Aruba Network Automation Essentials

Igor Chinchak
Aruba Edge Expert & Certified Instructor
Computer Data Inc
Igor Chinchak, Aruba Expert and Certified Instructor

– Aruba-Certified Edge Expert #104
– Cisco Certified Internetwork Expert Wireless #61135
– Experienced in automation with Ansible and Python

– https://www.credly.com/users/igor-chinchak
Aruba Network Automation Essentials

What to expect

PART 1: Jan 24, 2022 | 9-11AM PST

• Part 1 Introduces network automation, Python, Ansible

PART 2: Jan 31, 2022 | 9-11AM PST

• Part 2 Explains how to use network automation with Aruba products
Overview / Summary

- AOS-CX Automation
- AOS8 Automation
- Aruba Central Automation
- Aruba ClearPass Automation
- Aruba Airwave Automation
- Aruba Instant AP Automation
Aruba AOS-CX SDK Overview

PyaoScx modules that can be called upon to access the REST API and configure various features

Simplifies API usage
- Support both v1 and v10.0.4 APIs
- All functions in all modules fully documented

Open-Source and reliable through direct support from Aruba
AOSCX REST API Python SDK - Pyaoscx

Pyaoscx contains Python modules that can be called upon to access the REST API and configure various features on the switches.
DEMO 1
AOS-CX automation examples
AOS8
Automation
# AOS8 API Endpoints

<table>
<thead>
<tr>
<th>Login</th>
<th>• Login is a special endpoint that is required to access any configuration element using GET or POST method with any API endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logout</td>
<td>• To logout from an existing session, Logout API endpoint is used</td>
</tr>
<tr>
<td>Configuration APIs</td>
<td>• These APIs are used to configure the system using the POST method or fetch the configured data from the system data-stores using the GET method</td>
</tr>
<tr>
<td>Monitoring APIs</td>
<td>• Apart from the various endpoints available for GET, monitoring can also be achieved using the CLI &quot;show commands&quot; over APIs</td>
</tr>
</tbody>
</table>
| Show Command API | • It mimics the CLI show commands over API  
• Only GET method is supported for this type of API |
| Write Memory | • Write Memory API is used to deploy the pending changes made to the Mobility Conductor through POST API calls. |
AOS8 Automation Overview
Understanding Sequential Configuration

Create an SSID ➔ Server Group ➔ AAA profile ➔ SSID profile

Sequence Matters!

AP Group ➔ Virtual AP Profile
Aruba AOS 8 API Example

```python
import requests,urllib3
urllib3.disable_warnings()

#login API
login_url = "https://192.168.1.72:4343/v1/api/login"
querystring = {"username":"admin","password":"admin1"}
headers = {"Accept": "application/json"}
session = requests.session()
login= session.post(login_url, data=querystring, verify=False)
print(login.json())
UIDARUBA = login.json()['_global_result']['UIDARUBA']
print(UIDARUBA)
print("status code: " + login.status_code)

#configuration API - get vlan info
url1 = "https://192.168.1.72:4343/v1/configuration/object/vlan_id"
full_url=url1+'&UIDARUBA='+UIDARUBA
print(full_url)
querystring = {"config_path": "/mm"}
vlans = session.get(full_url, headers=headers, params=querystring)
print(vlans.json())

#logout API
logout_url="https://192.168.1.72:4343/v1/api/logout"
logout= session.post(logout_url, verify=False)
print(logout.status_code)
```

#OUTPUT

```
#login API
{'_global_result': {'status': '0', 'status_str': "You've logged in successfully.", 'UIDARUBA': 'MGNmNGYyM2EtYjlhMy00YjgzLWIyMTgtMDJk', 'X-CSRF-Token': 'OWQxYzViMTUtMjYwYi00MDQ0LTkwZWUtZTY3'}}
status code: 200

#configuration API - get vlan info
{'_data': {'vlan_id': [{'id': 1, '_flags': {'readonly': True, 'system': True}}]}}

#logout API
200
```
DEMO 2
AOS8 automation examples
Aruba Central
Automation
Overview – Aruba Central API Call

