ACI Advanced Monitoring and Troubleshooting

Sadiq Memon (CCIE No. 47508)
Joseph Ristaino (CCIE No. 41799)
Carlo Schmidt (CCIE No. 41842)
ACI Advanced Monitoring and Troubleshooting

Sadiq Memon, Joseph Ristaino, Carlo Schmidt

Copyright© 2021 Cisco Systems, Inc.

Published by: Cisco Press

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

Library of Congress Control Number: 2020941959


Warning and Disclaimer

This book is designed to provide information about in-depth monitoring and troubleshooting techniques related to Cisco’s Application Centric Infrastructure (ACI) to guide readers in learning to design, deploy, and maintain the ACI fabric. This book can also help in preparing and attaining advanced certification such as CCIE Data Center. This book was written based on ACI Release 3.2(-) as that release was the preferred long-lived release over the course of developing the content. Therefore, the vast majority of features and examples covered in the book reference ACI Release 3.2(-), and they can still be applied to later releases. However, newer features are identified where applicable, along with the supported version in order to provide more in-depth information. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an “as is” basis. The authors, Cisco Press, and Cisco Systems, Inc. shall have neither liability for nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided “as is” without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services. The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. Screenshots and icons reprinted with permission from the Microsoft Corporation. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.
Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc., cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through email at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Editor-in-Chief: Mark Taub
Alliances Manager, Cisco Press: Ron Fligge
Product Line Manager: Brett Bartow
Executive Editor: James Manly
Managing Editor: Sandra Schroeder
Development Editor: Christopher A. Cleveland
Senior Project Editor: Lori Lyons
Copy Editor: Kitty Wilson

Technical Editors: Mioljub Jovanovic, Joe LeBlanc
Editorial Assistant: Cindy Teeters
Cover Designer: Chuti Prasertsith
Production Manager: Aswini Kumar, codeMantra
Composition: codeMantra
Indexer: Cheryl Ann Lenser
Proofreader: Gill Editorial Services
About the Authors

Sadiq Memon, CCIE No. 47508, is a Lead Solutions Integration Architect (Automotive) with Cisco Customer Experience (CX). He has over 30 years of diversified experience in information technology with specialization and expertise in data center and enterprise networking. Sadiq joined Cisco in 2007, and as a Cisco veteran of over 13 years, he has worked with various large enterprise customers, including automotive, financials, manufacturing, and government in designing, implementing, and supporting end-to-end architectures and solutions. Sadiq was part of the Cisco Advanced Services Tiger Team during the early ACI incubation period. He has published a series of short videos covering ACI configuration on YouTube and has presented ACI/Cloud-related topics at Cisco Live! Sadiq was the technical editor for the Cisco Press book Deploying ACI and possesses multiple IT industry certifications from leading companies such as Cisco (CCIE, CCNA), VMware (VCP-DCV), Microsoft, and Citrix. Sadiq holds a bachelor's degree in computer systems engineering from NED University of Engineering & Technology, Karachi, Pakistan.

Joseph Ristaino, CCIE No. 41799, is a Technical Leader with the ACI Escalation Team in RTP, North Carolina. He joined Cisco in 2011 after graduating from Wentworth Institute of Technology with a bachelor's degree in computer networking. Joseph started with Cisco on the Server Virtualization TAC team, specializing in UCS and virtualization technologies. He has in-depth knowledge of compute/networking technologies and has been supporting customers for over eight years as they implement and manage data center deployments around the globe. Joseph now works closely with the ACI Technical Support teams to provide assistance on critical customer issues that go unsolved and has been working on ACI since its inception in 2014. Joseph lives with his wife in Durham, North Carolina.

Carlo Schmidt, CCIE No. 41842, is a Data Center Solutions Architect. He works with global enterprises, designing their next-generation data centers. Carlo started at Cisco in 2011, on the Data Center Switching TAC team. In that role, he focused on Nexus platforms and technologies such as FCoE, fabric path, and OTV. In 2016, he migrated to the ACI TAC team, where he specialized in customer problem resolution as well as improving product usability. In 2019 Carlo decided to take his knowledge and lessons learned from his eight years in Cisco TAC to a presales role as a Solutions Architect. Carlo is based out of Research Triangle Park, North Carolina.
About the Technical Reviewers

**Mioljub Jovanovic**, CCIE No. 17631, is certified in Routing & Switching and in Data Center. He is a Principal Engineer at Cisco Systems, working for Customer Experience organization, with more than 20 years of professional experience with Cisco networking products in solutions. Among other responsibilities, Mio’s role included training and support for initial ACI global deployments. Between 2015 and 2019, he presented ACI training and troubleshooting sessions at multiple Cisco Live conferences and other technical seminars.

As a Data Center Technical Leader in the CX DC EMEAR group, Mio coached and mentored Cisco support engineers on ACI, HyperFlex, CNAE, Tetration, FlexPod, vBlock and solutions involving Cisco UCS and Nexus and MDS platforms. Prior to his TL role in the DC Solutions team, Mio worked as Network Management senior TAC engineer, specializing in SNMP and network services platforms. Mio’s passions are Service Assurance, Day-2 Operations, Model-Driven Telemetry, Linux, Angular, and Python.

**Joe LeBlanc**, CCIE No. 41523, is a Technical Leader in the Intent-Based Networking Group at Cisco Systems. He has been supporting ACI customer escalations in the engineering group since FCS of the solution in 2014. Prior to that role, Joe worked in the Technical Assistance Center on the Server Virtualization team, supporting UCS and Nexus 1000v products.
Dedications

Sadiq H Memon:
This book is dedicated to my parents, Abdul Majeed Memon and Saeeda Memon, for their day and night gracious prayers. My beloved wife, Nazish Memon, and my kids, Nibras Memon, Ali Memon, and Ahmed Memon, for their utmost support and encouragement throughout the extended period of writing this book. My management at Cisco for their continuous support in excelling my career. And last but not least, the trust and support from all my auto customers, especially from Tony Cataldo (Manager Network Engineering from a renowned U.S.-based auto company). Without all their support, I don't think I would have been able to propose and author this book successfully.

Joseph Ristaino:
This book is dedicated to my wife, Katie, for her endless support, and to all the friends I've made at Cisco. Because of them, this book has become a reality.

Carlo Schmidt:
I dedicate this book to all the amazing mentors, managers, and coworkers who have supported me during my time at Cisco. Without their encouragement, and their countless after-work hours teaching me how to become a better engineer, I would have never had the ability to co-author this book with Sadiq and Joey. I also dedicate this book to my wife, Ally, who supported me through many late nights of researching, writing, and reviewing.
Acknowledgments

We would like to specially thank the technical editors Mioljub Jovanovic and Joe LeBlanc for providing their expert-level technical knowledge in editing the book. Being well-known for their subject matter expertise in Cisco and outside with ACI technology, both of the technical editors paid close attention in reviewing the material and were very blunt in identifying our mistakes and shortcomings; they helped make the content accurate and valuable for readers.

We would also like to acknowledge and appreciate Cisco Data Center Business Unit (DCBU) for the guidance and knowledge sharing. As the owner and developer of ACI, Cisco DCBU has empowered us to learn and excel this technology since the day of its inception, and this helped us successfully finish up this book. Cisco DCBU included us as part of Cisco Tiger Team in learning and developing the initial content of ACI before it was even publicly announced.

Big applause goes out to the production team for this book. James Manly and Christopher Cleveland have been incredibly professional and a pleasure to work with. This book could not have been completed successfully without their constant push and support.

Last but not least, we would like to acknowledge the services and support of our beloved friend and coworker Andy Gossett, whose in-depth technical expertise has not only helped in writing this book but has in general been a great help to Cisco teams and the extended customer base.
## Contents at a Glance

Foreword by Yusuf Bhaiji  xxviii
Foreword by Ronak Desai  xxix
Introduction  xxx

### Part I  Introduction to ACI

- **Chapter 1**  Fundamental Functions and Components of Cisco ACI  1
- **Chapter 2**  Introduction to the ACI Policy Model  31
- **Chapter 3**  ACI Command-Line Interfaces  67
- **Chapter 4**  ACI Fabric Design Options  85
- **Chapter 5**  End Host and Network Connectivity  185
- **Chapter 6**  VMM Integration  249
- **Chapter 7**  L4/L7 Service Integration  299
- **Chapter 8**  Automation and Orchestration  343

### Part II  Monitoring and Management Best Practices

- **Chapter 9**  Monitoring ACI Fabric  405
- **Chapter 10**  Network Management and Monitoring Configuration  509

### Part III  Advanced Forwarding and Troubleshooting Techniques

- **Chapter 11**  ACI Topology  589
- **Chapter 12**  Bits and Bytes of ACI Forwarding  611
- **Chapter 13**  Troubleshooting Techniques  717
- **Chapter 14**  The ACI Visibility & Troubleshooting Tool  771
- **Chapter 15**  Troubleshooting Use Cases  791

### Appendix A  Answers to Chapter Review Questions  861

Index  873
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword by Yusuf Bhaiji</td>
<td>xxviii</td>
</tr>
<tr>
<td>Foreword by Ronak Desai</td>
<td>xxix</td>
</tr>
<tr>
<td>Introduction</td>
<td>xxx</td>
</tr>
</tbody>
</table>

### Part I  Introduction to ACI

#### Chapter 1  Fundamental Functions and Components of Cisco ACI  1

- ACI Building Blocks  8
  - Hardware Specifications  8
    - Nexus 9000 Platform  9
  - APIC Controller  12
- ACI Key Concepts  14
  - Control Plane  15
  - Data Plane  17
  - VXLAN  17
- Tenant  18
- VRF  19
- Application Profile  20
- Endpoint Group  21
- Contracts  22
- Bridge Domain  24
- External Routed or Bridged Network  25

### Chapter 2  Introduction to the ACI Policy Model  31

- Key Characteristics of the Policy Model  32
  - Management Information Tree (MIT)  33
- Benefits of a Policy Model  37
- Logical Constructs  37
- Tenant Objects  38
- VRF Objects  39
- Application Profile Objects  40
- Endpoint Group Objects  41
Chapter 4  ACI Fabric Design Options  85

Physical Design  85
  Single-Versus Multiple-Fabric Design  87
  Dark Fiber  90
  Dense Wavelength-Division Multiplexing (DWDM)  92
  Ethernet over MPLS (EoMPLS) Pseudowire  92
  Multi-Pod  97
  ACI Multi-Pod Use Cases  100
  ACI Multi-Pod Scalability  103
  Inter-Pod Connectivity Deployment Considerations  104
  APIC Cluster Deployment Considerations  113
  Multi-Site  116
  Cisco ACI Multi-Site Orchestrator  120
  Cisco ACI Multi-Site Deployment Considerations  122
  Migration Scenarios  124
  Deployment Best Practices  128
  General Best Practices for Cisco ACI Multi-Site Design  129
  Remote Leaf  131
  Hardware and Software Support  134
  Recommended QOS Configuration for a Remote Leaf Solution  134
  Discovery of a Remote Leaf  136
  Remote Leaf Control Plane and Data Plane  138
  Remote Leaf Design Considerations  141
  ACI Multi-Pod and Remote Leaf Integration  143

Logical Design  149
  Design 1: Container-as-a-Service Using the OpenShift Platform and Calico CNI  149
  Business Case  149
  Design Solution  150
  Design 2: Vendor-Based ERP/SAP Hana Design with ACI  165
  Business Case  165
  Design Solution  165
  Design 3: vBrick Digital Media Engine Design with ACI  175
  Business Case  176
  Design Solution  176
Summary 180
Review Key Topics 181
Review Questions 181

Chapter 5  End Host and Network Connectivity 185

End Host Connectivity 185
VLAN Pool 186
Domain 186
Attachable Access Entity Profiles (AAEPs) 186
Switch Policies 187
Switch Policy Groups 187
Switch Profiles 187
Interface Policies 188
Interface Policy Groups 188
Interface Profiles 189
Virtual Port Channel (VPC) 191
Configuring VPC 192
Defining the VPC Domain 193
Creating an Interface Policy 195
Creating a Switch Profile 196
Port Channel 197
Configuring a Port Channel 198
Access Port 201
Configuring an Access Port 202
Best Practices in Configuring Access Policies 206
Policy Best Practices 206
Domain Best Practices 206
AAEP Best Practices 207
Compute and Storage Connectivity 207
FEX Connectivity 207
Cisco Blade Chassis Servers UCS B-Series 208
Standalone Rack-Mount Servers 209
Connecting Storage in ACI 209
L4/L7 Service Device Connectivity 210
Connecting Firewalls 211
Connecting Load Balancers 212
Network Connectivity 213
   Connecting an External Bridge Network 213
   Extending EPGs Outside the ACI Fabric 213
   Extending an ACI Bridge Domain Outside the Fabric 216
   Connecting an External Routed Network 218
   External Layer 3–Supported Routing Protocols 220
   Configuring MP-BGP Spine Route Reflectors 221
   Configuring External Routed Networks 222
   GOLF 227
   Network Connectivity Between Pods and Sites 228
   IPN Connectivity Considerations for Remote Leafs 237
Diagnosing Connectivity Problems 242
Summary 245
Review Questions 245

