SDN: SOFTWARE DEFINED NOW (ACI)

05.03.2024 | Padova

ANDREA DAINESE

NTS

- **Senior Cyber Security / DevNetSecOps Consultant**
- 15+ years' experience in designing, securing, and automating complex IT and OT infrastructures
- Focused on cyber security strategies, and GDPR/ISO 27001 compliance
- Cisco (CCIE), VMware, Red Hat... certified
- Father of Unified Networking Lab (UNetLab)

https://adainese.it
https://www.linkedin.com/in/adainese/

NOTES

NTS

- Part of each network automation engineer is to find doc (a.k.a. googling and/or reverse engineering).
- Network automation engineers needs to write code and write prototypes (a.k.a. spaghetti-code).
- Developers can turn prototypes into robust software. That’s not a task for network automation engineers.
- Developers cannot write network automation software because they lack skills on networking, on vendor products. But they write better code.

NETWORK EXPERTS
ARE NOT
SOFTWARE DEVELOPERS

PREPARING THE ENVIRONMENT

VISUAL STUDIO CODE

NTS

- https://code.visualstudio.com/Download
- Plugins:
  - Ansible by Red Hat
  - HashiCorp Terraform by HashiCorp
  - Jinja by wholroyd
  - TextFSM Template Syntax by andytruett
  - Pylance by Microsoft
  - Python by Microsoft
  - Remote - SSH by Microsoft
  - YAML by Red Hat
  - Shell-format by foxundermoon
  - Postman by Postman
  - Mintlify Doc Writer by Mintlify

VISUAL STUDIO CODE

NTS

Preparation:

- Exchange SSH keys to automatically login to the remote host using the ansible user.
- Use remote explorer to add a remote host
- Install the suggested extensions to the remote host.

ACI SIMULATOR

NTS

Resources: 12 vCPU, 32 GB RAM, 100 GB HDD

Links:

- Application Centric Infrastructure Simulator
- Cisco ACI Simulator VM Installation Guide
- ACI Simulator VM on VMware ESXi

ACI SANDBOX

NTS

- Url: https://sandboxapicdc.cisco.com/
- Username: `admin`
- Password: `!v3G@!4@Y`

INTRODUCTION

APPROACHES

NTS

Declarative (Terraform)

The desired state of the infrastructure is defined in code and Terraform manages the necessary changes to reach that state.
Terraform takes an immutable approach to infrastructure which is defined as instances that do not change over time or are unable to be changed.

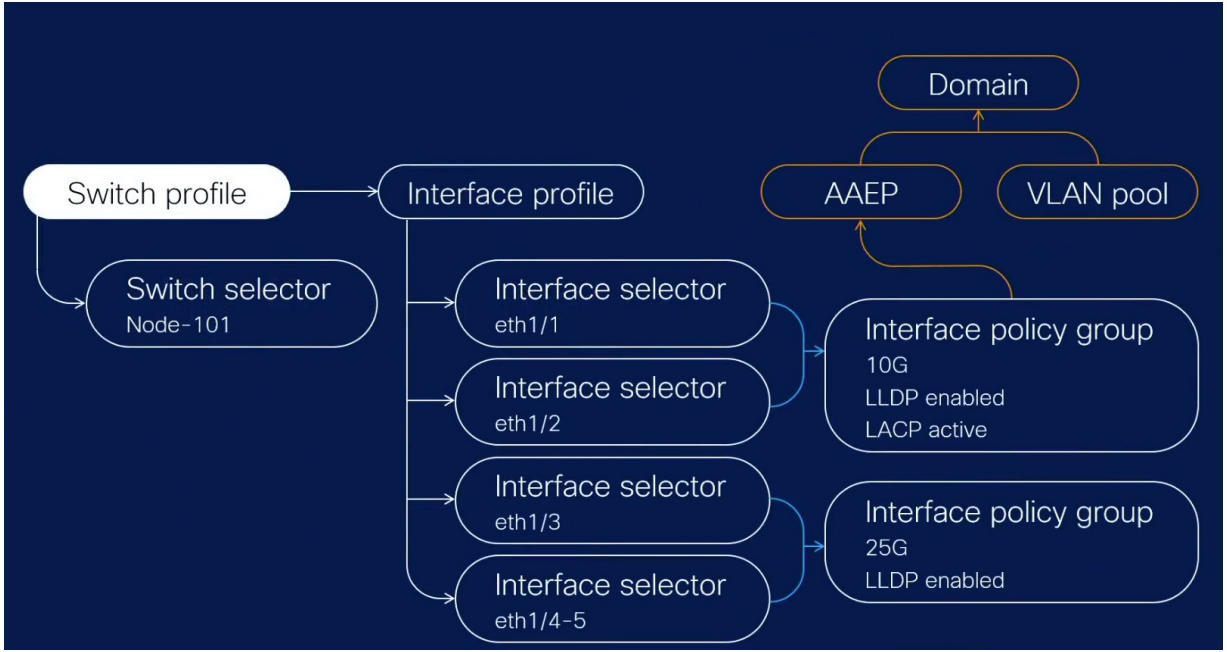
Procedural (Ansible)

