Cisco Application Centric Infrastructure Toolkit: Simplify Cisco ACI Operations

Guide

November 2014

Contents

Introduction

Cisco® Application Centric Infrastructure (ACI) technology enables you to integrate virtual and physical workloads in an easy-to-use and highly programmable multihypervisor fabric.

Programming the Cisco ACI fabric requires some understanding of the object model to configure representational state transfer (REST) calls correctly. The object model is a very powerful logical representation of the fabric. The task of learning the object model may seem complex at the beginning because it covers every element of the fabric.

The Cisco ACI toolkit is a set of Python libraries that makes programming the fabric a much more intuitive process. The Cisco ACI toolkit also provides a command-line interface (CLI) similar to that of Cisco NX-OS Software, which simplifies the operator's interaction with the fabric.

The Cisco ACI toolkit Python libraries can be downloaded from github.

The Cisco ACI toolkit offers these main benefits:

- It makes writing Python scripts easier because fewer classes are exposed and naming is simplified.
- It makes programming the fabric with scripts easier to learn than programming using REST calls because naming conventions are easier to learn.
- It preserves the concepts of Cisco ACI but makes them easier to learn.
- It provides a CLI application that enables interaction with the fabric that is similar to that with Cisco NX-OS.

Cisco ACI Model

This section summarizes some of the main concepts of Cisco ACI that you should be familiar with to understand what the ACI toolkit does.

Figure 1 shows the main elements of a Cisco ACI configuration. In this configuration, each tenant has two virtual routing and forwarding (VRF) instances, which are subdivided into bridge domains, and each bridge domain has multiple endpoint groups (EPGs).

Figure 1. Cisco ACI Configuration with Two Tenants

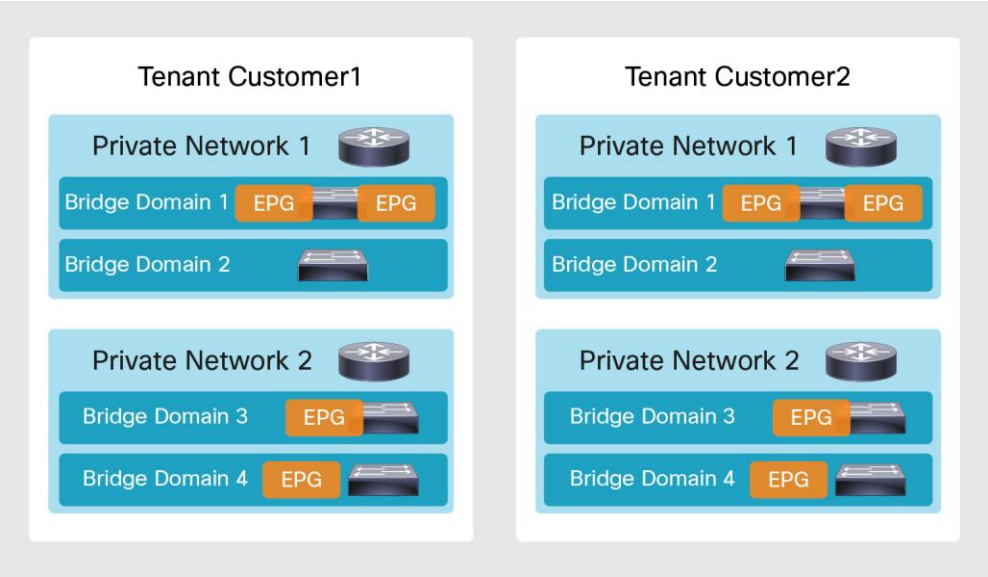
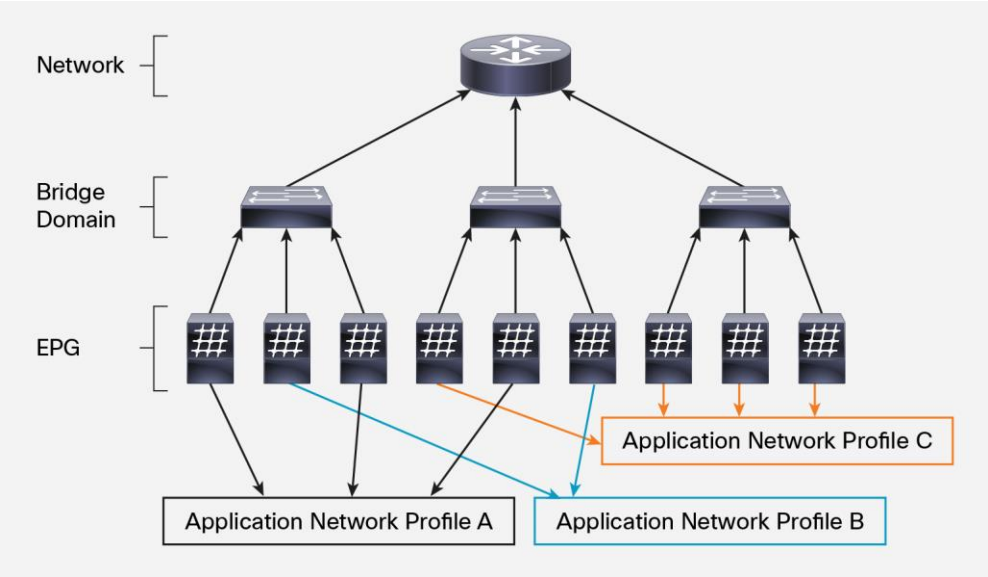