Application HTTP Client → Central API GW HTTP Server

- **HTTPS**
- **Enable API GW**
- **Create an App**
- **Validate Access Token**
- **OAuth 2.0 Implementation to get Access Token**
- **Get Request + Access Token**
- **200 OK Response Code + Data in JSON format**
- **Post Request + Data in JSON format + Access Token**
- **Validate Access Token**
- **200 OK Response Code + Data in JSON format**
Aruba Central SDK Overview

Aruba Central REST API

• For configuration, on-demand polling, monitoring data via HTTP Requests.
• GitHub - pycentral
• PyPi - pycentral

Webhook API

• Integrate external application and implement actions based on real-time alerts.
• GitHub - webhooks/python-webclient

Streaming API

• Subscribe to select topics, get statistics and state changes over Secure WebSocket (WSS).
• GitHub - streaming-api-client
Pycentral enables the user to interact with the Aruba Central API and provides a framework to allow the user to interact programmatically.
DEMO 3

Aruba Central automation examples
Aruba ClearPass
Automation
Aruba ClearPass API Overview

Enable API: Guest > Administration > API Services > API Clients
Aruba ClearPass API Overview

Create a service: CPPM > Configuration > Service Templates & Wizards > OAuth2 API User Access
DEMO 4

Aruba ClearPass automation examples
Aruba Airwave

Automation
Aruba Airwave API Overview

Overview

AirWave APIs are split into Query APIs, Search and Report APIs, and APIs to Batch Execute AP Commands.

Query APIs

- **AP Status** - Provides a high-level summary of AirWave's current status.
- **Folder List** - Provides a full (or optionally partial) list of folders on the AirWave server, and a high-level summary of the folder's current status.
- **Alert List** - Provides a list of all AirWave alerts.
- **AP List** - Provides a full (or optionally partial) list of managed Access Points on AirWave. A partial list of APs can be obtained by providing one or more AP IDs, as described below.
- **AP SSID List** - Provides a full (or optionally partial) list of SSIDs of managed Access Points on an AMP. A partial list of SSIDs of APs can be obtained by providing one or more AP IDs, as described below.
- **AP Detail** - Provides detailed information about managed APs, including associated clients and neighboring rogue access points.
- **AP Log** - Provides a specified number of the most recent log messages for APs. This API requires one or more AP IDs to be supplied.
- **Rogue Detail** - Provides detailed information about rogue access points, including a history of individual discovery events. This API requires one or more rogue_ap_ids.
- **Client Detail** - Provides detailed information about wireless clients, including a history of associations. This API requires one or more client MAC addresses.
- **User Info** - Provides authorization information about the currently logged in user.

Configuration APIs

- **Modify Template Variables** - Enables an external application to update API's template variables in AirWave.
- **Report AP Whitelist** - Enables an external application to create, update, and delete AP Whitelists on AirWave.

Search APIs

- **AP Search** - Provides an interface to AirWave's AP search functionality. This API requires a query string and returns an XML version of the AirWave WebUI search results.
- **Client Search** - Provides an interface to AirWave's client search functionality. This API requires a query string and returns an XML version of the AirWave WebUI's search results.
- **VPN User Search** - Provides an interface to AirWave's VPN User search functionality. This API requires a query string and returns an XML version of the AirWave WebUI's search results.

Report APIs

- **Latest Report** - Provides an XML version of the latest generated report for a given report definition.

Querying by Field

AirWave has two methods for querying the XML API: the URL interface, and the deprecated XML POST interface.

