Cisco Application Centric Infrastructure

Design Guide
Contents

Introduction ................................................................................................................. 4
   Benefits .................................................................................................................. 4
   Topology and Design Principles ........................................................................... 5
   Segmentation: Endpoint Groups ........................................................................... 6
   Extension to Virtualized Servers and VLAN and VxLAN Normalization ........... 6
   Scalable Forwarding Model: No Gateway Bottleneck ......................................... 7
   Full Visibility of Virtual Machine Location in Network Infrastructure ............... 8
   Management Model ............................................................................................... 10
   Management Tools ................................................................................................. 13

Hardware and Software Support ................................................................................. 13
   Configuration Considerations for a Modular Spine ............................................. 14
   Cisco APIC and NX-OS Software ......................................................................... 14

Cisco ACI Model for Defining Workload Connectivity .............................................. 15
   Policy Definition .................................................................................................... 15
      Workload Classification ...................................................................................... 15
      Endpoint Groups and Contracts ....................................................................... 16
      Any-to-Any Communication .............................................................................. 17
      Application Policy Enforcement ....................................................................... 18
      Rendering to VRF Instances, VLANs and so on ............................................ 18
   Application Network Profile .................................................................................. 18
   Routing and Switching in the Policy Model: Network and Bridge Domain ........... 19
   Using REST to Program the Network ................................................................... 20
      Examples of Tools for Sending REST Calls .................................................... 21
      REST Syntax in Cisco ACI ................................................................................ 24
      Modeling Tenants in XML ............................................................................... 25
      Defining the Relationship Amongst EPGs (Providers and Consumers).......... 26
      A simple Any-to-Any Policy .............................................................................. 26

Cisco ACI Architecture Design Principles .................................................................. 27
   Cisco APIC .............................................................................................................. 27
      Central Point of Management but no Centralized Control Plane .................... 27
      Controller Availability ....................................................................................... 28
      Fabric ................................................................................................................. 28
      Zero Touch Provisioning .................................................................................... 29
      What is an Integrated Overlay ......................................................................... 29
      Routed Design with VxLAN Overlays .............................................................. 30
      Leaf-and-Spine Design ...................................................................................... 30
      Mapping Database ............................................................................................. 31
      Unknown Unicast and Broadcast ..................................................................... 31
   Access Layer ........................................................................................................... 32
      Use of VLANs as a Segmentation Mechanism ................................................ 32
      VLANs and VxLANs Namespaces .................................................................... 33
      Concept of Domain ............................................................................................. 34
      Policy-based Configuration of Access Ports .................................................... 35
      PortChannels ....................................................................................................... 39
      Virtual PortChannels .......................................................................................... 41

Physical Topology ......................................................................................................... 43
   Cisco APIC Design Considerations ........................................................................ 44
   Spine Design Considerations ............................................................................... 45
   Leaf design considerations ..................................................................................... 45

Multi-tenancy considerations ....................................................................................... 45

Initial Configuration Steps .......................................................................................... 47
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Management</td>
<td>47</td>
</tr>
<tr>
<td>Inband Management</td>
<td>47</td>
</tr>
<tr>
<td>Out-of-Band Management</td>
<td>49</td>
</tr>
<tr>
<td>Fabric Bring up Considerations</td>
<td>52</td>
</tr>
<tr>
<td>Preprovisioning Switch Profiles for Each Leaf</td>
<td>53</td>
</tr>
<tr>
<td>Preprovisioning vPC Domains for Each Leaf Pair</td>
<td>53</td>
</tr>
<tr>
<td>Preprovisioning Interface Policies</td>
<td>54</td>
</tr>
<tr>
<td>Preprovision Interface Policy Groups</td>
<td>55</td>
</tr>
<tr>
<td>Preprovisioning Virtual Machine Mobility Domains for Multihypervisor Deployments</td>
<td>57</td>
</tr>
<tr>
<td>VMM Domain</td>
<td>57</td>
</tr>
<tr>
<td>AEP for Virtualized Servers Connectivity</td>
<td>59</td>
</tr>
<tr>
<td>Preprovisioning Physical Domains for Each Tenant</td>
<td>60</td>
</tr>
<tr>
<td>Preprovisioning an Outside Physical Domain per Tenant</td>
<td>62</td>
</tr>
<tr>
<td>Automation of Preprovisioning Processes</td>
<td>62</td>
</tr>
<tr>
<td>Configuring a Virtual Topology</td>
<td>62</td>
</tr>
<tr>
<td>Bridge Domain</td>
<td>63</td>
</tr>
<tr>
<td>Hardware Proxy</td>
<td>63</td>
</tr>
<tr>
<td>Flooding Mode</td>
<td>63</td>
</tr>
<tr>
<td>fvCtx</td>
<td>63</td>
</tr>
<tr>
<td>Endpoint Connectivity</td>
<td>64</td>
</tr>
<tr>
<td>Connecting a Physical Server</td>
<td>64</td>
</tr>
<tr>
<td>Connecting a Virtual Server</td>
<td>65</td>
</tr>
<tr>
<td>EPG, Bridge Domain and Routing Instance</td>
<td>65</td>
</tr>
<tr>
<td>External Connectivity</td>
<td>66</td>
</tr>
<tr>
<td>Redundancy</td>
<td>67</td>
</tr>
<tr>
<td>Configuring Virtual and Physical Servers Connectivity</td>
<td>68</td>
</tr>
<tr>
<td>Teaming Considerations for Physical Servers</td>
<td>68</td>
</tr>
<tr>
<td>Connecting Virtualized Servers</td>
<td>71</td>
</tr>
<tr>
<td>Hypervisor without a Virtual Machine Manager</td>
<td>71</td>
</tr>
<tr>
<td>Virtual Distributed Switch</td>
<td>71</td>
</tr>
<tr>
<td>Virtual Distributed Switch Across Multiple Layer 2 Hops</td>
<td>74</td>
</tr>
<tr>
<td>Virtual Distributed Switch Uplink Configuration</td>
<td>75</td>
</tr>
<tr>
<td>Conclusion</td>
<td>75</td>
</tr>
<tr>
<td>For More Information</td>
<td>75</td>
</tr>
</tbody>
</table>
Introduction

Cisco® Application Centric Infrastructure (ACI) technology enables you to integrate virtual and physical workloads in an easy-to-use and highly programmable multi-hypervisors fabric that is excellent for any multi-service or cloud datacenter. The Cisco ACI fabric consists of discrete components that operate as routers and switches but is provisioned and monitored as a single entity. The operation is like a single switch and router that provides advanced traffic optimization, security, and telemetry functions, stitching together virtual and physical workloads.

This document describes how to implement a fabric like the one depicted in Figure 1.

Figure 1. Cisco ACI Fabric

The Cisco ACI software development team is guided by the following principles:

- Agile development
- Release early and release often

The user deploying Cisco ACI will most likely adopt them too because with Cisco ACI organizations can create and decommission networks in real time.

Benefits

The main purpose of a datacenter fabric is to move traffic from physical and virtualized servers, bring it in the best possible way to its destination and while doing so apply meaningful services such as:

- Traffic optimization that improves application performance
- Telemetry services that go beyond classic port counters
- Overall health monitoring for what constitutes an application
- Applying security rules embedded with forwarding

The main benefits of using a Cisco ACI fabric are the following:

- Single point of provisioning either via GUI or via REST API
- Connectivity for physical and virtual workloads with complete visibility on virtual machine traffic
- Hypervisors compatibility and integration without the need to add software to the hypervisor
- Ease (and speed) of deployment
- Simplicity of automation
- Multitenancy (network slicing)
- Capability to create portable configuration templates
- Hardware-based security
- Elimination of flooding from the fabric
- Ease of mapping application architectures into the networking configuration
- Capability to insert and automate firewall, load balancers and other L4-7 services
- Intuitive and easy configuration process

**Topology and Design Principles**

With Cisco ACI the topology is prescriptive and automatically enforced with autodiscovery, zero-touch provisioning and a built-in cable-plan. The Cisco ACI topology consists of a set of leaf devices connected to a set of spine devices in a full bipartite graph, or Clos architecture using 40 Gigabit links.

All leaf devices are connected to all spine devices and all spine devices are connected to all leaf devices, links between spines devices or between leaf devices are disabled if present.

Leafs devices can connect to any device, and they are the place at which policies are enforced. Leafs devices also provide the capability to route and to bridge to external network infrastructures (campus, WAN, Multiprotocol Label Switching [MPLS] virtual private network [VPN] cloud and so on). In this case they are sometimes referred to as border leaf devices.

For example, the following endpoints that can be connected to leaf devices:

- Virtualized servers
- Bare-metal servers
- Mainframes
- L4-7 services nodes
- IP Storage devices
- Switches
- Routers

Spines devices constitute the backbone of the fabric and they provide the mapping database function. The section titled “Fabric” under “ACI Architecture Design Principles” provides additional details.

The Cisco ACI Controller is called Cisco® Application Policy Infrastructure Controller (APIC) and it is the management and control point for fabric-wide configurations. The controller distributes policies in the fabric and to virtual switches supporting OpFlex, although virtual switches do not need to support OpFlex in order to be integrated with the fabric.

Cisco APIC is not in the datapath; in fact, it can be disconnected, reconnected and moved to a different port while the fabric is processing traffic without any impact on data traffic. The datapath runs routing protocols on a per-node (spine or leaf) basis without any involvement of the controller. The controller is completely separated from the data path and forwarding decisions.
Cisco APIC servers can be connected on any ports of the fabric leaf switches with 10 Gigabit Ethernet interfaces, and it will automatically discover the leaf devices attached to it, and subsequently the rest of the fabric. You do not need to configure anything on the leaf or spine devices for the discovery to occur.

**Segmentation: Endpoint Groups**

You can think of the Cisco ACI fabric logically as a giant switch/router that is also a context in which to indicate application connectivity relationships according to the policy model. Traditionally, segmentation has been performed with VLANs that incidentally are also broadcast and flooding domains. With Cisco ACI the two concepts are decoupled (Figure 2).

**Figure 2.** Cisco ACI as a Telemetry Device

Bridge domains are the elements that provide a flooding and broadcast domain when required. When flooding is not required Bridge Domains simply act as a container for one or more subnets. Endpoint Groups (EPGs), which are like port-groups or port-profiles, provide segmentation among workloads. EPGs contain one or more virtual and physical servers that require similar policy and connectivity. Example of this would be application tiers, development stages or security zones.

The Cisco ACI fabric allows you to define the communication path among EPGs, just as you do by stitching virtual lines between VLANs using IP routing and access control lists (ACLs).

**Extension to Virtualized Servers and VLAN and VxLAN Normalization**

Segmentation extends from the fabric to the virtualized servers so that the Cisco ACI fabric can provide meaningful services (such as traffic load balancing, segmentation, filtering, traffic insertion, and monitoring) to workloads.

Figure 3 shows that the fabric provides a distributed policy consisting of two EPGs connected by a firewall. Each EPG in the picture can belong to one or more subnets.
Virtual workloads are connected into port-groups that are synchronized with the EPGs and send traffic tagged with VLAN IDs or Virtual Extensible LANs (VxLANs) VNIDs to the leaf devices. VLANs and VxLANs are dynamically generated and the user does not maintain them. They have local significance to the leaf switch and VMM domain, and they exclusively serve the purpose of segmenting traffic on the link between the server and the leaf.

**Figure 3. The Fabric Provides Distributed Policy**

Policy enforcement (insertion of workloads into the correct EPG and stitching them to the correct destination EPG and security, QoS, logging, etc…) is performed at the leaf.

**Scalable Forwarding Model: No Gateway Bottleneck**

With Cisco ACI all workloads are equal, regardless of whether they are virtual or physical workloads. The VLANs or VxLANs used by virtualized workloads are remapped to a bridge domain as necessary. As a result, communication between virtual and physical workloads doesn’t go through a gateway bottleneck, but directly along the shortest path to where the workload is (Figure 4).

In addition to this the default gateway is on the leaf switches both for virtual and for physical workloads.
Figure 4. Workloads are Mapped to Bridge Domains

Full Visibility of Virtual Machine Location in Network Infrastructure
Cisco ACI uses Cisco Discovery Protocol (CDP) and Link Layer Discovery Protocol (LLDP) information to discover the presence of virtualized hosts and to deploy policies only to the ports on which a virtualized host is present.

The information exchanged between Cisco ACI and the virtual machine managers enables Cisco ACI to display the exact location of the virtual machines are in the fabric, the physical ports to which they are connected and so on (Figure 5 and 6).

