Cisco APO Use Case Runbook

## Technology: DC

## Level: Walk

## Name: **Python ACI Toolkit**

**ACI Toolkit**

*Scenario: a Mid-to-Large Commercial Account seeks to accelerate their ACI rollout with automated tooling. ACI toolkit provides documentation, provisioning, and compliance scripts to accelerate ACI deployment.*

Customer Story: Customer A has 1000 VMs across their data centers. Setting the needed policies required **500+ hours** to create. However, after implementing the ACI Toolkit, Customer A was able to reduce the time required **to just 8 hours yielding a 98% savings** in time and allowing their team to focus on new projects to move the company into new business models.

Partner Story: VAR Partner A taught their deployment engineers to use the toolkit and was able to reduce the time to deploy the end customers ACI fabric, reduce configuration errors, and document how the system was deployed, increasing the services margin on the deal.

**Code Exchange Link:** **Data Center Ansible Configuration** -<https://developer.cisco.com/codeexchange/github/repo/datacenter/acitoolkit>

## **Use Case Playbook**

The ACI Toolkit is a set of python libraries that allow basic configuration of the Cisco APIC controller. It is intended to allow users to quickly begin using the REST API and accelerate the learning curve necessary to begin using the APIC.

The full documentation is published at the following link: <http://datacenter.github.io/acitoolkit/>

This lab requires Python and the [setuptools package](https://pypi.python.org/pypi/setuptools) as well as the library dependencies listed unter the “requirements.txt” file. If you have git installed, clone the repository at git clone <https://github.com/datacenter/acitoolkit.git>. If you don't have git, [download a zip copy of the repository](https://github.com/datacenter/acitoolkit/archive/master.zip) and extract. For the latest build of this project is also available as a Docker image from Docker Hub at “docker pull dockercisco/acitoolkit“

## Installing

After downloading, install using setuptools.

cd acitoolkit

python setup.py install

If you plan on modifying the actual toolkit files, you should install the developer environment that will link the package installation to your development directory. This is not required to run the labs, but will be useful for future expansion of the toolkit.

# Usage

A tutorial and overview of the acitoolkit object model can be found in the Documentation section at: <http://datacenter.github.io/acitoolkit/>. Each individual toolkit is documented in the <https://github.com/datacenter/acitoolkit/tree/master/docs/source> folder. (NOTE: We may want to build the current document set and include as a pdf file for the Use Case)

# You can also use a Docker container to run the ACI toolkit using the Docker Image built from the ‘Dockerfile’ in the code repository. Run: ‘docker run -it --name acitoolkit dockercisco/acitoolkit’

The labs are run against the Cisco DevNet ACI Sandbox. Don’t use the ACI Simulator as it requires a jumpbox, but use the reservable ACI Lab at: <https://devnetsandbox.cisco.com/RM/Diagram/Index/87944d49-5c56-49d2-b208-ae7f9bc8d930?diagramType=Topology>

**Overview:**

The ACI Sandbox provides a developer with an environment to design, develop and test using the ACI RESTful APIs over http/https with XML and JSON encodings. The central SDN Controller of the ACI solution, the Application Policy Infrastructure Controller (APIC) manages and configures the policy on each of the switches in the ACI fabric. Diagram and network element information about the ACI Sandbox Topology can be found [here](https://devnetsandbox.cisco.com/sandbox-instructions/ACI_Simulator_Reservation_2.2/ACI-Reserved-Sandbox-TopologyV4.jpg).

**Lab Walk Through:**

Information about device functionality that can be exercised in the ACI Sandbox can be found on the Sandbox homepage.

**Connecting to the APIC Controller:**

This ACI Sandbox Lab contains a dedicated Hardware Fabric topology with ACI Server Instance.

* IP: **10.10.20.42** (After VPN connection)
* SSH Username: **admin**
* SSH Password: **Cisco123!**

**Lab Architecture:**  Diagram and network element information about the [ACI Sandbox Topology](https://devnetsandbox.cisco.com/sandbox-instructions/ACI_Hardware_Reservation/ACI-Sandbox-TopologyV3.12v.jpg). The ACI Sandbox contains physical network elements and servers that developers can reserve and remotely access so they can develop, debug and test their ACI application.

