Rest AnyCli testing (Service CliCommand)

Contents

Purpose

The purpose of this document is to demonstrate the use of the rest API to access switch information. The usual way is to access the Rest API to access the data in json format, however not all switch features are exposed via the API. Aruba has introduced the service cli to access information and settings not yet exposed via Rest. The examples are mostly bash command lines, however python is used to extract the result from the information returned (because we can use Json module) The actual cli output is base64 encoded so we have to use the base64 program from coreutils to translate the data

The rest API and anycli provide a more programmatic and reliable method to configure the switch and extract data compared to cli screen scraping scripts.

NOTE: You should be able to cut and paste the commands in the document but sometimes extra characters get included. (I don't know why). If one of the

commands does not work and others do. You may need to type it in directly to the shell.

If you get stuck at the > prompt press ctrl +d

A good getting started guide is here. You should work through these examples before proceeding.

http://arubaos-switch-rest-guide.readthedocs.io/en/latest/

You should find the API guide with this search

ArubaOS-Switch REST API and JSON Schema Reference Guide 16.04

Requirements

ArubaOS switch running v16.04 code

Linux host Ubuntu 15.10 was used for testing but any version with curl and Python installed will probably work.

```
ubuntu@ubuntu-VirtualBox:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 15.10
Release:        15.10
Codename:       wily
```

Linux host with curl and python 2.7 installed. In this testing I used Ubuntu

Base64 program

The anycli API calls return base64 encoded data.

https://askubuntu.com/questions/178521/how-can-i-decode-a-base64-string-from-the-command-line

Step 1 Get a session token

In this example we have enabled the rest API on the switch and we are use the switch user credentials of user = admin and password = Password

curl --insecure -X POST https://192.168.1.251/rest/v1/login-sessions -H "Content-Type: application/json" -H "Accept: application/json" -d '{"userName":"admin", "password":"Password"}'

export cookie=sessionId=<cookie>

Show LLDP Info Remote example

ubuntu@ubuntu-VirtualBox:~$ curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show lldp info remote"}' https://192.168.1.251/rest/v3/cli | python -c 'import json,sys;obj=json.load(sys.stdin);print obj["result_base64_encoded"]'| base64 --decode

This returns the output exactly how it would be displayed on the CLi

```
LLDP Remote Devices Information

LocalPort | ChassisId          PortId            PortDescr SysName
--------- + ------------------ ------------------ --------- ------------------
7         | 186472-c6e63a      18 64 72 c6 e6 3a  eth0      18:64:72:c6:e6:3a
10        | b05ada-983160      10                 10        2930home
```

Curl commands

Get a session token

curl --insecure -X POST https://192.168.1.251/rest/v1/login-sessions -H "Content-Type: application/json" -H "Accept: application/json" -d '{"userName":"admin", "password":"Password"}'

Export the token in bash variable

export cookie=sessionId=<cookie>

curl -s --insecure --cookie $cookie -X GET https://192.168.1.251/rest/v1/system | json_reformat

curl -s --insecure --cookie $cookie -X GET https://192.168.1.251/rest/v1/vlans | json_reformat

Show IP command

Show ip is an example of an anycli command

curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show ip"}' https://192.168.1.251/rest/v3/cli -v | json_reformat