Should virtual machines move from one server to a different one within the same leaf or across multiple leaf switches the system preserves virtual machine visibility.
Figure 5. Visibility of Virtual Machines in the Network Infrastructure

Figure 6. Location of Virtual Machines

<table>
<thead>
<tr>
<th>LEARNING SOURCE</th>
<th>HOSTING SERVER</th>
<th>REPORTING CONTROLLER TYPE</th>
<th>REPORTING CONTROLLER NAME</th>
<th>INTERFACE</th>
</tr>
</thead>
<tbody>
<tr>
<td>learned vmm</td>
<td>172.28.80.71</td>
<td>vCenter</td>
<td>vCenter</td>
<td>Node-1/192.168.0.11, Nod</td>
</tr>
<tr>
<td>vmm</td>
<td>172.28.80.71</td>
<td>vCenter</td>
<td>vCenter</td>
<td>Node-1/192.168.0.11</td>
</tr>
<tr>
<td>learned vmm</td>
<td>172.28.80.72</td>
<td>vCenter</td>
<td>vCenter</td>
<td>Node-1/192.168.0.11, Nod</td>
</tr>
<tr>
<td>learned vmm</td>
<td>172.28.80.71</td>
<td>vCenter</td>
<td>vCenter</td>
<td>Node-1/192.168.0.11, Nod</td>
</tr>
</tbody>
</table>
Management Model

Among its many innovations, Cisco ACI is changing network management from a traditional feature-by-feature, link-by-link approach to a declarative model, in which the controller relies on each node to render the declared desired endstate.

The user configures policies on Cisco APIC and Cisco ACI propagates the policy configuration through the OpFlex protocol to all the leaf devices in the fabric (Figure 7).

**Figure 7.** Cisco ACI Propagates Policies to All the Leaf Devices in the Fabric

If the server and the software switching on the server support OpFlex the policy can also be applied within the server.

Each networking element (physical or virtual) then renders the policies according to the local capabilities (Figure 8).
The user can define configurations on the Cisco APIC controller in several ways (Figure 9):

- Using the easy-to-use Graphical User Interface running on the same appliance that provides the controller function
- Using representational state transfer (REST) calls with intuitive XML or JavaScript Object Notation (JSON) formatted payloads that are sent to the Cisco APIC: these can be sent in many ways, using tools such as Google's POSTMAN or Python scripts that send REST calls
- Using a custom-built Graphical User Interface that sends REST calls
- Using the command-line interface (CLI) to navigate the object model from the Cisco APIC
- Using Python scripts that use the associated Cisco ACI libraries
Even if spine and leaf devices are receiving the policy configurations from the controller, you can still connect to each device through the console or the management (mgmt0) port and use the well-known Cisco NX-OS Software CLI to monitor how policies are rendered.

As Figure 10 shows, whatever you configure with REST calls is translated in Virtual Routing and Forwarding (VRF) instances, VLANs, VxLANs, routes, and so on, all familiar concepts to the network administrator.

Each tool has its areas of strength and weakness, most likely this is how different teams will use the tools:

- GUI: it is used mostly for the infrastructure administration and for monitoring and troubleshooting purposes. It is also used to generate templates.
- CLI on Cisco APIC: the main use is to create shell scripts and for troubleshooting
- POSTMAN and other REST tools: the main use is for testing and to define configurations to be automated
• Scripts based on XML, JSON REST calls: simple scripts for the operator for CLI like operations without the need to really understand Python

• True Python scripts: the main use case is to create comprehensive provisioning. Using the SDK provided with Cisco ACI can do this.

• PHP and Web pages with embedded REST calls: mostly used to create simple user interface for operators or IT customers.

• Advanced orchestration tools like Cisco ® Intelligent Automation for Cloud or UCS Director: for end-to-end provisioning of compute and network

Management Tools
With Cisco ACI Cisco embraces and accelerates the adoption of automation in the datacenter. As a result the tools that the networking team can use for configuration encompass the following ones:

• CLI as usual (each device in the fabric is accessible through Secure Shell [SSH], or the console and has the traditional Cisco NX-OS show commands)

• Intuitive Graphical User Interface that guides the user through the tasks of managing fabrics of various sizes

• REST calls with XML or JSON, which can be sent with various tools, such as POSTMAN

• Python scripts: with the libraries provided by Cisco or by simply using scripts that originate REST calls.

Configuration of the fabric with the new model is simple and fast and requires little new learning because the XML format is simple to read. For Python very little new knowledge needs to be acquired except that you need a script on your computer that converts XML files to REST calls.

Furthermore Cisco ACI is designed so that every network team can easily create a portal of its own to give different administrators specialized views of the fabric.

Hardware and Software Support
The solution described in this document requires the following components:

• Spine switches: The spine provides the mapping database function and connectivity among leaf switches. At the time of this writing these can be either the Cisco Nexus ® N9K-C9508 switch equipped with N9K-X9736PO linecards or fixed form-factor switches such as the Cisco Nexus N9K-C9336PO ACI spine switch. Spine switches provide high-density 40 Gigabit Ethernet connectivity between leaf switches. The Cisco Nexus 9336PO form factor is well suited for smaller deployments because it provides 36 ports of 40 Gigabit Ethernet. The Cisco Nexus 9508 provides 288 40 Gigabit Ethernet ports.

• Leaf switches: The leaf provides physical and server connectivity and policy enforcement. At the time of this writing, these can be fixed form factor switches such as the Cisco Nexus N9K-C9396PX, the N9K-C9396TX and N9K-C93128TX switches. The choice of leaf switches provides the option to use 10GBASE-T or Enhanced Small Form-Factor Pluggable (SFP+) connectivity to the servers. Leaf switches can be used in two modes: as standalone Cisco NX-OS devices, or as devices that are part of the Cisco ACI fabric (with an ACI version of the NX-OS Software).

• Cisco APIC: The controller is the point of configuration of policies and the place where statistics are archived and processed to provide visibility, telemetry, application health information and overall management of the fabric. Cisco APIC is a physical server appliance like a UCS ® C220 M3 rack server with two 10 GigabitEthernet interfaces that are meant to be connected to the leaf switches and with Gigabit...
Ethernet interfaces for out-of-band management. Two controller models are available: Cisco APIC-M and APIC-L.

- **40 Gigabit Ethernet cabling**: Leaf and spine switches can connect at 40 Gbps with multimode fiber by using the new Cisco 40Gbps short-reach (SR) bidirectional (BiDi) Quad SFP (QSFP) optics modules, which do not require new cabling. With these optics modules, you can connect equipment at distances up to 100 meters on OM3 cabling and up to 125 meters or more on OM4 cabling. Other QSFP options are also available for 40-Gbps links. For more information about 40 Gbps cabling options see:
- **Classic 10 Gigabit Ethernet cabling**: Cabling to the server with 10 Gigabit Ethernet can be implemented with SFP+ fiber or copper or with 10GBASE-T technology.

**Configuration Considerations for a Modular Spine**

In the Cisco Nexus 9508 care must be taken with the placement of the fabric cards: if you are using fewer than the total of six fabric cards, make sure to use the even slots.

Also refer to the hardware installation guide at:


The usual commands such as `show environment power` are available by connecting to the console of the Cisco Nexus 9508 or by opening a virtual shell session after logging into the bash shell.

**Cisco APIC and NX-OS Software**

Cisco Nexus 9000 series switches can be deployed in two modes:

- **Standalone mode**: in this mode the switch provides functionalities that are similar to the other Cisco Nexus switches with the addition of programmability, linux containers, python shell and so on. This design guide is not based on the use of standalone mode.
- **Fabric mode**: in this mode the switch operates as part of a fabric. This design guide is based on the use of the Nexus 9000 series switches in ACI mode.

Software for leaf and spine switches is managed directly from the controller. The Firmware Management view allows you to create upgrade policies (Figure 11).
Cisco ACI Model for Defining Workload Connectivity

This section describes how to define workload connectivity in Cisco ACI.

Policy Definition

With Cisco ACI the communication path between workloads is defined by establishing which server can talk to which other server and what the servers can talk about (for example the Layer 4 ports that should be used). Servers can be in the same bridge domain, in different bridge domains, in the same subnet, or in different subnets.

The definition of the connectivity abstracts the details of subnets, VLANs and bridge domains and instead expresses them in terms of policy “contracts” between EPGs.

The benefit this approach is that a configuration template becomes portable regardless of what is the specific IP addressing scheme that a datacenter uses.

Workload Classification

Workloads are associated with EPGs with bindings:

- A static binding defines the physical port on a leaf (and, potentially, the VLAN) that is associated with a given EPG
- The definition of the virtual machine mobility (VMM) domain association defines a dynamic binding through which a virtualized server tags traffic with a VLAN or a VxLAN that has been negotiated with Cisco APIC to segment the traffic.

Figure 12 shows the mapping of an EPG to multiple ports and VLANs. Each VLAN has only local significance and doesn’t change the relationship of this EPG with other EPGs.
Fig. 12. EPG Mapped to Multiple Ports and VLANs

Endpoint Groups and Contracts

Each EPG is a collection of servers (physical and virtual).

Each contract is defined by a name and by filters (ACLs). Filters are a collection of layer 4 protocol and ports.

Each EPG is a provider and a consumer of policy contracts.

As a first approximation you can think of the configuration of EPGs providing and consuming contracts similar to the configuration of VRFs that import and export route distinguishers.

Differently from the example of VRFs, an EPG provides a contract when it has a listening socket for incoming requests. As an example an EPG that hosts web servers should be configured as a provider of a contract that includes port 80 and 443. The client side EPG instead is a consumer of a web contract.

By defining which contracts each EPG provides and consumes the relationship between EPGs is established.

You can consider classic routing configurations in which everybody can talk to anybody as the equivalent to a contract called “anyany” that all EPGs provide and consume.

Figure 13 shows an example of a network topology with two EPGs. EPG-A provides the contract called A-to-B and EPG-B consumes it.

Cisco ACI uses a whitelist model: two EPGs cannot talk unless a contract expresses which traffic is allowed. The firewall in the picture represents the default filtering that occurs via the contract.
Any-to-Any Communication

If you want to use the Cisco ACI fabric as a simple routed or switched fabric you can configure contracts that are imported and exported by each EPG, and you can map each EPG to familiar constructs such as VLANs (Table 1).

You would still get the following benefits, among others:

- Programmability with REST APIs from a single configuration point
- Visibility and telemetry features
- Scalability
- Capability to use VLANs that have local significance but are mapped to the same bridge domain
- No need to configure HSRP
- Virtual workloads connectivity

Table 1. Configuring Any-to-Any Communication

<table>
<thead>
<tr>
<th>EPG “VLAN 10”</th>
<th>VLAN10</th>
<th>Default</th>
<th>ALL</th>
<th>ALL</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPG “VLAN 20”</td>
<td>VLAN20</td>
<td>Default</td>
<td>ALL</td>
<td>ALL</td>
<td></td>
</tr>
<tr>
<td>EPG “VLAN 30”</td>
<td>VLAN30</td>
<td>Default</td>
<td>ALL</td>
<td>ALL</td>
<td></td>
</tr>
</tbody>
</table>

© 2014 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.
**Application Policy Enforcement**

When you define a configuration, it is expressed in terms of a policy that defines:

- Which servers can talk to each other
- What the servers can talk about (for instance which Layer 4 ports can be used, potentially defined as “any any” to allow all communication)

Policies are enforced at the leaf.

Two levels of control settings define when the policy is pushed to the leaf and when the leaf can apply it in hardware:

- Resolution: immediate or on-demand mode
- Deployment: immediate or on-demand mode

If you set the immediate mode, the policy is immediately deployed to the leaf (deployment) or immediately applied to the hardware (resolution). If you set it the on-demand mode, the system waits for a workload to appear that requires that policy.

**Rendering to VRF Instances, VLANs and so on**

The segmentation needs expressed as EPGs and their binding and classification requirements are rendered on each leaf with well-known constructs such as VLANs or VRF instances. Communication policy enforcement also uses well-known constructs such as inbound and outbound 5-tuple match permit and denies and is powered by additional application-specific integrated circuits (ASICs) developed by Cisco.

With the CLI you can verify how the policy has been translated into the hardware for a given leaf.

To accomplish this just ssh to the management port of any leaf or spine on port 1025 and provide the correct credentials.

You can then change the prompt to the virtual shell (vsh) and issue commands such as the following ones:

```
leaf1# show system internal epm vlan all
leaf1# show vrf all
leaf1# show vlan all-ports
```

**Application Network Profile**

An “application” in the context of Cisco ACI is the end-to-end components that make up a business application. An application network profile is a collection of EPGs, contracts and connectivity policy. In other words is a collection of groups of workloads that together form what the administrator defines as the application.

The meaning of “application” varies depending on the team in which the administrator works. Here are some possible definitions of application in an organization:

- Business or service owner: An application is a platform for a business function such as an ordering application, a contact center application or a scheduling application
- IT application teams: An application is some component of a business application, such as a quoting application
- Middleware team: An application is a platform like IBM WebSphere or a Java engine
- Network team: An application is a webserver, a database server
Cisco ACI doesn’t define or discover applications; it offers a construct called an application network profile that defines the workloads (that is, the group of servers in a group of EPGs) that make up a given application.

For instance, a service owner may define all the workloads in an ordering application as the Ordering Application. This relationship doesn’t prevent these workloads from talking to other workloads outside the application profile; it just provides a convenient configuration container.

