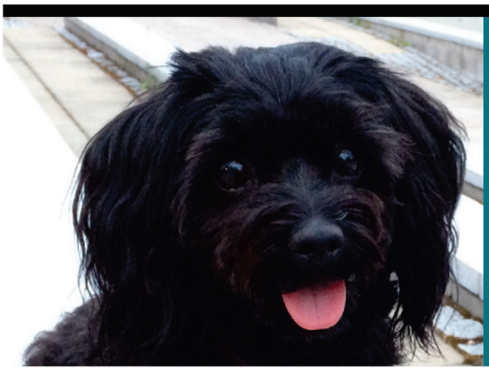
The Network Times
Handbook Series – Part IV.

Object-Based Approach to Cisco ACI

The Logic Behind the Application Centric Infrastructure



Fabric Access Policies, Tenants, VRFs, Bridge-Domains, EPGs, and Contracts are basic building-blocks (Objects) of Cisco ACI. This book translates the GUI-based configurations into JSON data-format in order to make it easier to understand the relationship between objects and how the APIC configuration can be build using REST API calls.

Toni Pasanen, CCIE No. 28158

Object-Based Approach to Cisco ACI

The Logic Behind the Application Centric Infrastructure

Toni Pasanen, CCIE 28158

First edition - 17 December 2020

About the Author:

Toni Pasanen. CCIE No. 28158 (RS), Distinguished Engineer at Fujitsu Finland. Toni started his IT-carrier in 1998 at Tieto, where he worked as a Service Desk Specialist moving via the LAN team to the Data Center team as a 3rd. Level Network Specialist. Toni joined Teleware (Cisco Learning partner) in 2004, where he spent two years teaching network technologies focusing on routing/switching and MPLS technologies. Toni joined Tieto again in 2006, where he spent the next six years as a Network Architect before joining Fujitsu. In his current role, Toni works closely with customers helping them in selecting the right network solutions not only from the technology perspective but also from the business perspective. He is also the author of books "*Virtual Extensible LAN – VXLAN: The Practical Guide to Understand VXLAN Solution - 2019*", "*LISP Control-Plane in Campus Fabric - 2020*", and *"VXLAN Fabric with BGP EVPN Control-Plane – 2020"*.

About This Book

First of all, I wrote this book for myself because I wanted to understand the logic behind the Cisco Application Centric Infrastructure (ACI) solution better. Fabric Access Policies, Tenants, VRFs, Bridge-Domains, EPGs, and Contracts are basic building-blocks (Objects) of Cisco ACI. This book translates the GUI-based configurations into JSON data-format to make it easier to understand the relationship between objects and how the APIC configuration can be build using REST API calls. The book is meant to be neither a design guide nor a best practice guide. Instead, it should give readers a clear idea of ACI logic and lower the learning curve when you step into the ACI world. There are also a couple of CLI examples but not many. The reason why I left out the CLI –based configuration is not that I don't see it as a useful tool for managing ACI. I think that it is a very powerful troubleshooting tool. However, I have used the ACI simulator and its CLI is very limited. I am running ACI Simulator as VM in ESXi host but it can also be used tested in Cisco DevNet sandbox:
https://devnetsandbox.cisco.com/RM/Topology. ACI Simulator is available as an always-on or reserved mode. The reserved ACI Simulator also includes a Windows client with Postman REST API Client so you can also test to do the configuration with it. I left out Multi-Site and Multi-Pod chapters as well as layer 4-7 integration because I wanted to focus on the ACI basics and keep the number of pages in the book reasonably small.

Disclaimers

The content of this book is based on the author's own experience and testing results. This book is meant to be neither design nor an implementation guide. After reading this book, readers should do their technology validation before using it in a production environment.

Table of Contents

Book Structure

Chapter 1 – Initial Fabric Setup:

This chapter discusses the basic fabric installation steps. The first step is to configure APIC and define the IP addressing scheme used within the fabric (Inter-Switch, TEPs, Mgmt-Net/Gw, BD Mcast. The next step is to configure the Date and Time settings (timezone and NTP Servers). The last section introduces how to define the configuration back servers and how to schedule backups. The configuration is done by using APIC GUI but they are also mapped into APIC Management Information Model (MIM). Also, the JSON body generated by GUI is discussed shortly.

Chapter 2 – Fabric Access Policies:

The focus is to explain what is needed to attach hosts to the fabric access port. It starts by explains how to configure VLAN Pool and attach it to the domain. Then it discusses Interface policies (speed, CDP/LLDP, and so forth) and how they are grouped under the Interface Policy Group which eventually is attached to the set of physical interfaces and bind into INterface Profile. This chapter also explains how the VLAN settings are combined with the physical Interface by using Attachable Access Entity Point. Also, this chapter introduces how to define switch policies like STP and Netflow global settings and how these are bind to a set of switches defined under the Switch Profile. The last section introduces the APIC built-in Object Store called Visore and APIC Management Information Model. As in every chapter, configurations are explained by using Objects and their dependencies which are then reflected GUI configuration and REST API call using JSON data format. The REST API client Postman and the APIC built-in API Inspector are also introduced.

Chapter 3 - Tenant Networking:

VRF and Bridge Domain: This chapter explains the networking structure within Tenant. It introduces Virtual Routing and Forwarding (VRF) instance and its relationship to Bridge-Domain. The naming structure of classes and objects are also discussed.