Ansible, uses an imperative (or procedural) approach.
Tasks are defined in code and executed in order to reach the desired state.
There is no hidden magic, and the troubleshooting is “simple”.
In network automation, the immutable approach can be implemented by golden configuration.

COMPLEXITY

NTS

12 STEPS TO CONFIGURE A PORT CHANNEL VIA GUI

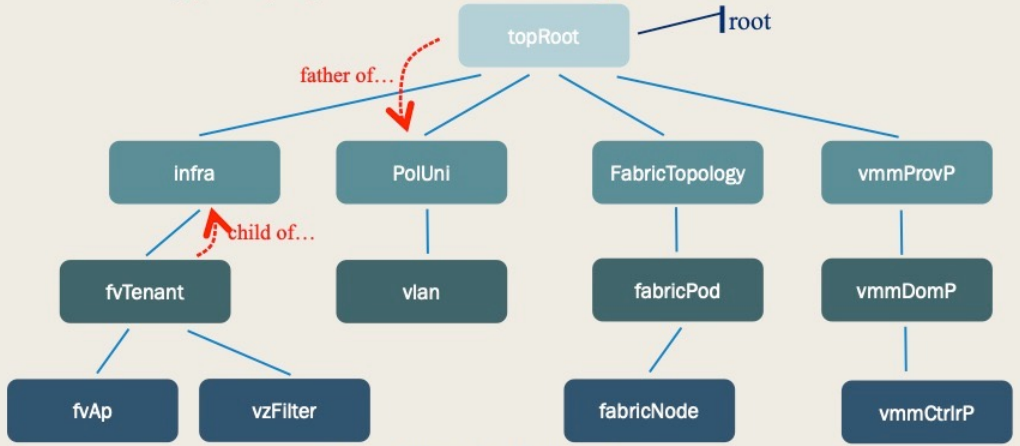


COMPLEXITY

NTS

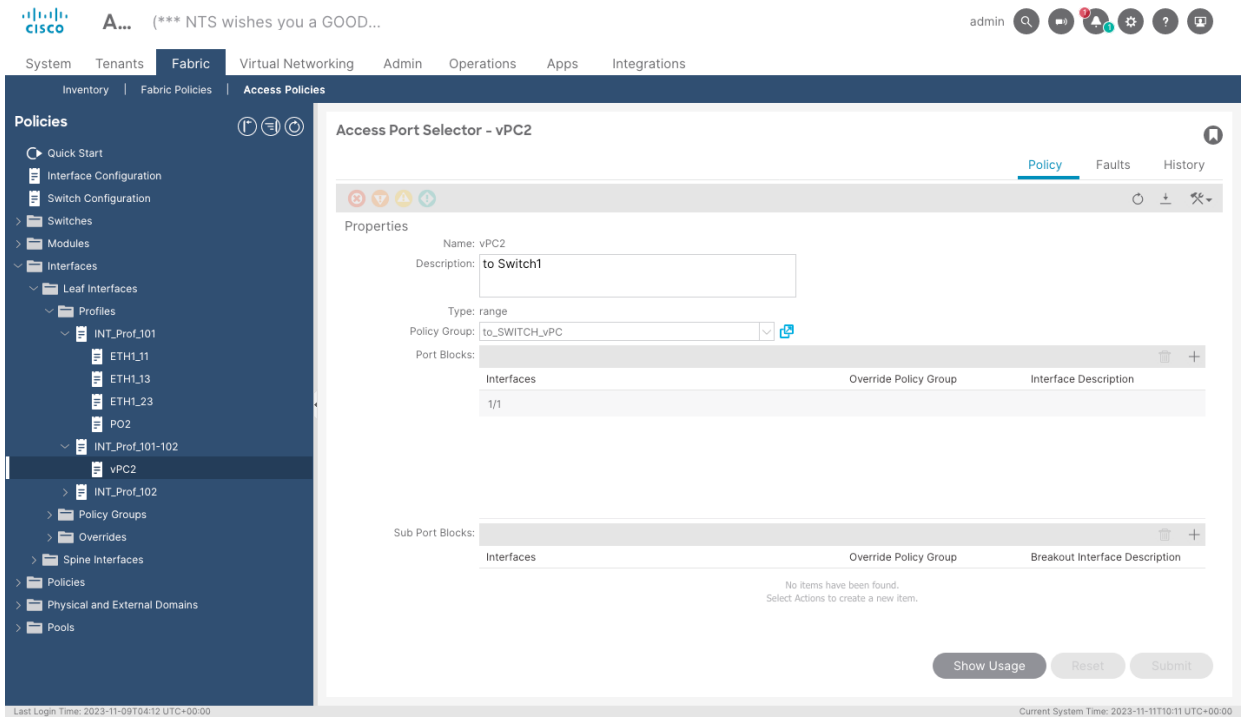
CISCO ACI, HOW TO USE IT! BY MARIO ROSI

- ✓ MIT is a modelled representation of mgmt, application, network, virtualization items of ACI data center
  - ✓ *classes (entities that are instantiated as MOs): pkg_name + class_name*
    - ✓ Package (pkg_name) identifies the functional area
    - ✓ Class (class_name) identifies object type
      - ✓ *e.g.: fvTenant, fv as fabric virtualization, Tenant as class_name*