Chapter 6  VMM Integration 249

Virtual Machine Manager (VMM) 249
   VMM Domain Policy Model 250
   VMM Domain Components 250
   VMM Domains 250
   VMM Domain VLAN Pool Association 252
   Attachable Access Entity Profile Association 252
   VMM Domain EPG Association 253
   EPG Policy Resolution and Deployment Immediacy 255
VMware Integration 257
   Prerequisites for VMM Integration with AVS or VDS 257
   Guidelines and Limitations for VMM Integration with AVS or VDS 257
   ACI VMM Integration Workflow 258
   Publishing EPGs to a VMM Domain 258
   Connecting Virtual Machines to the Endpoint Group Port Groups on vCenter 259
   Verifying VMM Integration with the AVS or VDS 259
   Verifying the Virtual Switch Status 259
   Verifying the vNIC Status 260
Microsoft SCVMM Integration 260
   Mapping ACI and SCVMM Constructs 261
Mapping Multiple SCVMMs to an APIC 262
Verifying That the OpFlex Certificate Is Deployed for a Connection from
the SCVMM to the APIC 262
Verifying VMM Deployment from the APIC to the SCVMM 263
OpenStack Integration 263
Extending OpFlex to the Compute Node 264
ACI with OpenStack Physical Architecture 264
OpFlex Software Architecture 265
OpenStack Logical Topology 265
Mapping OpenStack and ACI Constructs 266
Prerequisites for OpenStack and Cisco ACI 267
Guidelines and Limitations for OpenStack and Cisco ACI 268
Verifying the OpenStack Configuration 270
Configuration Examples for OpenStack and Cisco ACI 271
Kubernetes Integration 272
Planning for Kubernetes Integration 272
Prerequisites for Integrating Kubernetes with Cisco ACI 273
Provisioning Cisco ACI to Work with Kubernetes 274
Preparing the Kubernetes Nodes 277
Installing Kubernetes and Cisco ACI Containers 279
Verifying the Kubernetes Integration 280
OpenShift Integration 281
Planning for OpenShift Integration 282
Prerequisites for Integrating OpenShift with Cisco ACI 283
Provisioning Cisco ACI to Work with OpenShift 284
Preparing the OpenShift Nodes 287
Installing OpenShift and Cisco ACI Containers 290
Updating the OpenShift Router to Use the ACI Fabric 291
Verifying the OpenShift Integration 291
VMM Integration with ACI at Multiple Locations 292
Multi-Site 292
Multiple Virtual Machine Managers Across Sites 292
Single Virtual Machine Manager Across Sites 295
Remote Leaf 295
Summary 298
Chapter 7 L4/L7 Service Integration  299

Service Insertion  299
The Service Graph  300
  Managed Mode Versus Un-Managed Mode  301
  L4–L7 Integration Use Cases  302
  How Contracts Work in ACI  303
The Shadow EPG  306
Configuring the Service Graph  307
  Step 1: Create an L4–L7 Device  307
  Step 2: Create a Service Graph Template  308
  Step 3: Deploy the Service Graph from the Template  308
  Step 4: Configure the L4–L7 Parameters (Managed Mode Only)  310
Verifying the Service Graph Configuration  310
Service Graph Design and Deployment Options  312
  Firewall as Default Gateway for Client and Server (Routed Mode)  312
  Firewall Not the Default Gateway for Clients (Routed Mode)  312
  Route Peering with a Firewall (Routed Mode)  314
  Service Graph with Firewall (Transparent Mode)  316
  Service Graph with ADC (One-Arm Mode with S-NAT)  316
  Service Graph with ADC (Two-Arm Mode)  316
  Service Graph with Two Service Nodes (Firewall with NAT and ADC in Two-Arm Mode)  317
  Service Graph with Two Service Nodes (Firewall with No NAT and ADC in Two-Arm Mode)  319
  Service Graph with Two Service Nodes (Firewall with No NAT and ADC in One-Arm Mode)  319
  Service Graph with an Intrusion Prevention System (IPS)  319
Policy-Based Redirect (PBR)  322
  PBR Design Considerations  323
  PBR Design Scenarios  324
  PBR Service Graph with an ADC (One-Arm Mode and No S-NAT)  324
  PBR Service Graph with a Firewall (Two-Arm Mode and Routed)  324
Configuring the PBR Service Graph  325
Service Node Health Check 326
L4–L7 PBR Tracking 326
L4–L7 PBR Threshold 326
L4–L7 PBR Health Groups 327
Common Issues in the PBR Service Graph 328
Unnecessary Layer 2 Traffic Redirection Toward the Service Node 328
Inability to Ping the Consumer Connector 329
Routing on a Service Node 330
L4/L7 Service Integration in Multi-Pod and Multi-Site 332
Multi-Pod 332
Anycast Services in Multi-Pod 334
Multi-Site 338
Review Questions 342

Chapter 8  Automation and Orchestration 343

The Difference Between Automation and Orchestration 343
Benefits of Automation and Orchestration 344
Example 1 345
Example 2 347

REST API 349
Automating Tasks Using the Native REST API: JSON and XML 351
API Inspector 351
Object (Save As) 353
Visore (Object Store Browser) 355
MOQuery 357
Automation Use Cases 364

Automating Tasks Using Ansible 372
Ansible Support in ACI 375
Installing Ansible and Ensuring a Secure Connection 378
APIC Authentication in Ansible 382
Automation Use Cases 384
Use Case 1 384
Use Case 2 388
Orchestration Through UCS Director  392
   Management Through Cisco UCS Director  392
   Automation and Orchestration with Cisco UCS Director  393
   Automation Use Cases  395
Summary  402
Review Questions  402

Part II  Monitoring and Management Best Practices

Chapter 9  Monitoring ACI Fabric  405
Importance of Monitoring  405
Faults and Health Scores  407
   Faults  407
   Health Scores  411
   Health Score Used in Proactive Monitoring  413
   Health Score Used in Reactive Monitoring  414
   Health Score with Interface Errors  414
ACI Internal Monitoring Tools  415
   SNMP  415
      Interface Failures Example  418
   Syslog  420
       Example: Leaf Membership Failure  423
       Example: Spine/IPN Failure  423
   NetFlow  426
       Example: Network Visibility on a Border Leaf  428
ACI External Monitoring Tools  430
   Network Insights  430
      Network Insights for Resources (NIR)  431
      Network Insights Advisor (NIA)  432
       Example: Application Intermittent Disconnect Issue (Standalone Compute)  433
       Example: Application Connectivity Issue (Virtual Compute)  435
   Network Assurance Engine  437
   NAE Installation  439
   NAE Configuration and Initial Setup  440
       Example: Subnet Reachability Issue  450
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tetration</td>
<td>453</td>
</tr>
<tr>
<td>Software Agents</td>
<td>455</td>
</tr>
<tr>
<td>Hardware Agents</td>
<td>455</td>
</tr>
<tr>
<td>Tetration Installation and Configuration</td>
<td>455</td>
</tr>
<tr>
<td>Tetration System Monitoring</td>
<td>461</td>
</tr>
<tr>
<td>Configuring Email Alerts</td>
<td>463</td>
</tr>
<tr>
<td>Enabling Syslog</td>
<td>464</td>
</tr>
<tr>
<td>Tetration Scopes</td>
<td>465</td>
</tr>
<tr>
<td>Tetration Applications</td>
<td>465</td>
</tr>
<tr>
<td>Tetration Code Upgrades</td>
<td>467</td>
</tr>
<tr>
<td>Tetration Patch Upgrade</td>
<td>467</td>
</tr>
<tr>
<td>Tetration Cluster Reboot</td>
<td>469</td>
</tr>
<tr>
<td>Tetration Cluster Shutdown</td>
<td>469</td>
</tr>
<tr>
<td>Example: Workload Security with Tetration</td>
<td>470</td>
</tr>
<tr>
<td>Monitoring Through the REST API</td>
<td>473</td>
</tr>
<tr>
<td>Monitoring an APIC</td>
<td>475</td>
</tr>
<tr>
<td>Monitoring CPU and Memory</td>
<td>475</td>
</tr>
<tr>
<td>Monitoring Disk Utilization</td>
<td>477</td>
</tr>
<tr>
<td>Monitoring Interfaces</td>
<td>478</td>
</tr>
<tr>
<td>Monitoring the APIC Cluster State</td>
<td>481</td>
</tr>
<tr>
<td>Monitoring Leafs and Spines</td>
<td>482</td>
</tr>
<tr>
<td>Monitoring CPU Utilization</td>
<td>482</td>
</tr>
<tr>
<td>Monitoring Memory Utilization</td>
<td>485</td>
</tr>
<tr>
<td>Monitoring Power Supply Unit (PSU) Status</td>
<td>486</td>
</tr>
<tr>
<td>Monitoring Fan Status</td>
<td>488</td>
</tr>
<tr>
<td>Monitoring Module Status</td>
<td>489</td>
</tr>
<tr>
<td>Monitoring Leaf/Spine Membership Status in a Fabric</td>
<td>491</td>
</tr>
<tr>
<td>Monitoring Interface Status</td>
<td>496</td>
</tr>
<tr>
<td>Monitoring Applications</td>
<td>499</td>
</tr>
<tr>
<td>Monitoring Application Traffic Status</td>
<td>499</td>
</tr>
<tr>
<td>Monitoring External Network Connectivity</td>
<td>502</td>
</tr>
<tr>
<td>Monitoring the PBR Service Graph</td>
<td>504</td>
</tr>
<tr>
<td>Summary</td>
<td>505</td>
</tr>
<tr>
<td>Review Questions</td>
<td>506</td>
</tr>
</tbody>
</table>
Chapter 10  Network Management and Monitoring Configuration  509

Out-of-Band Management  509
   Creating Static Management Addresses  510
   Creating the Management Contract  510
   Choosing the Node Management EPG  513
   Creating an External Management Entity EPG  513
   Verifying the OOB Management Configuration  515

In-Band Management  517
   Creating a Management Contract  517
   Creating Leaf Interface Access Policies for APIC INB Management  518
   Creating Access Policies for the Border Leaf(s) Connected to L3Out  520
   Creating INB Management External Routed Networks (L3Out)  522
   Creating External Management EPGs  524
   Creating an INB BD with a Subnet  527
   Configuring the Node Management EPG  529
   Creating Static Management Addresses  530
   Verifying the INB Management Configuration  530

AAA  533
   Configuring Cisco Secure ACS  533
   Configuring Cisco ISE  542
   Configuring AAA in ACI  547
   Recovering with the Local Fallback User  550
   Verifying the AAA Configuration  550

Syslog  551
   Verifying the Syslog Configuration and Functionality  555

SNMP  556
   Verifying the SNMP Configuration and Functionality  562

SPAN  566
   Access SPAN  567
   Fabric SPAN  571
   Tenant SPAN  572
   Ensuring Visibility and Troubleshooting SPAN  575
   Verifying the SPAN Configuration and Functionality  576
NetFlow 577
- NetFlow with Access Policies 580
- NetFlow with Tenant Policies 582
- Verifying the NetFlow Configuration and Functionality 585
Summary 587

Part III Advanced Forwarding and Troubleshooting Techniques

Chapter 11 ACI Topology 589
- Physical Topology 589
- APIC Initial Setup 593
- Fabric Access Policies 595
  - Switch Profiles, Switch Policies, and Interface Profiles 595
  - Interface Policies and Policy Groups 596
  - Pools, Domains, and AAEPs 597
- VMM Domain Configuration 601
- VMM Topology 601
- Hardware and Software Specifications 603
- Logical Layout of EPGs, BDs, VRF Instances, and Contracts 605
  - L3Out Logical Layout 606
Summary 608
Review Key Topics 608
References 609