Base64 encoded response

Trying 192.168.1.251...
* Connected to 192.168.1.251 (192.168.1.251) port 443 (#0)
* found 187 certificates in /etc/ssl/certs/ca-certificates.crt
* found 758 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / RSA_AES_256_GCM_SHA384
* server certificate verification SKIPPED
* server certificate status verification SKIPPED
* common name: rbhome (does not match '192.168.1.251')
* server certificate expiration date OK
* server certificate activation date OK
* certificate public key: RSA
* certificate version: #3
* subject: CN=rbhome,OU=lab,O=home,L=nz,ST=ak,C=nz
* start date: Thu, 03 Aug 2017 22:09:21 GMT
* expire date: Fri, 03 Aug 2018 23:59:59 GMT
* issuer: CN=rbhome,OU=lab,O=home,L=nz,ST=ak,C=nz
* compression: NULL
* ALPN, server did not agree to a protocol
> POST /rest/v3/cli HTTP/1.1
> Host: 192.168.1.251
> User-Agent: curl/7.43.0
> Accept: */*
> Cookie:
sessionId=HzdWgBfNeMSZR0TuXGMW2IuGgltS1VEFaSw35YzhAuBdDiezA2kUQOnC0ZQ5o89
> Content-Length: 17
> Content-Type: application/x-www-form-urlencoded
>
} [17 bytes data]
* upload completely sent off: 17 out of 17 bytes
< HTTP/1.1 200 OK
< Server: eHTTP v2.0
< Connection: keep-alive
< Content-Type: application/json
< Transfer-Encoding: chunked
< RequestId:
<
{ [1335 bytes data]
* Connection #0 to host 192.168.1.251 left intact
{
   "uri": "/cli",
   "cmd": "show ip",
   "result_base64_encoded":
"CiBJbnRlcm5ldCAoSVApIFNlcnZpY2UKCiAgSVAgUm91dGluZyA6IEVuYWJsZWQgCgoKICB
EZWZhdWx0IFRUTCAgICAgOiA2NCAgIAogIEFycCBBZ2UgICAgICA6IDIwICAKICBEb2
1haW4gU3VmZml4ICAgOiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICBETlMgc2
VydmVyICAgICAgOiAxOTIuMTY4LjEuMSAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgC
gogICAgICAgICAgICAgICAgICAgICAgIHwgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
AgICAgICAgICAgICAgIFByb3h5IEFSUCAKICBWTEFOICAgICAgICAgICAgICAgICB8IElQIE
NvbmZpZyAgSVAgQWRkcmVzcyAgICAgIFN1Ym5ldCBNYXNrICAgICBTdGQgIExvY2FsCiAg

LS0tLS0tLS0tLS0tLS0tLS0tLS0gKyAtLS0tLS0tLS0tIC0tLS0tLS0tLS0tLS0tLSAtLS0tLS0tLS0tLS
0tLS0gLS0tLS0tLS0tLQogIERFRkFVTFRfVkxBTiAgICAgICAgIHwgTWFudWFsICAgICAxOTIu
MTY4LjEuMjUxICAgMjU1LjI1NS4yNTUuMCAgICBObyAgICBObwogIENBUHMgICAgICAgIC
AgICAgICAgICAgIHwgRGlzYWJsZWQgCiAgc2VydmVyICAgICAgICAgICAgICAgfCBEaXNhYmxlZ
CAKICBNZ21udCAgICAgICAgICAgICAgICB8IERpc2FibGVkIAogIGVtcGxveWVlICAgICAgIC
AgICAgICAgIHwgRGlzYWJsZWQgCiAgQllPRCAgICAgICAgICAgICAgICAgfCBEaXNhYmxlZCAKI
CBHdWVzdCAgICAgICAgICAgICAgICB8IERpc2FibGVkIAogIFZMQU4zNCAgICAgICAgICAgI
CAgIHwgRGlzYWJsZWQgCiAgVkxBTjM1ICAgICAgICAgICAgICAgfCBEaXNhYmxlZCAKICB
WTEFOMzYgICAgICAgICAgICAgICB8IERpc2FibGVkIAogIFROLVRSQU5TUE9SVCAgICAgI
CAgIHwgRGlzYWJsZWQgCiAgVE5QUklOVCAgICAgICAgICAgICAgfCBEaXNhYmxlZCAKIA
oKAA==",
  "status": "CCS_SUCCESS",
  "error_msg": ""
}

We can see the result in the field keyed by "result_base64_encoded": but the information is not particularly user friendly. If we use grep we can just extract the useful bit of the response and strip out all the header info

curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show ip"}' https://192.168.1.251/rest/v3/cli | grep -Po '"result_base64_encoded":.*?[^\\]",'