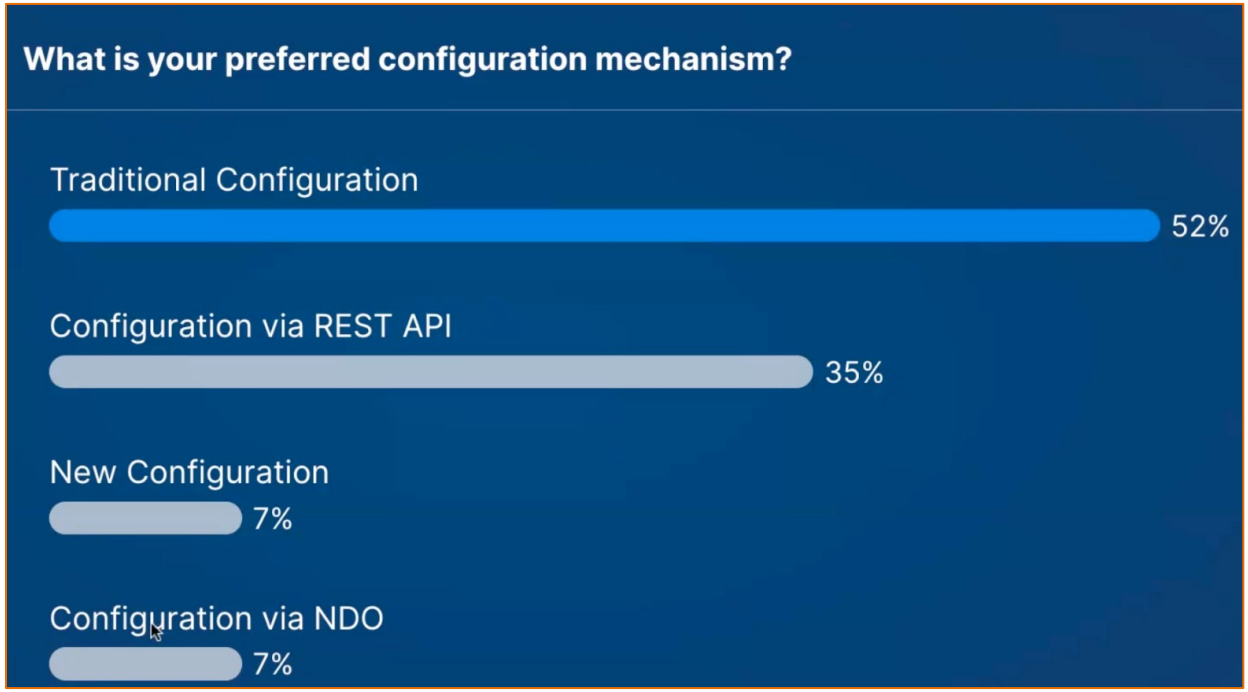
OPERATING

NTS



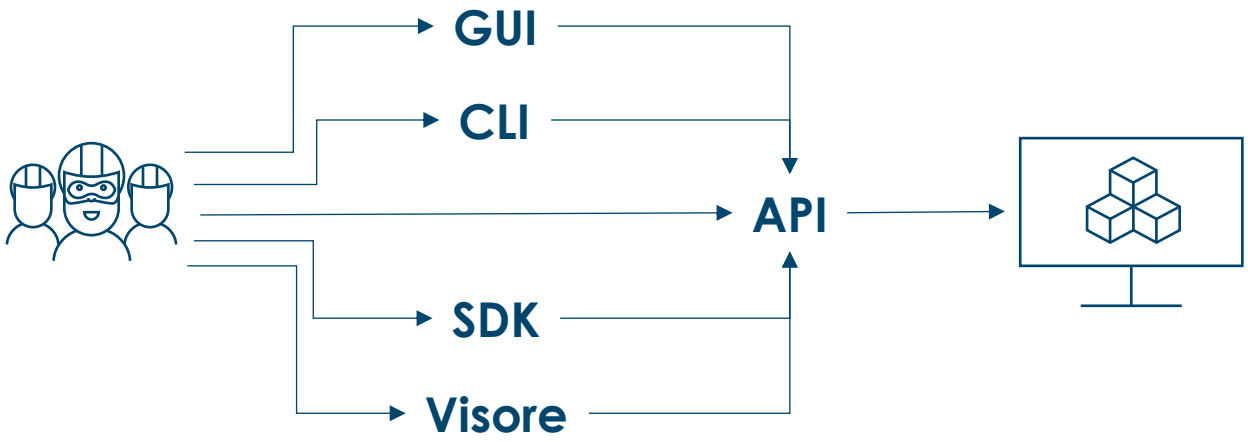
OPERATING

NTS



OPERATING

- Web user interface (GUI)
- Command Line Interface (CLI)
- Automation tools via SDK
- Automation tools via native API



FRAMEWORKS & TOOLS

NTS

- Ansible
- Terraform
- Python (raw)
- Python SDK (Cobra)

- *Postman*
- *GUI*
- *ACI Toolkit*

FRAMEWORKS & TOOLS

NTS

Performance	**1 tenant**	**11 tenants**
▪ Ansible:	0m18.422s	3m22.484s
▪ Terraform (1 tenant):	0m18.229s	-
▪ Python (raw):	0m3.056s	0m33.475s
▪ Python SDK (Cobra):	0m0.906s	0m3.911s

Complexity:

- Ansible: low, well documented
- Python (raw): medium, “self-documented”
- Terraform: high, well documented
- Python SDK (Cobra): highest, no documentation at all

A REAL CASE SCENARIO

A REAL CASE SCENARIO

NTS

Phases:

- Phase 1: a single site, dual pod Cisco ACI fabric replaces a legacy NX-OS based fabric.
- Phase 2: a second site, single pod Cisco ACI fabric replaces the disaster recovery site.