In general, you should think of the application profile as a hierarchical container of EPGs, along with their connectivity and policy requirements. It enables the fabric to provide meaningful information about the health status of the networking components that make a given application.

**Note:** To deploy Cisco ACI, you don’t need to know the mapping of applications. There are several documented methods for mapping existing network constructs within the ACI policy model.

**Routing and Switching in the Policy Model: Network and Bridge Domain**

All application configurations in Cisco ACI are part of a tenant.

The application abstraction demands that EPGs always be part of an application network profile, and the relationship between EPGs through contracts can span application profiles and even tenants.

Bridging domains and routing instances to move IP packets across the fabric provide the transport infrastructure for the workloads defined in the EPGs.

Within a tenant, you define one or more Layer 3 networks (VRF instances), one or more bridge domains per network, and EPGs to divide the bridge domains.

Figure 14 shows the creation of two tenants. Each tenant has two VRF instances, which are further subdivided into bridge domains, and each bridge domain has multiple EPGs.

**Figure 14.** Creation of Two Tenants
Figure 15 shows the relationship of the networks, bridge domains, and EPGs.

**Figure 15.** Relationships 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 (this is rarely needed). The bridge domain is not a VLAN, although it can act similar to a VLAN; you instead should think of it as a distributed switch, which, on a leaf, can be translated locally as a VLAN with local significance.

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

Whenever you create an EPG, you need to reference a 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 relationship among application components.

**Using REST to Program the Network**

The Cisco ACI management model is designed for automation. Every configurable element is part of the object tree known as the Management Information tree (MIT).

The networking elements in Cisco ACI can be configured using the GUI, REST calls, or a CLI that operates on the managed objects.

A RESTful web service (also called a RESTful web API) is a web service implemented using HTTP and the principles of REST. It is a collection of resources:

- The base universal resource identifier (URI) for the web service, such as [http://example.com/resources/](http://example.com/resources/)
- The Internet media type of the data supported by the web service; often JSON, XML, or YAML, but it can be any other valid Internet media type
● The set of operations supported by the web service using HTTP methods (for example, GET, PUT, POST, or DELETE)

Standard HTTP requests have the following semantics:

● POST: Target domain name, class, and options specify roots of subtrees at which modifications are applied. Data carried in the request is in the form of structured text (XML or JSON) subtrees. You can, therefore, specify a parent tree node target and affect N subtrees under that node. POST also can run methods on objects in scope.

● GET: Specify retrieval of objects in the scope of the URL and the options provided.

Configuration of the network using REST calls doesn’t require a deep understanding of the object tree; you can easily change sample configurations and reapply them. A popular tool for doing this is POSTMAN (http://www.getpostman.com/).

The typical sequence of configuration is:

1. Authenticate: Call https://<IP of APIC controller>/api/mo/aaaLogin.xml with a payload that in XML is <aaaUser name='username' pwd='password'/>. This call returns a cookie value that the browser uses for the next calls.

2. Send HTTP POST to apply the configuration: The URL of the POST message varies depending on the object, the following is an example: https://<IP of APIC controller>/api/mo/uni.xml, where api indicates that this call is to the API, mo indicates that this call is to modify a managed object, uni (universe) refers to the root of the object tree, and .xml indicates that the payload is in XML format. If the end of URL were .json, that would mean that the payload is in JSON format.

3. Verify the HTTP status code: You want a response of 200 OK.

Note: This design guide focuses on configuration using XML rather than JSON

You can also easily delete configurations with HTTP DELETE calls and retrieve information with GET calls.

Examples of Tools for Sending REST Calls

REST calls can be sent in a variety of ways. The most common use:

● The GUI: Using the GUI, you can use REST without having to create a REST call manually. You will most commonly use the GUI for infrastructure configurations such as initial setup, creation of PortChannels, virtual PortChannel (vPC) allocation of VLAN and VxLAN name space, creation of users, monitoring, and so on. You can also use the GUI to generate configurations that you then save in XML format for automation purposes.

● POSTMAN with a Chrome browser: You will most commonly use Python to test tenant configurations that you want to automate and provide to the users of the network infrastructure.

● Python scripts: You will most commonly use Python scripts for very large configurations that you want to create rapidly, and that you may have already tested individually using POSTMAN or cURL.

● Scripts in general

● Automation tools: For instance, you can use Cisco UCS Director or Cisco Intelligent Automation for Cloud (IAC).

POSTMAN makes sending REST calls very easy. Figure 16 shows the field in which to enter the URL, the space to place the XML payload, and the Send button for sending the post to the controller.
You can also use Python (or any scripting language) by creating a simple script that transforms XML files into REST calls. You can invoke this script from your computer to create configuration settings.

The following script, xml2REST.py, takes as input the Domain Name System (DNS) name or IP address of the server and the name of a text file that includes the XML configuration settings. This script also includes the credentials for Cisco APIC, which you may have to change.

```python
#!/usr/bin/python
import glob
import json
import os
import os.path
import requests
import sys
import time
import xml.dom.minidom
import yaml

try:
    xmlFile = sys.argv[1]
except Exception as e:
    print str(e)
    sys.exit(0)

apic = sys.argv[2]
```
auth = {
    'aaaUser': {
        'attributes': {
            'name':'admin',
            'pwd':'P@ssw0rd'
        }
    }
}
status = 0
while( status != 200 ):
    url = 'http://%s/api/aaaLogin.json' % apic
    while(1):
        try:
            r = requests.post( url, data=json.dumps(auth), timeout=1 )
            break;
        except Exception as e:
            print "timeout"
    status = r.status_code
    print r.text
    cookies = r.cookies
    time.sleep(1)

def runConfig( status ):
    with open( xmlFile, 'r' ) as payload:
        if( status==200):
            time.sleep(5)
        else:
            raw_input( 'Hit return to process %s' % xmlFile )

    data = payload.read()
    print '++++++++ REQUEST (%s) ++++' % xmlFile
    print data
    print '-------- REQUEST (%s) --------' % xmlFile
    url = 'http://%s/api/node/mo/.xml' % apic
    r = requests.post( url, cookies=cookies, data=data )
    result = xml.dom.minidom.parseString( r.text )
    status = r.status_code
    print '++++++++ RESPONSE (%s) ++++' % xmlFile
    print result.toprettyxml()
    print '-------- RESPONSE (%s) --------' % xmlFile
    print status
runConfig( status )
With Python, you can perform very complex configurations near instantaneously.

Alternatively, if you want to perform a CLI-based configuration from your desktop to the controller, you can also use cURL as in the following example.

First get the authentication token:

```bash
curl -X POST http://<APIC-IP>/api/aaaLogin.xml -d '<aaaUser name="admin" pwd="P@ssw0rd" />' - cookie.txt
```

Save it to a file; then perform any operation with POSTMAN:

```bash
curl -b cookie.txt -X POST http://<APIC-IP>/api/mo/uni/tn-finance.xml -d '<fvTenant />'
```

**REST Syntax in Cisco ACI**

The format for the URL to place REST calls in Cisco ACI is as follows:

```
http://host[[:port]]/api/{mo|class}/{dn|className}.{json/xml}[?options]
```

- **/api/** specifies that the message is directed to the API.
- **mo | class** specifies whether the target of the operation is a managed object (MO) or an object class.
- **dn** specifies the distinguished name (DN) of the targeted MO.
- **className** specifies the name of the targeted class. This name is a concatenation of the package name of the object queried and the name of the class queried in the context of the corresponding package. For example, the class `aaa:User` results in a `className` of `aaaUser` in the URI.
- **json | xml** specifies whether the encoding format of the command or response HTML body is JSON or XML.

Simple operations are performed as follows.

**To create a tenant (fvTenant):**

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

**To create an application network profile (fvAp):**

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

**To add EPGs (fvAEPg):**

```xml
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>
```
Modeling Tenants in XML

The example shown here creates a tenant with the necessary bridging domain and routing instance.

In this example, the tenant provides connectivity to servers through subnets 10.0.0.1/24 and 20.0.0.1/24. The default gateway can be either 10.0.0.1 or 20.0.0.1.

Servers can connect to EPG VLAN10 or EPG VLAN20.

The EPG is also created as a port group in a VMware Virtual Distributed Switch (VDS) on VMware ESX. The virtual machine manager and Cisco APIC negotiate to determine which VLAN or VxLAN to use for the communication in this port group.

In the example, the meaning of the following fields is 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"/>
      <fvSubnet ip="20.0.0.1/24" 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>
    </fvAp>
  </fvTenant>
</polUni>
```
**Defining the Relationship Amongst EPGs (Providers and Consumers)**

The communication path between EPGs is managed using the concept of contracts. Contracts define the protocols and Layer 4 ports that can be used for the communication path between two EPGs.

The following example of contract defines a Permit All filter, where:

- `vzBrCP` is the name of the contract.
- `vzSubj` refers to the subject and is the name of the container of filters, which are similar to an ACL but more powerful in that they allow for separate inbound and outbound filtering.
- `vzRsSubfiltAtt` refers to a filter; the default filter is **permit any any**.

```xml
<vzBrCP name="A-to-B">
    <vzSubj name="any">
        <vzRsSubjFiltAtt tnVzFilterName="default"/>
    </vzSubj>
</vzBrCP>
```

The relationship between contracts is defined according to which EPG provides the contract and which EPG consumes the contract. The following example illustrates how EPG-A can be made to talk to EPG-B, where:

- `fvRsProv` indicates the name of the contract that EPG-A provides,
- `fvRsCons` indicates the name of the contract that EPG-B consumes,

```xml
<fvAp name="web-and-ordering">
    <fvAEPg name="EPG-A">
        <fvRsProv tnVzBrCPName="A-to-B" />
    </fvAEPg>
    <fvAEPg name="EPG-B">
        <fvRsCons tnVzBrCPName="A-to-B"/>
    </fvAEPg>
</fvAp>
```

**A simple Any-to-Any Policy**

The configuration described in the previous section instantiates a bridge domain for use by the tenant and a routing instance and default gateway.

Servers can then be associated with EPGs VLAN10 and VLAN20.

If the servers are in the same EPG, they can talk without any further configuration, but if they are part of different EPGs, the administrator has to configure explicit contracts and define which EPG can talk with which EPG.

The example in this section completes the previous configuration and enables any-to-any communication among EPGs just as a conventional routing and switching infrastructure would provide.

