWORD}"

    # Login URL
    login_url = f"https://{host}:{api_port}/v1/api/login"

    # Authenticate
    response = session.post(
        login_url, data=creds, headers=http_headers, verify=SSL_VERIFY
    )

    # Verify Authentication Worked
    if response.status_code == 200:
        auth_response = response.json().get("_global_result")
        if auth_response.get("X-CSRF-Token"):
            # Bind headers to requests' session object
            session.headers["X-CSRF-Token"] = auth_response["X-CSRF-Token"]
```

At this point we can just Auth by calling our function

```
import aruba_auth
import requests

host = "aruba.lasthop.io"
api_port = "4343"

session = requests.Session()
session.headers["Accept"] = "application/json"
uid_aruba = aruba_auth.auth(session, host, api_port=api_port)

print(uid_aruba)
```

Whew, we have a working and reusable authentication.

Now, let's start trying to use the API.

Constructing the URL in the request (what to query)?

response = session.get(full_url, verify=SSL_VERIFY)

Which action:
HTTP Get

What we are querying?

https://aruba.lasthop.io:4343/v1/configuration/object/int_vlan

If we want different information, we change the URL

https://aruba.lasthop.io:4343/v1/configuration/object/int_vlan

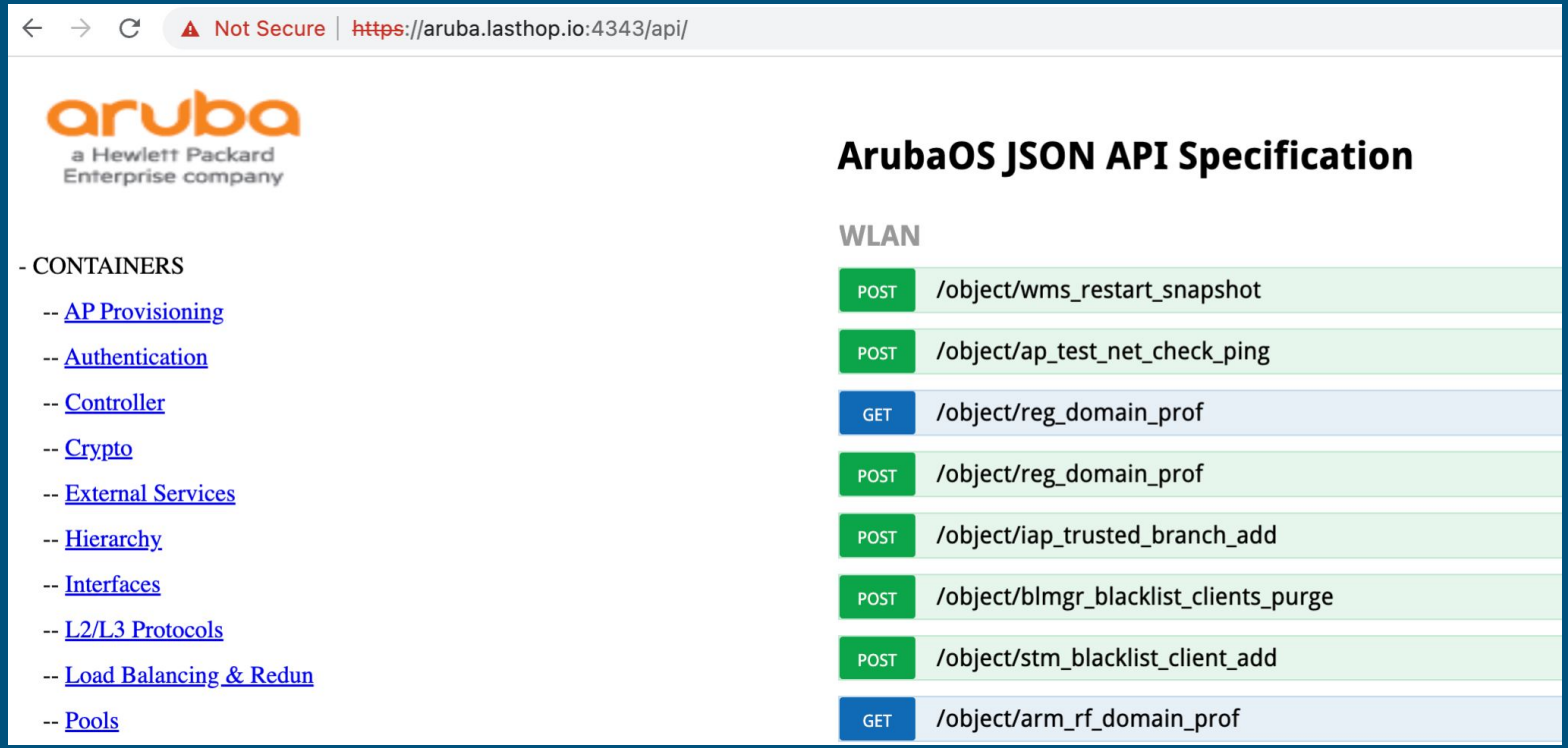
Different URL

https://aruba.lasthop.io:4343/v1/configuration/object/ap_group

Remember to add UIDARUBA=<session-id> to the URL on AOS <= 8.6.X.X

```
response = session.get(full_url, verify=SSL_VERIFY)
```

How to find the URLs: API Documentation



Testing It Out.