WBS:

- Network assessment
- Design
- Bring up
- Configuration
- Testing
- L3 migration
- L2 migration

A REAL CASE SCENARIO

NTS

Phase 1

Network assessment:

- Automatic discovery and documentation via Python scripts (see NetDoc)
- Output: CSV files (VLANs, IP addresses, VRFs, interface configurations…)

Design: modelling was part of the design phase (process analysis, standardization, normalization…)

Configuration:

- Automatic configuration via Python scripts
- CSV file from network assessment phase were used to automatically configure the fabric, via Python scripts.

A REAL CASE SCENARIO

NTS

Phase 1

Testing: the fabric has been automatically built multiple time until:

- All requested features were 100% working.
- The customer was satisfied by naming convention.
- The customer was satisfied by Python scripts (used also to operate the fabric in the future).

L3 migration: 1-click + testing

L2 migration: customer was able to independently migrate devices.

Expected: 2 years | **Completed in:** 6 months*

A REAL CASE SCENARIO

NTS

Phase 2

Network assessment: the customer was autonomous
Design
Bring up
Configuration: the customer was autonomous
Testing: the customer was autonomous
L3 migration: the customer was autonomous
L2 migration: the customer was autonomous

A REAL CASE SCENARIO

NTS

Final notes

- The modelling phase was crucial.
- Scripts used to assess and operate on the main site were used in the second site without changes.
- The completion time does not include L2 migrations (*)
- Fast doesn't mean easy: spend time to show improvements to the management (KPI)

HOW TO START

HOW TO START

- Task automation / Process automation / Orchestration
- Process analysis
- Standardization
- Modelling
- Documentation

VISORE

NTS

URL and Response of last query

Response Type

JSON XML

URL Copy URL

/api/node/class/fvTenant.json?&order-by=fvTenant.modTs|desc

Response Copy Response

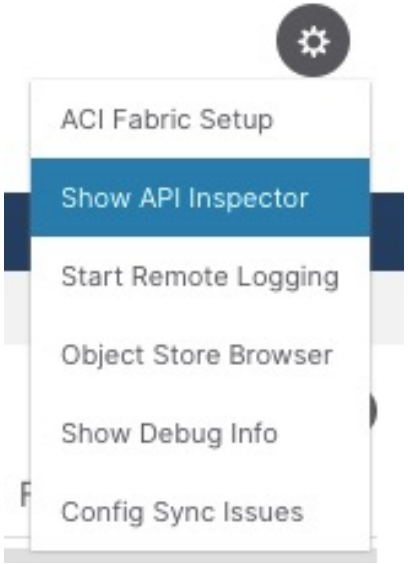

```
{
   "totalCount": "8",
   "imdata": [
      {
         "fvTenant": {
            "attributes": {
               "annotation": "orchestrator:terraform",
               "childAction": "",
               "descr": "Terraform tenant 1",
               "dn": "uni/tn-terraform1",
               "extMngdBy": "",
               "lcOwn": "local",
               "modTs": "2023-05-04T07:26:20.081+00:00",
               "monPolDn": "uni/tn-common/monepg-default",
               "name": "terraform1",
```