Chapter 4 - Application Profile:

This chapter continues where chapter 3 ends. The focus is to explain the concept of an Application Profile and those relationships with End Point Groups (EPG). It also explains how EPG are attached to Bridge-Domain.

Chapter 5 - Filters, Contracts, and Subject:

The previous chapter of the book discusses EPGs. This chapter explains how to filter entries are grouped under one filter object and how it is used to create the Contract which eventually is added between EPGs.

Chapter 6 - Binding EPGs to Interface and Domain:

This chapter finalizes the process of building an application environment by binding the logical EPG into a physical Interface.

Chapter 7 – Automatization Basics:

This short chapter shows how the application environment including networking, application profiles, EPGs, contracts, and interface binding can be done by one REST API call.

Chapter 8 – Basic Object Monitoring:

This chapter shows some methods to take a look at the object dependencies using APC GUI. Also, it introduces how the configuration can be verified by using APIC CLI.

Chapter 9 - External L2 Connection (L2OUT):

This chapter introduces two methods for creating a switched external connection. It starts by explaining how EPGs can be extended to traditional DC networks in the case where there is no need to filter traffic between hosts in the same segment located in ACI and traditional DC. This mostly applies to the transition project. Then it explains how we can control traffic between hosts in the same segment but locate in ACI and traditional DC by using extending Bridge-Domain into traditional DC and by creating dedicated external EPG. Chapter one explains the concept of domain and introduced the Physical Domain, this chapter discusses Layer 2 External Domain (L2OUT).

Chapter 10 - External L2 Connection (L3OUT):

This chapter explains how we implement the internal BGP process in ACI and how to build external BGP peering. It also explains the purpose of external EPG and how to allow traffic flows between internal and external EPGs.

Chapter 6: Binding EPG to Interface and Domain

Introduction

We have so far created an Application Profile *ap-NWKT* where we have two EPGs; *epg-AppSrv-EPG* and *epg-WebSrv-EPG*, in which Inter-EPG traffic is filtered with a contract *brc-Web-to-App_SQL*.

This section explains how to configure interface eth1/1 on Leaf-101 for WebSrv VM hosted by ESXi-Host 1 and interface eth1/1 on Leaf-102 for AppSrv VM hosted by ESXi-Host 2. In figure 6-1, we have EPG objects *epg-AppSrv-EPG* and *epg-WebSrv-EPG* under the class *fvAEPg (Application Profile)*. We are binding both EPGs to object *Phys-Standalone-ESXi_PHY* with a relationship *fvRsDomAtt* (9). By doing this we allow VLANs from 300 – 399 to be assigned to interfaces to which we are going to connect our servers. We create an object *rspathAtt-[topology/pod-1/paths-101/pathep-[eth1/1]* for WebSrv connection and bind it to object *epg-WebSrv-EPG* via relationship *FvRsPathAtt*. We also create an object *rspathAtt-[topology/pod-1/paths-102/pathep-[eth1/1]* for AppSrv and bind it to object *epg-AppSrv-EPG* via relationship *FvRsPathAtt*. At this phase, we are allowing WebSrv and AppSrv to communicate based on filters bind to contract *brc-Web-to-App_SQL* between them.

End Point Groups (EPGs) used in ACI and *Scalable Group Tag* (SGT) used in SD-Access have the same role, they give a location independent identity for traffic that is not based on the location of the source IP address, and which is carried within the data frame (VXLAN-GPE). In another word, the identity of users/traffic is segregated from the network topology. The difference lays in the filtering method, ACI uses Contracts while SD-Access uses a traffic matrix. However, both methods are eventually nothing more than a set of access-lists.