**Good Citizen Code of Conduct:**

This APIC controller and Fabric topology resource is a dedicated instance.Please, do not erase resources you have not created yourself. Also, do not perform performance testing against this dedicated instance of ACI.

## Labs:

You should now be ready to complete the 3 Labs:

**Lab1** is a script to log on to the APIC and displays all of the Interfaces. We will use the ‘reports’ tool at: <https://github.com/datacenter/acitoolkit/tree/master/applications/reports>. This lab should take about 30 minutes.

We will go to the ‘./acitoolkit/applications/reports’ directory from where you installed the repo. Run the aci-report-logical.py’ program against the sandbox ACI controller. This should return a table of the Tenants and their Descriptions.

For example: to run a logical report for the ACI Sandbox run: “python aci-report-logical.py -u https://10.10.20.42 -l admin -p Cisco123!”.

Also run the “python aci-report-logical.py -h” to call out the various reports such as endpoints, epg’s, contracts, etc…

You should also run the ‘aci-report-security-audit.py’ and ‘aci-report-switch.py’ programs to see how you can gather that data as well.

Then you can run the aciReportGui.py version. This version runs a Flask web server that puts an HTML-based frontend on the program. Launch the program ‘aciReportGui.py’ and then point a web browser to ‘localhost:5000’. NOTE: will be different if in a container and will require the container IP instead of localhost.

**Lab 2** implements a set of functions that returns information from the APIC Controller using a cli wrapper. This is done using the cli tool at: <https://github.com/datacenter/acitoolkit/tree/master/applications/cli>

This lab should take about 30 minutes.

For this lab we will need to run the ‘acitoolkitcli.py’ program in the “./acitoolkit/applications/cli” folder. First run this with the ‘-h’ flag to see the required fields.

Edit the ‘test\_cli.txt’ file to initially limit the number of items returned - perhaps to just show bridgedomain/tenant/app/epg/interface for the first run, and name the file ‘myfirsttest\_cli.txt’.

Run the test with the command “python acitoolkitclil.py -l admin -p Cisco123! -u https://10.10.20.42 -t myfirsttest\_cli.txt” NOTE: The order of the flags seems to be required to be -l -p -u -o -t for this command.

Edit the ‘test\_cli.txt’ file to try different command sets and discover how easy it is to document the ACI install before and after you make changes. NOTE: the initial ‘test\_cli.txt’ file does contain configuration changes and you should experiment with these at your peril!

**Lab3** implements an event feed to track changes to the config. We will launch a web server that can track the changes using the tool at: <https://github.com/datacenter/acitoolkit/tree/master/applications/eventfeeds> This lab should take about 30 minutes.

To launch the server, go to the “./acitoolkit/applications/eventfeeds” folder and run the ‘eventsfeed.py’ program. Again, using the ‘-h’ command will show you how to pass parameters to the program. Run the “python eventfeeds.py -u https://10.10.20.42 -l admin -p Cisco123!” to launch the server that is available at localhost:5000 with your web browser.

You can now see the Events Feeds from the Web Interface. Explore the data and look at how the interface was built using the HTML code for the web interface in the “templates” folder and the CSS file in the “static” folder.

Once you complete the three Learning labs you will be ready for the challenge.

## Completion Challenge (Thoughts)

In order to provide a Proof of Performance Challenge we are going to set the following goal. Take what you have learned in the 3 labs and configure your lab to add the required fields to automatically configure your APIC.

The challenge is to automate the necessary components - IP addresses, bridge domains, tenants, app, epg, contracts, context, and vlans - to complete the setup using components from the ACItoolkit. Feel free to use any design you wish that works within the ACI Sandbox.

Once you have completed the challenge, please host a 2 minute demo of your application running the configuration and display the configuration from the web interface. Once you record the video, send the link to {{ [email\_addr@cisco.com](mailto:email_addr@cisco.com) }} where your video will be reviewed for credit. We suggest doing a Webex demo and recording it. Share the recording link via the email for review.