ACI Programmability

Information and Inspiration To Get Started

Quinn Snyder, Developer Advocate and Evangelist
@qsnyder
DEVNET-DC

Agenda

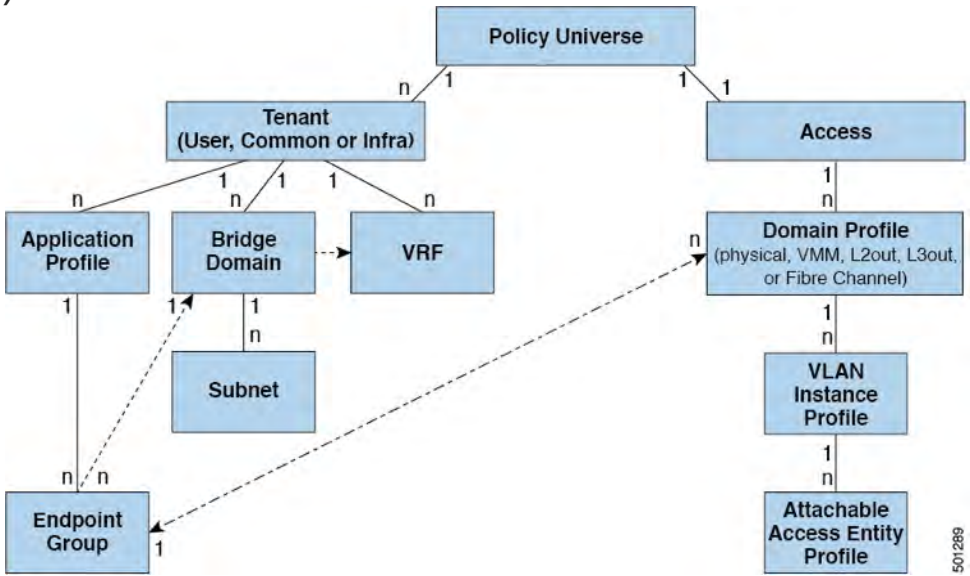
- What is the ACI Object Model?
  - Overview
  - Visore Viewer Exploration
- The ACI RESTful API
  - URI Construction
  - Authentication (Postman v. Python)
- Toolkits, SDKs, 3rd Party Tools
- More Information

What is the ACI Object Model?

The ACI Object Model

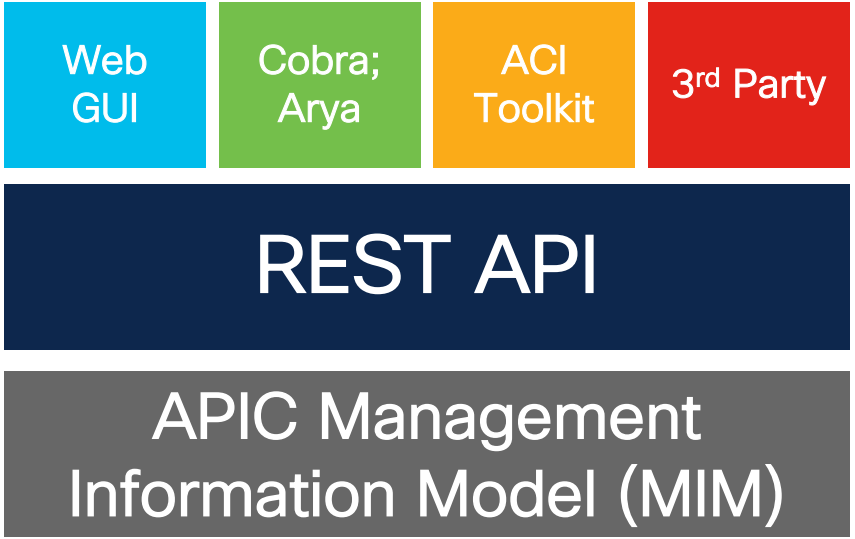
The Foundation of Everything in ACI

- Everything in ACI is an object (MO)
  - Object “class” identifies its type
- Parent/child relationships exist between objects
  - 1-1; 1-N depending on class
- When assembled, creates the MIT/MIM (Management Information Tree/Model)
- Everything builds from “root”
  - Seen as “uni” (policy universe)