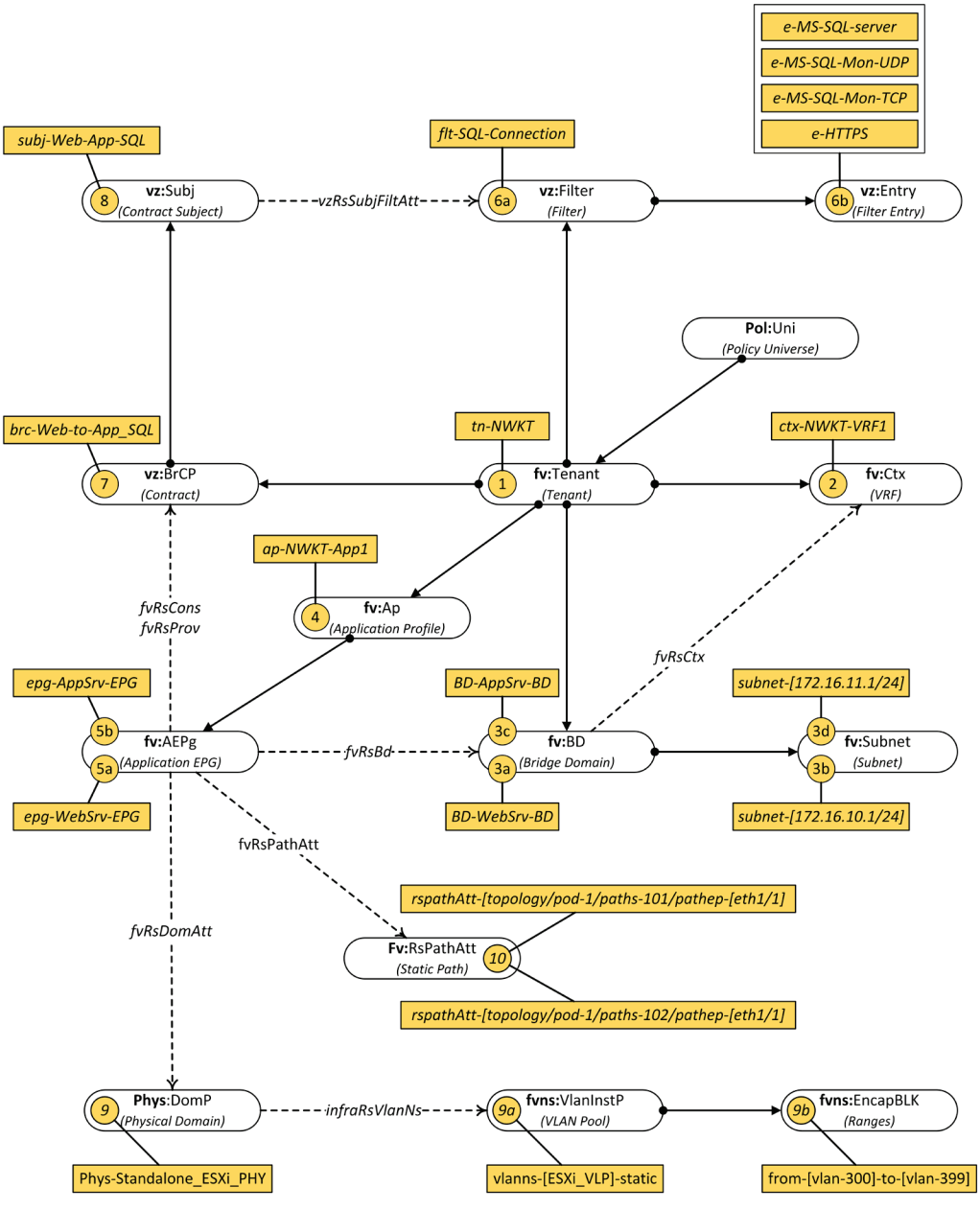


Figure 6-1: *APIC Management Information Model*

Figure 6-2 merges objects and their dependencies to the more traditional looking logical network diagram.