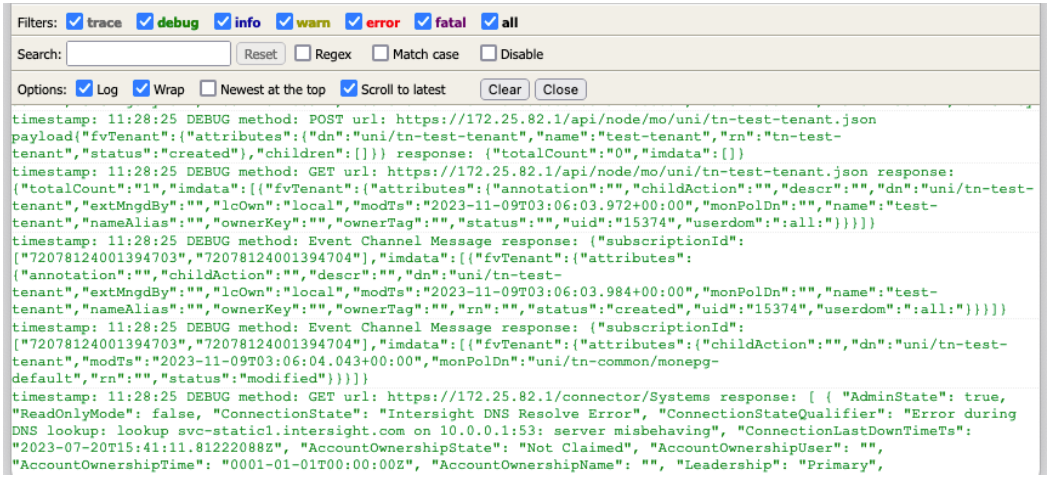

Close

https://172.25.82.1/visore.html#/&class=fvTenant

API INSPECTOR

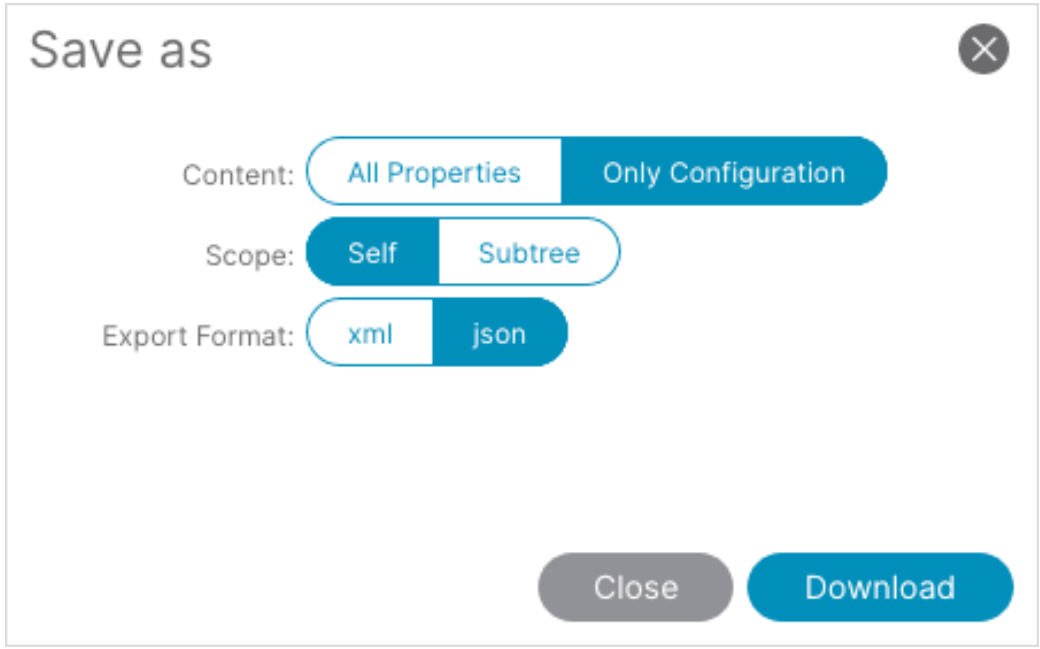
The API Inspector provides a real-time display of REST API commands that the APIC processes to perform GUI interactions. API Inspector displays API communication in a dedicated web browser page while the user navigates the GUI.





CONFIG EXPORT

Each object can be exported in XML or JSON format. The export can possibly include all child objects.
The export can be used as a template to rebuild the same object, or a similar one.