Chapter 12 Bits and Bytes of ACI Forwarding 611
- Limitations of Traditional Networks and the Evolution of Overlay Networks 611
- High-Level VXLAN Overview 613
- IS-IS, TEP Addressing, and the ACI Underlay 615
  - IS-IS and TEP Addressing 615
  - FTags and the MDT 618
- Endpoint Learning in ACI 626
  - Endpoint Learning in a Layer 2–Only Bridge Domain 627
  - Council of Oracle Protocol (COOP) 632
  - Updating the Managed Object (MO) Tree 634
  - Endpoint Learning in a Layer 3–Enabled Bridge Domain 635
  - Fabric Glean 640
Remote Endpoint Learning 641
Endpoint Mobility 645
Anycast Gateway 647
Virtual Port Channels in ACI 649
Routing in ACI 651
Static or Dynamic Routes 651
Learning External Routes in the ACI Fabric 656
Transit Routing 659
Policy Enforcement 661
Shared Services 664
L3Out Flags 668
Quality of Service (QoS) in ACI 669
Externally Set DSCP and CoS Markings 671
EPG QoS 671
Custom QoS Policy 671
Contract QoS 671
CoS Preservation in ACI 672
iTraceroute Class 672
QoS and Multi-Pod 672
DSCP Class-to-CoS Translation Policy 674
Multi-Pod 674
Multi-Site 680
Remote Leaf 684
Forwarding Scenarios 686
ARP Flooding 686
Layer 2 Known Unicast 688
ARP Optimization 690
Layer 2 Unknown Unicast Proxy 690
L3 Policy Enforcement When Going to L3Out 693
L3 Policy Enforcement for External Traffic Coming into the Fabric 695
Route Leaking/Shared Services 695
Consumer to Provider 695
Provider to Consumer 698
Multi-Pod Forwarding Examples 698
  ARP Flooding 700
  Layer 3 Proxy Flow 700
Multi-Site Forwarding Examples 703
  ARP Flooding 703
  Layer 3 Proxy Flow 705
Remote Leaf 707
  ARP Flooding 707
  Layer 3 Proxy Flow 710
Summary 713
Review Key Topics 713
References 714
Review Questions 714

Chapter 13 Troubleshooting Techniques 717
General Troubleshooting 717
  Faults, Events, and Audits 718
  moquery 722
  iCurl 724
  Visore 726
Infrastructure Troubleshooting 727
  APIC Cluster Troubleshooting 727
  Fabric Node Troubleshooting 734
How to Verify Physical- and Platform-Related Issues 737
  Counters 737
  CPU Packet Captures 743
  ASIC 744
  ASIC Interface 744
  Application 745
  SPAN 748
Troubleshooting Endpoint Connectivity 751
  Endpoint Tracker and Log Files 752
  Enhanced Endpoint Tracker (EPT) App 756
  Rogue Endpoint Detection 758
Troubleshooting Contract-Related Issues 759
  Verifying Policy Deny Drops 764
Embedded Logic Analyzer Module (ELAM) 765
Summary 769
Review Key Topics 769
Review Questions 769

Chapter 14 The ACI Visibility & Troubleshooting Tool 771
Visibility & Troubleshooting Tool Overview 771
Faults Tab 772
Drop/Stats Tab 773
  Ingress/Egress Buffer Drop Packets 774
  Ingress Error Drop Packets Periodic 774
  Storm Control 774
  Ingress Forward Drop Packets 775
  Ingress Load Balancer Drop Packets 776
Contract Drops Tab 777
  Contracts 777
  Contract Considerations 778
Events and Audits Tab 779
Traceroute Tab 780
Atomic Counter Tab 782
Latency Tab 785
SPAN Tab 786
Network Insights Resources (NIR) Overview 787
Summary 790

Chapter 15 Troubleshooting Use Cases 791
Troubleshooting Fabric Discovery: Leaf Discovery 792
  Solution 794
Troubleshooting APIC Controllers and Clusters: Clustering 795
  Solution 798
Troubleshooting Management Access: Out-of-Band EPG 799
  Solution 801
Troubleshooting Contracts: Traffic Not Traversing a Firewall as
  Expected 801
  Solution 803
Troubleshooting Contracts: Contract Directionality   804
   Solution   807
Troubleshooting End Host Connectivity: Layer 2 Traffic Flow Through ACI   807
   Solution   810
Troubleshooting External Layer 2 Connectivity: Broken Layer 2 Traffic Flow Through ACI   812
   Solution 1   813
   Solution 2   813
Troubleshooting External Layer 3 Connectivity: Broken Layer 3 Traffic Flow Through ACI   814
   Solution   816
Troubleshooting External Layer 3 Connectivity: Unexpected Layer 3 Traffic Flow Through ACI   816
   Solution   820
Troubleshooting Leaf and Spine Connectivity: Leaf Issue   821
   Solution   822
Troubleshooting VMM Domains: VMM Controller Offline   826
   Solution 1   829
   Solution 2   829
Troubleshooting VMM Domains: VM Connectivity Issue After Deploying the VMM Domain   829
   Solution 1   830
   Solution 2   831
   Solution 3   831
Troubleshooting L4–L7: Deploying an L4–L7 Device   832
   Solution   834
Troubleshooting L4–L7: Control Protocols Stop Working After Service Graph Deployment   834
   Solution   836
Troubleshooting Multi-Pod: BUM Traffic Not Reaching Remote Pods   837
   Solution 1   839
   Solution 2   839
Troubleshooting Multi-Pod: Remote L3Out Not Reachable   839
   Solution   841
Troubleshooting Multi-Site: Using Consistency Checker to Verify State at Each Site   841
   Solution   842
Troubleshooting Programmability Issues: JSON Script Generates Error  844
   Solution  844
Troubleshooting Multicast Issues: PIM Sparse Mode Any-Source Multicast (ASM)  846
   Solution  847
Summary  860

Appendix A  Answers to Chapter Review Questions  861

Index  873
Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a `show` command).
- **Italic** indicates arguments for which you supply actual values.
- Vertical bars (`|`) separate alternative, mutually exclusive elements.
- Square brackets (`[]`) indicate an optional element.
- Braces (`{}`) indicate a required choice.
- Braces within brackets (`{{ }}`) indicate a required choice within an optional element.
Foreword by Yusuf Bhaiji

*ACI Advanced Monitoring and Troubleshooting* is an excellent self-study material for the latest blueprint of CCIE Data Center certification exam (v3.0). Whether you are studying to attain CCIE certification or are just seeking to gain a better understanding of Cisco ACI technology in designing, implementing, maintaining, and troubleshooting, you will benefit from the information presented in this book.

The authors have used a unique approach in explaining concepts and the architecture of the ACI technology carefully crafted into an easy-to-follow guide. The book provides readers a comprehensive and all-inclusive view of the entire range of Cisco ACI solutions in a single binder.

As an early-stage exam-preparation guide, this book presents a detailed and comprehensive introduction to the technologies used to build scalable software-defined networks and also covers the topics defined in the CCIE exam blueprint.

Cisco Press books are designed to help educate, develop, and excel the community of IT professionals in not only traditional networking technologies but also in today’s state-of-the-art software-defined networking techniques.

Most networking professionals use a variety of learning methods to keep them up to the mark with the latest technologies. Cisco Press titles are a prime source of content for some individuals and can also serve as an excellent supplement to other forms of learning. Training classes, whether delivered in a classroom or online, are a great way to quickly acquire knowledge on newer technologies. Hands-on practice is essential for anyone seeking to build or acquire new skills.

The author (Sadiq Hussain Memon) and his co-authors have a very distinct style and have proven their skills by writing on a difficult subject using real-world examples and use cases. A must-read and an essential part of your exam preparation toolkit and a valuable addition to your personal library.

**Yusuf Bhaiji**
Director of Certifications
Cisco Systems
Foreword by Ronak Desai

When Cisco built the Application Centric Infrastructure (ACI), it expanded the influence of Data Center operators by providing them with an agile and accessible framework on which they could build and operate their networks. My own journey with Cisco Data Center began soon after I joined the company in 2002, when it acquired Andiamo, where I was a lead engineer. After joining Cisco, I worked on building the MDS 9000 and Nexus 7000 series, which evolved into the first line of products for Cisco’s then-new Data Center business unit. After successfully delivering MDS and Nexus I was asked to be founding employee on the ACI team and have been driving engineering there since day one.

In the past eight years, I have seen the ACI products mature and become part of the critical infrastructure for hospitals, emergency systems, banks, mobile networks, and large-scale enterprises. “ACI Anywhere” is recognized as the best SDN solution for private and public cloud.

So, I am honored to be the one to introduce you to this book, which will help you take the best advantage of this powerful networking platform.

Throughout my years at Cisco, I have pleasure to work with Sadiq Memon, Joey Ristaino, and Carlo Schmidt countless occasions. As invaluable members of the Data Center Networking Group, and their collective experience with the ACI solution, makes them incredible resources to anyone who wants to learn about the ins and outs of the infrastructure.

This book is accessible to network professionals just beginning with ACI, as well as to ACI veterans looking for insight and advanced tips. Readers seeking a deeper analysis can opt to dive into later chapters where the authors collaborate with technical engineers to effectively communicate key technical concepts. Here, readers can build upon their foundational knowledge with more hands-on application-based learning.

Readers will also find valuable the advice based on personal experiences and challenges our authors faced in the data center field. These vignettes provide readers with in-depth examinations into real-world cases with step-by-step instructions and troubleshooting advice. Even readers familiar with the ACI fabric will find that they can extend their knowledge with these critical insights into ACI monitoring and troubleshooting.

By the end of this book, engaged readers will be proficient with ACI technology and have an in-depth understanding of troubleshooting and monitoring best practices for the ACI fabric, giving them the competitive edge to grow their business.

Ronak Desai
VP of Engineering for the Data Center Networking Business Unit
Cisco Systems
Introduction

Application Centric Infrastructure (ACI) is a software-defined network offering from Cisco that addresses the challenges of application agility needs in data centers. ACI was announced on November 6, 2013, and it has been widely deployed on large number of customer data centers globally since then. The demand to monitor and troubleshoot this unique and modern form of network infrastructure has increased exponentially from every corner of the world. This book was written with the goal of helping guide data center professionals understand the crucial topics of ACI with real-world examples from field experiences. The Cisco Data Center Business Unit and industry leaders were consulted for technical accuracy of the content of this book.

Who Should Read This Book?

This book is intended for data center architects, engineers, software developers, network and virtualization administrators, and, most importantly, operations team members striving to better understand and manage this new form of software-defined networking.

The content of the book will help you confidently deploy, support, monitor, and troubleshoot ACI fabric and its components. It also introduces some of the newer concepts in this technology by relating them to traditional networking terminology and experiences. The readers should be at the intermediate to expert level. This book assumes common knowledge of Cisco NX-OS and network switching and routing concepts. A typical reader should at least possess a Cisco CCNA certification and be responsible for day-to-day operations of networks and applications. Because of its in-depth and advanced subject matter, this book can also be used as a reference guide for CCIE Data Center certification.

This book is also a good preparatory reference for those taking the Cisco DCACIA (300-630) exam toward the Cisco Certified Specialist—ACI Advanced Implementation certification. Where applicable, portions of some chapters are marked with a Key Topic icon to highlight concepts you should know for the exam. Chapters 1, 2, 4, 5, 7, 8, 9, 12, and 13 also provide some review questions to help you prepare for this exam. This book can also help you prepare for the CCIE Data Center (v3.0) exam.

How This Book Is Organized

This book is divided into three major sections:

Part I, “Introduction to ACI”: This section includes the following chapters:

- Chapter 1, “Fundamental Functions and Components of ACI”: This chapter provides a high-level overview of the core functions and components of Cisco Application Infrastructure (ACI). This chapter also covers key concepts of control and data plane protocols used in ACI fabric, such as IS-IS, MP-BGP EVPN, COOP, and VXLAN, along with logical constructs in configuring application-hosting infrastructure, such as tenants, VRF instances, application profiles, endpoint groups, bridge domains, external routed or bridge networks, and contracts.
Chapter 2, “Introduction to the ACI Policy Model”: Cisco ACI is a policy-based object model, and it is important to understand how this model works. This chapter outlines the physical and logical constructs of ACI and their relationships in developing the overall application framework through software-defined policies.

Chapter 3, “ACI Command-Line Interfaces”: Traditionally, network engineers have been comfortable in using command-line interfaces (CLIs) on network devices. This chapter describes the different CLIs that can be used to monitor and troubleshoot both APICs and ACI fabric switches.