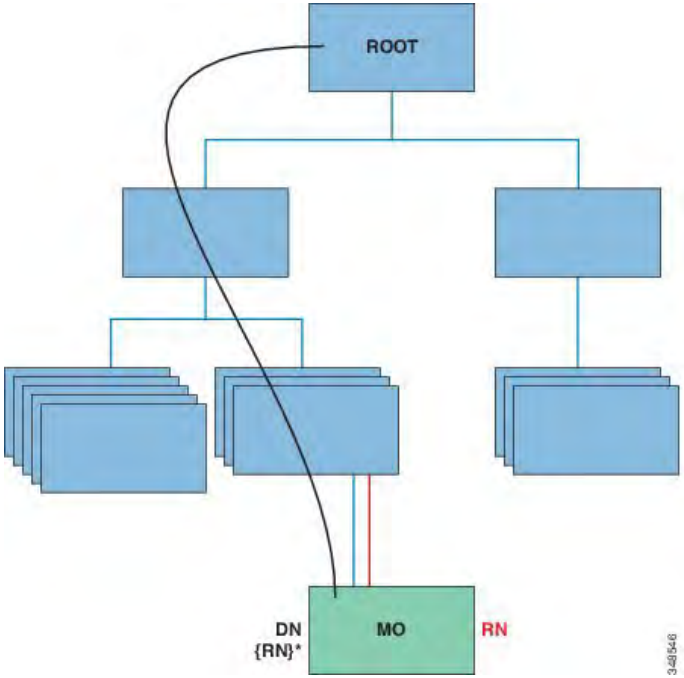


REST API All the Things

- ACI designed as “API first”; nothing without API
- Everything built on top of the REST API; most expose hierarchy
  - APIC GUI
  - Cobra SDK
  - ACI Toolkit
  - Plugins (CNI; vSphere)
  - 3rd Party Tools (Ansible, Terraform)



Its All Relatively Distinguished



- Objects have 2 names
  - Distinguished Name (DN)
    - Unique identification within MIT
    - Series of Relative Names building to “uni” (root)
  - Relative Name (RN)
    - Identify object related to “siblings”
    - Unique within a parent object, but can be used in other classes