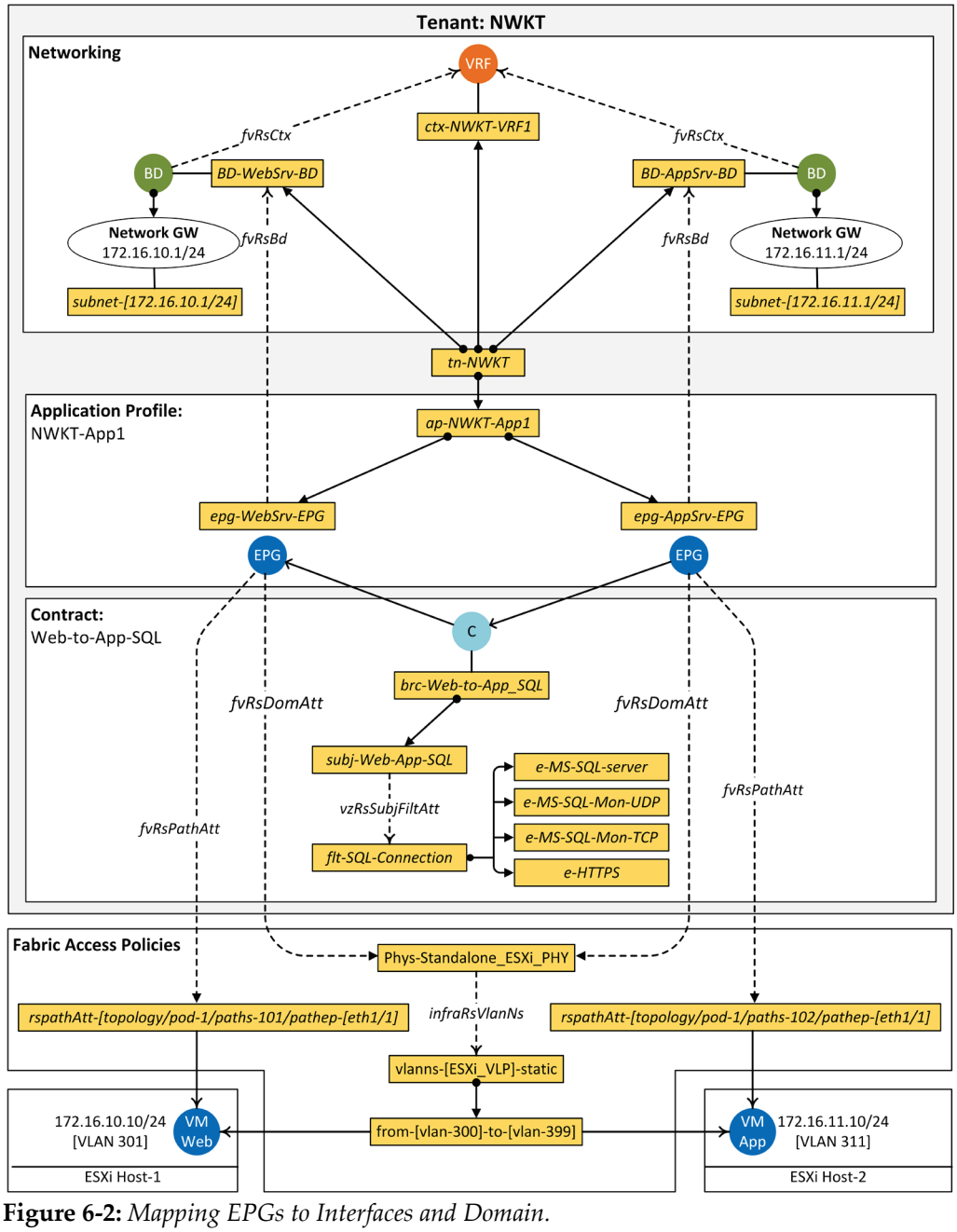


Figure 6-2: *Mapping EPGs to Interfaces and Domain.*

Binding Domain to EPG with REST API

Examples 6-1 and 6-2 shows the URL of a REST API call and the message body in JSON format for attaching an object *phys-Standalone_ESXi_PHY* to EPGs.

```
https://192.168.10.65/api/node/mo/uni/tn-NWKT/ap-NWKT-App1/epg-AppSrv-
EPG.json
```

```
{
    "fvRsDomAtt": {
        "attributes": {
            "resImedcy": "immediate",
            "tDn": "uni/phys-Standalone_ESXi_PHY",
            "status": "created"
        },
        "children": []
    }
}
```

Example 6-1: *Mapping EPG epg-AppSrv-EPG to Physical Domain.*

```
https://192.168.10.65/api/node/mo/uni/tn-NWKT/ap-NWKT-App1/epg-WebSrv-
EPG.json
```

```
{
    "fvRsDomAtt": {
        "attributes": {
            "resImedcy": "immediate",
            "tDn": "uni/phys-Standalone_ESXi_PHY",
            "status": "created"
        },
        "children": []
    }
}
```

Example 6-2: *Mapping EPG epg-WebSrv-EPG to Physical Domain.*

Binding Interface and VLAN Id to EPG with REST API

Examples 6-3 shows the URL of a REST API call and the message body in JSON for binding an object *epg-WebSrv-EPG* to object *rspathAtt-[topology/pod-1/paths-101/pathep-[eth1/1]]* which defies the leaf switch (Leaf-101) and its interface (eth1/1) and VLAN Id (301) used in 802.1Q tag.

```
https://192.168.10.65/api/node/mo/uni/tn-NWKT/ap-NWKT-App1/epg-WebSrv-EPG/rspathAtt-
[topology/pod-1/paths-101/pathep-[eth1/1]].json
```


```
{
    "fvRsPathAtt": {
        "attributes": {
            "dn": "uni/tn-NWKT/ap-NWKT-App1/epg-WebSrv-EPG/rspathAtt-[topology/pod-
1/paths-101/pathep-[eth1/1]]",
            "encap": "vlan-301",
            "tDn": "topology/pod-1/paths-101/pathep-[eth1/1]",
            "rn": "rspathAtt-[topology/pod-1/paths-101/pathep-[eth1/1]]",
            "status": "created"
        },
        "children": []
    }
}
```


Example 6-3: *Mapping EPG epg-WebSrv-EPG to Physical Domain.*

Examples 6-4 shows the URL of a REST API call and the message body in JSON for binding an object *epg-AppSrv-EPG* to object *rspathAtt-[topology/pod-1/paths-102/pathep-[eth1/1]]* which defies the leaf switch (Leaf-102) and its interface (eth1/1) and VLAN Id (311) used in 802.1Q tag.

```
https://192.168.10.65/api/node/mo/uni/tn-NWKT/ap-NWKT-App1/epg-AppSrv-EPG/rspathAtt-
[topology/pod-1/paths-102/pathep-[eth1/1]].json
```


```
{
    "fvRsPathAtt": {
        "attributes": {
            "dn": "uni/tn-NWKT/ap-NWKT-App1/epg-AppSrv-EPG/rspathAtt-[topology/pod-
1/paths-102/pathep-[eth1/1]]",
            "encap": "vlan-311",
            "tDn": "topology/pod-1/paths-102/pathep-[eth1/1]",
            "rn": "rspathAtt-[topology/pod-1/paths-102/pathep-[eth1/1]]",
            "status": "created"
        },
        "children": []
    }
}
```


Example 6-4: *Mapping EPG epg-WebSrv-EPG to Physical Domain.*

Binding Domain to EPG with APIC GUI

Navigate to the *Domain (VMs and Bare-Metal)* sub-folder under the *AppSrv-EPG* folder. Click the Tools icon, and select *Add Physical Domain Association*.