Chapter 4, “ACI Fabric Design Options”: To monitor and troubleshoot the ACI fabric and its components, it is important to understand ACI fabric design. This chapter explains in detail various design options, starting from physical designs such as stretching ACI fabric using transit leafs, multi-pod, multi-site, and remote leafs. The chapter also demonstrates logical designs, covering Kubernetes using Calico CNI, ERP SAP HANA, and vBrick Digital Media Engine.

Chapter 5, “End Host and Network Connectivity”: This chapter describes compute, storage, and service device (load balancer and firewall) connectivity to ACI leaf switches using either Access ports, port channel, or virtual port channel. The chapter also covers switch and router connectivity between external networks and the ACI fabric. Finally, it also covers connectivity between ACI pods, sites, and remote leafs.

Chapter 6, “VMM Integration”: Virtual Machine Manager (VMM) provides visibility into the virtualization layer. This chapter explains the integration of various hypervisors and container platforms into ACI to extend the networking stack up to the end-host level.

Chapter 7, “L4/L7 Service Integration”: Layer 4 to Layer 7 services such as load-balancing and firewall services are essential components between application tiers for efficient and secure service delivery. Cisco ACI offers seamless integration of L4/L7 services, and these services can be stitched using service chaining or through policy-based routing and service graphs.

Chapter 8, “Automation and Orchestration”: ACI technology enables automation and orchestration for speedy deployment of ACI. This chapter explains the difference between automation and orchestration and how the REST API works in ACI. It provides examples of automation scripts using JSON and XML. It explains Ansible, which is widely used as a data center automation tool, and provides examples for ACI- and non-ACI-based infrastructure. This chapter also provides details about UCS Director and examples for orchestrating various components of application-hosting infrastructure.

Part II, “Monitoring and Management Best Practices”: This section includes the following chapters:

Chapter 9, “Monitoring ACI Fabric”: Proper monitoring solutions can enable businesses to run their operations smoothly by minimizing service downtime and providing immediate ROI on software-defined application hosting infrastructure, such as
Cisco ACI. This chapter outlines the key concepts of ACI monitoring, such as using faults and health scores, built-in and external tools, and the REST API to monitor ACI.

- **Chapter 10, “Network Management and Monitoring Configuration”:** This chapter covers the configuration of ACI management, such as in-band and out-of-band management and AAA, along with monitoring protocols such as syslog, SNMP, SPAN, and NetFlow. Network management and monitoring configurations are provided, along with verification steps.

**Part III, “Advanced Forwarding and Troubleshooting Techniques”:** This section includes the following chapters:

- **Chapter 11, “ACI Topology”:** To help lay a foundation for the following chapters, this chapter describes the lab infrastructure used for the rest of the Part III chapters.

- **Chapter 12, “Bits and Bytes of ACI Forwarding”:** The book covers many aspects of ACI, but to truly understand how the fabric works, you have to deep dive into the bits and bytes of forwarding. This chapter builds a strong foundation for VXLAN forwarding and the additional bits used in the iVXLAN header to enable policy enforcement and other ACI features. This chapter provides a variety of forwarding examples that demonstrate the packet life cycle through the ACI fabric.

- **Chapter 13, “Troubleshooting Techniques”:** This chapter highlights a variety of troubleshooting techniques that can be used to manage ACI fabric. The chapter begins by explaining system logs, such as fault, event, and audit logs, and then it dives deeper into specific components in the fabric to help build additional confidence for troubleshooting critical events.

- **Chapter 14, “The ACI Visibility & Troubleshooting Tool”:** The Visibility & Troubleshooting tool has been part of the APIC for many ACI releases. This chapter provides an overview of how the tool works and examples of how it can ease the troubleshooting process.

- **Chapter 15, “Troubleshooting Use Cases”:** This book demonstrates many ways to manage, monitor, and troubleshoot the ACI fabric. This chapter provides focused troubleshooting scenarios, illustrating problems and resolutions based on real-world issues seen in customer deployments. Each scenario outlines the problem faced, as well as how to troubleshoot the type of problem to isolate the issue using ACI tools.
## Figure Credits

<table>
<thead>
<tr>
<th>Figure</th>
<th>Selection Title</th>
<th>Attribution/Credit Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 8-6</td>
<td>Creating Tenant t01 Using Postman</td>
<td>Screenshot © 2020 Postman, Inc.</td>
</tr>
<tr>
<td>Figure 9-8</td>
<td>Fabric Node Unreachable System Message</td>
<td>Screenshot © 2005-2020 Splunk Inc.</td>
</tr>
<tr>
<td>Figure 9-11</td>
<td>Viewing NetFlow Information from the Border Leaf 201 CLI</td>
<td>Screenshot © 2020 Zoho Corp.</td>
</tr>
<tr>
<td>Figure 9-12</td>
<td>Viewing NetFlow Information in NetFlow Analyzer - 1</td>
<td>Screenshot © 2020 Zoho Corp.</td>
</tr>
<tr>
<td>Figure 9-13</td>
<td>Viewing Top Conversation</td>
<td>Screenshot © 2020 Zoho Corp.</td>
</tr>
<tr>
<td>Figure 9-14</td>
<td>Viewing NetFlow Information in NetFlow Analyzer - 2</td>
<td>Screenshot © 2020 Zoho Corp.</td>
</tr>
<tr>
<td>Figure 9-39</td>
<td>Tetration Software Agent in Windows</td>
<td>Screenshot © Microsoft 2020</td>
</tr>
<tr>
<td>Figure 9-40</td>
<td>Attaching a Datastore ISO File to a CD/DVD Drive</td>
<td>Screenshot © Microsoft 2020</td>
</tr>
<tr>
<td>Figure 9-41</td>
<td>Mapping Alert Types to Publisher Types</td>
<td>Screenshot © Microsoft 2020</td>
</tr>
<tr>
<td>Figure 9-42</td>
<td>Email Alerts</td>
<td>Screenshot © Microsoft 2020</td>
</tr>
<tr>
<td>Figure 9-43</td>
<td>Configuring Syslog in Tetration</td>
<td>Screenshot © Microsoft 2020</td>
</tr>
<tr>
<td>Figure 9-44</td>
<td>Enabling Alert Types</td>
<td>Screenshot © Microsoft 2020</td>
</tr>
<tr>
<td>Figure 15-46</td>
<td>JSON Syntax Error</td>
<td>Screenshot © 2020 Postman, Inc.</td>
</tr>
<tr>
<td>FIG15-47</td>
<td>JSON Syntax, Including the attributes Tag</td>
<td>Screenshot of JSON Syntax, Including the attributes Tag © 2020 Postman, Inc.</td>
</tr>
</tbody>
</table>
Cisco ACI virtual machine (VM) networking supports hypervisors from multiple vendors. It allows for multivendor hypervisors along with programmable and automated access to high-performance scalable virtualized data center infrastructure. In this chapter, you will learn about Virtual Machine Manager (VMM) and its integration into Cisco Application Centric Infrastructure (ACI) from the following virtualization-supported products and vendors:

- VMware
- Microsoft
- OpenStack
- Kubernetes
- OpenShift

You will also learn about VMM integration with ACI at multiple locations.

**Virtual Machine Manager (VMM)**

VMM integration enables the ACI fabric to extend network policies and policy group definitions into the virtualization switching layer on end hosts. This integration automates critical network plumbing steps that typically create delays in the deployment of overall virtual and compute resources in legacy network environments. VMM integration into ACI also provides value in getting visibility up to the virtualization layer of the application, which is a perpetually conflicting factor between network and server virtualization teams.
VMM Domain Policy Model

VMM domain profiles (vmmDomP) specify connectivity policies that enable virtual machine controllers to connect to the ACI fabric. Figure 6-1 shows the general hierarchy of VMM configuration.

![Policy Universe (polUni)](image)

**Figure 6-1**  VMM Policy Model

VMM Domain Components

VMM domains enable an administrator to configure connectivity policies for virtual machine controllers in ACI. The essential components of an ACI VMM domain policy include the following:

- VMM domain
- VLAN pool association
- Attachable access entity profile association
- VMM domain endpoint group (EPG) association

VMM Domains

VMM domains make it possible to group VM controllers with similar networking policy requirements. For example, VM controllers can share VLAN pools and application EPGs. The Cisco Application Policy Infrastructure Controller (APIC) communicates
with the VM controller to publish network configurations such as port groups, which are then applied to the virtual workloads. The VMM domain profile includes the following essential components:

- **Credential**: Associates a valid VM controller user credential with an APIC VMM domain.
- **Controller**: Specifies how to connect to a VM controller that is part of a policy enforcement domain. For example, the controller specifies the connection to a VMware vCenter instance that is part of a VMM domain.

**Note** A single VMM domain can contain multiple instances of VM controllers, but they must be from the same vendor (for example, VMware, Microsoft).

An APIC VMM domain profile is a policy that defines a VMM domain. The VMM domain policy is created on an APIC and pushed into the leaf switches. Figure 6-2 illustrates VM controllers of the same vendor as part of the same VMM domain.

**Figure 6-2  VMM Domain Integration**

VMM domains provide the following:

- A common layer in the ACI fabric that enables scalable fault-tolerant support for multiple VM controller platforms.
- VMM support for multiple tenants within the ACI fabric.
VMM domains contain VM controllers such as VMware vCenter or Microsoft System Center Virtual Machine Manager (SCVMM) and the credentials required for the ACI API to interact with the VM controllers. A VMM domain enables VM mobility within the domain but not across domains. A single VMM domain can contain multiple instances of VM controllers, but they must be from the same vendor. For example, a VMM domain can contain many VMware vCenter instances managing multiple controllers, each running multiple VMs; however, it cannot contain Microsoft SCVMM instances. A VMM domain inventories controller elements (such as pNICs, vNICs, and VM names) and pushes policies into the controllers, creating port groups or VM networks and other necessary elements. The ACI VMM domain listens for controller events such as VM mobility events and responds accordingly.

**VMM Domain VLAN Pool Association**

A VLAN pool specifies a single VLAN ID or a range of VLAN IDs for VLAN encapsulation. It is a shared resource that can be consumed by multiple domains, such as physical, VMM, or external domains.

In ACI, you can create a VLAN pool with allocation type static or dynamic. With static allocation, the fabric administrator configures a VLAN; with dynamic allocation, the APIC assigns the VLAN to the domain dynamically. In ACI, only one VLAN or VXLAN pool can be assigned to a VMM domain.

A fabric administrator can assign a VLAN ID statically to an EPG. However, in this case, the VLAN ID must be included in the VLAN pool with the static allocation type, or the APIC will generate a fault. By default, the assignment of VLAN IDs to EPGs that are associated with the VMM domain is done dynamically by the APIC. The APIC provisions VMM domain VLAN IDs on leaf switch ports based on EPG events, either statically binding or based on VM events from controllers such as VMware vCenter or Microsoft SCVMM.

**Attachable Access Entity Profile Association**

An attachable access entity profile (AAEP) associates a VMM domain with the physical network infrastructure where the vSphere hosts are connected. The AAEP defines which VLANs will be permitted on a host-facing interface. When a domain is mapped to an endpoint group, the AAEP validates that the VLAN can be deployed on certain interfaces. An AAEP is a network interface template that enables the deployment of VM controller policies on a large set of leaf switch ports. An AAEP specifies which switches and ports are available and how they are configured. The AAEP can be created on-the-fly during the creation of the VMM domain itself.
VMM Domain EPG Association

Endpoint groups regulate connectivity and visibility among the endpoints within the scope of the VMM domain policy. VMM domain EPGs behave as follows:

- The APIC pushes these EPGs as port groups into the VM controller.
- An EPG can span multiple VMM domains, and a VMM domain can contain multiple EPGs.

The ACI fabric associates EPGs to VMM domains, either automatically through an orchestration component such as VMware vRealize suite (vRA/vRO) or Microsoft Azure, or when an APIC administrator creates such configurations. An EPG can span multiple VMM domains, and a VMM domain can contain multiple EPGs.

In Figure 6-3, endpoints (EPs) of the same color are part of the same EPG. For example, all the gray EPs are in the same EPG, even though they are in different VMM domains.

Figure 6-3  VMM Domain EPG Association

Note  Refer to the latest Verified Scalability Guide for Cisco ACI at the Cisco website for virtual network and VMM domain EPG capacity information.
Figure 6-4 illustrates multiple VMM domains connecting to the same leaf switch if they do not have overlapping VLAN pools on the same port. Similarly, the same VLAN pools can be used across different domains if they do not use the same port of a leaf switch.