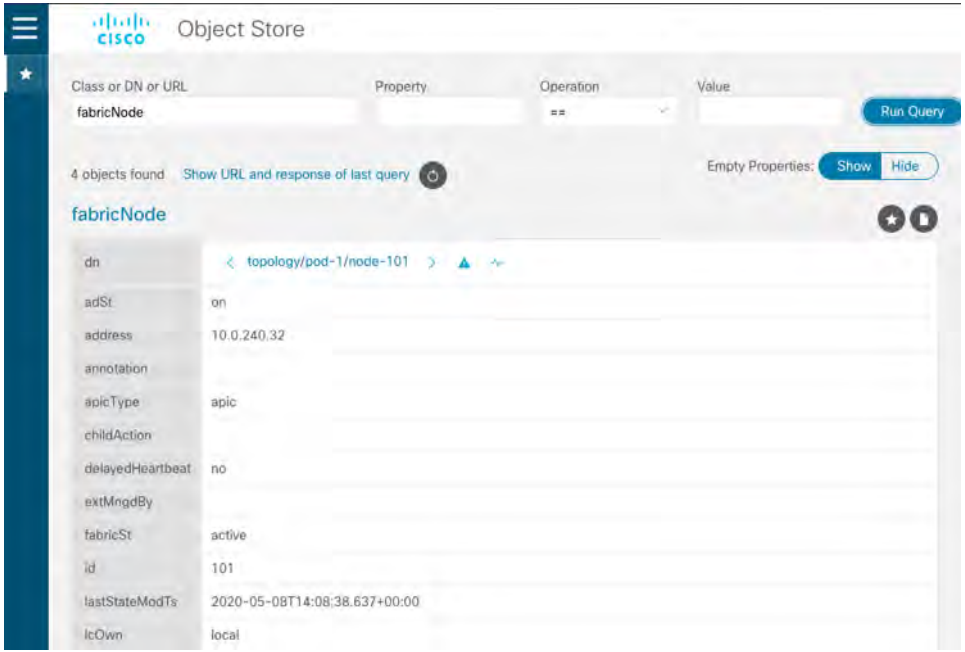
Sample ACI Object Names

Object	Example RN	Example DN	Class
System	uni	uni	uni
Tenant	tn-**Heroes**	uni/tn-Heroes	fvTenant
VRF/Context	ctx-**Development**	uni/tn-Heroes/ctx-Development	fvCtx
Bridge Domain	BD-**Web**	uni/tn-Heroes/BD-Web	fvBD
Subnet	subnet-**10.1.2.1/24**	uni/tn-Heroes/BD-Web/subnet-10.1.2.1/24	fvSubnet
Application Profile	ap-**Save_The_Planet**	uni/tn-Heroes/ap-Save_The_Planet	fvAp
EPG	epg-**Database**	uni/tn-Heroes/ap-Save_The_Planet/epg-Database	fvAEPg
Client Endpoint	cep-**0000.1111.2222**	uni/tn-Heroes/ap-Save_The_Planet/epg-Database/cep-0000.1111.2222	fvCEp
Filter	flt-**HTTP**	uni/tn-Heroes/flt-HTTP	vzFilter
Contract	brc-**Web_Services**	uni/tn-Heroes/brc-Web_Services	vzBrCP
Contract Subject	subj-**HTTP**	uni/tn-Heroes/brc-Web_Services/subj-HTTP	vzSubj

Full ACI Model Reference: https://developer.cisco.com/site/aci/docs/apis/apic-mim-ref/

Visore: Object Model (and API) Browser

- Web page hosted on APIC
  - `http(s)://<apic-ip>/visore.html`
  - Recently updated; options have moved
- Navigate the object model
  - Search by class, DN
  - Move up and down the MIT
- Expose ACI REST API calls

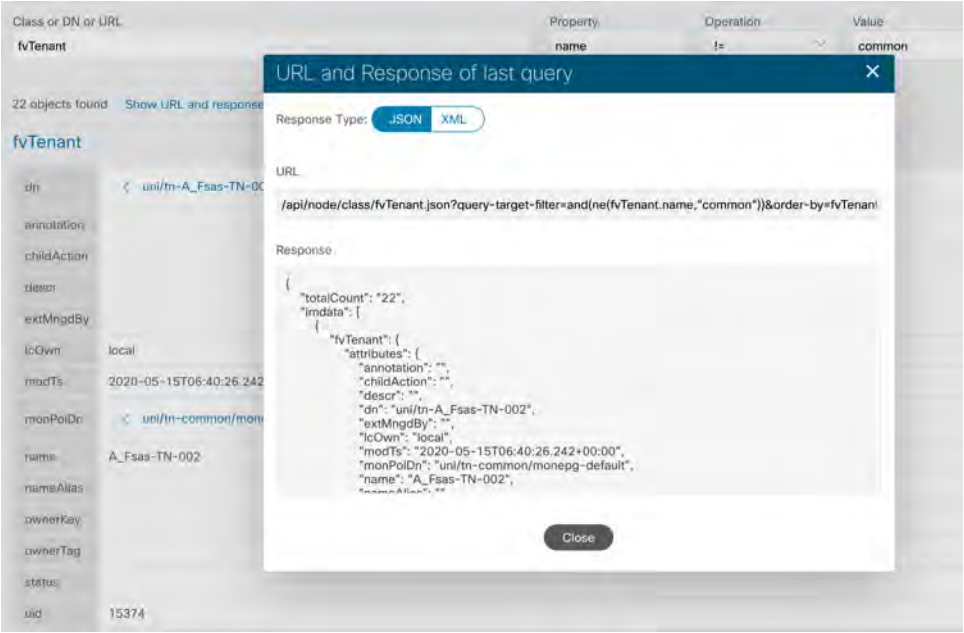


Visore: Searching by Class

- Useful for finding all instances of a specific type (autocomplete)
  - Example: All application profiles all EPGs, all client endpoints

- Find all tenants (other than common)
  - Class: fvTenant
  - Property: name != common
  - Display URI to view ACI REST API call