CISCO ACI API ARE “SELF-DOCUMENTED”

SCRIPTING

SCRIPTING

NTS

Login (session timeout is 10 minutes)

```
import requests
username = "admin"
password = "password"
ip = "172.25.82.1"
verify = False

url = f"https://{ip}/api/aaaLogin.json
payload = {"aaaUser": {"attributes": {"name":
    username, "pwd": password}}}

s = requests.session()
r = s.post(url, verify=verify, json=payload)
```

SCRIPTING

NTS

Output

```
r.status_code # HTTP error
r.text        # Unparsed text, only for debug
r.json()      # Text in JSON format
r.cookies     # Useless with sessions
token = r.json()["imdata"][0]["aaaLogin"]
    ["attributes"]["urlToken"]
```

SCRIPTING

NTS

Get tenants

```
url = f"https://{ip}/api/node/class/
    fvTenant.json?challenge={token}"
r = s.get(url, verify=verify)
```

Print a list of tenant

```
for t in r.json()["imdata"]:
    list(t.keys())[0]
```

SCRIPTING

NTS

Create a tenant

```
name = "MyNewTestTenant"
url = f"https://{ip}/api/mo/uni/
    tn-{name}.json?challenge={token}"
payload = {"fvTenant": {"attributes":
    {"name": name}}}

r = s.post(url, verify=verify, json=payload)
assert r.status_code == 200
```

CISCO ACI SDK

CISCO ACI SDK

NTS

A software development kit (SDK or devkit) is typically a set of software development tools that allows the creation of applications for a certain software or hardware.

Cisco ACI SDK

- *ACI Toolkit: the acitoolkit **exposes a small subset** of that model in a way that is meant to provide an introduction to the ACI concepts.*
- Cobra: Cobra is the officially supported python bindings for Cisco APIC REST API.
  Note: documentation covers the high-level classes, only.
- *Arya (APIC Rest Python Adapter): converts APIC object documents from their XML or JSON form into the equivalent Python code leveraging the Cobra SDK.*

CISCO ACI SDK

Cobra is the officially supported python bindings for Cisco APIC REST API. It provides a pythonic library that allows developers to quickly develop and test applications to program the Cisco ACI through the APIC.

The Cisco APIC Python SDK (“cobra”) comes in two installable .whl files that are part of the cobra namespace, they operate as one virtual namespace.

Packages are downloadable from the APIC:

https://172.25.82.1/cobra/_downloads/

CISCO ACI SDK

NTS

Installing Cobra

```
$ pip install aci*whl
```

Cobra example: login

```
import urllib3
from cobra.mit.access import MoDirectory
from cobra.mit.session import LoginSession
urllib3.disable_warnings()

session = LoginSession(f"https://{ip}",
    username, password)

moDir = MoDirectory(session)
moDir.login()
```

CISCO ACI SDK

NTS

Cobra example: get tenants

```
fvTenant_objs =
    moDir.lookupByClass("fvTenant")
for tenant in fvTenant_objs:
    print(tenant.name)
```

Cobra example: get a single tenant

```
dnQuery = DnQuery("uni/tn-test-tenant")
dnQuery.subtree = "children"
for tenant in moDir.query(dnQuery):
    print(tenant.name)
```

CISCO ACI SDK

NTS

Cobra example: add a tenant

```
from cobra.mit.request import ConfigRequest
from cobra.model.fv import Tenant
from cobra.internal.codec.xmlcodec
    import toXMLStr
from cobra.internal.codec.jsoncodec
    import toJSONStr

uniMo = moDir.lookupByDn("uni")
fvTenantMo = Tenant(uniMo, "TenantName")

print(toJSONStr(fvTenantMo, prettyPrint=True))

vmmCfg = ConfigRequest()
vmmCfg.addMo(fvTenantMo)
moDir.commit(vmmCfg)
```

CONCLUSIONS

CONCLUSIONS

NTS

Cisco ACI, the industry's most secure, open, and comprehensive Software-Defined Networking (SDN) solution, enables automation that accelerates infrastructure deployment and governance, simplifies management to easily move workloads across a multifabric and multicloud frameworks, and proactively secures against risk arising from anywhere. It radically simplifies, optimizes, and expedites the application deployment lifecycle.

In short: SDN cannot be built via WebUI (or CLI).