```
POST to http://apic1/api/mo/uni.xml
<polUni>
    <fvTenant dn="uni/tn-Customer1" name="Customer1">
        <vzBrCP name="ALL">
            <vzSubj name="any">
                <vzRsSubjFiltAtt tnVzFilterName="default"/>
            </vzSubj>
        </vzBrCP>
    </fvTenant>
</polUni>
```
Cisco ACI Architecture Design Principles

Cisco ACI technology is based on Layer 3 traffic forwarding with VxLAN encapsulation.

The Cisco ACI architecture is based on the following networking architecture:

- A controller that is not involved in traffic forwarding
- Zero-touch provisioning for all loopbacks, Layer 3 links, and routing within the fabric
- Leaf-and-spine architecture
- Host-based forwarding by lookup in the mapping database that provides IP-to-VxLAN tunnel endpoint mapping information to the leaf
- Hardware-based mapping database lookup
- New equal-cost multipath (ECMP) algorithm that goes beyond Layer 4 port hashing and that can handle very large (elephant) flows
- Anycast gateway

Cisco APIC

Cisco APIC is deployed as a cluster of three or more appliances, although you can still create new policies even with a single controller, and the fabric can forward traffic even without controllers.

Cisco APIC is shipped as a physical server appliance on Cisco UCS C220 M3 server with two 10 Gigabit Ethernet interfaces that must be connected to any two leaf switches and with Gigabit Ethernet interfaces for out-of-band management.

Central Point of Management but no Centralized Control Plane

Cisco APIC provides several functions and constitutes the central point of management of the fabric. The controller is not in the path of the traffic; it is just a management entity that stores and distributes policies and allows you to configure the fabric. The controller can be disconnected, moved to a different port, etc. without any impact on the fabric. This can be thought of as a controller that focuses on what should connect, rather than how things connect.
Distributed forwarding control plane is maintained for scale, with each node in the fabric running a full control plane. This control plane performs routing and switching and implements the policies that the controller has communicated. You can log into each device and see how the devices have been configured. You can issue `show` commands and view the MAC address tables, VRF instances, etc. just as you can do on any Cisco NX-OS device.

If there’s a problem in the network, such as disconnected links, the fabric can handle this problem without the intervention of the controller. The policies may have already been pushed to the leaf switches and just not applied in hardware (this behavior is configuration dependent), in these cases workload mobility will work just fine without the controller.

**Controller Availability**

You can think of the fabric as a giant telemetry device that collects information for Cisco APIC to provide meaningful data to run your data center and your applications. Cisco APIC is the database for all this data, just as it is for the policies that govern the fabric.

Because of this design, the Cisco APIC database is based on these principles:

- High Performance Computing (HPC) type clustering with all active nodes
- High availability (three controllers are recommended, although the fabric can be managed with just one)
- Low latency
- Incremental scalability
- Eventual consistency based on Pormise Theory
- Partition tolerance

The controller automatically archives and replicates the following data:

- Policies
- Statistics
- Endpoint database

The fabric continues to forward traffic even in the absence of the controller.

**Fabric**

This section describes the fabric control plane and data plane.

The Cisco ACI network solution uses an overlay, based on VxLAN, to virtualize the physical infrastructure. This overlay, like most overlays, requires the data path at the edge of the network to map from the tenant endpoint address in the packet, its identifier, to the location of the endpoint, its locator. This mapping occurs in the tunnel endpoint (TEP).

Cisco ACI uses a combination of a centralized database of the mappings implemented in the packet data path, at line rate, coupled with a caching mechanism, again in the data path, at the tunnel endpoint. The solution supports very large topologies with no real-time software intervention and near-instantaneous response to moves and changes.

The Cisco ACI fabric is a 40-Gbps IP fabric that supports routing to the edge (100-Gbps capable). It is a bipartite graph topology with distinct leaf and spine nodes. The spine layer requires a minimum of two nodes (for 1+1 redundancy).
The spine can support an arbitrary number of tiers if required.


**Zero Touch Provisioning**

When using Cisco ACI, you don’t have to configure the following items:

- You don’t need to configure /30.
- You don’t need to configure the routing for the infrastructure.
- You don’t need to specify the subnets to advertise in an Interior Gateway Protocol (IGP) area.
- You don’t need to specify loopback addresses for IGP announcements.
- You don’t need to specify the interfaces on which to peer.
- You don’t need to tune the routing timers.
- You don’t need to verify cabling and neighbors.
- You don’t need to remove VLANs from trunks.

All of these configurations are set automatically when you connect leaf and spine nodes together.

The Cisco ACI fabric is designed to provide a zero-touch operation experience with:

- A logically central but physically distributed controller for policy, bootstrap, and image management
- Easy startup with topology autodiscovery, automated configuration, and infrastructure addressing using industry-standard protocols: IS-IS, LLDP, and Dynamic Host Configuration Protocol (DHCP)
- A simple and automated policy-based upgrade process and automated image management

Cisco APIC is a physically distributed but logically centralized controller that provides DHCP, bootstrap configuration, and image management to the fabric for automated startup and upgrades.

After LLDP discovery, Cisco APIC learns all neighboring connections dynamically. These connections are validated against a loose specification rule that the user provides through REST calls or through the GUI.

If a rule mismatch occurs, a fault occurs, and the connection is blocked. In addition, an alarm is created indicating that the connection needs attention. The Cisco ACI fabric operator has the option of importing the names and serial numbers of all the fabric nodes from a simple text file into Cisco APIC, or discovering the serial numbers automatically and assigning names from the Cisco APIC GUI, CLI, or API.

**What is an Integrated Overlay**

The Cisco ACI solution uses an overlay, based on VxLAN, to virtualize the physical infrastructure. This overlay, like most overlays, requires the data path at the edge of the network to map from the Tenant end-point address in the packet, a.k.a. its "identifier", to the location of the end-point, a.k.a. its "locator". This mapping occurs in a function called a “Tunnel End-Point” or TEP.

The VxLAN header carries several fields. The VNID field is the 24-bit virtual network ID carried in the VxLAN header. It represents one of three possible things: the Virtual L3 context or VRF for this packet, the bridge domain or BD for this packet. The VNID is a VRF when the packet is routed and is a BD when the packet is bridged.

The reason for using an Integrated Overlay is to provide the fabric the capability to be virtualized into multiple virtual fabrics, while preserving the ability to apply proper Quality of Service and advanced traffic telemetry features.
Routed Design with VxLAN Overlays

The Cisco ACI fabric decouples the tenant endpoint address, its identifier, from the location of that endpoint, which is defined by its locator, or VxLAN termination endpoint address.

Forwarding within the fabric is between VxLAN tunnel endpoints (VTEPs) and uses an extender VxLAN header format referred to as the VxLAN policy header. The mapping of the internal tenant MAC or IP address to a location is performed by the VTEP using a distributed mapping database. Cisco ACI supports full Layer 2 and Layer 3 forwarding semantics; no changes are required to applications or endpoint IP stacks.

The default gateway for each bridge domain is a pervasive switch virtual interface (SVI) configured on top-of-rack (ToR) switches wherever the bridge domain of a tenant is present. The pervasive SVI has an anycast gateway per subnet, which is global across the fabric. The fabric routes if the destination MAC address is a router MAC address; otherwise, it bridges.

Cisco ACI forwards based on the destination IP address within and between subnets. Bridge semantics are preserved for traffic with a subnet (no time-to-live [TTL] decrement, no MAC address header rewrite, etc.). Non-IP packets are forwarded using the MAC address. In this case, the fabric learns the MAC address for non-IP packets, and it learns the IP address for all other packets.

Leaf-and-Spine Design

The fabric is based on a leaf-and-spine architecture.

Cisco ACI topologies have two types of functions: functions for leaf devices and functions for spine devices:

- **Leaf devices:** These devices have ports connected to Classic Ethernet devices (servers, firewalls, router ports, and so on) and 40 Gigabit Ethernet uplink ports connected to the fabric cloud. Leaf switches are at the edge of the fabric and contain the TEP function. They are also responsible for routing or bridging the tenant packet and for applying network policies. Leaf devices can map an IP or MAC address to the destination VTEP. Leaf devices can be used either as regular Cisco NX-OS devices or in ACI mode as leaf devices.

- **Spine devices:** These devices exclusively interconnect leaf devices. Spine devices also provide the mapping database function. The hardware used for the spine is designed for this function. The hardware includes specific line cards for the Cisco Nexus 9508 Switch and a ToR switch with 40 Gigabit Ethernet ports.

Besides forwarding traffic, the leaf discovers the endpoints and informs the spine. As a result, the spine creates the mapping between the endpoint and the VTEP.

The leaf is also the place where policies are applied to traffic.

All leaf devices connect to all spine devices, and all spine devices connect to all leaf devices, but no direct connectivity is allowed between spine devices or between leaf devices. You may have topologies in which certain leaf devices are not connected to all spine devices, but traffic forwarding won’t be as effective as when each leaf is attached to each spine.
Mapping Database
The mapping database is a database maintained by the fabric that contains the mapping for each endpoint attached to the network (identifier) and the address of the tunnel endpoint that it sits behind (locator). The endpoint address is both the MAC address and the IP address of the endpoint plus the logical network that it resides in (VRF instance). The mapping database in the spine is replicated for redundancy and it is synchronized across all spines.

When an ingress leaf switch forwards a packet, it checks the local cache of the mapping database. If it does not find the endpoint address it is looking for, it encapsulates the packet to the proxy function residing in the spine switch and forwards it as unicast. The spine switch, upon receiving a packet addressed to its proxy function, looks up the destination identifier address in its forwarding tables, which contain the entire mapping database. On the basis of the result, it reencapsulates the packet using the correct destination locator while retaining the original ingress source locator address in the VxLAN encapsulation. This packet is in turn forwarded as unicast packet to the intended destination. This allows for the elimination of Address Resolution Protocol (ARP) flooding.

The entries in the mapping database can expire. The default timer is 900 seconds. After 75 percent of this value is reached, three Address Resolution Protocol (ARP) requests are sent as unicast in a staggered fashion (with a time delta between the requests) as a probe to the MAC address of the endpoint to check for the endpoint’s existence. If there is no ARP response, then the endpoint is removed from the local table.

Unknown Unicast and Broadcast
Traffic forwarding in Cisco ACI operates as follows:

- Cisco ACI routes traffic destined for the router MAC address.
- Cisco ACI bridges traffic that is not destined for the router MAC address.

In both cases, the traffic traverses the fabric encapsulated in VxLAN to the VTEP destination IP address of the endpoint.

Cisco ACI doesn’t use flooding by default, but this behavior is configurable.

These are the options for Layer 2 unknown unicast frames:

- Flood: If the flood option is enabled in a bridge domain, the packet is flooded in the bridge domain by using a multicast tree rooted in the spine that is scoped to the bridge domain.
- No-flood (default): The packet is looked up in the spine, and if it is not found in the spine, it is dropped.

These are the options for Layer 2 multicast frames:

- Flood (default): Flood in the bridge domain.
- Drop: Drop the packet.

These are the options for Layer 3 unknown unicast frames:

- Drop: If the destination IP address is not found in the leaf cache, forward the packet to the spine. If the spine finds the address in the database, forward the packet; otherwise, drop the packet.

These are the forwarding options for ARP:

- Flood: Use traditional ARP flooding.
- Hardware-based forwarding (default): Send ARP to the destination endpoint.

All these options are configurable in the GUI (or using REST) as shown in Figure 17.
Notice that the bridge domain must be associated with a router instance for the subnets to be instantiated. The other fields control the way that unknown unicast traffic and multicast traffic is forwarded.

The Unicast Routing field controls whether this is a pure Layer 2 bridge domain or whether it provides a pervasive default gateway.

**Figure 17.** Forwarding Options

![Bridge Domain - tenant1-bridgedomain](image)

**Access Layer**

The configuration of Cisco ACI access policies requires an understanding of VLAN and VxLAN name spaces. Access policies also make configuration of PortChannels and vPCs easy to accomplish.

**Use of VLANs as a Segmentation Mechanism**

In Cisco ACI the VLANs used between a server and a leaf have local significance and they are used exclusively to segment traffic coming from the servers. Cisco ACI has been designed so that when using virtualized workloads you don’t have to enter VLAN numbers manually per each port-group. Whenever possible one should leverage the dynamic negotiation of VLANs between the virtualized server and the Cisco ACI fabric.

Figure 18 shows how a virtualized server tags traffic with a VLAN or a VxLAN and sends it to the leaf. The tenant configuration defines the VLAN or VxLAN that belongs to the EPG.
Figure 18. Use of VLANs for Segmentation

In the case of physical workloads you should use VLANs to map traffic coming from a trunk into the correct EPG.

**VLANs and VxLANs Namespaces**

A single fabric may have multiple VMM domains, each consuming 4000 VLANs (EPGs), so sometimes you may want to reuse a VLAN range multiple times. The same pool of VLANs can be reused as long as it is associated with a different set of leaf switches.

Alternatively, if you use VxLANs between the virtualized server and the Cisco ACI network, there is less need for reuse because the addressing space is larger.

In spanning-tree networks, the user must specify which VLANs belong to which ports by using the `switchport trunk allowed vlan` command. In Cisco ACI, you specify a domain (physical or virtual), and you associate this domain with a range of ports. Unlike with traditional Cisco NX-OS operations, in Cisco ACI the VLANs used for port groups on virtualized servers are dynamically negotiated between Cisco APIC and the virtual machine manager (Figure 19).

Figure 19. Reusing VLANs with Dynamic Negotiation
**Concept of Domain**

Whether you connect physical or virtual servers to the Cisco ACI fabric, you define a physical or a virtual domain. Virtual domains reference a particular virtual machine manager (for example, VMware vCenter 1 or data center ABC) and a particular pool of VLANs or VxLANs that will be used. A physical domain is similar to a virtual domain except that there’s no virtual machine manager associated with it.

The person who administers the VLAN or VxLAN space is the infrastructure administrator. The person who consumes the domain is the tenant administrator. The infrastructure administrator associates domains with a set of ports that are entitled or expected to be connected to virtualized servers or physical servers through an attach entity profile (AEP).

You don’t need to understand the details of the AEP except that it encapsulates the domain. The AEP can include boot policies for the virtualized server to boot from the network, and you can include multiple domains under the same AEP and authorize virtualized servers of different kinds.

The following example of an AEP specifies that Cisco ACI should expect a VMware ESX server managed by VMware vCenter 1 on port 1/3 on leaf 101. Normally you would specify a greater range of ports (for example, the VMware vMotion domain).