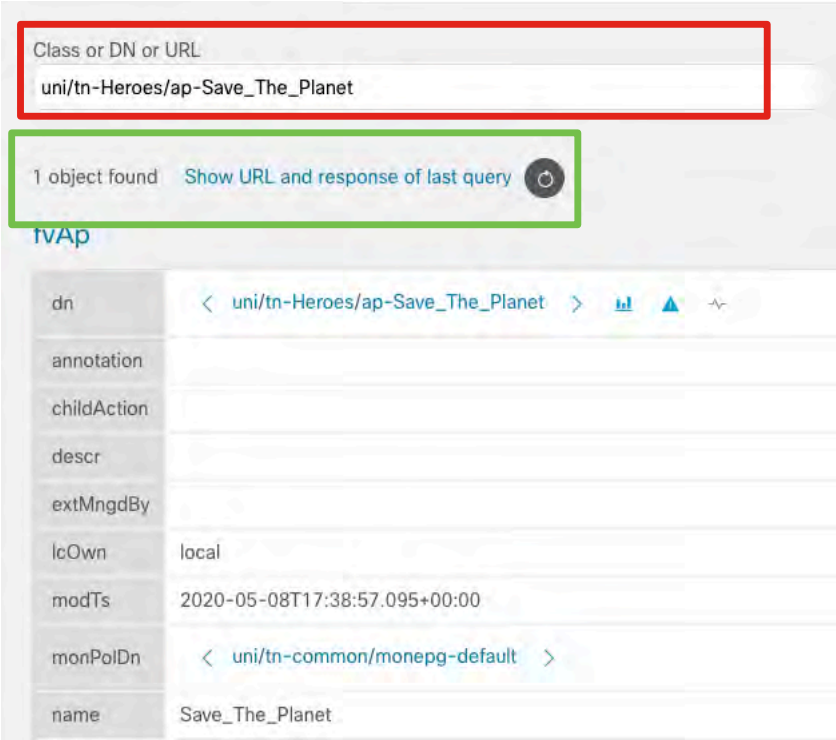
Visore: URL Response



- URL for RESTful call no longer displayed inline
- Reponse pop-up provides selectable options
- Displays JSON'd (or XML'd) output of response – handy to compare

Visore: Searching by DN

- Useful for finding a specific object and/or children
  - Does not autocomplete like class query
  - Example: app profile “Save_The_Planet”
- Display application profile
  - DN: uni/tn-Heroes/ap-Save_The_Planet
  - Display URI to view ACI REST API call



The ACI RESTful API

ACI REST API URI Construction

https://<ADDRESS>/api/<QUERY TYPE>/<IDENTIFIER>.<FORMAT>[?<QUERY PARAMS>]

- api – The main entry point for ACI API REST requests.
- QUERY TYPE
  - **node/class** – Query and return all instances of a given class
  - **node/mo** – Target a specific instance of an object from the MIT
- IDENTIFIER – Class Name or Distinguished Name
- FORMAT – Identify XML or JSON as type of content
  - Used instead of HTTP Headers
- [?<QUERY PARAMS>] – optional parameters that impact returned results; scoping filters
  - Example: query-target – Return Scope (self, children, subtree)

```
https://10.10.10.1/api/class/pcAggrIf.json?query-target=subtree
```

ACI REST API URI Scoping Filters

Filter Type	Syntax	Description
query-target	{self | children | subtree}	This filter defines the scope of the query.
target-subtree-class	<class name>	This filter returns only elements that include the specified class.
query-target-filter	<filter expressions>	This filter returns only elements that match conditions.
rsp-subtree	{no | children | full}	This filter specifies the child object level included in the response.
rsp-subtree-class	<class name>	This filter returns only specified classes.
rsp-subtree-filter	<filter expressions>	This filter returns only classes that matching conditions.
rsp-subtree-include	{faults | health :stats: ...}	This filter returns additional objects.
order-by	<classname.property>| {asc| desc}	This filter sorts the response based on the property values.