"result_base64_encoded":"CiBJbnRlcm5ldCAoSVApIFNlcnZpY2UKCiAgSVAgUm91dGluZyA6IE
VuYWJsZWQgCgoKICBEZWZhdWx0IFRUTCAgICAgOiA2NCAgIAogIEFycCBBZ2UgICAgICA
gICA6IDIwICAKICBEb21haW4gU3VmZml4ICAgOiAgICAgICAgICAgICAgICAgICAgICAgICA
gICAKICBETlMgc2VydmVyICAgICAgOiAxOTIuMTY4LjEuMSAgICAgICAgICAgICAgICAgIC
AgICAgICAgICAgICAgCgogICAgICAgICAgICAgICAgICAgICAgICAgICAgIHwgICAgICAgICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIFByb3h5IEFSUCAKICBWTEFOICAgICAgICA
gICAgICAgICB8IElQIENvbmZpZyAgSVAgQWRkcmVzcyAgICAgIFN1Ym5ldCBNYXNrICAgIC
BTdGQgIExvY2FsCiAgLS0tLS0tLS0tLS0tLS0tLS0gKyAtLS0tLS0tLS0tLS0tLS0tLS
0tLSAtLS0tLS0tLS0tLS0tLS0gLS0tLS0tLS0tLQogIERFRkFVTFRfVkxBTiAgICAgICAgIHwgTW
FudWFsICAgICAxOTIuMTY4LjEuMjUxICAgMjU1LjI1NS4yNTUuMCAgICBObyAgICBObwogI
ENBUHMgICAgICAgICAgICAgICAgICAgICAgICAgIHwgRGlzYWJsZWQgCiAgc2VydmVyICAgICAgICAgICAgICAgICAgICA
gICAgfCBEaXNhYmxlZCAKICBNZ21udCAgICAgICAgICAgICAgICB8IERpc2FibGVkIAogIGVt
cGxveWVlICAgICAgICAgICAgICAgICAgIHwgRGlzYWJsZWQgCiAgQllPRCAgICAgICAgICAgICAgICAgfCBEaXNhYmxlZCAg
fCBEaXNhYmxlZCAKICBHdWVzdCAgICAgICAgICAgICAgICB8IERpc2FibGVkIAogIFZMQU4
zNCAgICAgICAgICAgICAgICAgIHwgRGlzYWJsZWQgCiAgVkxBTjM1ICAgICAgICAgICAgICAgfC
BEaXNhYmxlZCAKICBWTEFOMzYgICAgICAgICAgICAgICB8IERpc2FibGVkIAogIFROLVRS
QU5TUE9SVCAgICAgICAgICAgICAgICAgIHwgRGlzYWJsZWQgCiAgVE5QUklOVCAgICAgICAgICAgICAgICAgICAgICAgICAgICAgf
CBEaXNhYmxlZCAKIAoKAA==",

We now have mostly useful data and we could paste this into an online decoder to translate it. You can try that here. Simply select all the data after “result_base64_encoded”: between the quotes (CiB… to KAA==) and paste into the online form

https://www.base64decode.org/

It should return the information exactly like the Cli

Internet (IP) Service

IP Routing : Enabled

Default TTL : 64
Arp Age : 20
Domain Suffix :
DNS server : 192.168.1.1

| Proxy ARP
VLAN | IP Config IP Address Subnet Mask Std Local
---------- + ---------- ---------- ---------- ----------
DEFAULT_VLAN | Manual 192.168.1.251 255.255.255.0 No No
CAPs | Disabled
server | Disabled
Mgmnt | Disabled
employee | Disabled
BYOD | Disabled
Guest | Disabled
VLAN34 | Disabled
VLAN35 | Disabled
VLAN36 | Disabled
TN-TRANSPORT | Disabled
TNPRINT | Disabled

Pipe the output to base64 decode

This is a fairly clumsy way to extract data so we need to decode this as part of the program. At this point I ran out of ideas in the bash shell and had to introduce python programming to solve the problem. This link (https://askubuntu.com/questions/178521/how-can-i-decode-a-base64-string-from-the-command-line) gave me a simple answer to create a one line program. We can pipe the output to python which will extract the data keyed by **["result_base64_encoded"]** . I am sure there will be a way to parse the data from this output in bash but python already has a json module that can easily extract the data.

curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show ip"}' https://192.168.1.251/rest/v3/cli | python -c 'import json,sys;obj=json.load(sys.stdin);print obj["result_base64_encoded"]'

CiBJbnRlcm5ldCAoSVApIFNlcnZpY2UKCiAgSVAgUm91dGluZyA6IEVuY
WJsZWQgCgoKICBEZWZhdWx0IFRUTCAgICAgOiA2NCAgIAogIEFycCB
BZ2UgICAgICAgICA6IDIwICAKICBEb21haW4gU3VmZml4ICAgOiAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAKICBETlMgc2VydmVyICAg