![Figure 6-4 VMM Domain EPG VLAN Consumption](image)

EPGs can use multiple VMM domains in the following ways:

- An EPG within a VMM domain is identified by an encapsulation identifier that is either automatically managed by the APIC or statically selected by the administrator. An example for a VLAN is a virtual network ID (VNID).
- An EPG can be mapped to multiple physical (for bare-metal servers) or virtual domains. It can use different VLAN or VNID encapsulations in each domain.

**Note** By default, an APIC dynamically manages the allocation of a VLAN for an EPG in a VMM integration. VMware vSphere Distributed Switch (VDS) administrators have the option of configuring a specific VLAN for an EPG. In that case, the VLAN is chosen from a static allocation block within the pool associated with the VMM domain.

Applications can be deployed across VMM domains, as illustrated in Figure 6-5. While live migration of VMs within a VMM domain is supported, live migration of VMs across VMM domains is not supported.
Whenever an EPG associates to a VMM domain, the administrator can choose the policy resolution and deployment preferences to specify when it should be pushed and programmed into leaf switches. This approach provides efficient use of hardware resources because resources are consumed only when demanded. You should be aware of picking one option over the other, depending on the use case and scalability limits of your ACI infrastructure, as explained in the following sections.

Resolution Immediacy

The Resolution Immediacy option defines when policies are downloaded to the leaf software based on the following options:

- **Pre-provision**: This option specifies that a policy (such as VRF, VLAN, VXLAN binding, contracts, or filters) is downloaded to the associated leaf switch software even before a VM controller is attached to the distributed virtual switch (DVS), such as a VMware (VDS), defined by an APIC through the VMM domain.

- This option helps when management traffic between hypervisors and VM controllers such as VMware vCenter is also using the APIC-defined virtual switch.

- When you deploy a VMM policy such as VLAN or VXLAN on an ACI leaf switch, an APIC must collect CDP/LLDP information from hypervisors through...
the VM controller and ACI leaf switch to which the host is connected. However, if the VM controller is supposed to use the same VMM policy to communicate with its hypervisors or even an APIC, the CDP/LLDP information for hypervisors can never be collected because the required policy is not deployed yet.

- With the Pre-provision immediacy option, policy is downloaded to the ACI leaf switch software, regardless of CDP/LLDP neighborship and even without a hypervisor host connected to the VMM domain-defined DVS.

- **Immediate**: This option specifies that a policy (such as VRF, VLAN, VXLAN binding, contracts, or filters) is downloaded to the associated leaf switch software upon ESXi host attachment to a DVS. LLDP or OpFlex permissions are used to resolve the VM controller to leaf switch attachments.

  - The policy is downloaded to a leaf when you add a host to the VMM domain-defined DVS. CDP/LLDP neighborship from host to leaf is required.

- **On Demand**: This option specifies that a policy (such as VRF, VLAN, VXLAN binding, contracts, or filters) is pushed to the leaf node only when a host running hypervisor is attached to a DVS and a VM is placed in the port group (EPG).

  - The policy is downloaded to a leaf when a host is added to the VMM domain-defined DVS and a virtual machine is placed in the port group (EPG). CDP/LLDP neighborship from host to leaf is required.

With both the Immediate and On Demand options for resolution immediacy, if the hypervisor running on the host and leaf lose LLDP/CDP neighborship, the policies are removed from the leaf switch software.

**Deployment Immediacy**

After the policies are downloaded to the leaf software through the Resolution Immediacy option, you can use Deployment Immediacy to specify when the policy is pushed to the hardware policy content-addressable memory (CAM). Two options are available:

- **Immediate**: This option specifies that the policy is programmed into the hardware policy CAM as soon as the policy is downloaded in the leaf software. You should be aware of your ACI infrastructure scalability limits when choosing this option.

- **On Demand**: This option specifies that the policy is programmed in the hardware policy CAM only when the first packet is received through the data path. This process helps optimize the hardware resources.

**Note** When you use On Demand deployment immediacy with MAC-pinned VPCs, the EPG contracts are not pushed to the leaf ternary content-addressable memory (TCAM) until the first endpoint is learned in the EPG on each leaf. This can cause uneven TCAM utilization across VPC peers. (Normally, the contract would be pushed to both peers.)
VMware Integration

When integrating your VMware infrastructure into Cisco ACI, you have two options for deploying virtual networking:

- VMware vSphere Distributed Switch (VDS)
- Cisco Application Virtual Switch (AVS)

These two options provide similar basic virtual networking functionality; however, the AVS option provides additional capabilities, such as VXLAN and microsegmentation support.

Prerequisites for VMM Integration with AVS or VDS

The prerequisites for VMM integration with AVS or VDS are as follows:

- You need to decide whether to use VLAN or VXLAN encapsulation or multicast groups.
- A virtual machine manager must be already deployed, such as vCenter.
- The VMM must be accessible by the APIC through either the out-of-band or in-band management network.
- For Cisco AVS deployment, a vSphere Installation Bundle (VIB) must be installed on all hypervisor hosts to be added to the AVS.
- For a VXLAN deployment, you need to know whether intermediate devices have Internet Group Management Protocol (IGMP) snooping on or off by default.

Guidelines and Limitations for VMM Integration with AVS or VDS

The guidelines and limitations for VMM integration with AVS or VDS are as follows:

- When utilizing VLANs for VMM integration, whether with Cisco AVS or VMware VDS, the range of VLANs to be used for port groups must be manually allowed on any intermediate devices.
- For VMM integration with VLANs and the Resolution Immediacy setting On Demand or Immediate, there can be a maximum of one hop between a host and the compute node.
- For VMM integration with VXLAN, only the infrastructure VLAN needs to be allowed on all intermediate devices.
- For VMM integration with VXLAN, if the Infra bridge domain subnet is set as a querier, the intermediate devices must have IGMP snooping enabled for traffic to pass properly.
To log in to the APIC GUI, choose Tenants > Infra > Networking > Bridge Domains > default > Subnets > 10.0.0.30/27.

For VMM integration with VXLAN and UCS-B, IGMP snooping is enabled on the UCS-B by default. Therefore, you need to ensure that the querier IP address is enabled for the Infra bridge domain. The other option is to disable IGMP snooping on the UCS and disable the querier IP address on the Infra bridge domain.

ACI VMM Integration Workflow

Figure 6-6 illustrates the ACI VMM integration workflow steps.

Publishing EPGs to a VMM Domain

This section details how to publish an existing EPG to a VMM domain. For an EPG to be pushed to a VMM domain, you must create a domain binding within the tenant EPG by following these steps:

Step 1. From the menu bar, choose Tenants > All Tenants.
Step 2. From the Work pane, choose the Tenant_Name.
Step 3. From the Navigation pane, choose Tenant_Name > Application Profiles > Application_Profile_Name > Application EPGs > Application_EPG_Name > Domains (VMs and bare-metal servers).
Step 4. From the Work pane, choose Actions > Add VM Domain Association.
Step 5. In the Add VM Domain Association dialog box, choose the VMM domain profile that you created previously. For Deployment and Resolution
Immediacy, Cisco recommends keeping the default option, On Demand. This provides the best resource usage in the fabric by deploying policies to leaf nodes only when endpoints assigned to this EPG are connected. There is no communication delay or traffic loss when you keep the default selections.

**Step 6.** Click Submit. The EPG is now available as a port group to your VMM.

### Connecting Virtual Machines to the Endpoint Group Port Groups on vCenter

To connect virtual machines to the endpoint group port groups on vCenter, do the following:

**Step 1.** Connect to vCenter by using the VMware VI Client.

**Step 2.** From the Host and Clusters view, right-click on your virtual machine and choose Edit Settings.

**Step 3.** Click on the network adapter and from the Network Connection drop-down box, choose the port group that corresponds to your EPG. It should appear in the format of TENANT | APPLICATION_PROFILE | EPG | VMM_DOMAIN_PROFILE.

If you do not see your Cisco ACI EPG in the Network Connection list, it means one of the following:

- The VM is running on a host that is not attached to the distributed switch managed by the APIC.
- There may be a communication between your APIC and vCenter either through the OOB or the INB management network.

### Verifying VMM Integration with the AVS or VDS

The following sections describe how to verify that the Cisco AVS has been installed on the VMware ESXi hypervisor.

#### Verifying the Virtual Switch Status

To verify the virtual switch status, follow these steps:

**Step 1.** Log in to the VMware vSphere client.

**Step 2.** Choose Networking.

**Step 3.** Open the folder for the data center and click the virtual switch.

**Step 4.** Click the Hosts tab. The VDS Status and Status fields display the virtual switch status. Ensure that the VDS status is Up, which indicates that OpFlex communication has been established.
Verifying the vNIC Status

To verify the vNIC status, follow these steps:

Step 1. In the VMware vSphere client, click the Home tab.
Step 2. Choose Hosts and Clusters.
Step 3. Click the host.
Step 4. In the Configuration tab, select the Hardware panel and choose Networking.
Step 5. In the View field, click the vSphere Distributed Switch button.
Step 6. Click Manage Virtual Adapters. The vmk1 displays as a virtual adapter with an IP address.
Step 7. Click the newly created vmk interface to display the vmknic status.

Note: Allow approximately 20 seconds for the vmk to receive an IP address through DHCP.

Microsoft SCVMM Integration

Figure 6-7 shows a representative topology for a Microsoft SCVMM integration with Cisco ACI. Hyper-V clustering connectivity between SCVMM virtual machines and the APIC can run over the management network.
Figure 6-8 illustrates the workflow for integrating Microsoft SCVMM with Cisco ACI. The following sections describe the steps in this workflow.

**Mapping ACI and SCVMM Constructs**

Figure 6-9 shows the mapping of Cisco ACI and the SCVMM constructs (SCVMM controller, cloud, and logical switches).
One VMM domain cannot map to the same SCVMM more than once. An APIC can be associated with up to five SCVMM controllers. For additional information on other limitations, see the *Verified Scalability Guide for Cisco ACI* on the Cisco website.

**Mapping Multiple SCVMMs to an APIC**

When multiple SCVMMs are associated with an APIC, the OpFlex certificate from the first SCVMM controller must be copied to the secondary controller and other controllers, as applicable. You use the `certlm.msc` command on the local SCVMM controller to import the certificate to the following location:

```
Certificates - Local Computer > Personal > Certificates
```

The same OpFlex certificate is deployed on the Hyper-V servers that are managed by this SCVMM controller. You use the `mmc` command to install the certificate on the Hyper-V servers.

**Verifying That the OpFlex Certificate Is Deployed for a Connection from the SCVMM to the APIC**

You can verify that the OpFlex certificate is deployed for a connection from the SCVMM to the APIC by viewing the `Cisco_APIC_SCVMM_Service` log file, which is located in the `C:\Program Files (x86)\ApicVMMService\Logs\` directory. In this file, ensure that the correct certificate is used and also check to make sure there was a successful login to the APIC (see Example 6-1).

**Example 6-1  Viewing the Cisco_APIC_SCVMM_Service Log File**

```
4/15/2017 2:10:09 PM-1044-13||UpdateCredentials|| new: EndpointAddress: Called_from_SCVMM_PS,
Username ApicAddresses 10.10.10.1;10.10.10.2;10.10.10.3 CertName: OpflexAgent
4/15/2017 2:10:09 PM-1044-13||UpdateCredentials|| ########
4/15/2017 2:10:09 PM-1044-13||UpdateCredentials|| oldreg_apicAddresses is
4/15/2017 2:10:09 PM-1044-13||UpdateCredentials|| Verifying APIC address 10.10.10.1
4/15/2017 2:10:09 PM-1044-13||GetInfoFromApic|| HostAddr 10.10.10.1
4/15/2017 2:10:09 PM-1044-13||PopulateCertsAndCookies|| URL:/api/node/class/infraWiNode.xml
4/15/2017 2:10:09 PM-1044-13||PopulateCertsAndCookies|| Searching Cached Store Name: My
4/15/2017 2:10:09 PM-1044-13||PopulateCertsAndCookies|| Using Certificate CN=OpflexAgent, C=USA, S=MI, O=CX, E=aci@lab.local in Cached Store Name:My
```
Verifying VMM Deployment from the APIC to the SCVMM

You can verify that the OpFlex certificate is deployed on the Hyper-V server by viewing log files in the C:\Program Files (x86)\ApicHyperAgent\Logs directory. In this file, ensure that the correct certificate is used and ensure that the connection with the Hyper-V servers on the fabric leafs is established. In addition, ensure that a VTEP virtual network adapter is added to the virtual switch and an IP address is assigned to the VTEP adapter.