ACI REST API CRUD Operations

HTTP Method	Details
GET	Return an object by DN or all instances of a class
POST	Create a new instance of an object or Update details about an existing object.
DELETE	Delete an object

ACI API with Postman

Postman: Manage Environments for Credentials

- Add variables for host, and credentials
- Reference anywhere with `{{variable name}}` syntax

ACI AlwaysOn Login

POST https://{{apic}}/api/aaaLogin.json

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "aaaUser": {
3         "attributes": {
4             "name": "{{username}}",
5             "pwd": "{{password}}"
6         }
7     }
8 }
```

MANAGE ENVIRONMENTS

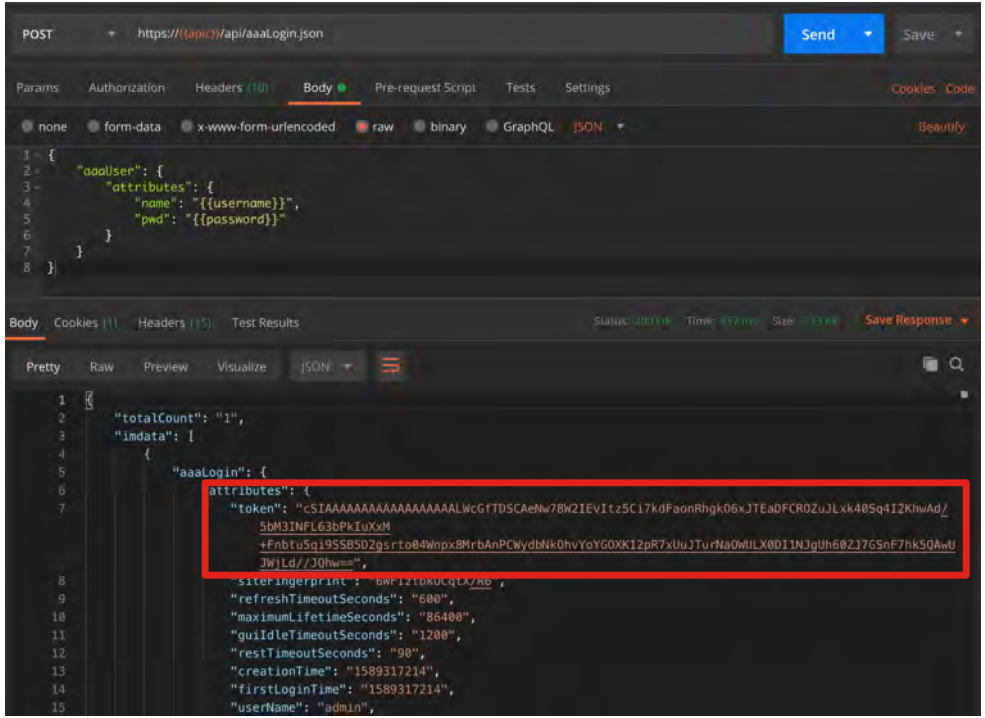
Environment Name

ACI Always-On Sandbox

VARIABLE	INITIAL VALUE	CURRENT VALUE
apic	sandboxapicdc.cisco.co	sandboxapicdc.cisco.com
username	admin	admin
password	ciscopsdt	ciscopsdt
port	443	443

Postman: APIC Login Request

- APIC uses a ticket (token) for authenticating API calls
- **POST** to `/api/aaaLogin.json` with credentials to receive token
- Once logged in, Postman automatically includes token in further requests as a session cookie



Postman: Not the Cookie Monster

- Postman automatically saves the token as ‘APIC-Cookie’
- This cookie will be appended to the header in subsequent requests

Body | Cookies (1) | Headers (15) | Test Results | Status: 200 OK | Time: 984 ms | Size: 2.13 KB | Save Response

Name	Value	Domain	Path	Expires	HttpOnly	Secure
APIC-cookie	cSIAAAAAAAAAAA AAAAAAALWcGfT DSCAeNw7BW2IE vitz5Ci7kdFaonRh gkO6xJTEaDFCRO ZujLxk40Sq4I2Kh wAd%2F5bM3INF L63bPkIuXxM%2 BFnbtu5qi9SSB5 D2gsrto04Wnpx8 MrbAnPCWydbN kOhvYoYGOXK12 pR7xUujTurNaO WULX0DI1NJgUh 60Zj7GSnF7hk5Q AwUJWjLd%2F%2 FIOhw%3D%3D	sandboxapicdc.cisco.com	/	Session	true	true

ACI API with Python

Python: `requests` Makes It Easy

- requests allows you to focus on outcome, not language
- Have to handle cookie tracking and refresh if needed
- Use of “cookie jar” to append login token to header for subsequent request