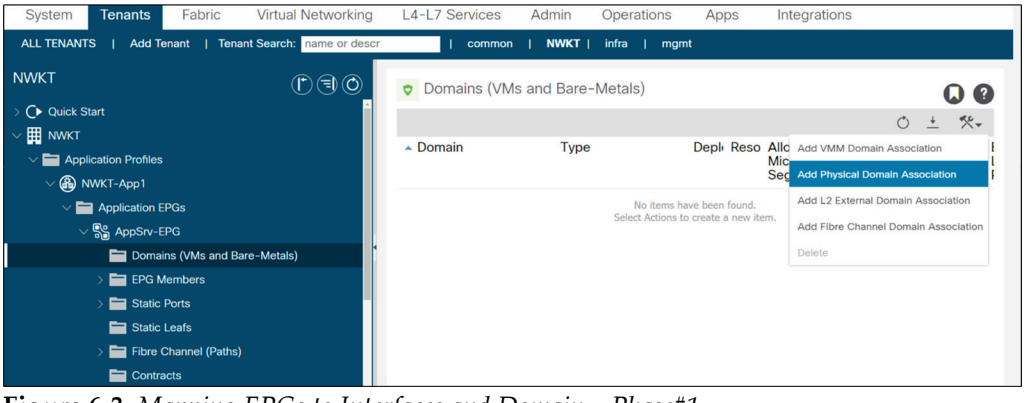


Figure 6-3: *Mapping EPGs to Interfaces and Domain – Phase#1.*

Select *Standalone_ESXi_PHY* from the *Physical Domain Profile* drop-down menu. This creates a relationship *fvRsDomAtt* between objects. Click the Expand icon next to the drop-down menu of *Physical Domain Profile* in order to check what VLAN pool is mapped into the domain.

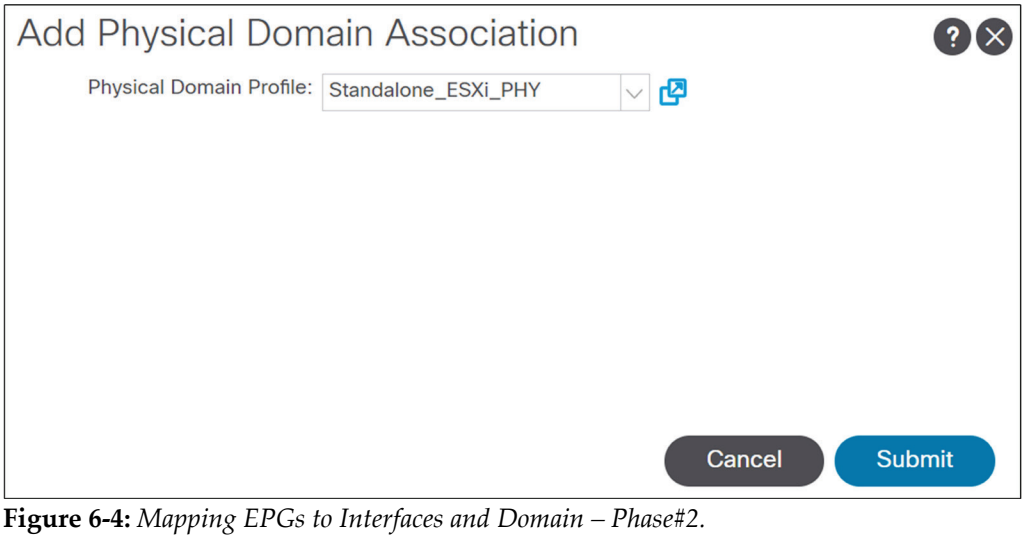


Figure 6-4: *Mapping EPGs to Interfaces and Domain – Phase#2.*

Click the Expand icon next to the drop-down menu of *VLAN Pools* in order to check what VLAN pool is mapped into the domain.

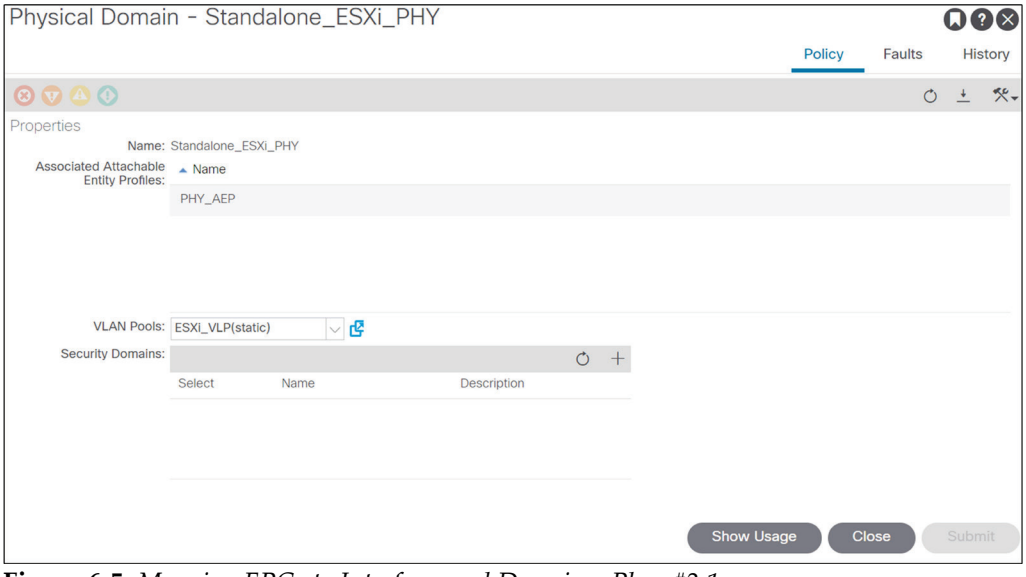


Figure 6-5: *Mapping EPGs to Interfaces and Domain – Phase#2.1.*

We can see that the VLAN Range is 300 – 399.