When using the URL interface, all the query parameters are contained in the URL of an HTTP GET request. For example, to query for the most recent log messages of APs with ids of 12 and 13, the URL would be:

https://example.host.com/wln_log.xml?id=12&13

When using the XML POST interface, the query is made by submitting a small XML document in an HTTP POST request. To accomplish the same query as above, the POST parameter "ap" would be set to the following XML string:

```xml
<ap id="12"/>
```
Aruba Airwave API Overview

```xml
<amp:amp_list version="1" xsi:schemaLocation="http://www.airwave.com amp_list.xsd">
  <amp id="3">
    <device_category>controller</device_category>
    <firmware>8.6.0.2</firmware>
    <folder id="6">My-Controller</folder>
    <group id="3">My-Controller</group>
    <is_up>true</is_up>
    <lan_ip>10.1.19.3</lan_ip>
    <lan_mac>00:59:56:A5:73:99</lan_mac>
    <last_contacted>1642700002</last_contacted>
    <last_reboot>1642189319</last_reboot>
    <mfr>Aruba</mfr>
    <model id="683">MM-VA</model>
    <monitor_only>true</monitor_only>
    <name>p13t1-mm</name>
    <operating_mode>ap</operating_mode>
    <planned_maintenance_mode>false</planned_maintenance_mode>
    <reboot_count/>
    <serial_number>MM771D31D</serial_number>
    <snmp_uptime>510942</snmp_uptime>
    <syscontact/>
    <syslocation/>
    <upstream_device_id/>
    <upstream_port_index/>
  </amp>
  <amp id="5">
    <client_count>0</client_count>
    <device_category>controller</device_category>
    <firmware>8.6.0.2</firmware>
    <folder id="6">My-Controller</folder>
    <group id="3">My-Controller</group>
    <is_up>true</is_up>
    <lan_ip>10.1.10.100</lan_ip>
    <lan_mac>00:08:8B:39:E7</lan_mac>
    <last_contacted>1642700017</last_contacted>
    <last_reboot>1642619557</last_reboot>
    <mfr>Aruba</mfr>
    <model id="493">7030</model>
    <monitor_only>true</monitor_only>
    <name>p13t1-mc</name>
    <operating_mode>ap</operating_mode>
    <planned_maintenance_mode>false</planned_maintenance_mode>
  </amp>
</amp:amp_list>
DEMO 5
Aruba Airwave automation examples
Using REST API on Aruba Instant

Step 1: Enable REST API on Instant AP

By default, REST API on Aruba Instant AP is disabled

(InstantAP)(config)#allow-rest-api
(InstantAP)(config)#end
(InstantAP)#commit-apply

Step 2: Login to the master Instant AP

Sample curl command to log in to the master Instant AP

```bash
curl "https://<Master-iap-ip>:4343/rest/login" -H "Content-Type:application/json" --data '{"user": "<username>", "passwd": "<password>"}' --insecure
```

{ "Status": "Success", "sid": "m7z17bicqELh4g5bBSNJ" }
Using REST API on Aruba Instant

Step 3: Run REST API requests to Instant AP

Send a request to Aruba Instant AP with data in JSON format and SID

curl "https://<Master-iap_ip>:4343<API>?
sid=<SID>" -H "Content-Type: application/json" --data @<json_payload_file> --insecure

{ "Status":0,  
"message": "Success" }

Step 4: Logout from the master Instant AP

To close all the interactions, you need to logout

curl "https://<Master-iap-ip>:4343/rest/logout" -H "Content-Type: application/json"--data '{"sid":"<sid>"}’ – insecure -k

{ "Status": 0,  
"message": "User logout successfully“ }

The --insecure option can be used with the curl command if the certificate of the Instant AP cannot be validated
DEMO 6

Aruba Instant AP automation examples
Resources

Aruba Developer hub
- developer.arubanetworks.com

Ansible & Aruba
- www.ansible.com/integrations/networks/aruba

Airheads Developer Community
- community.arubanetworks.com/discussion

Aruba Solution Exchange
- ase.arubanetworks.com

Github
- github.com/aruba