```
import requests
import json

requests.packages.urllib3.disable_warnings()

encoded_body = json.dumps({"aaaUser": {"attributes":
{"name":"admin","pwd": "ciscopsdt”}}})

resp =
requests.post("https://sandboxapicdc.cisco.com/api/aaaLogin.
json", data=encoded_body, verify=False)

header = {"Cookie": "APIC-cookie=" + resp.cookies["APIC-
cookie"]}

tenants =
requests.get("https://sandboxapicdc.cisco.com/api/node/class
/fvTenant.json", headers=header, verify=False)
```

Python: `requests` Makes It Easy

- Breaking it all down:

JSON-encoded body of the username and password

HTTP POST of body to APIC aaaLogin.json URI

Storing returned token as APIC-Cookie header

```
import requests
import json

requests.packages.urllib3.disable_warnings()

encoded_body = json.dumps({"aaaUser": {"attributes":
{"name":"admin","pwd": "ciscopsdt”}}})

resp =
requests.post("https://sandboxapicdc.cisco.com/api/aaaLogin.
json", data=encoded_body, verify=False)

header = {"Cookie": "APIC-cookie=" + resp.cookies["APIC-
cookie"]}

tenants =
requests.get("https://sandboxapicdc.cisco.com/api/node/class
/fvTenant.json", headers=header, verify=False)
```

ACI REST API Takeaways

- Leverages token in cookie or certificate based authentication – need to account for this
- Uses `.json` and `.xml` within URI instead of `Content-Type` and `Accept` headers to indicate data format
- API will target specific class type or managed object (mo) via DN
- Scoping filters help target the information you want
- REST API Guide available on [Cisco.com]

SDKs, Toolkits, and 3rd Party Tools

ACI Network Programmability Scripting Options

Direct API

Pros:

Limitless options

Any language/method

Cons:

Raw API syntax

Session Management

Individual Atomic Actions

Software Development Kit

Pros:

Language Wrapper of API

Simplifies Syntax and Management

Cons:

Availability

Atomic API Interactions

"Toolkits"

Pros:

Encapsulate common use cases

Less code

Cons:

Not 100% Coverage

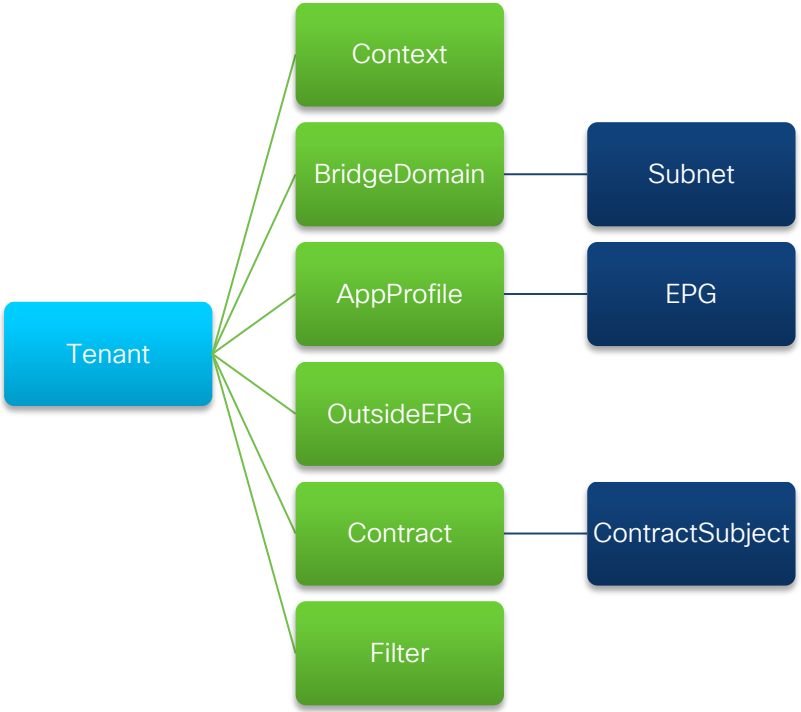
Availability

ACI Toolkit

- Python Libraries for Working with APIC Controller
- Designed to quickly enable users to use REST APIs
- Available on GitHub
  - https://github.com/datacenter/acitoolkit
- Docs
  - http://acitoolkit.readthedocs.io

ACI Toolkit: Object Models

- ACI Toolkit provides a simple, user friendly object model
- Python classes for developer to work with
- Three areas of objects
  - **Application Topology Object Model**
  - Interface Object Model
  - Physical Topology Model



* Partial representation of the Application Topology Object Model

ACI Toolkit: Batteries Included Programmability

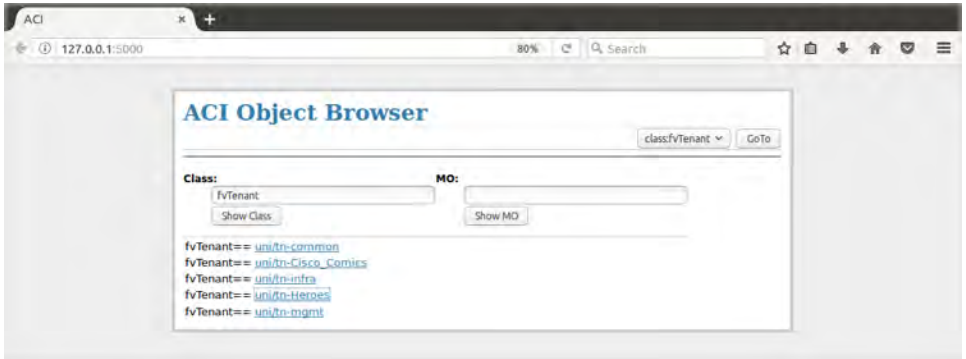
Toolkit Library