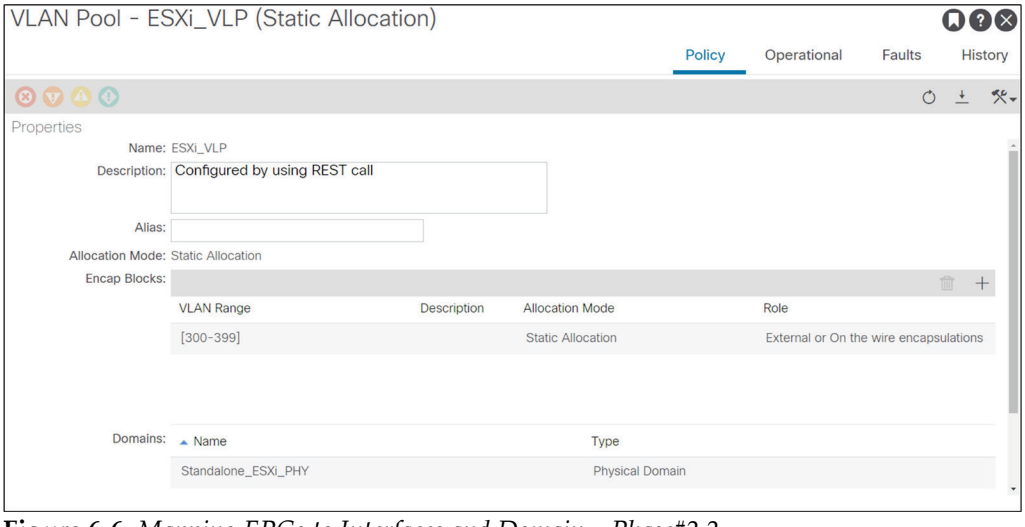


Figure 6-6: *Mapping EPGs to Interfaces and Domain – Phase#2.2.*

Figure 6-7 shows that the physical domain *Standalone_ESXi_PHY* is attached to EPG *AppSrv-EPG*.

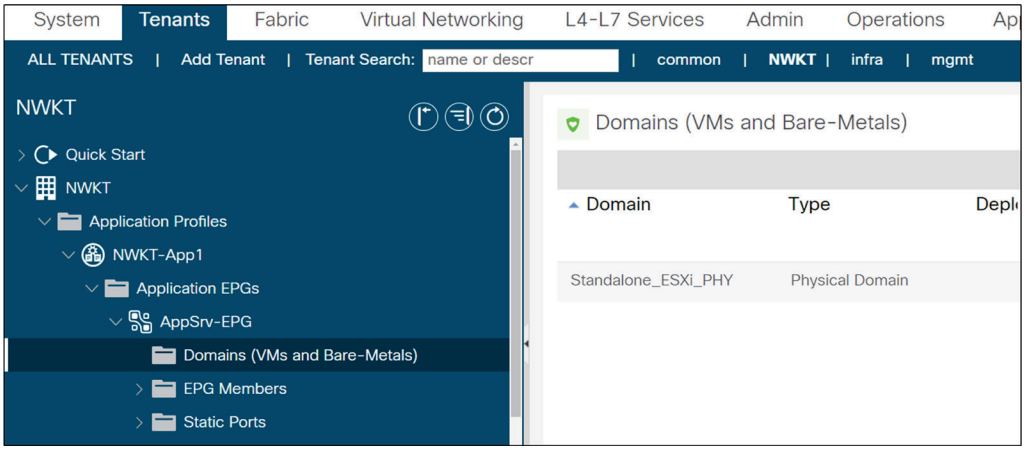


Figure 6-7: *Mapping EPGs to Interfaces and Domain – Verification.*

Binding Interface and VLAN Id to EPG with APIC GUI

Navigate to the *Static Ports* sub-folder under the *AppSrv-EPG* folder. Click the Tools icon, and select *Deploy Static EPG on PC, VPC, or Interface*.

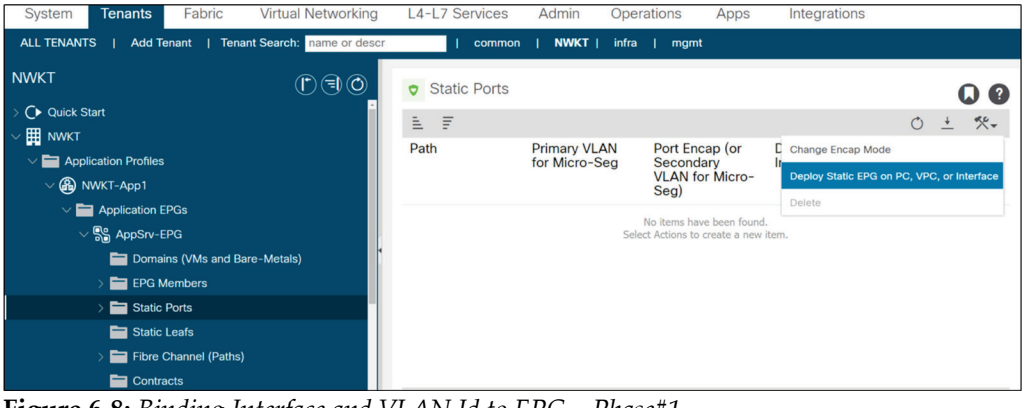


Figure 6-8: *Binding Interface and VLAN Id to EPG – Phase#1.*

Select *Port* from the *Path Type* options. Select *Leaf-102 (Node-102)* from the *Node:* drop-down menu and select *eth1/1* from the *Path:* drop-down menu. Type VLAN-Id 311 to *Integer Value* field under the *Port Encap* Key. Click the Submit button.