```
uid_aruba = auth(session, host=host, api_port=api_port)
uid_aruba_qs = f"UIDARUBA={uid_aruba}"

# Test a GET operation
base_url = f"https://{host}:{api_port}/v1/configuration/"
relative_url = "object/ap_group"

# Adding the UID_ARUBA query string for compatibility with 8.6.X.X
full_url = f"{base_url}{relative_url}?{uid_aruba_qs}"

response = session.get(full_url, verify=False)
rich.print(response.json())
```

Exercise1:

Using the existing "auth" function, connect to the Aruba Controller.

Construct a query to retrieve: "object/hostname".

Use rich.print to print the result to the screen.

Use the code in "day4/aruba_get/aruba_get.py" as a model for this.

GitHub: {{ repo }}/day4/aruba_get/exercise1.txt

Executing a Show Command.

This returns JSON.

```
# Test a GET operation
base_url = f"https://{host}:{api_port}/v1/configuration/"

# Adding the UID_ARUBA query string for compatibility with 8.6.X.X
command = "show+ap+database-summary"
relative_url = f"showcommand?command={command}&{uid_aruba_qs}"
full_url = f"{base_url}{relative_url}"

response = session.get(full_url, verify=False)
rich.print(response.json())
```

Exercise2:

Using the existing "auth" function, connect to the Aruba Controller.

Construct a "showcommand" to retrieve: "show vlan".

Use rich.print to print the result to the screen.

Use the code in "day4/aruba_get/aruba_showcommand.py" as a model for this.

GitHub: {{ repo }}/day4/aruba_get/exercise2.txt

Adding in config_path

```
config_path = "?config_path=/mm/mynode"
config_path = "?config_path=/md"
config_path = "?config_path=/md/40Lab"
config_path = "?config_path=/md/40Lab/VH/20:4c:03:39:5a:fc"
```

```
# Test a GET operation
base_url = f"https://{host}:{api_port}/v1/configuration/"
relative_url = "object/ap_group"

# Default config_path
config_path = "?config_path=/mm/mynode"
# Adding the UID_ARUBA query string for compatibility with 8.6.X.X
url_and_qs = f"{relative_url}{config_path}&{uid_aruba_qs}"
full_url = f"{base_url}{url_and_qs}"

response = session.get(full_url, verify=False)
```

Adding in config_path

Exercise1:

Using the existing "auth" function, connect to the Aruba Controller.

Construct a config_path for each of the two Aruba controllers (all the way down to each controller's MAC address). The two controller MAC addresses are: "20:4c:03:39:5a:fc", "20:4c:03:58:70:72"

Using this config_path, query the controller for "object/hostname" and use rich.print() to print these results to the screen.

Use the code in "day4/config_path/cfg_path_example.py" for reference code for config_path construction.

GitHub: {{ repo }}/day4/config_path/exercise1.txt

Constructing a More General Aruba Get Function

```python
def get_request(session, host, api_port=4343, relative_url="", config_path="", uid_aruba=""):

    base_url = f"https://{host}:{api_port}/v1/configuration/"

    if config_path:
        query_string = f"?config_path={config_path}"
        if uid_aruba:
            query_string += f"&UIDARUBA={uid_aruba}"
    else:
        if uid_aruba:
            query_string = f"?UIDARUBA={uid_aruba}"
        else:
            query_string = ""

    full_url = f"{base_url}{relative_url}{query_string}"
    return session.get(full_url, verify=False)
```

Exercise2:

Repeat "config_path" exercise1 except use the "get_request" function located in the aruba_auth module.

GitHub: {{ repo }}/day4/config_path/exercise2.txt

Where are we at:

1. We are using Python to connect to the Aruba Controller.
2. We can authenticate and have created an auth function to make this authentication process easier going forward.
3. We can perform information retrieval (GET operations) using either "object" or "showcommand" patterns.
4. We have learned about using config_path and how it can be used to qualify the "object" information retrieval.
5. We have created a get_request function to simplify information retrieval for the API GET "object" case.

Review Exercise1

Using both the "auth" function function, connect to the Aruba Controller.

Retrieve "object/ssid_prof" using a config_path of "/md/40Lab/VH" and using the "get_request" function.

From the returned data structure extract the "profile-names" and put them into a simple list.

Use rich.print to print this profile_name_list to the screen.

Note, remember your process for dealing with complex data structures. You will need to step multiple layers into this data structure to extract the field that you require.

GitHub: {{ repo }}/day4/review_exercises/exercise1.txt

GitHub: {{ repo }}/day4/review_exercises/exercise2.txt

Review Exercise2

Using {{ repo }}/day4/aruba_get/aruba_showcommand.py as a model, connect to the Aruba Controller and execute the following "showcommand":

`show configuration node-hierarchy`

Parse the data-structure that is returned and extract both the "Config Node" and the "Type" fields.

For any "Config Node" with a type of "Device", print both the "Config Node" path and the "Type". The output should look similar to the following:

Config Node: /md/40Lab/VH/20:4c:03:39:5a:fc --> Type: Device
Config Node: /md/40Lab/VH/20:4c:03:58:70:72 --> Type: Device

Review Exercise3 (Optional)

Construct a reusable function that allows you to simplify the execution of "showcommand" via the Aruba API.

GitHub: {{ repo }}/day4/review_exercises/exercise3.txt