```
>>> from acitoolkit.acitoolkit import *
>>> from credentials import *
>>>
>>> session = Session(URL, LOGIN, PASSWORD)
>>> session.login()
<Response [200]>
>>>
>>> tenant_list = Tenant.get(session)
>>> for tenant in tenant_list:
...     print(tenant)
...
common
infra
mgmt
Heroes
>>>
```

Sample Scripts

```
$ python acitoolkit/samples/aci-show-tenant-health.py
APIC login username: admin
APIC URL: https://apic
APIC Password:
tenant                                                    current_health
---------                                                 ----
common                                                    100
infra                                                     100
Heroes                                                    95
mgmt                                                      100
$
```

Toolkit Applications



Cobra Python SDK and PyACI

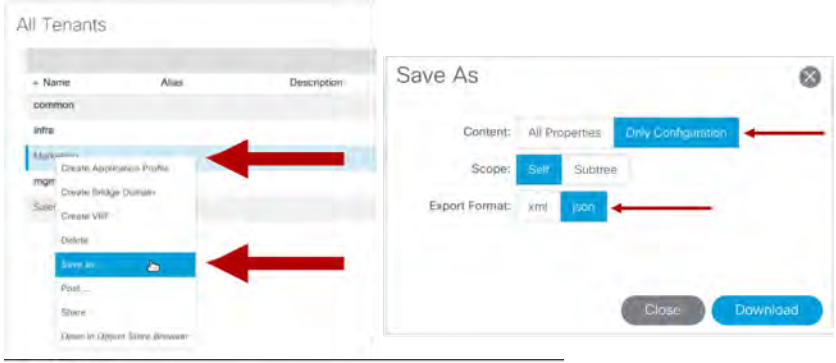
- Full SDK and Pythonic bindings for building ACI apps
- Cobra Python packages
  - **acicobra**: for interacting with APIC
  - **acimodel**: a model of the MIT
- Cobra download from APIC controller
  - `https://<apic address>/cobra/_downloads`