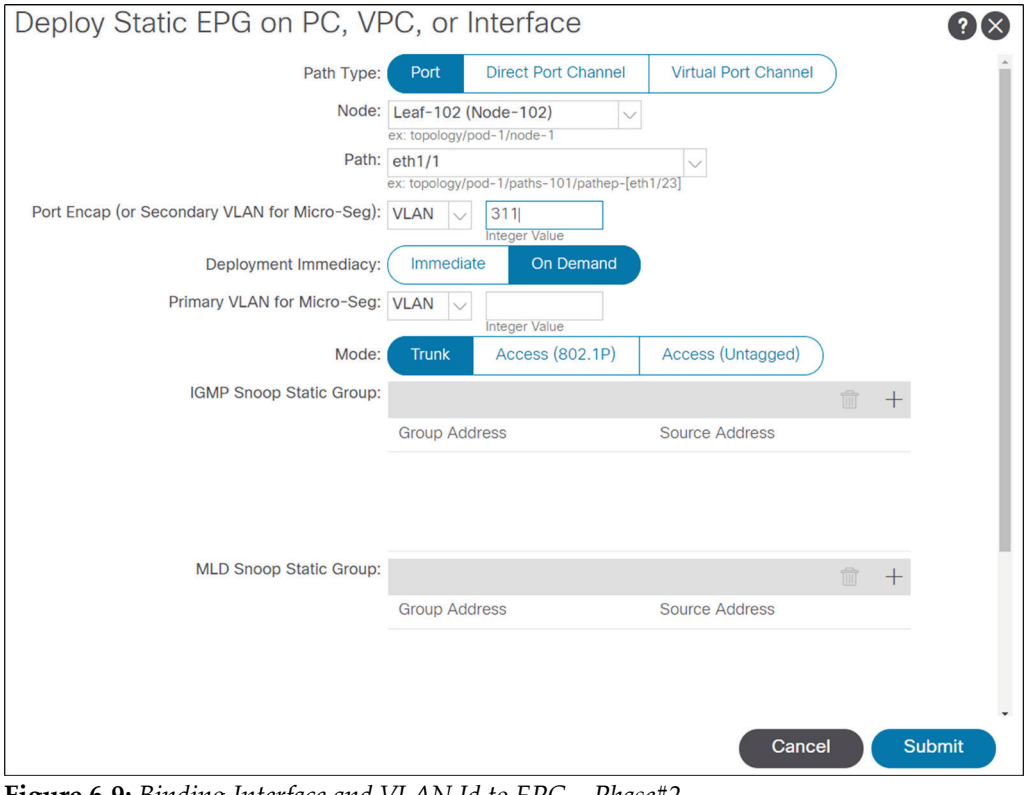


Figure 6-9: *Binding Interface and VLAN Id to EPG – Phase#2.*

Figure 6-10 shows that the object *epg-AppSrv-EPG* is now bound to eth1/1 of Leaf-102 with VLAN Id 311.

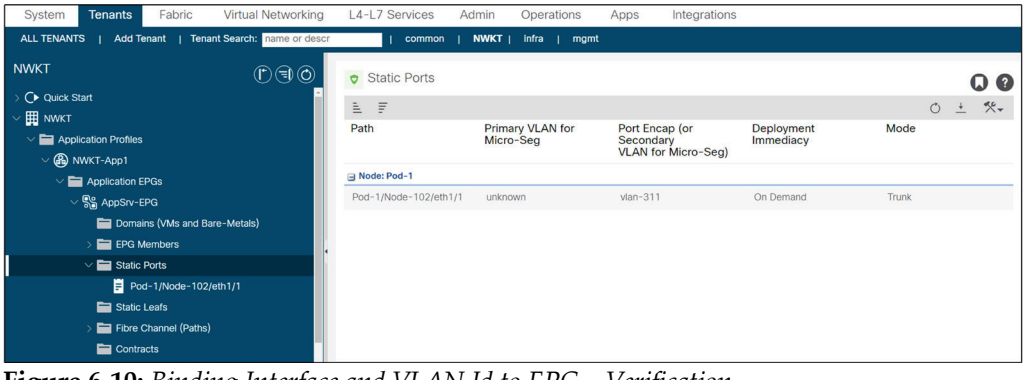


Figure 6-10: *Binding Interface and VLAN Id to EPG – Verification.*

Figure 6-11 summarizes what we have done. The information can be seen by moving the pointer above the object symbol. The left-hand side Bare-Metal Server belongs to domain *Standalone_ESXi_PHY* and it is attached to interface 1/1 of Leaf-101. The server is assigned to EPG *WebSrv-EPG* which belongs to Application Profile *NWKT-App1*. The EPG belongs to *BD WebSrv-BD* that is part of the VRF *NWKT-VRF1*. The right-hand side Bare-Metal Server also belongs to domain *Standalone_ESXi_PHY* and it is attached to interface 1/1 of Leaf-102. The server is assigned to EPG *AppSrv-EPG* which belongs to Application Profile *NWKT-App1*. The EPG belongs to BD *AppSrv-BD* that is part of the VRF *NWKT-VRF1*. There is also a Contract *Web-to-App_SQL* between these two EPGs which defines the policy between servers attached to EPGs. EPG *WebSrv-EPG* is a *Contract Provider* and EPG *WebSrv-EPG* is a *Contract Consumer*.