In the SCVMM, check for the following:

- Under Fabric > Logical Switches, verify that apicVswitch_VMMdomainName is deployed from the APIC to the SCVMM.
- Under Fabric > Logical Networks, verify that apicLogicalNetwork_VMMdomainName is deployed from the APIC to the SCVMM.
- Under Fabric > Port Profiles, verify that apicUplinkPortProfile_VMMdomainName is deployed. If it is not deployed, right-click the host under Servers and choose Properties. Go to Virtual Switches and ensure that the physical adapters are attached to the virtual switches.

Note In the APIC GUI, the Hyper-V servers and the virtual machines do not appear in the Microsoft SCVMM inventory until you ensure that these points for the SCVMM are satisfied.

OpenStack Integration

OpenStack defines a flexible software architecture for creating cloud-computing environments. The reference software-based implementation of OpenStack allows for multiple Layer 2 transports, including VLAN, GRE, and VXLAN. The Neutron project within OpenStack can also provide software-based Layer 3 forwarding. When OpenStack is used with ACI, the ACI fabric provides an integrated Layer 2/3 VXLAN-based overlay networking capability that can offload network encapsulation processing from the compute nodes to the top-of-rack or ACI leaf switches. This architecture provides the flexibility of software overlay networking in conjunction with the performance and operational benefits of hardware-based networking.
Extending OpFlex to the Compute Node

OpFlex is an open and extensible policy protocol designed to transfer declarative networking policies such as those used in Cisco ACI to other devices. By using OpFlex, you can extend the policy model native to ACI all the way down into the virtual switches running on OpenStack Nova compute hosts. This OpFlex extension to the compute host allows ACI to use Open vSwitch (OVS) to support common OpenStack features such as source Network Address Translation (SNAT) and floating IP addresses in a distributed manner.

The ACI OpenStack drivers support two distinct modes of deployment. The first approach is based on the Neutron API and Modular Layer 2 (ML2), which are designed to provide common constructs such as network, router, and security groups that are familiar to Neutron users. The second approach is native to the group-based policy abstractions for OpenStack, which are closely aligned with the declarative policy model used in Cisco ACI.

ACI with OpenStack Physical Architecture

A typical architecture for an ACI fabric with an OpenStack deployment consists of a Nexus 9000 spine/leaf topology, an APIC cluster, and a group of servers to run the various control and compute components of OpenStack. An ACI external routed network connection as a Layer 3 connection outside the fabric can be used to provide connectivity outside the OpenStack cloud. Figure 6-10 illustrates OpenStack infrastructure connectivity with ACI.

![OpenStack Physical Topology with ACI](image)
OpFlex Software Architecture

The ML2 framework in OpenStack enables the integration of networking services based on type drivers and mechanism drivers. Common networking type drivers include local, flat, VLAN, and VXLAN. OpFlex is added as a new network type through ML2, with an actual packet encapsulation of either VXLAN or VLAN on the host defined in the OpFlex configuration. A mechanism driver is enabled to communicate networking requirements from the Neutron servers to the Cisco APIC cluster. The APIC mechanism driver translates Neutron networking elements such as a network (segment), subnet, router, or external network into APIC constructs in the ACI policy model.

The OpFlex software stack also currently utilizes OVS and local software agents on each OpenStack compute host that communicates with the Neutron servers and OVS. An OpFlex proxy from the ACI leaf switch exchanges policy information with the agent OVS instance in each compute host, effectively extending the ACI switch fabric and policy model into the virtual switch. Figure 6-11 illustrates the OpenStack architecture with OpFlex in ACI.

Figure 6-11  OpenStack Architecture with OpFlex in ACI

OpenStack Logical Topology

The logical topology diagram in Figure 6-12 illustrates the connections to OpenStack network segments from Neutron/controller servers and compute hosts, including the distributed Neutron services.
Mapping OpenStack and ACI Constructs

Cisco ACI uses a policy model to enable network connectivity between endpoints attached to the fabric. OpenStack Neutron uses more traditional Layer 2 and Layer 3 networking concepts to define networking configuration. The OpFlex ML2 driver translates the Neutron networking requirements into the necessary ACI policy model constructs to achieve the desired connectivity. The OpenStack Group-Based Policy (GBP) networking model is quite similar to the Cisco ACI policy model. With the Cisco ACI unified plug-in for OpenStack, you can use both ML2 and GBP models on a single plug-in instance.

Note Only ML2 or GBP can be used for any given OpenStack project. A single project should not mix ML2 and GBP configurations.

Table 6-1 illustrates the OpenStack Neutron constructs and the corresponding APIC policy objects that are configured when they are created. In the case of GBP deployment, the policies have a direct mapping to the ACI policy model. Table 6-2 shows the OpenStack GBP objects and their corresponding ACI objects.
Table 6-1  OpenStack Neutron Objects and Corresponding APIC Objects

<table>
<thead>
<tr>
<th>Neutron Object</th>
<th>APIC Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Neutron Instance)</td>
<td>VMM Domain</td>
</tr>
<tr>
<td>Project</td>
<td>Tenant + Application Network Profile</td>
</tr>
<tr>
<td>Network</td>
<td>EPG + Bridge Domain</td>
</tr>
<tr>
<td>Subnet</td>
<td>Subnet</td>
</tr>
<tr>
<td>Security Group + Rule</td>
<td>N/A (Iptables rules maintained per host)</td>
</tr>
<tr>
<td>Router</td>
<td>Contract</td>
</tr>
<tr>
<td>Network:external</td>
<td>L3Out/Outside EPG</td>
</tr>
</tbody>
</table>

Table 6-2  OpenStack GBP Objects and Corresponding APIC Objects

<table>
<thead>
<tr>
<th>GBP Object</th>
<th>APIC Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy Target</td>
<td>Endpoint</td>
</tr>
<tr>
<td>Policy Group</td>
<td>Endpoint Group (fvAEPg)</td>
</tr>
<tr>
<td>Policy Classifier</td>
<td>Filter (vzFilter)</td>
</tr>
<tr>
<td>Policy Action</td>
<td>--</td>
</tr>
<tr>
<td>Policy Rule</td>
<td>Subject (vzSubj)</td>
</tr>
<tr>
<td>Policy Ruleset</td>
<td>Contract (vzBrCP)</td>
</tr>
<tr>
<td>L2 Policy</td>
<td>Bridge Domain (fvBD)</td>
</tr>
<tr>
<td>L3 Policy</td>
<td>Context (fvCtx)</td>
</tr>
</tbody>
</table>

Prerequisites for OpenStack and Cisco ACI

Keep in mind the following prerequisites for OpenStack and Cisco ACI:

- **Target audience**: It is important to have working knowledge of Linux, the intended OpenStack distribution, the ACI policy model, and GUI-based APIC configuration.

- **ACI Fabric**: ACI fabric needs to be installed and initialized with a minimum APIC version 1.1(4e) and NX-OS version 1.1.1(4e). For basic guidelines on initializing a new ACI fabric, see the relevant documentation. For communication between multiple leaf pairs, the fabric must have a BGP route reflector enabled to use an OpenStack external network.

- **Compute**: You need to have a controller and servers connected to the fabric, preferably using NIC bonding and a VPC. In most cases the controller does not need to be connected to the fabric.
L3Out: For external connectivity, one or more Layer 3 Outs (L3Outs) need to be configured on the ACI.

VLAN mode: For VLAN mode, a non-overlapping VLAN pool of sufficient size should be allocated ahead of time.

Guidelines and Limitations for OpenStack and Cisco ACI

The following sections describes the guidelines and limitations for OpenStack and Cisco ACI.

Scalability Guidelines

There is a one-to-one correlation between the OpenStack tenant and the ACI tenant, and for each OpenStack tenant, the plug-in automatically creates ACI tenants named according to the following convention:

collection_system_id_openstack_tenant_name

You should consider the scalability parameters for supporting the number of required tenants.

It is important to calculate the fabric scale limits for endpoint groups, bridge domains, tenants, and contracts before deployment. Doing so limits the number of tenant/project networks and routers that can be created in OpenStack. There are per-leaf and per-fabric limits. Make sure to check the scalability parameters for the deployed release before deployment. In the case of GBP deployment, it can take twice as many endpoint groups and bridge domains as with ML2 mode. Table 6-3 and Table 6-4 list the APIC resources that are needed for each OpenStack resource in GBP and ML2 configurations.

Table 6-3  OpenStack GBP and ACI Resources

<table>
<thead>
<tr>
<th>GBP Resource</th>
<th>APIC Resources Consumed</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3 policy</td>
<td>One context</td>
</tr>
<tr>
<td>L2 policy</td>
<td>One bridge domain</td>
</tr>
<tr>
<td></td>
<td>One endpoint group</td>
</tr>
<tr>
<td></td>
<td>Two contracts</td>
</tr>
<tr>
<td>Policy group</td>
<td>One endpoint group</td>
</tr>
<tr>
<td>Ruleset</td>
<td>One contract</td>
</tr>
<tr>
<td>Classifier</td>
<td>Two filters (forward and reverse)</td>
</tr>
</tbody>
</table>

Note: Five overhead classifiers are created
Table 6-4  OpenStack ML2 and ACI Resources

<table>
<thead>
<tr>
<th>ML2 Resource</th>
<th>APIC Resources Consumed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>One bridge domain</td>
</tr>
<tr>
<td></td>
<td>One endpoint group</td>
</tr>
<tr>
<td>Router</td>
<td>One contract</td>
</tr>
<tr>
<td>Security groups</td>
<td>N/A (no filters are used)</td>
</tr>
</tbody>
</table>

Availability Guidelines

For redundancy, you can use bonded interfaces (VPCs) by connecting two interfaces to two leaf switches and creating a VPC in ACI. You should deploy redundant OpenStack controller nodes to avoid a single point of failure. The external network should also be designed to avoid a single point of failure and service interruption.

NAT/External Network Operations

The OpFlex driver software can support external network connectivity and Network Address Translation (NAT) functions in a distributed manner using the local OVS instance on each OpenStack compute node. This distributed approach increases the availability of the overall solution and offloads the central processing of NAT from the Neutron server Layer 3 agent that is used in the reference implementation. You can also provide direct external connectivity without NAT or with a mix of NAT and non-NAT external connectivity.

Subnets Required for NAT

Unlike with the standard Neutron approach, three distinct IP subnets are required to take full advantage of external network functionality with the OpFlex driver:

- **Link subnet:** This subnet represents the actual physical connection to the external next-hop router outside of the fabric to be assigned to a routed interface, subinterface, or SVI.

- **Source NAT subnet:** This subnet is used for Port Address Translation (PAT), allowing multiple virtual machines to share an outside-routable IP address. A single IP address is assigned to each compute host, and Layer 4 port number manipulation is used to maintain unique session traffic.

- **Floating IP subnet:** With OpenStack, the term *floating IP* is used when a virtual machine instance is allowed to claim a distinct static NAT address to support inbound connections to the virtual machine from outside the cloud. The floating IP subnet is the subnet assigned within OpenStack to the Neutron external network entity.
Optimized DHCP and Metadata Proxy Operations

The OpFlex driver software stack provides optimized traffic flow and distributed processing to provide DHCP and metadata proxy services for virtual machine instances. These services are designed to keep processing and packet traffic local to the compute host as much as possible. The distributed elements communicate with centralized functions to ensure system consistency. You should enable optimized DHCP and metadata services when deploying the OpFlex plug-in for OpenStack.

Physical Interfaces

OpFlex uses the untagged fabric interface for an uplink trunk in VLAN mode. This means the fabric interface cannot be used for PXE because PXE usually requires an untagged interface. If you require PXE in a VLAN mode deployment, you must use a separate interface for PXE. This interface can be connected through ACI or an external switch. This issue is not present in VXLAN mode since tunnels are created using the tagged interface for an infrastructure VLAN.

Layer 4 to Layer 7 Services

Service insertion in OpenStack is done through a physical domain or device package. You should check customer requirements and the plug-in mode (GBP or ML2) to plan how service insertion/chaining will be done. The OpenStack Neutron project also defines Layer 4 to Layer 7 extension APIs, such as LBaaS, FWaaS, and VPNaaS. The availability of these extensions depends on the device vendor. Check the vendor for the availability of these extensions.

Blade Servers

When deploying on blade servers, you must make sure there is no intermediate switch between the fabric and the physical server interfaces. Check the OpenStack ACI plug-in release notes to make sure a particular configuration is supported. At this writing, there is limited support for B-Series blade servers, and the support is limited to VLAN mode only.