Figure 2 shows the relationship of the networks, bridge domains, and EPGs.

Figure 2. Relationship of Networks, Bridge Domains, and EPGs



A bridge domain is a container for subnets that can act as a broadcast or flooding domain if broadcast or flooding is enabled.

The bridge domain references a VRF instance called the Layer 3 network. The subnets and gateways for the workloads are defined as part of the bridge domain.

The relationships among the various objects are as follows: the EPG points to a bridge domain, and the bridge domain points to a Layer 3 network.

EPGs are grouped into application network profiles. The application profile can span multiple bridge domains. By grouping EPGs into application profiles, the administrator makes the network aware of the relationships among application components.

Sample Configuration with REST Calls (Without Cisco ACI Toolkit)

This section illustrates how to configure Cisco ACI with REST calls. This section does not use the Cisco ACI toolkit, but it is useful for comparison with the same configuration performed using the toolkit.

You can configure Cisco ACI by using REST calls with an XML or JavaScript Object Notation (JSON) payload that modifies the elements in the object tree.

Simple operations are performed as shown here.

To create a tenant (**fvTenant**), use this code:

```
POST to http://apic1/api/mo/uni.xml
<fvTenant name='Tenant1' status='created,modified'>
</fvTenant>
```

To create an application network profile (**fvAp**), use this code:

```
POST to http://apic1/api/mo/uni.xml
<fvTenant name='Tenant1' status='created,modified'>
   <fvAp name='WebApp'>
   </fvAp>
</fvTenant>
```

To add EPGs (**fvAEPg**), use this code:

```
POST to http://apic1/api/mo/uni.xml
<fvTenant name='Tenant1' status='created,modified'>
<fvAp name='WebApp'>
   <fvAEPg name="WEB" status="created,modified"/>
</fvAp>
</fvTenant>
```

The following example shows the configuration of a tenant. In the example, the following fields are defined as follows:

- **fvCtx** indicates the routing instance.
- **fvBD** is the bridge domain.
- **fvRsCtx** is the pointer from the bridge domain to the routing instance.
- **fvSubnet** is the list of subnets and default gateways for the bridge domain.
- **fvRsDomAtt** is the reference to the virtual machine mobility domain.

```
POST to http://apic1/api/mo/uni.xml
<polUni>
    <fvTenant dn="uni/tn-Customer1” name="Customer1">
    <fvCtx name="customer1-router"/>
    <fvBD name="BD1”>
```

```
        <fvRsCtx tnFvCtxName="customer1-router" />
        <fvSubnet ip="10.0.0.1/24" scope="public"/>
      </fvBD>
     <fvAp name="web-and-ordering">
       <fvAEPg name="VLAN10”>
         <fvRsBd tnFvBDName="BD1"/>
         <fvRsDomAtt tDn="uni/vmmp-VMware/dom-Datacenter"/>
       </fvAEPg>
       <fvAEPg name="VLAN20”>
         <fvRsBd tnFvBDName="BD1"/>
         <fvRsDomAtt tDn="uni/vmmp-VMware/dom-Datacenter"/>
       </fvAEPg>
  </fvTenant>
  </polUni>
```

Using Python with the Cisco ACI Toolkit

You can configure Cisco ACI in multiple ways:

- Using the GUI
- Using REST calls
- Using the Python software development kit (SDK)

The Cisco ACI toolkit is a library for use with the Python SDK for Cisco ACI. Using the Cisco ACI toolkit makes creating the scripts easier. Python scripts written with the Cisco ACI toolkit may also be easier to understand than the XML payload used in REST calls.