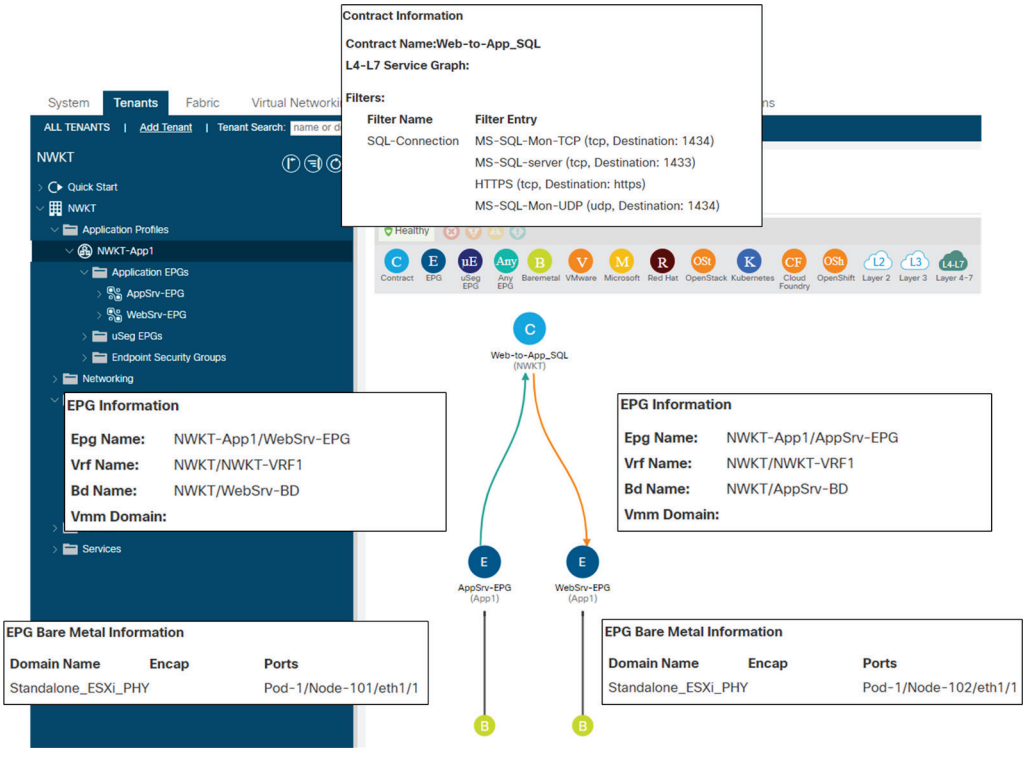
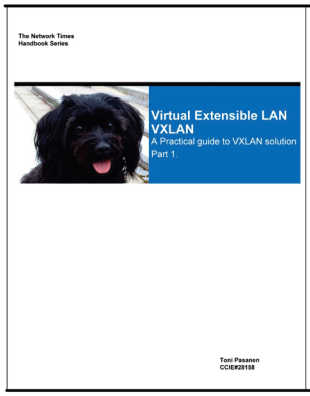


Figure 6-11: *Binding Interface and VLAN Id to EPG – Verification.*

This book is part of **The Network Times** Handbook Series.



Virtual Extensible LAN (VXLAN):

A Practical guide to VXLAN solution

Amazon

Pages: 368
ISBN-10 : 170326536X
ISBN-13 : 978-1703265361
Paperback: 19:90€/21:00$
Kindle Edition: 7:70$

Leanpub.com

PDF Edition: 7:99$

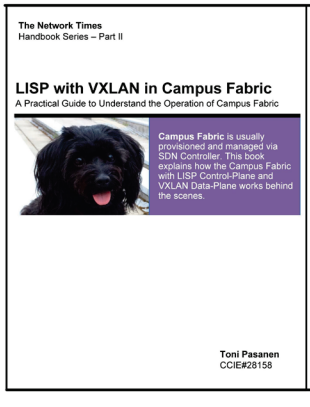
VXLAN Fabric with BGP EVPN Control-Plane: Design Considerations

Amazon

Pages: 271
ISBN-13 : 979-8683023690
Paperback: 13:30€/15:00$
Kindle Edition: 6:60$

Leanpub.com

PDF Edition: 5:99$



LISP Control-Plane in Campus Fabric:

A Practical Guide to Understand the Operation of Campus

Amazon

Paperback : 264 pages
ISBN-13 : 979-8615059186
Paperback: 14:54€/15:00$
Kindle Edition: 6:60$
Price: 14:54€

Leanpub.com

PDF Edition: 4:99$