Verifying the OpenStack Configuration

Follow these steps to verify the OpenStack configuration:

**Step 1.** Verify that a VMM domain was created for the OpenStack system ID defined during installation. The nodes connected to the fabric that are running the OpFlex agent should be visible under Hypervisors. The virtual machines running on the hypervisor should be visible when you select that hypervisor. All networks created for this tenant should also be visible under the DVS submenu, and selecting the network should show you all endpoints connected to that network.

**Step 2.** Look at the health score and faults for the entity to verify correct operation. If the hypervisors are not visible or appear as being disconnected, check the OpFlex connectivity.
Step 3. Verify that there is a tenant created for the OpenStack tenant/project. All the networks created in OpenStack should show up as endpoint groups and corresponding bridge domains. Choose the Operational tab for the endpoint group to see all of the endpoints for that endpoint group.

Step 4. Check the Health Score tab and Faults tab to make sure there are no issues.

Configuration Examples for OpenStack and Cisco ACI

The following sections provide configuration examples for OpenStack and Cisco ACI.

Optimized Metadata and DHCP

In the configuration file, optimized DHCP is enabled by default in the OpFlex OpenStack plug-in. To disable optimized DHCP, add the following line:

```
enable_optimized_dhcp = False
```

In the configuration file, the optimized metadata service is disabled by default. To enable the optimized metadata, add the following line:

```
enable_optimized_metadata = True
```

External Network/NAT Configuration

You can define external network connectivity by adding an apic_external_network section to the configuration file, as in this example:

```
[apic_external_network:DC-Out]
preexisting=True
external_epg=DC-Out-EPG
host_pool_cidr=10.10.10.1/24
```

In this example, host_pool_cidr defines the SNAT subnet. You define the floating IP subnet by creating an external network in Neutron or an external policy in GBP. The name of the external network or policy should use the same name as apic_external_network that is defined in the file (in this case, DC-Out).

It is possible to disable NAT by adding enable_nat = False in the apic External network section. You can have multiple external networks using different Layer 3 Outs on ACI, and you can have a mix of NAT and non-NAT external networks.

In GBP deployment, network subnets for policy groups are carved out of the default_ip_pool setting defined in the plug-in configuration file, as in this example:

```
[group_policy_implicit_policy]
default_ip_pool = 192.168.10.0/16
```

This pool is used to allocate networks for created policy groups. You must make sure that the pool is large enough for the intended number of groups.
Kubernetes Integration

Kubernetes is a portable, extensible open-source platform that automates the deployment, scaling, and management of container-based workloads and services in a network. Beginning with Cisco APIC Release 3.0(1), you can integrate Kubernetes on bare-metal servers into Cisco ACI.

To integrate Kubernetes with Cisco ACI, you need to execute a series of tasks. Some of them you perform in the network to set up the Cisco APIC; others you perform on the Kubernetes server. Once you have integrated Kubernetes, you can use the Cisco APIC to view Kubernetes in the Cisco ACI.

The following sections show the workflow for integrating Kubernetes and provide specific instructions for setting up the Cisco APIC. However, it is assumed that you are familiar with Kubernetes and containers and can install Kubernetes. Specific instructions for installing Kubernetes are beyond the scope of this book.

The following are the basic tasks involved in integrating Kubernetes into the Cisco ACI fabric:

**Step 1.** Prepare for the integration and set up the subnets and VLANs in the network.

**Step 2.** Fulfill the prerequisites.

**Step 3.** To provision the Cisco APIC to integrate with Kubernetes, download the provisioning tool, which includes a sample configuration file, and update the configuration file with information you previously gathered about your network. Then run the provisioning tool with the information about your network.

**Step 4.** Set up networking for the node to support Kubernetes installation. This includes configuring an uplink interface, subinterfaces, and static routes.

**Step 5.** Install Kubernetes and Cisco ACI containers.

**Step 6.** Use the Cisco APIC GUI to verify that Kubernetes has been integrated into Cisco ACI.

The following sections provide details on these steps.

Planning for Kubernetes Integration

Various network resources are required to provide capabilities to a Kubernetes cluster, including several subnets and routers. You need the following subnets:

- **Node subnet:** This subnet is used for Kubernetes control traffic. It is where the Kubernetes API services are hosted. Make the node subnet a private subnet and make sure that it has access to the Cisco APIC management address.

- **Pod subnet:** This is the subnet from which the IP addresses of Kubernetes pods are allocated. Make the pod subnet a private subnet.
Note This subnet specifies the starting address for the IP pool that is used to allocate IP addresses to pods and your Cisco ACI bridge domain IP address. For example, if you define it as 192.168.255.254/16, this is a valid configuration from a Cisco ACI perspective. However, your containers will not get an IP address because there are no free IP addresses after 192.168.255.254 in this subnet. We suggest always using the first IP address in the pod subnet, which in this example would be 192.168.0.1/16.

- **Node service subnet**: This subnet is used for internal routing of load-balanced service traffic. Make the node service subnet a private subnet.

Note Much as with the pod subnet, you should configure the service subnet with the first IP address of the allocated subnet.

- **External service subnets**: These subnets are pools from which load-balanced services are allocated as externally accessible service IP addresses.

Note The externally accessible service IP addresses could be globally routable. You should configure the next-hop router to send traffic destined for these IP addresses to the fabric. There are two such pools: One is used for dynamically allocated IP addresses, and the other is available for services to request a specific fixed external IP address.

You need the following VLANs for local fabric use:

- **Node VLAN**: This VLAN is used by the physical domain for Kubernetes nodes.
- **Service VLAN**: This VLAN is used for delivery of load-balanced service traffic.
- **Infrastructure VLAN**: This is the infrastructure VLAN used by the Cisco ACI fabric.

**Prerequisites for Integrating Kubernetes with Cisco ACI**

Ensure that the following prerequisites are in place before you try to integrate Kubernetes with the Cisco ACI fabric:

- A working Cisco ACI installation
- An attachable entity profile (AEP) set up with interfaces that are desired for the Kubernetes deployment
- An L3Out connection, along with a Layer 3 external network to provide external access
- Virtual routing and forwarding (VRF)
The VRF and L3Out connection in Cisco ACI that are used to provide outside connectivity to Kubernetes external services can be in any tenant. The most common usage is to put the VRF and L3Out in the common tenant or in a tenant that is dedicated to the Kubernetes cluster. You can also have separate VRFs—one for the Kubernetes bridge domains and one for the L3Out—and you can configure route leaking between them.

- Any required route reflector configuration for the Cisco ACI fabric
- A next-hop router that is connected to the Layer 3 external network and that is capable of appropriate external access and configured with the required routes

In addition, the Kubernetes cluster must be up through the fabric-connected interface on all the hosts. The default route should be pointing to the ACI node subnet bridge domain. This is not mandatory, but it simplifies the routing configuration on the hosts and is the recommend configuration. If you choose not to use this design, all Kubernetes-related traffic must go through the fabric.

**Provisioning Cisco ACI to Work with Kubernetes**

You can use the acc_provision tool to provision the fabric for the Kubernetes VMM domain and generate a .yaml file that Kubernetes uses to deploy the required Cisco ACI container components. The procedure to accomplish this is as follows:

**Step 1.** Download the provisioning tool from

https://software.cisco.com/download/type.html?mdfid=285968390&i=rm and then follow these steps:

- **a.** Click APIC OpenStack and Container Plugins.
- **b.** Choose the package that you want to download.
- **c.** Click Download.

**Step 2.** Generate a sample configuration file that you can edit by entering the following command:

```
terminal$ acc-provision--sample
```

This command generates the aci-containers-config.yaml configuration file, which looks as follows:

```yaml
# # Configuration for ACI Fabric
#
aci_config:
  system_id: mykube # Every opflex cluster must have a distinct ID
```
apic_hosts: # List of APIC hosts to connect for APIC API
  - 10.1.1.101

vmm_domain: # Kubernetes VMM domain configuration
  encap_type: vxlan # Encap mode: vxlan or vlan
  mcast_range: # Every opflex VMM must use a distinct range
    start: 225.20.1.1
    end: 225.20.255.255

# The following resources must already exist on the APIC, they are used, but not created by the provisioning tool.

  aep: kube-cluster # The AEP for ports/VPCs used by this cluster
  vrf:
    name: mykube-vrf
    tenant: common # This can be system-id or common

l3out:
  name: mykube_l3out # Used to provision external IPs
  external_networks:
    - mykube_extepg # Used for external contracts

# Networks used by Kubernetes

# net_config:
  node_subnet: 10.1.0.1/16 # Subnet to use for nodes
  pod_subnet: 10.2.0.1/16 # Subnet to use for Kubernetes Pods
  extern_dynamic: 10.3.0.1/24 # Subnet to use for dynamic external IPs
  extern_static: 10.4.0.1/24 # Subnet to use for static external IPs
  node_svc_subnet: 10.5.0.1/24 # Subnet to use for service graph

  kubeapi_vlan: 4001 # The VLAN used by the physdom for nodes
  service_vlan: 4003 # The VLAN used by LoadBalancer services
  infra_vlan: 4093 # The VLAN used by ACI infra

# Configuration for container registry
# Update if a custom container registry has been setup

# registry:
  image_prefix: noiro # e.g: registry.example.com/noiro
  # image_pull_secret: secret_name # (if needed)
Chapter 6: VMM Integration

**Note**  Do not modify the Cisco ACI bridge domain configuration that is pushed by the acc-provisioning tool. Setting the bridge domain to flood results in a broken environment.

**Step 3.** Edit the sample configuration file, providing information from your network, and save the file.

**Step 4.** Provision the Cisco ACI fabric by using the following command:

```
acc-provision -c aci-containers-config.yaml -o aci-containers.yaml -f kubernetes-<version> -a -u [apic username] -p [apic password]
```

This command generates the file aci-containers.yaml, which you use after installing Kubernetes. It also creates the files user-[system id].key and user-[system id].crt, which contain the certificate used to access the Cisco APIC. Save these files in case you change the configuration later and want to avoid disrupting a running cluster because of a key change.

**Note**  The file aci-containers.yaml is security sensitive. It contains keys necessary for connecting to the Cisco APIC administration API.

**Note**  Currently, the provisioning tool supports only the installation of a single Kubernetes cluster on a single or multi-pod Cisco ACI fabric. However, you can run the tool as often as needed to install multiple Kubernetes clusters. A single Cisco ACI installation can support more than one Kubernetes cluster.

**Step 5.** (Optional) Configure advanced optional parameters to adjust to custom parameters other than the ACI default values or base provisioning assumptions. For example, if your VMM’s multicast address for the fabric is different from 225.1.2.3, you can configure it by using the following:

```
aci_config:
  vmm_domain:
    mcast_fabric: 225.1.2.3
```

If you are using VLAN encapsulation, you can specify the VLAN pool for it, as follows:

```
aci_config:
  vmm_domain:
    encap_type: vlan
```
Preparing the Kubernetes Nodes

When you are done provisioning Cisco ACI to work with Kubernetes, you can start preparing the networking construct for the Kubernetes nodes by following this procedure:

**Step 1.** Configure your uplink interface with or without NIC bonding, depending on how your AAEP is configured. Set the MTU on this interface to 1600.

**Step 2.** Create a subinterface on your uplink interface on your infrastructure VLAN. Configure this subinterface to obtain an IP address by using DHCP. Set the MTU on this interface to 1600.

**Step 3.** Configure a static route for the multicast subnet 224.0.0.0/4 through the uplink interface used for VXLAN traffic.

**Step 4.** Create a subinterface (for example, kubeapi_vlan) on the uplink interface on your node VLAN in the configuration file. Configure an IP address on this interface in your node subnet. Then set this interface and the corresponding node subnet router as the default route for the node.

**Note** Many Kubernetes installer tools look specifically for the default route to choose interfaces for API server traffic and other traffic. It's possible to install with the default route on another interface. To accomplish this, you set up a number of static routes into this interface and override your installer configuration. However, we recommend setting up the default route through the node uplink.
Step 5. Create the /etc/dhcp/dhclient-eth0.4093.conf file with the following content, inserting the MAC address of the Ethernet interface for each server on the first line of the file:

Note If you have a single interface, you could name the file dhclient.conf without the added interface name, as in dhclient-eth0.4093.conf.

```plaintext
send dhcp-client-identifier 01:<mac-address of infra VLAN interface>;
request subnet-mask, domain-name, domain-name-servers, host-name;
send host-name <server-host-name>;

option rfc3442-classless-static-routes code 121 = array of unsigned integer 8;
option ms-classless-static-routes code 249 = array of unsigned integer 8;
option wpad code 252 = string;

also request rfc3442-classless-static-routes;
also request ms-classless-static-routes;
also request static-routes;
also request wpad;
also request ntp-servers;
```

The network interface on the infrastructure VLAN requests a DHCP address from the APIC infrastructure network for OpFlex communication. Make sure the server has a dhclient configuration for this interface to receive all the correct DHCP options with the lease.