ICAgOiAxOTIuMTY4LjEuMSAgICAgICAgICAgICAgICAgICAgICAgICAgICAgI
CAgCgogICAgICAgICAgICAgICAgICAgICAgIHwgICAgICAgICAgICAgIC
AgICAgICAgICAgICAgICAgICAgICAgICAgICAgIFByb3h5IEFSUCAKICB
WTEFOICAgICAgICAgICAgICAgICB8IElQIENvbmZpZyAgSVAgQWRkcm
VzcyAgICAgIFN1Ym5ldCBNYXNrICAgICBTdGQgIExvY2FsCiAgLS0tLS0t
LS0tLS0tLS0tLS0gKyAtLS0tLS0tIC0tLS0tLS0tLS0tLSAtLS0t
LS0tLS0tLS0gLS0tLS0tLS0tLQogIERFRkFVTFRfVkxBTiAgICAgICAgI
HwgTWFudWFsICAgICAxOTIuMTY4LjEuMjUxICAgMjU1LjI1NS4yNTUu
MCAgICBObyAgICBObwogIENBUHMgICAgICAgICAgICAgICAgICAgIHwgRGl
zYWJsZWQgCiAgc2VydmVyICAgICAgICAgICAgICAgfCBEaXNhYmxlZC
AKICBNZ21udCAgICAgICAgICAgICAgICB8IERpc2FibGVkIAogIGVtcGxv
eWVlICAgICAgICAgICAgIHwgRGlzYWJsZWQgCiAgQllPRCAgICAgICAgI
CAgICAgICAgICAgfCBEaXNhYmxlZCAKICBHdWVzdCAgICAgICAgICAgICA
gICB8IERpc2FibGVkIAogIFZMQU4zNCAgICAgICAgICAgICAgIHwgRGlz
YWJsZWQgCiAgVkxBTjM1ICAgICAgICAgICAgICAgfCBEaXNhYmxlZCA
KICBWTEFOMzYgICAgICAgICAgICAgICB8IERpc2FibGVkIAogIFROLVR
SQU5TUE9SVCAgICAgICAgIHwgRGlzYWJsZWQgCiAgVE5QUklOVCAgI
CAgICAgICAgICAgfCBEaXNhYmxlZCAKIAoKAA==

This is better as we just get the data we need. Now we just have to pipe this output to the base64 utility

We can now test with a few other Cli commands

```
curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show config"}'
https://192.168.1.251/rest/v3/cli | python -c 'import json,sys;obj=json.load(sys.stdin);print
obj["result_base64_encoded"]'| base64 --decode
curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show ip route"}'
https://192.168.1.251/rest/v3/cli | python -c 'import json,sys;obj=json.load(sys.stdin);print
obj["result_base64_encoded"]'| base64 --decode
curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show interface brief"}'
https://192.168.1.251/rest/v3/cli | python -c 'import json,sys;obj=json.load(sys.stdin);print
obj["result_base64_encoded"]'| base64 --decode
curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show interface 7"}'
https://192.168.1.251/rest/v3/cli | python -c 'import json,sys;obj=json.load(sys.stdin);print
obj["result_base64_encoded"]'| base64 --decode
curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show flash"}'
https://192.168.1.251/rest/v3/cli | python -c 'import json,sys;obj=json.load(sys.stdin);print
obj["result_base64_encoded"]'| base64 --decode
curl -s --insecure --cookie $cookie -X POST -d '{"cmd":"show lldp info remote"}'
https://192.168.1.251/rest/v3/cli | python -c 'import json,sys;obj=json.load(sys.stdin);print
obj["result_base64_encoded"]'| base64 --decode
```

Direct Rest API calls

Some examples of direct rest API calls. These will return json formatted outputso we don’t need to use python and the base64 utilty

curl -s --insecure --cookie $cookie -X GET https://192.168.1.251/rest/v3/system | json_reformat

curl -s --insecure --cookie $cookie -X GET https://192.168.1.251/rest/v3/vlans | json_reformat

curl -s --insecure --cookie $cookie -X GET https://192.168.1.251/rest/v3/vlans-ports | json_reformat

curl -s --insecure --cookie $cookie -X GET https://192.168.1.251/rest/v3/qos/dscp-map | json_reformat

curl -s --insecure --cookie $cookie -X GET https://192.168.1.251/rest/v3/qos/policies | json_reformat

curl -s --insecure --cookie $cookie -X GET https://192.168.1.251/rest/v3/port-statistics | json_reformat

curl -s --insecure --cookie $cookie -X GET https://192.168.1.251/rest/v3/device_profiles | json_reformat