```xml
<infraInfra dn="uni/infra">
  <!-- attachable entity, i.e. Domain information -->
  <infraAttEntityP name="Entity_vCenter1_Domain">
    <infraRsDomP tDn="uni/vmmp-VMware/dom-vCenter1" />
  </infraAttEntityP>
  <!-- Policy Group, i.e. a bunch of configuration bundled together -->
  <infraFuncP>
    <infraAccPortGrp name="vCenter1_Domain_Connectivity">
      <infraRsAttEntP tDn="uni/infra/attentp-Entity_vCenter1_Domain" />
    </infraAccPortGrp>
  </infraFuncP>
  <infraAccPortP name="Leaf101esxports">
    <infraHPortS name="line1" type="range">
      <infraPortBlk name="block0" fromPort="3" toPort="3" />
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-vCenter1_Domain_Connectivity" />
    </infraHPortS>
  </infraAccPortP>
  <infraNodeP name="Leaf101">
    <infraLeafS name="line1" type="range">
      <infraNodeBlk name="block0" from_="101" to_="101" />
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-Leaf101esxports" />
  </infraNodeP>
</infraInfra>
```
Using the AEP, if you simply need to add ports to the configuration, you can edit the interface profile infraAccPortP and add lines such as `<infraHPortS name="line2" type="range">` with new interface ranges.

Policy-based Configuration of Access Ports
The infrastructure administrator configures ports in the fabric for speed, Link Aggregation Control Protocol (LACP) mode, LLDP and Cisco Discovery Protocol, etc.

In Cisco ACI, the configuration of physical ports is designed to be extremely simple for both small- and large-scale data centers.

The underlying philosophy of Cisco ACI is that the infrastructure administrator categorizes servers based on their requirements: virtualized servers with hypervisor A connected at a Gigabit Ethernet, nonvirtualized servers running OS A connected at 10 Gigabit Ethernet, etc.

Cisco ACI provides a way to keep this level of abstraction when defining the connection of the servers to the fabric (Figure 20). The infrastructure administrator prepares a template of configurations for servers connected with active-standby teaming, PortChannels, and vPCs and bundles all the settings for the ports into a policy group. The administrator then creates objects that select interfaces of the fabric in ranges that share the same policy-group configuration.

**Figure 20.** Defining the Connection of Servers to the Fabric

The logic is better understood by following an example of configuration.

In the fabric access policy, under the switch profiles, you define one profile per switch: leaf101, leaf102, and so on (Figure 21).
Figure 21. Defining One Profile per Switch

You have now created objects that represent each leaf device. You could also create an object that represents two leaf devices if you wanted. You can then create profiles that categorize ports into different groups and later add these to the switch.

Figure 22 illustrates this point. If you highlight the object that represents the leaf of interest, you can then add interface profiles to it.

Figure 22. Adding Interfaces to a Leaf

The interface profile consists of a range of interfaces with similar configurations. For example, the range kvmportsonleaf101 may select ports 1 through 10.

The configuration of the ports is based on the policy group (Figure 23). The policy group is a template of configurations such as speed, Cisco Discovery Protocol, LLDP, Spanning Tree Protocol, LACP and AEP (see the previous section).
To associate the configuration with interfaces on the leaf switches, you need to create an interface profile. For instance, assume that port 1/15 on leaf 101 is attached to a physical server. You can create an interface profile object called **physicalserversonleaf101** and add port 1/15 to it. You can add more ports later to apply the same configuration to all ports connected to the physical servers (Figure 24).

**Figure 24. Creating an Interface Profile**

For this selection of ports to be carved out of leaf 101, you need to add it to the switch profile that identifies leaf101 (Figure 25).
Figure 25. Adding the Interface Profile to the Switch Profile

Figure 26 shows the relationship between leaf switches, ports, AEP, domains, and VLAN pools. This figure illustrates the following points:

- The infrastructure administrator can create a range of VLANs.
- This VLAN range is associated with a physical domain.
- This association is encapsulated in an AEP (which is configured in the Global Policy area of the GUI).
- The left portion of the figure shows how the AEP is associated with an interface.
- The interface profile selects an interface number.
- The switch profile selects a switch number.
- The policy group is basically the interface configuration, which may include an AEP (and, as a result of the various links, also includes the set of VLANs).

Figure 26. Relationships Between Leaf Switches, Ports, AEP, Domains, and VLAN Pools

This configuration can be achieved with a single REST call (Figure 27).
Figure 27.  REST Call

The advantage of this approach is that you can effectively apply configurations in a more logical manner. For instance, if you want to add one port to the set of physical servers, you just need to add an interface to the interface profile. If you want to change the physical ports settings, you just make that change in the policy group. If you want to add a VLAN to the range you, just modify the physical domain.

Furthermore, you can create policies that apply not to just one switch at a time but to multiple switches, which is very useful for the configuration of vPCs.

**PortChannels**

Policy groups can be any of three types, defined when the policy group is created:

- Regular port
- PortChannel
- Virtual PortChannel

You can create PortChannels in Cisco ACI more quickly and easily than on a regular switch. The reason is that with the policy model, you just need to create a selection of interfaces and associate that with the same policy group. Each policy group of type PortChannel is a different channel group.

The LACP active or passive configuration is managed through the interface policy configuration (which is referenced by the policy group).

Figure 28 shows how to create the LACP configuration.
Figure 28.  Creating the LACP Configuration

Figure 28 shows how to create the PortChannel group.

Figure 29.  Creating the LACP Configuration

You can do everything in a single REST call:

```
http://10.51.66.236/api/mo/uni.xml
<infraInfra>

<infraNodeP name="leafs101">
    <infraLeafS name="leafsforpc" type="range">
        <infraNodeBlk name="line1" from="101" to="101" />
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-ports22and23" />
</infraNodeP>

<infraAccPortP name="ports22and23">
    <infraHPortS name="line1" type="range">
```

© 2014 Cisco and/or its affiliates. All rights reserved. This document is Cisco Public Information.
Note: A bundle group defined with the setting lagT="link" indicates that this configuration is for a PortChannel. If the setting instead were lagT="node", it would be a configuration for a vPC.

Virtual PortChannels
Creating vPCs in Cisco ACI is also simpler than creating them in regular Cisco NX-OS configurations because there are fewer possibilities for mistakes and because you can use switch selectors to configure ports on multiple switches simultaneously.

Configuring vPCs with Cisco ACI is different than configuring them in other NX-OS platforms because of the following:

- There is no need for a vPC peer-link
- There is no need for a vPC peer keepalive

To configure a vPC in Cisco ACI, in addition to creating a policy group of type vPC and creating switch and interface profiles, you need to create a vPC protection policy, which tells Cisco ACI which pair of switches form a vPC domain (Figure 30).

Figure 30. Creating a vPC Protection Policy
With this policy in place, all you need to do is create one policy group of type vPC per channel group and reference it from the interface profile.

You also may want to create a switch profile that encompasses the two leaf switches that form the vPC domain to add all the interface configurations under the same switch profile object.

All of this configuration can be accomplished with multiple REST calls or with a single REST call with all the pieces together.

This configuration creates the vPC domain:

```xml
POST to api/mo/uni/fabric.xml
<polUni>
  <fabricInst>
    <fabricProtPol name="FabricPolicy">
      <fabricExplicitGEp name="VpcGrpPT" id="101">
        <fabricNodePEp id="103"/>
        <fabricNodePEp id="105"/>
      </fabricExplicitGEp>
    </fabricProtPol>
  </fabricInst>
</polUni>
```

This configuration creates the vPC channel group; the keyword `lagT="node"` indicates that this is a vPC:

```xml
POST to api/mo/uni.xml
<polUni>
  <infraInfra dn="uni/infra">
    <infraFuncP>
      <infraAccBndlGrp name="vpcgroup1" lagT="node">
      </infraAccBndlGrp>
    </infraFuncP>
  </infraInfra>
</polUni>
```

This configuration associates ports and switches with the policy:

```xml
POST to api/mo/uni.xml
<polUni>
  <infraInfra dn="uni/infra">
    <infraAccPortP name="interface7">
      <infraHPortS name="ports-selection" type="range">
        <infraPortBlk name="line1" fromCard="1" toCard="1" fromPort="7" toPort="7">
        </infraPortBlk>
      </infraHPortS>
    </infraAccPortP>
    <infraNodeP name="leaf103andleaf105">
      <infraLeafS name="leafs103and105" type="range">
      </infraLeafS>
    </infraNodeP>
  </infraInfra>
</polUni>
```
Physical Topology

Cisco ACI uses a spine-and-leaf topology. All leaf nodes connect to all spine nodes, but a full mesh is not required. Spine nodes don’t connect to each other, and leaf nodes don’t connect to each other.

As Figure 31 shows, a simple topology can consist of a pair of spine switches (such as the Cisco Nexus 9336PQ switches) with leaf devices dual-connected at 40 Gigabit Ethernet to each spine device. Servers can be connected to two leaf devices, potentially in a PortChannel or vPC. Any leaf switch can also be a border leaf switch for outside connectivity from each tenant. All devices can be connected through the mgmt0 port to an out-of-band management network, which can be used to access the switch CLI or to connect to the REST API of Cisco APIC.

Figure 31. Simple Physical Topology

This topology can be physically all in one room or it can be split in two rooms for instance like depicted in Figure 32.
Cisco APIC Design Considerations

Cisco APIC should be dual-attached to two leaf devices. No configuration is required to build the bonding interface; the 10 Gigabit Ethernet ports of the Cisco APIC appliance are preconfigured for bonding.

The fabric needs at least one Cisco APIC server to provide switch bootup, policy management, and fault and statistics correlation. Three controllers are recommended for redundancy, although you can still provision and configure policies for the fabric with a single controller. Three controllers provide optimal redundancy and support both Cisco APIC software upgrades and failure scenarios. More than three controllers can be used for geographically redundancy and in cases in which you need additional transactional scale (high transaction rate for the API for policy creation or monitoring of the network).

The members of the Cisco APIC cluster do not need to form a full cluster prior to switch node bootstrapping. The controller cluster is designed to operate in split-brain mode, which will occur on bootup and during a partitioning network failure (large-scale failure). The cluster can form part way through or at the end of network activation if required.

Connectivity between Cisco APIC cluster members takes place through the management port and infrastructure VRF, so an out-of-band management network is not needed for the cluster to form, but again the cluster does not have to form before each individual node can initiate the fabric and switch.

When you define the Cisco APIC cluster, you will be asked how many members you want to be present at steady state. This number tells the cluster how many other nodes to expect so that each node can track bootup scenarios (only the first node has been attached), partitioned fabrics, and other cases in which only a subset of the total target number of Cisco APIC nodes is active.

When all nodes are active, the distributed management information tree (DMIT) for the Cisco APIC cluster has the database shards (containers for the managed objects representing the system and policy) replicated across the servers and assigns one of the shard copies as the primary, with transactions performed against that copy. If three servers are defined in a cluster, when all three are active each will support transactions against one-third of the DMIT. If only two servers are active, each has half of the shards marked as primary, and the system load is shared across the two active Cisco APIC nodes.
Spine Design Considerations
The main function of the spine is to provide the mapping database in case a leaf hasn’t learned yet about the mapping of an endpoint and to forward traffic among leaf switches.

The mapping database is stored in a redundant fashion within each spine, and it is replicated among spine switches so that if a spine disappears, traffic forwarding continues.

Modular spine switches have greater mapping database storage capacity; in fact, the mapping database is sharded across fabric cards, so the more fabric cards, the more endpoints can be stored. The use of more fabric cards also depends on the forwarding capacity that you want to give to line cards.

No direct link is allowed or required between the spine switches.

Leaf design considerations
Leaf switches provide physical and virtual server connectivity. They terminate VLAN and VxLAN and reencapsulate the traffic in a normalized VxLAN header, and they are the enforcement point for policies.

Typically, you want to look at leaf switches in pairs, because of the likelihood that you are going to connect servers with PortChannels in vPC mode to the leaf switches.

Cisco ACI leaf switches support vPC interfaces similar to Cisco Nexus Family switches (IEEE 802.3ad PortChannels with links split across two devices); however, with Cisco ACI a peer link is not needed to connect the leaf switches.

It’s easy to define peers of leaf switches using switch profiles.

Leaf switches can be the attachment point simultaneously for workloads and for the border leaf to provide connectivity to the WAN or to an MPLS VPN cloud.

For policy propagation to the leaf, you can choose among these three modes, depending on the trade-off you want to make between scalability and immediacy:

- Policy preconfiguration: Cisco APIC pushes all policies to all leaf switches in a VMM domain, and policies are immediately programmed.
- No policy prepopulation: Policy is requested from Cisco APIC when a notification is received or data-path detection occurs for new endpoint. Packets are dropped until the policy programming is complete.
- Policy prepopulation with on-demand configuration (default): Cisco APIC pushes all policies to all leaf switches in a VMM domain. The policy is programmed when the VMM is notified or the data plane learns of a new endpoint. During the configuration stage, the packets are forwarded, and the policy applied on the egress leaf.

Multi-tenancy considerations
The Cisco ACI fabric has been designed for multitenancy.

To create a tenant, a simple REST call like the following is needed:

http://10.51.66.236/api/mo/uni.xml
<polUni>
<!-- Tenant Customer1 -->
<fvTenant dn="uni/tn-Customer1" name="Customer1">
<fvCtx name="customer1-router"/>
</!-- bridge domain -->
<fvBD name="BD1">
  <fvRsCtx tnFvCtxName="customer1-router" />
  <fvSubnet ip="10.0.0.1/24" scope="public"/>
  <fvSubnet ip="20.0.0.1/24" scope="private"/>
  <fvSubnet ip="30.0.0.1/24" scope="private"/>
</fvBD>
</!-- Security -->
<aaaDomainRef dn="uni/tn-Customer1/domain-customer1" name="customer1"/>
</fvTenant>
</polUni>

The example shows a REST call to create a tenant named Customer1, to associate a VRF instance named customer1-router and a bridge domain named BD1, and to create three subnets: 10.0.0.1, 20.0.0.1, and 30.0.0.1. These subnets are also the default gateways for the tenant. In addition, the security domain associated with the tenants provides the capability to scope the operations of administrators.

The tenant administrator cannot see the full fabric. This administrator can use some resources, such as physical ports and VLANs, to exit the fabric or connect to the outside world and can extend the definition of EPGs to virtualized servers.

The infrastructure administrator manages the entire fabric and can control and scope the domains of VLAN and VxLAN name spaces that a given tenant can use.

Resources in the fabric may be dedicated to a given tenant, and other resources may be shared. An example of a dedicated resource is a nonvirtualized server. Examples of shared resources are virtualized servers and ports to connect outside the fabric.

To simplify the task of associating these resources with tenants, Cisco suggests the following:

- Create one physical domain per tenant for nonvirtualized servers. A physical domain is a VLAN name space. The VLAN name space is used to further divide servers into EPGs. The physical domain is then associated with a set of ports that a tenant is eligible to use.

- Create one physical domain per tenant for external connectivity. This approach defines a set of VLANs that can be used to stitch the virtual data center with an MPLS VPN cloud or across data centers. Multiple physical domains are then grouped into a single AEP because the ports used to connect to the outside are shared across multiple tenants.

Creation of one single VMM domain per tenant is theoretically possible, but practically this approach is not feasible because administrators will want to share the VMM across multiple tenants. In this case, the best way to aggregate VMM domains is to associate the same VMM domain with all the leaf ports that can be connected to virtualized servers in the same mobility domain.
Initial Configuration Steps

The infrastructure administrator manages the initial configuration of Cisco ACI. The fabric is self-discovered when Cisco APIC is attached to a leaf, and the administrator validates the legitimate nodes from the GUI or with scripts.

There is no need to configure/30 on the uplinks or to configure loopback addresses and so on. This configuration is performed automatically through self-service provisioning.

Items that you may want to check as part of the initial phase of the deployment include:

- Clock synchronization: If the clocks on the nodes and the controller are configured with times and dates that are too far apart, discovery may not occur.
- Out-of-band management: Ensure that you have out-of-band management in place to connect to Cisco APIC and potentially to the mgmt0 ports of the leaf and spine switches (at the time of this writing, Gigabit Ethernet is required for out-of-band management).

Network Management

Cisco APIC configures automatically an Infrastructure VRF that is used for inband communication between Cisco APIC and the switch node communication, and it is non routable outside the fabric.

The Cisco APIC serves as DHCP and TFTP server for the fabric. Cisco APIC assigns the TEP addresses for each switch. Core Links are unnumbered.

Cisco APIC allocates three types of IP addresses from private address space:

- Switch Tunnel End point IP (TEP) - Switches inside a POD share common prefix
- Cisco APIC IP address
- vLeaf Tunnel End point IP (VTEP) - VTEPs behind a Leaf share common prefix

In addition to this you can attach a management station to the Cisco ACI fabric and that management station can talk to the fabric nodes or to Cisco APIC inband on the Tenant called “mgmt”.

You can also configure out of band management, i.e. you can control the mgmt0 IP address for all the Cisco ACI fabric nodes from Cisco APIC. These addresses also belong to the Tenant “mgmt”.

You can access all nodes via ssh to port 1025 and then issue the command “vsh” to get an NX-OS prompt and command line.

Inband Management

In addition, you can create an in-band management network to log into the nodes through the Cisco ACI fabric itself. The configuration is as follows:

```xml
POST http://192.168.10.1/api/policymgr/mo/.xml
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <mgmtMgmtP name="default">
      <mgmtInB name="default" encap="vlan-1324"/>
    </mgmtMgmtP>
  </fvTenant>
</polUni>
```
POST http://192.168.10.1/api/policymgr/mo/.xml
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <!-- Addresses for APIC in-band management network -->
    <fvnsAddrInst name="apicInb" addr="10.13.1.254/24">
      <fvnsUcastAddrBlk from="10.13.1.1" to="10.13.1.3"/>
    </fvnsAddrInst>
    <!-- Addresses for switch in-band management network -->
    <fvnsAddrInst name="switchInb" addr="10.13.1.254/24">
      <fvnsUcastAddrBlk from="10.13.1.17" to="10.13.1.21"/>
    </fvnsAddrInst>
  </fvTenant>
</polUni>

POST http://192.168.10.1/api/policymgr/mo/.xml
<!-- api/policymgr/mo/.xml -->
<infraInfra>
  <!-- Management node group for APICs -->
  <mgmtNodeGrp name="apic">
    <infraNodeBlk name="all" from="1" to="3"/>
    <mgmtRsGrp tDn="uni/infra/funcprof/grp-apic"/>
  </mgmtNodeGrp>
  <!-- Management node group for switches -->
  <mgmtNodeGrp name="switch">
    <infraNodeBlk name="all" from="101" to="105"/>
    <mgmtRsGrp tDn="uni/infra/funcprof/grp-switch"/>
  </mgmtNodeGrp>
</infraInfra>

POST http://192.168.10.1/api/policymgr/mo/.xml
<!-- api/policymgr/mo/.xml -->
<infraInfra>
  <!-- Functional profile -->
  <infraFuncP>
    <!-- Management group for APICs -->
    <mgmtGrp name="apic">
      <!-- In-band management zone -->
      <mgmtInBZone name="default">
        <mgmtRsInbEpg tDn="uni/tn-mgmt/mgmtp-default/inb-default"/>
        <mgmtRsAddrInst tDn="uni/tn-mgmt/addrinst-apicInb"/>
      </mgmtInBZone>
    </mgmtGrp>
  </infraFuncP>
</infraInfra>
Out-of-Band Management

You can assign IP addresses for out-of-band management to the nodes of the Cisco ACI fabric with a single REST call or with a set of REST calls as shown here. Out-of-band management configurations refer to the mgmt0 ports, and they are considered part of the mgmt tenant.

This configuration defines the range of IP addresses to use for the Cisco APIC cluster and for the nodes that are part of the fabric:
POST http://192.168.10.1/api/policymgr/mo/.xml
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <!-- Addresses for APIC out-of-band management network -->
    <fvnsAddrInst name="apicOob" addr="192.168.10.254/24">
      <fvnsUcastAddrBlk from="192.168.10.1" to="192.168.10.3"/>
    </fvnsAddrInst>
    <!-- Addresses for switch out-of-band management network -->
    <fvnsAddrInst name="switchOob" addr="192.168.10.254/24">
      <fvnsUcastAddrBlk from="192.168.10.17" to="192.168.10.21"/>
    </fvnsAddrInst>
  </fvTenant>
</polUni>

This configuration defines the selection of nodes to which you want the configuration to be applied:

POST http://192.168.10.1/api/policymgr/mo/.xml
<!-- api/policymgr/mo/.xml -->
<polUni>
  <infraInfra>
    <!-- Management node group for APICs -->
    <infraNodeGrp name="apic"> <!--
      <infraNodeBlk name="all" from="1" to="3"/> -->
    <infraRsGrp tDn="uni/infra/funcprof/grp-apic"/>
    </infraNodeGrp>
    <!-- Management node group for switches -->
    <infraNodeGrp name="switch">
      <infraNodeBlk name="all" from="101" to="105"/>
      <infraRsGrp tDn="uni/infra/funcprof/grp-switch"/>
    </infraNodeGrp>
  </infraInfra>
</polUni>

This configuration defines the association between nodes and the EPG that represents them:

POST http://192.168.10.1/api/policymgr/mo/.xml
<!-- api/policymgr/mo/.xml -->
<polUni>
  <infraInfra>
    <!-- Functional profile -->
    <infraFuncP>
      <!-- Management group for APICs -->
      <infraGrp name="apic"> <!--
        <infraOoBZone name="default"/>
      </infraGrp>
      <!-- Out-of-band management zone -->
      <infraOoBZone name="default"/>
    </infraFuncP>
  </infraInfra>
</polUni>
This configuration defines the filters to control access to the mgmt0 interface:

```xml
POST http://192.168.10.1/api/policymgr/mo/.xml
<-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <!-- Out-of-band binary contract profile -->
    <vzOOBBrCP name="default">
      <vzSubj name="all">
        <!-- Allow everything -->
        <vzRsSubjFiltAtt tnVzFilterName="default"/>
      </vzSubj>
    </vzOOBBrCP>
    <!-- Management profile -->
    <mgmtMgmtP name="default">
      <!-- Out-of-band EPG -->
      <mgmtOoB name="default">
        <mgmtRsOoBProv tnVzOOBBrCPName="default"/>
      </mgmtOoB>
    </mgmtMgmtP>
    <!-- The other end -->
    <mgmtExtMgmtEntity name="default">
      <mgmtInstP name="default">
        <mgmtSubnet ip="10.30.0.0/16"/>
        <mgmtRsOoBCons tnVzOOBBrCPName="default"/>
      </mgmtInstP>
    </mgmtExtMgmtEntity>
  </fvTenant>
</polUni>
```
You can then verify the IP addresses for the nodes in the GUI as shown in Figure 33.

**Figure 33.** Verifying the IP Addresses for Nodes

**Fabric Bring up Considerations**

As previously indicated, fabric activation is automatic; the administrator only needs to give an ID to each node as the controller discovers it.

You should give the spine switches a number in the top range of the IDs or in the very lowest range (101 to 109), so that all leaf switches are numbered with a continuous range, to make range configurations more readable.

When the switches boot up, they send LLDP packets and a DHCP request. Cisco APIC operates as the TFTP and DHCP server. Cisco APIC provides the switches with a TEP address, switch image, and global configuration.

The infrastructure administrator sees the leaf and spine switches as they are discovered, validates their serial numbers, and decides whether to accept them into the fabric. Figure 34 shows the fabric inventory with the nodes that have been accepted by the administrator.

**Figure 34.** Fabric Inventory
Preprovisioning Switch Profiles for Each Leaf

After bringing up the fabric, the administrator should create two types of switch profile lists:

- One switch profile per leaf switch: For example, switch101, switch102, and switch103 have switch profiles that select leaf 101, leaf 102, and leaf 103.
- One switch profile per pair of leaf switches: For example, leaf 101 and leaf 102 are selected by switch profile switch101and102, and leaf 103 and leaf 104 are selected by switch profile switch103and104.

When you need to add a range of ports to a leaf, you can add it to the profile that you have already defined. Similarly, when you need to add to a vPC, you can simply add the interface profile to the pair of leaf switches that you have predefined.

Figure 35 shows the configuration.

**Figure 35.** Preprovisioning Switch Profiles

Preprovisioning vPC Domains for Each Leaf Pair

As part of the initial configuration, you can divide the leaf switches into pairs of vPC domains by creating a vPC protection policy. You should pair the leaf switches in the same way as you paired them in the switch profiles: that is, you could create vpcdomain1, vpcdomain2, etc., where vpcdomain1 selects leaf switches 101 and 102, vpcdomain2 selects leaf switches 103 and 104, etc. (Figure 36)
Preprovisioning Interface Policies

Interface policies control the configuration of features such as LLDP and Cisco Discovery Protocol.

You should create from the very beginning one object with LLDP enabled and one with LLDP disabled, and you should create one object with Cisco Discovery Protocol enabled and one with Cisco Discovery Protocol disabled, and so on. The advantage of this approach is that later when you configure interfaces, you can simply select a predefined status of LLDP or Cisco Discovery Protocol and so on.

The example in Figure 37 clarifies this approach.

Under Interface Policies, under Link Level, you can preconfigure Fast Ethernet links or Gigabit Ethernet links. Under Cisco Discovery Protocol Interface (CDP Interface), you can specify the configuration for the protocol enabled and for the protocol disabled. Under LACP, you can specify the configuration for LACP active (and define the other options that you want for this configuration: maximum and minimum number of links and so on).
Preprovision Interface Policy Groups

Policy groups can be used as the following:

- Templates of configurations for the interfaces: the collection of features that should be applied to a given interface; these features are a list of pointers to the interface profiles that you defined in the previous section
- Templates of channel groups (when using PortChannel or vPC policy groups)

The most meaningful way to define policy groups is to consider the server types you are planning to connect and then create categories. For instance, you might create categories such as the following:

- Linux Kernel-based Virtual Machine (KVM) servers connected at 1 Gigabit Ethernet without teaming
- Linux KVM servers connected at 1 Gigabit Ethernet with PortChannels
- Microsoft Hyper-V servers connected at 10 Gigabit Ethernet
- Microsoft Hyper-V servers connected at 10 Gigabit Ethernet with PortChannels

For each category of devices, you define the policy group.

Figure 38 shows an example of a policy group that defines a PortChannel with a speed of 1 Gigabit Ethernet on the physical interfaces.
Policy groups also include references to the AEP (you don’t have to add the AEP right away; you can add it or change it later).

Policy groups can be associated with interfaces and with switches by using the interface profiles and the switch profiles.
Preprovisioning Virtual Machine Mobility Domains for Multihypervisor Deployments

Cisco APIC is designed to provide full visibility into virtualized servers and to provide connectivity for virtual machines.

Multiple tenants share virtual machines on the same set of virtualized servers. The VLAN allocation to segment these virtual machines must be dynamic: the virtual machine managers and Cisco APIC negotiate the VLAN tagging that is used.

VMM domains are mobility domains and are not associated with a particular tenant. Instead, you may want to group VMM domains into a single AEP that identifies a common mobility domain for a set of virtual machine managers. For example, one mobility domain may span leaf101 through leaf 110, so the AEP with the VMM domain for VMware vCenter, Linux KVM, and Microsoft System Center Virtual Machine Manager (SCVMM) is applied to all the ports across these leaf switches.

The AEP for the VMM domain must then be attached to the set of interfaces at which virtualized hosts will connect. This attachment is achieved by defining a policy group, an interface profile, and a switch profile.

VMM Domain

A VMM domain is defined as a virtual machine manager and the pool of VLANs and VxLANs that this virtual machine manager is going to use for the purpose of sending traffic to the leaf switches. The VMM domain is associated with an AEP and a policy group and the interfaces at which it is attached to define where virtual machines can move.

In the virtual machine networking view, you can create multiple virtual machine provider domains, which define the virtual machine manager and the data center with which Cisco APIC interfaces, and the VLAN pool that this VMM domain is entitled to use (Figure 40).

The VLAN pool should be dynamic, to allow the virtual machine manager and Cisco APIC to allocate VLANs as needed for the port groups that are going to be used.

Figure 40. Creating Virtual Machine Provider Domains
For each VMM domain, Cisco APIC creates a virtual switch in the hypervisor. For example, in VMware vCenter, if the user configures two VMM domains, which in the example are associated with the same VMware vCenter but with different data centers, Cisco APIC creates two VDSs (Figure 41).

**Figure 41.** For Each VMM Domain, Cisco APIC Creates a Virtual Switch in the Hypervisor

You should create nonoverlapping VLAN ranges for VMM domains that are present on the same leaf switches (Figure 42).
**Figure 42.** Create nonoverlapping VLAN Ranges for VMM Domains on the Same Leaf Switches

For practical reasons, you may want to bundle multiple VMM domains of different types in the same AEP. For instance, your application will likely consist of a mix of workloads: Linus KVM, Microsoft Hyper-V, and VMware vCenter. Each defines a VMM domain, but together these domains are present on the same leaf.

Therefore, you may want to create an AEP named VMM1, which includes:

- VMM domain vCenter Datacenter1
- VMM domain Hyper-V
- VMM domain KVM

Their VLAN ranges are nonoverlapping.

So you might organize your VLAN pools as follows:

- vlan-pool-HyperV1
- vlan-pool-KVM1
- vlan-pool-vCenter1
- vlan-pool-HyperV2
- vlan-pool-KVM2
- vlan-pool-vCenter2
- Etc.

Again, vlan-pool-HyperV1, vlan-pool-KVM1, vlan-pool-vCenter1 etc. are nonoverlapping.
You would have three different VMM domains for the respective virtual machine managers:

- VMM domain HyperV1
- VMM domain vCenter1
- VMM domain KVM1
- Etc.

You would then bundle the three hypervisor types for an application into the same AEP, so you would end with a configuration with these AEPs:

- VMMdomain1 (which consists of VMM domain HyperV1, vCenter1, and KVM1)
- VMMdomain2 (which consists of VMM domain HyperV2, vCenter2, and KVM2)

The AEP provides the domain-to-physical infrastructure connectivity information. It provides the span of the VLAN pool which is associated with the VMM and physical domains on the leaf switches and ports.

The AEP just deploys the VLAN name space (and associated VLANs) on the leaf. The VLANs are not actually provisioned or enabled on the port, so no traffic will flow without EPG provisioning. A particular VLAN is provisioned and enabled on the leaf port based on EPG events: either static EPG binding on a leaf port or LLDP discovery in the case of the VMM domain.

Besides enabling the VLAN name space, the AEP provides the following functions: a VMM domain automatically derives all the policies for the physical interfaces, such as the maximum transmission unit (MTU), LLDP, Cisco Discovery Protocol, and LACP, from the interface policy groups associated with the AEP.

**Preprovisioning Physical Domains for Each Tenant**

In contrast to VMM domains, physical domains can be specific to a given tenant, because in most deployments a physical server belongs to one tenant. You would then identify which set of statically allocated VLANs you want to use for physical server connectivity.

For example, a tenant may want to partition the physical servers into different EPGs; in that case, you would allocate more than one VLAN for this purpose (Figure 43).
As Figure 44 shows, on Cisco APIC you would create one physical domain per tenant, and each physical domain would consist of a static range of VLANs. Similarly, you would have one AEP for each physical domain.

In the examples, the physical domains include physical-workloads-tenant1 and physical-workloads-tenant2, and there is an AEP for tenant1 and an AEP for tenant2.

The AEP for the physical domains must then be attached to the set of interfaces to which the physical servers for the tenant are attached. This attachment is achieved by defining a policy group, an interface profile, and a switch profile.
With this configuration, the infrastructure administrator restricts the port and VLAN that a given tenant can use to connect a physical workload to the virtual topology.

**Preprovisioning an Outside Physical Domain per Tenant**

For each tenant, you should also create a dedicated VLAN or set of VLANs for outside connectivity. You would create something like this:

- Outside physical domain for tenant 1
- Outside physical domain for tenant 2
- Etc.

You would then create an AEP that bundles all these physical domains together as an outside connectivity AEP. The reason for using a single AEP for the outside ports is that multiple tenants will likely share the same set of ports for outside connectivity.

**Automation of Preprovisioning Processes**

All the preceding preprovisioning steps can easily be automated with a single script that takes as input the number of tenants and the IP addresses and credentials of the hypervisors. The only task that you would have to perform periodically is to assign a set of ports to a switch profile (leaf101, leaf102, etc.).

**Configuring a Virtual Topology**

Figure 45 shows a simple network topology.

The topology consists of an inside network with one bridge domain divided into EPGs: EPG A and EPG B. It also includes an extended Layer 2 network that includes local and remote workloads, further divided in EPGs - EPG C and EPG D - and connectivity to the outside through multiple Layer 3 hops with a Layer 3 interface.

This section explains the elements of this simple topology.

**Figure 45. Simple Network Topology**
**Bridge Domain**

The bridge domain can be compared to a giant distributed switch. Cisco ACI preserves the Layer 2 forwarding semantics even if the traffic is routed on the fabric. The TTL is not decremented for Layer 2 traffic, and the MAC addresses of the source and destination endpoints are preserved.

The XML configuration to create the bridge domain is as follows:

```xml
<fvBD name="Tenant1-BD"/>
```

**Hardware Proxy**

By default, Layer 2 unknown unicast traffic is sent to the spine proxy. This behavior is controlled by the hardware proxy option associated with a bridge domain: if the destination is not known, send the packet to the spine proxy; if the spine proxy also does not know the address, discard the packet (default mode).

The implicit configuration is as follows:

```xml
<fvBD arpFlood="no" name="tenant1-BD" unicastRoute="yes" unkMacUcastAct="proxy" unkMcastFlood="yes"/>
```

The advantage of the hardware proxy mode is that no flooding occurs in the fabric. The potential disadvantage is that the fabric has to learn all the endpoint addresses.

With Cisco ACI, however, this is not a concern for virtual and physical servers that are part of the fabric: the database is built for scalability to millions of endpoints. However, if the fabric had to learn all the IP addresses coming from the Internet, it would clearly not scale.

**Flooding Mode**

Alternatively, you can enable flooding mode: if the destination MAC address is not known, flood in the bridge domain. By default, ARP traffic is not flooded but sent to the destination endpoint. By enabling ARP flooding, ARP traffic is also flooded.

The configuration is as follows:

```xml
<fvBD arpFlood="yes" name="VLAN100" unicastRoute="no" unkMacUcastAct="flood" unkMcastFlood="yes"/>
```

This mode of operation is equivalent to that of a regular Layer 2 switch, except that in Cisco ACI this traffic is transported in the fabric as a Layer 3 frame with all the benefits of Layer 2 multipathing, fast convergence, and so on.

Hardware proxy and unknown unicast and ARP flooding are two opposite modes of operation. With hardware proxy disabled and without unicast and ARP flooding, Layer 2 switching would not work.

The advantage of disabling hardware-based proxy and using flooding for unknown hosts and ARP is that the fabric does not need to learn millions of source IP addresses coming from a given port.

**fvCtx**

In addition to the bridge domain, a tenant normally also is configured with a VRF instance for routing. You need to configure more router instances if you need more overlapping IP addresses.

The XML configuration to create the router instance is as follows:

```xml
<fvCtx name="Tenant1-router"/>
```
Endpoint Connectivity

Endpoint connectivity in the virtual network is defined by carving the bridge domain into EPGs and associating these EPGs with either a virtual machine manager or a physical server (static binding).

Connecting a Physical Server

The following configuration example provides connectivity for a physical server to an EPG. EPGs are always part of an application network profile `<fvAp>`, which in this case is called “test”.

Here, `<fvRsPathAtt>` indicates that the physical server connected to port 1/33 on leaf101 can send traffic untagged (mode="native"), and on leaf 101 the traffic from this server is tagged as `vlan-10` (which has local significance). All the traffic from this server is associated with the bridge domain “Tenant1-BD”.

```xml
Method: POST
http://10.51.66.243/api/mo/uni.xml
<polUni>
  <fvTenant dn="uni/tn-Tenant1" name="Tenant1">
    <fvAp name="test">
      <fvAEPg name="EPG-A">
        <fvRsBd tnFvBDName="Tenant1-BD"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/33]" encap="vlan-10" mode="native"/>
      </fvAEPg>
      </fvAp>
    </fvTenant>
  </polUni>
```

If hardware proxy is enabled for the bridge domain, as it is by default, the endpoints are discovered when they send the first frame, and they appear in the operational view under Client Endpoints (Figure 46).

**Figure 46.** Endpoint Discovery with Hardware Proxy Enabled
**Connecting a Virtual Server**

The following configuration example provides connectivity for a virtual server to an EPG. EPGs are always part of an application network profile `<fvAp>` that in this case is called "test".

The EPG called "EPG-A" appears on the virtualized server as a port group named Tenant1|test|EPG-A. The virtual server administrator would then associate the virtual machine with the port group.

```xml
Method: POST
http://10.51.66.243/api/mo/uni.xml
<polUni>
  <fvTenant dn="uni/tn-Tenant1" name="Tenant1">
    <fvAp name="test">
      <fvAEPg name="EPG-A">
        <fvRsBd tnFvBDName="Tenant1-BD" />
        <fvRsDomAtt tDn="uni/vmmp-VMware/dom-vCenter1"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

The virtual machines associated with the EPG show in the Operational view under Client-Endpoints. You also find on which virtualized servers they are located.

**EPG, Bridge Domain and Routing Instance**

For the EPG configuration to work, that is, for endpoints to be discovered and EPG to be propagated to the virtual machine manager, remember to verify the following (Figure 48):

- Associate the EPG with the bridge domain.
- Create a router in the tenant.
- Associate the bridge domain with the router.
- Enable unicast routing (if you want traffic to be routed: that is, if you want a pervasive gateway).
External Connectivity

The Cisco ACI fabric distinguishes internal endpoints from external routes. All the endpoints that are internal to the fabric are known by means of discovery of the endpoint itself. The external routes are known by peering with Open Shortest Path First (OSPF) or Border Gateway Protocol (BGP) with neighboring routers or by configuring static routes.

The configuration of Layer 3 connectivity requires identification of the leaf that will be the border leaf for this specific tenant, the interfaces that will be used, the IP addresses that should be used, and the routing instance of the tenant with which the routes should be associated.

In the configuration example that follows:

- **l3InstP** is an EPG for the traffic coming from the outside.
- **l3extSubnet** is a filter in case the user wants to categorize external traffic into multiple EPGs.
- **fvRsCons** defines the contract that is consumed by this EPG, and as a result it establishes the communication path to internal EPGs.
- **l3extLNodeP** is the location where you put the configuration for the border leaf (for example, for static routes).
- **l3extRsNodeL3OutAtt** identifies a particular leaf that you select as the border leaf for the tenant.
- **l3extLIfP** is the location where you can configure the ports and subinterfaces of the border leaf with IP addresses and so on.
- **l3extRsEctx** is that location where you associate the configuration with the routing instance for the tenant.

```xml
<fvTenant name="Tenant1">
    <l3extOut name="Internet-access-configuration">
        <l3extInstP name="outsideEPGforTenant1">
```
The same configuration can easily be achieved on the GUI (Figure 48).

**Figure 48. Configuring External Connectivity Using the GUI**

**Redundancy**

In Cisco ACI, the infrastructure administrator is in charge of helping ensure that the infrastructure has redundant controllers and that leaf and spine switches are connected for high availability. The tenant administrator is more focused on the logical flow of the configuration: on the sequence of routing and bridging and the relationship between EPGs.