For example, to create a tenant you would configure the following line:

```
tenant = Tenant(‘Tenant1’)
```

To create an application network profile, you use this code:

```
app = AppProfile(‘WebApp’, tenant)
```

To add EPGs, you use this code:

```
web_epg = EPG(‘WEB’, app)
```

To create a bridge domain, you use this code:

```
bd = BridgeDomain(‘BD1’)
```

To associate the EPG with the bridge domain, you use this code:

```
web_epg.add_bd(bd)
```

The following example shows the configuration of a tenant using the Cisco ACI toolkit:

```
from acitoolkit.acitoolkit import *
[…]
tenant = Tenant (‘Customer1’)
context = Context (‘customer1-router’, tenant)
bd = BridgeDomain(‘BD1’, tenant)
bd.add_context(context)
```

```
bd.add_subnet('10.0.0.1/24')
app = AppProfile('web-and-ordering', tenant)
vlan10 = EPG('VLAN10', app )
vlan10.add_bd(bd)
vlan20 = EPG('VLAN20', app )
vlan20.add_bd(bd)
```

CLI with the Cisco ACI Toolkit

The Cisco ACI toolkit also provides applications that use the toolkit libraries. One of these applications is a CLI that is similar to that of Cisco NX-OS. This CLI runs on the operator's PC. To launch the CLI application, you enter a command like the following from the PC:

```
python acitoolkitcli.py  -l username -p password -u https://APIC-IP
```

Sample configurations performed using the CLI are shown here.

To create a tenant, use this code:

```
fabric(config)# [no] tenant <tenant-name>
```

To switch to a tenant configuration mode (in the manner of a virtual device context [VDC]), use this code:

```
fabric# switchto <tenant-name>
```

To create an application network profile, use this code:

```
fabric-tenant(config)# [no] app <app-profile-name>
```

To add EPGs, use this code:

```
fabric-tenant(config-app)# [no] epg <epg-name>
```

To create a bridge domain, use this code:

```
fabric-tenant(config)# [no] bridgedomain <bd-name>
fabric-tenant(config-bd)# [no] context <context-name>
```

To associate the EPG with the bridge domain, use this code:

```
fabric-tenant(config-epg)# [no] bridgedomain <bd-name>
```

You will then be able to run **show** commands such as these:

```
show tenant [<tenant-name>]
fabric# show bridgedomain [<bd-name>] [detail]
fabric# show interface [detail]
fabric# show context
```

Interaction of the Toolkit with the Cisco Application Policy Infrastructure Controller

The Cisco ACI toolkit interacts with the Cisco Application Policy Infrastructure Controller (APIC) in the same way as the GUI does or as any other automation tool does. The Cisco ACI toolkit is based on the Python SDK for Cisco ACI, which sends REST calls to the Cisco APIC (Figure 3).

Figure 3. Toolkit Relationship to the SDK and Cisco APIC

Current Limitations

At the time of this writing, the Cisco ACI toolkit covers the following functions:

- Tenant creation
- Application profiles
- EPGs
- Interfaces
- PortChannels and virtual PortChannels (vPCs)
- Bridge domains
- Contexts
- Subnets
- Contracts
- Outside EPGs

The Cisco ACI toolkit currently doesn't cover the following:

- Service graphs
- Configuration of virtual machine manager (VMM) domains
- Cisco Switched Port Analyzer (SPAN) and atomic counter configurations
- Faults, statistics, and health scores

Conclusion

The Cisco ACI toolkit simplifies Cisco ACI fabric operations:

- It exposes a subset of the object tree.
- It provides classes and methods with simplified naming conventions.
- It provides a CLI that enables operators to interact with the fabric in the same way as they interact with a Cisco NX-OS switch.

For More Information

You can find the toolkit at https://github.com/datacenter/Simple-ACI-Toolkit and for more information, please refer to http://datacenter.github.io/acitoolkit/docsbuild/html/index.html.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.

Printed in USA C07-733328-00 11/14