  - Version available on DevNet to complete labs against sandbox
- PyACI Download
  - https://github.com/datacenter/pyaci
- Docs
  - https://pyaci.readthedocs.io/en/latest/
  - https://cobra.readthedocs.io

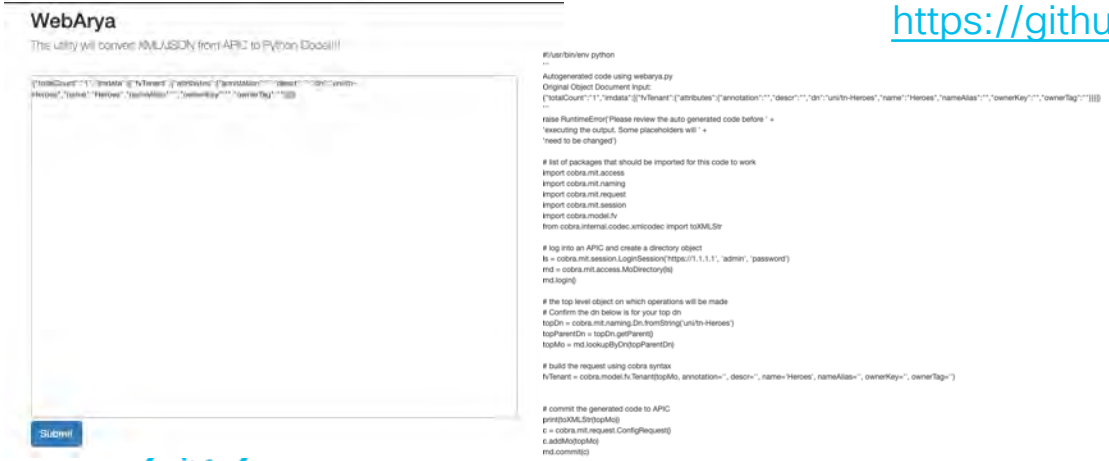
Cobra: Code Made Easier with ARYA

- Cobra provides full MIT access; can be intimidating
- Export valid object from APIC, run ARYA against export, profit!

- Cobra applies all created/modified configuration objects in single atomic commit
- Source code at https://github.com/datacenter/arya

Cobra: Webarya – Flask + ARYA

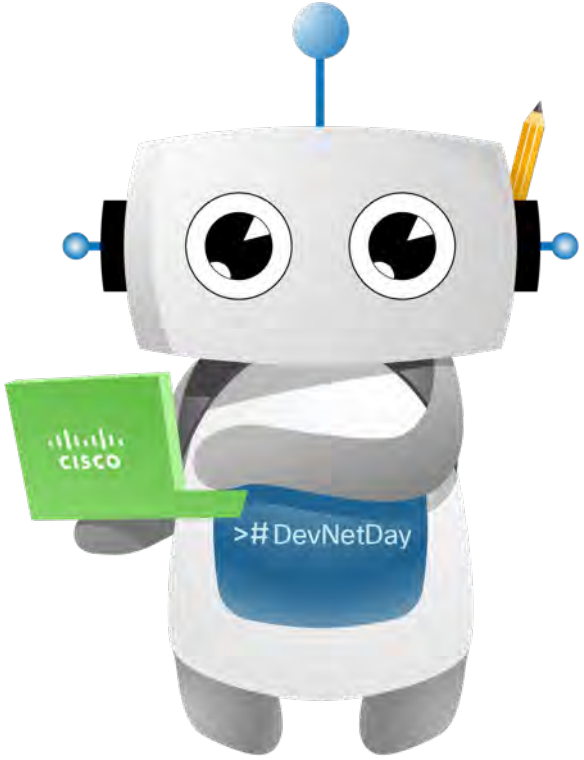




- Flask web front-end for ARYA
- Clone repo, install requirements, run locally
- Paste exported JSON/XML, receive Cobra code
- Source code at https://github.com/datacenter/webarya

Explore More

- REST API Documentation
  - http://cs.co/ACI_API
- ACI Toolkit Documentation
  - https://acitoolkit.readthedocs.io
- Cobra SDK Documentation
  - https://cobra.readthedocs.io
- ACI Programmability Learning Labs
  - http://cs.co/DevNet_ACI
- Always-On ACI Sandbox
  - http://cs.co/ACI_SBX
- ACI on DevNet
  - https://developer.cisco.com/aci



Thank you

Possibilities

#CiscoLive | #DevNetDay