A bridge domain is like a distributed switch, and fvCtx, that is, the routing instance, is like a distributed router, so no particular configuration is needed to help ensure redundancy except for the underlying infrastructure.
For external connectivity, the tenant administrator is responsible for helping ensure that the external uplinks are configured for redundancy.

**Configuring Virtual and Physical Servers Connectivity**

The tenant administrator is the operator responsible for associating EPGs with the ports to which physical servers are connected or with the virtual machine manager to which the EPGs must be extended through port groups.

For virtualized servers, the discovery of the virtual machines correlates with the LLDP or Cisco Discovery Protocol communication between the VDS created by Cisco APIC and the Cisco ACI fabric itself.

**Teaming Considerations for Physical Servers**

A server connects to an access switch with multiple network adapter ports. For a typical rack-mountable server, a typical configuration consists of the following ports:

- **Integrated lights-out (iLO) management port**: Replaces the console and provides the capability to turn the servers on and off
- **Dual lights-out management (LOM) port**: Connected redundantly to the management network
- **Quad-port Gigabit Ethernet adapter**: Used for production traffic

Modern installations consist of servers adopting 10 Gigabit Ethernet adapters with or without the capability to support Fibre Channel over Ethernet (FCoE) as follows:

- iLO
- Dual LOM
- Dual-port 10 Gigabit Ethernet adapter: Used for production traffic

Servers provide several options for teaming, with names that vary according to the vendor. The most common options include:

- **Active-standby mode**: In this mode one port is used during normal operation while the other waits in standby mode for failure scenarios.
- **Active-active transmit load balancing**: With this option, only one network interface card (NIC) can transmit and receive, and all NICs can transmit. This configuration enhances the server transmit performance, but it doesn't improve the receive bandwidth.
- **Static PortChannel**: This option is equivalent to channel-group mode on, which configures PortChannels without any negotiation protocol in place.
- **IEEE 802.3ad PortChannel**: This option enables the negotiation of the PortChannel between the server and the switch, thus allowing the server administrator to know whether the configuration was successful. It also gives the network administrator information about whether the server administrator configured teaming properly.

With vPC support on the switch, the last two options can be deployed with network adapters split onto two different leaf switches, thus achieving increased bandwidth and redundancy at the same time.

In addition, the teamed adapters can be virtualized with VLANs, with each VLAN appearing on the server as a physically separate adapter. This capability allows the consolidation of multiple adapters for increased aggregated bandwidth.
The choice of the teaming option depends on the topology and the configuration of the switching infrastructure. A vPC-based data center enables both static PortChannels and IEEE 802.3ad PortChannels with or without IEEE 802.1q VLAN partitioning on the NIC.

In the event that a given switch topology has not been enabled for vPC, the other teaming options remain valid.

The infrastructure administrator is responsible for the following operations:

- Creating the vPC policy group
- Associating the correct AEP with the policy group

The tenant administrator is responsible for associating the EPG with the path that includes the vPC through a static binding.

The configuration of the vPC domain (to be performed only once per pair of vPC leaf switches) is as follows:

```xml
http://10.51.66.236/api/mo/uni/fabric.xml
<fabricProtPol name="protocolpolicyforvpc">
  <fabricExplicitGEp name="myVpcGrp" id="101">
    <fabricNodePEp id="101"/>
    <fabricNodePEp id="102"/>
  </fabricExplicitGEp>
</fabricProtPol>
```

The configuration of the AEP defines the static range of VLANs that the tenant can use to segment workloads:

```xml
<infraAttEntityP name="PhysicalDomainTenant1">
  <infraRsDomP tDn="uni/phys-VLANdomainTenant1" />
</infraAttEntityP>
```

The configuration of the vPC ports to which servers are connected is as follows:

```xml
http://10.51.66.236/api/mo/uni.xml
<infraInfra dn="uni/infra">
  <infraNodeP name="leafs101and102">
    <infraLeafS name="leafs101and102" type="range">
      <infraNodeBlk name="line1" from_="101" to="102" />
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-portsforvpc101" />
  </infraNodeP>

  <infraAccPortP name="portsforvpc101">
    <infraHPortS name="ports" type="range">
      <infraPortBlk name="line1" fromCard="1" toCard="1" fromPort="7" toPort="7" />
      <infraRsAccBaseGrp tDn="uni/infra/functprof/accbundle-vpcgroup101-1" />
    </infraHPortS>
  </infraAccPortP>

  <infraFuncP>
```

```
</infraInfra>
```
The static binding to this path creates the association of the EPG with the vPC ports:

```
topology/pod-1/protpaths-101-102/pathep-[vpcgroup101-1]
```

The server must also be associated with a VLAN that has local significance, and it is used to associate the traffic with an EPG, as shown in Figure 49. If the server on port 1/33 needs to be plugged into EPG-A, you need to specify a binding to port 1/33, VLAN A, with native mode. The VLAN A number has local significance on the leaf, and any VLAN number is valid as long as it is different than that for VLAN B used to associate port 1/34 with EPG-B.

**Figure 49.** Associating a Server with a VLAN with Local Significance

```
• User Wants Physical Port 1/33 Associated with EPG-A
• Static Binding to Port 1/33 VLAN A
• User Wants Physical Port 1/34 Associated with EPG-B
• Static Binding to Port 1/34 VLAN B
```

The mode keyword "native" indicates that the port is an access port and not a trunk.

```
<polUni>
  <!-- Consumer Tenant Customer1 -->
  <fvTenant dn="uni/tn-CustomerA" name="CustomerA">
    <!-- Application Profile-->  
    <fvAp name="Email">
      <fvAEPg name="SMTP-servers">
        <fvRsBd tnFvBDName="BridgeDomainName" />
        <fvRsPathAtt tDn="topology/pod-1/protpaths-101-102/pathep-[vpcgroup101-1]
          encaps="vlan-10" mode="native"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```
Connecting Virtualized Servers
Connecting Cisco ACI to virtualized servers doesn’t require the use of software on virtualized hosts. This software, which is based on the Cisco Nexus 1000V Switch and is known as the Cisco Application Virtual Switch (AVS), can be used, but it is not necessary. This design guide describes a deployment without any additional software on the virtualized servers.

With the configuration defined in this design guide, EPGs simply become port groups on virtualized hosts.

Hypervisor without a Virtual Machine Manager
If the virtualized servers are used as individual hosts with virtual switches and without a VDS, the configuration consists of creation of multiple port groups - as many as there are EPGs, each with a VLAN.

The EPG static binding would match the leaf, port, and VLAN number as in the case of a nonvirtualized server. The link between the leaf and the server that is a trunk would be configured with mode "regular" instead of "native".

Virtual Distributed Switch
The integration of Cisco ACI with virtualized servers with a VDS provides more control of the virtual environment from Cisco APIC. Cisco APIC aggregates the information from virtualized servers, allowing the administrator to see where virtual machines are located in the fabric, the locations where the virtualized hosts are attached, and so on.

To configure the networking for virtualized servers, perform the following steps:

- Create a VMM domain by specifying a VLAN or VxLAN range, a virtual machine manager IP address, credentials, and the data center under which the VDS must be created. This configuration instantiates a VDS with the Cisco APIC name in the virtual machine manager.
- Associate the VMM domain with an AEP.
- Associate the AEP with a policy group.
- Associate the policy group with the interfaces to which the virtualized servers are attached.
The following is the configuration of the VLAN name space for virtualized hosts:

```xml
<polUni>
<infraInfra>
  <fvnsVlanInstP name="virtualdomain1" allocMode="dynamic">
    <fvnsEncapBlk name="encap" from="vlan-201" to="vlan-300"/>
  </fvnsVlanInstP>
</infraInfra>
<polUni>

The following is the configuration for the VMM domain:

```xml
<polUni>
  <vmmProvP vendor="VMware">
    <vmmDomP name="vCenter1">
      <infraRsVlanNs tDn="uni/infra/vlans-virtualdomain1-dynamic"/>
      <vmmUsrAccP name="vcenter1credentials" usr="root" pwd="vmware"/>
      <vmmCtrlrP name="vcenterappliance" hostOrIp="192.168.64.186">
        <vmmRsAcc tDn="uni/vmmp-VMware/dom-mininet/usracc-vcenter1credentials"/>
      </vmmCtrlrP>
    </vmmDomP>
  </vmmProvP>
<polUni>

The following is the configuration for the AEP:

```xml
<polUni>
  <infraInfra>
    <infraNodeP name="leaf101">
      <infraLeafS name="vmm" type="range">
        <infraNodeBlk name="vmm" from="101" to="101"/>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-esxhostsonleaf101"/>
    </infraNodeP>
    <infraAccPortP name="esxhostsonleaf101">
      <infraHPortS name="port1to10" type="range">
        <infraPortBlk name="line1"
          fromCard="1" toCard="1" fromPort="1" toPort="10"/>
      </infraPortBlk>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-esxportstenant1"/>
    </infraAccPortP>
    <infraHPortS>
      <infraAccPortP>
        <infraFuncP>
          <infraAccPortGrp name="esxportstenant1">
            <infraRsAttEntP tDn="uni/infra/attentp-vCenter1AEP"/>
          </infraAccPortGrp>
        </infraFuncP>
      </infraAccPortP>
    </infraHPortS>
  </infraInfra>
</polUni>
The tenant administrator simply associates the EPG with the VMM domain name. This association automatically propagates the EPG to the VDS with a name that includes the information about the tenant, the application profile, and the EPG as follows: “Tenant|ApplicationProfile|EPG”.

The configuration looks like the following example:

```xml
<infraInfra>
  <infraAttEntityP name="vCenter1AEP">
    <infraRsDomP tDn="uni/vmmp-VMware/dom-vCenter1" />
  </infraAttEntityP>
</infraInfra>
```

The virtual machine administrator associates the virtual machines with the port groups. Cisco ACI provides information identifying the physical ports on which they are located, the hosts on which they are running, etc., as shown in Figure 50.

![Cisco ACI Provides Information About Virtual Machines and Ports](image)

The discovery of the hosts is a result of a combination of control-plane and data-plane learning.

The control-plane learning can be one of the following:

- Out-of-band handshake: VMware vCenter APIs
- In-band handshake: OpFlex-enabled host (Cisco AVS, Microsoft Windows Server 2012, etc.) or through LLDP or Cisco Discovery Protocol
- Data-plane learning can be one of the following:
- Distributed switch learning
- LLDP used to resolve the virtual host ID to the attached port on the leaf node (non-OpFlex hosts)

Figure 51 shows the discovery of the ports to which the VMware ESX hosts are attached.

**Figure 51.** Discovery of Ports to Which VMware ESX Hosts Are Attached

![Diagram showing the discovery of ports to which VMware ESX hosts are attached.]

In the example in the figure:

- The AEP must be mapped to the ports.
- The VMware ESX host discovers the neighbors through LLDP and communicates this information to VMware vCenter, which in turn provides it to Cisco APIC

**Virtual Distributed Switch Across Multiple Layer 2 Hops**

If the virtualized hosts are connected to blade switches or other Layer 2 devices that are in the path between the virtualized host and the leaf, the discovery of the mapping is achieved in the following way:

- The virtualized host discovers the adjacent device through Cisco Discovery Protocol or LLDP (which depends on the VDS configuration).
- The leaf discovers the intermediate device through LLDP or Cisco Discovery Protocol.
- Cisco APIC correlates the two pieces of information to determine where the virtualized host is connected.

At the time of this writing, if you are connecting Cisco UCS B-Series Blade Servers with the Cisco UCS 6100 Series Fabric Interconnects, you need to enable Cisco Discovery Protocol for the virtual NIC (vNIC) in the Cisco UCS Manager service profile. This process requires the VDS to also run Cisco Discovery Protocol.
To enable Cisco Discovery Protocol on the VDS, you can either edit the properties of VDS directly from VMware vCenter or configure the policy group for the virtualized hosts so that Cisco Discovery Protocol is enabled. As a result:

- The leaf detects the fabric interconnect
- The virtualized hosts detect the fabric interconnect
- Cisco APIC correlates the two pieces of information to determine the fabric interconnect ports that are providing connectivity to the virtualized host

**Virtual Distributed Switch Uplink Configuration**

The policy group configuration not only controls the leaf configuration to the virtualized servers, but it also controls the VDS uplink configuration.

For the discovery and policy resolution to work, you need to enable LLDP and, in certain cases, Cisco Discovery Protocol on the VDS as shown in Figure 52.

**Figure 52.** Enabling LLDP and Cisco Discovery Protocol

![Figure 52](image)

**Conclusion**

Cisco ACI provides an advanced datacenter networking methodology that abstracts networking constructs from application deployment. In addition to this it provides a robust set of network telemetry, security, and L4-L7 automation. All of the features and capabilities of ACI are fully automatable through the REST API.

**For More Information**

For more information please refer to the following urls:

http://www.cisco.com/go/aci