Note The infrastructure VLAN interface in your environment may be a basic Linux-level subinterface, such as eth0.4093.

Step 6. If you have a separate management interface for the node being configured, configure any static routes that you need to access your management network on the management interface.

Step 7. Ensure that OVS is not running on the node.

Here is an example of the interface configuration (in /etc/network/interfaces):

```plaintext
# Management network interface (not connected to ACI)
auto ens160
iface ens160 inet static
   address  192.168.66.17
   netmask 255.255.255.0
   up route add -net 10.0.0.0/8 gw 192.168.66.1
dns-nameservers 192.168.66.1
```
ange to ACI
auto ens192
iface ens192 inet manual
   mtu 1600

# ACI Infra VLAN
auto ens192.3095
iface ens192.3095 inet dhcp
   mtu 1600
   up route add -net 224.0.0.0/4 dev ens192.3095
   vlan-raw-device ens192

# Node Vlan
auto ens192.4001
iface ens192.4001 inet static
   address 12.1.0.101
   netmask 255.255.0.0
   mtu 1600
   gateway 12.1.0.1
   vlan-raw-device ens192

## Installing Kubernetes and Cisco ACI Containers

After you provision Cisco ACI to work with Kubernetes and prepare the Kubernetes nodes, you can install Kubernetes and ACI containers. You can use any installation method you choose, as long as it is appropriate to your environment. This procedure provides guidance and high-level instruction for installation; for details, consult Kubernetes documentation.

When installing Kubernetes, ensure that the API server is bound to the IP addresses on the node subnet and not to management or other IP addresses. Issues with node routing table configuration and API server advertisement addresses are the most common problems during installation. If you have problems, therefore, check these issues first.

Install Kubernetes so that it is configured to use a Container Network Interface (CNI) plug-in, but do not install a specific CNI plug-in configuration through your installer. Instead, deploy the CNI plug-in. To install the CNI plug-in, use the following command:

```bash
collectl apply -f aci-containers.yaml
```

**Note** You can use this command wherever you have kubectl set up—generally from a Kubernetes master node. The command installs the following:

- ACI container host agent and OpFlex agent in a daemon set called aci-containers-host
- Open vSwitch in a daemon set called aci-containers-openvswitch
- ACI containers controller in a deployment called aci-containers-controller
- Other required configurations, including service accounts, roles, and security context
Verifying the Kubernetes Integration

After you have performed the steps described in the preceding sections, you can verify the integration in the Cisco APIC GUI. The integration creates a tenant, three EPGs, and a VMM domain. The procedure to do this is as follows:

**Step 1.** Log in to the Cisco APIC.

**Step 2.** Go to Tenants > tenant_name, where tenant_name is the name you specified in the configuration file that you edited and used in installing Kubernetes and the ACI containers.

**Step 3.** In the tenant navigation pane, expand the following: tenant_name > Application Profiles > application_profile_name > Application EPGs. You should see three folders inside the Application EPGs folder:

- **kube-default:** The default EPG for containers that are otherwise not mapped to any specific EPG.
- **kube-nodes:** The EPG for the Kubernetes nodes.
- **kube-system:** The EPG for the kube-system Kubernetes namespace. This typically contains the kube-dns pods, which provide DNS services for a Kubernetes cluster.

**Step 4.** In the tenant navigation pane, expand the Networking and Bridge Domains folders. You should see two bridge domains:

- **node-bd:** The bridge domain used by the node EPG
- **pod-bd:** The bridge domain used by all pods

**Step 5.** If you deploy Kubernetes with a load balancer, go to Tenants > common, expand L4-L7 Services, and perform the following steps:

- Open the L4-L7 Service Graph Templates folder; you should see a template for Kubernetes.
- Open the L4-L7 Devices folder; you should see a device for Kubernetes.
- Open the Deployed Graph Instances folder; you should see an instance for Kubernetes.

**Step 6.** Go to VM Networking > Inventory, and in the Inventory navigation pane, expand the Kubernetes folder. You should see a VMM domain, with the name you provided in the configuration file, and in that domain you should see folders called Nodes and Namespaces.
OpenShift Integration

OpenShift is a container application platform that is built on top of Docker and Kubernetes that makes it easy for developers to create applications and provides a platform for operators that simplifies deployment of containers for both development and production workloads. Beginning with Cisco APIC Release 3.1(I), OpenShift can be integrated with Cisco ACI by leveraging the ACI CNI plug-in.

To integrate Red Hat OpenShift with Cisco ACI, you must perform a series of tasks. Some tasks are performed by the ACI fabric administrator directly on the APIC, and others are performed by the OpenShift cluster administrator. After you have integrated the Cisco ACI CNI plug-in for Red Hat OpenShift, you can use the APIC to view OpenShift endpoints and constructs within the fabric.

**Note**  This section describes the workflow for integrating OpenShift with ACI. However, it is assumed that you are familiar with OpenShift and containers and have knowledge of installation. Specific instructions for installing OpenShift are beyond the scope of this book.

The following is a high-level look at the tasks required to integrate OpenShift with the Cisco ACI fabric:

**Step 1.** To prepare for the integration, identify the subnets and VLANs that you will use in your network.

**Step 2.** Perform the required Day 0 fabric configurations.

**Step 3.** Configure the Cisco APIC for the OpenShift cluster. Many of the required fabric configurations are performed directly with a provisioning tool (acc-provision). The tool is embedded in the plug-in files from www.cisco.com. Once downloaded and installed, modify the configuration file with the information from the planning phase and run the tool.

**Step 4.** Set up networking for the node to support OpenShift installation. This includes configuring an uplink interface, subinterfaces, and static routes.

**Step 5.** Install OpenShift and Cisco ACI containers.

**Step 6.** Update the OpenShift router to use the ACI fabric.

**Step 7.** Use the Cisco APIC GUI to verify that OpenShift has been integrated into the Cisco ACI.

The following sections provide details on these steps.
Planning for OpenShift Integration

The OpenShift cluster requires various network resources, all of which are provided by the ACI fabric integrated overlay. The OpenShift cluster requires the following subnets:

- **Node subnet**: This is the subnet used for OpenShift control traffic. This is where the OpenShift API services are hosted. The acc-provisioning tool configures a private subnet. Ensure that it has access to the Cisco APIC management address.

- **Pod subnet**: This is the subnet from which the IP addresses of OpenShift pods are allocated. The acc-provisioning tool configures a private subnet.

**Note**  This subnet specifies the starting address for the IP pool that is used to allocate IP addresses to pods as well as your ACI bridge domain IP address. For example, if you define it as 192.168.255.254/16, which is a valid configuration from an ACI perspective, your containers do not get IP addresses as there are no free IP addresses after 192.168.255.254 in this subnet. We suggest always using the first IP address in the pod subnet, which in this example is 192.168.0.1/16.

- **Node service subnet**: This is the subnet used for internal routing of load-balanced service traffic. The acc-provisioning tool configures a private subnet.

**Note**  Much as with the pod subnet, you should configure the node service subnet with the first IP address in the subnet.

- **External service subnets**: These are pools from which load-balanced services are allocated as externally accessible service IP addresses.

The externally accessible service IP addresses could be globally routable. Configure the next-hop router to send traffic destined for IP addresses to the fabric. There are two such pools: One is used for dynamically allocated IPs, and the other is available for services to request a specific fixed external IP address.

All of the aforementioned subnets must be specified on the acc-provisioning configuration file. The node pod subnets are provisioned on corresponding ACI bridge domains that are created by the provisioning tool. The endpoints on these subnets are learned as fabric endpoints and can be used to communicate directly with any other fabric endpoint without NAT, provided that contracts allow communication. The node service subnet and the external service subnet are not seen as fabric endpoints but are instead used to manage the cluster IP address and the load balancer IP address, respectively, and are programmed on Open vSwitch via OpFlex. As mentioned earlier, the external service subnet must be routable outside the fabric.

OpenShift nodes need to be connected on an EPG using VLAN encapsulation. Pods can connect to one or multiple EPGs and can use either VLAN or VXLAN encapsulation. In addition, PBR-based load balancing requires the use of a VLAN encapsulation to reach
the OpFlex service endpoint IP address of each OpenShift node. The following VLAN IDs are therefore required:

- **Node VLAN ID**: The VLAN ID used for the EPG mapped to a physical domain for OpenShift nodes
- **Service VLAN ID**: The VLAN ID used for delivery of load-balanced external service traffic
- **The fabric infrastructure VLAN ID**: The infrastructure VLAN used to extend OpFlex to the OVS on the OpenShift nodes

### Prerequisites for Integrating OpenShift with Cisco ACI

Ensure that the following prerequisites are in place before you try to integrate OpenShift with the Cisco ACI fabric:

- A working Cisco ACI fabric running a release that is supported for the desired OpenShift integration
- An attachable entity profile (AEP) set up with the interfaces desired for the OpenShift deployment (When running in nested mode, this is the AEP for the VMM domain on which OpenShift will be nested.)
- An L3Out connection, along with a Layer 3 external network to provide external access
- **VRF**

**Note**  The VRF and L3Out connection in Cisco ACI that are used to provide outside connectivity to OpenShift external services can be in any tenant. The most common usage is to put the VRF and L3Out in the common tenant or in a tenant that is dedicated to the OpenShift cluster. You can also have separate VRFs—one for the OpenShift bridge domains and one for the L3Out—and you can configure route leaking between them.

- Any required route reflector configuration for the Cisco ACI fabric

In addition, ensure that the subnet used for external services is routed by the next-hop router that is connected to the selected ACI L3Out interface. This subnet is not announced by default, so either static routes or appropriate configuration must be considered.

In addition, the OpenShift cluster must be up through the fabric-connected interface on all the hosts. The default route on the OpenShift nodes should be pointing to the ACI node subnet bridge domain. This is not mandatory, but it simplifies the routing configuration on the hosts and is the recommended configuration. If you do not follow this design, ensure that the OpenShift node routing is correctly used so that all OpenShift cluster traffic is routed through the ACI fabric.
Provisioning Cisco ACI to Work with OpenShift

You can use the acc_provision tool to provision the fabric for the OpenShift VMM domain and generate a .yaml file that OpenShift uses to deploy the required Cisco ACI container components. This tool requires a configuration file as input and performs two actions as output:

■ It configures relevant parameters on the ACI fabric.
■ It generates a YAML file that OpenShift administrators can use to install the ACI CNI plug-in and containers on the cluster.

Note  We recommended that when using ESXi nested for OpenShift hosts, you provision one OpenShift host for each OpenShift cluster for each ESXi server. Doing so ensures that, in the event of an ESXi host failure, a single OpenShift node is affected for each OpenShift cluster.

The procedure to provision Cisco ACI to work with OpenShift is as follows:

Step 1. Download the provisioning tool from https://software.cisco.com/download/type.html?mdfid=285968390&i=rm and then follow these steps:

a. Click APIC OpenStack and Container Plugins.
b. Choose the package that you want to download.
c. Click Download.

Step 2. Generate a sample configuration file that you can edit by entering the following command:

terminal$ acc-provision--sample

Note  Take note of the values if you are provisioning OpenStack to work with OpenShift.

This command generates the aci-containers-config.yaml configuration file, which looks as follows:

```yaml
# Configuration for ACI Fabric
#
aci_config:
  system_id: mykube  # Every opflex cluster must have a distinct ID
  apic_hosts:       # List of APIC hosts to connect for
    APIC API - 10.1.1.101
  vmm_domain:       # Kubernetes VMM domain configuration
```
encap_type: vxlan           # Encap mode: vxlan or vlan
mcast_range:               # Every opflex VMM must use a distinct range
    start: 225.20.1.1
    end: 225.20.255.255
# The following resources must already exist on the APIC,
# they are used, but not created by the provisioning tool.
aep: kube-cluster         # The AEP for ports/VPCs used by this cluster
vrf:
    name: mykube-vrf       # This VRF used to create all kubernetes EPs
    tenant: common        # This can be system-id or common
l3out:
    name: mykube_l3out    # Used to provision external IPs
    external_networks:
        - mykube_extepg   # Used for external contracts
# # Networks used by Kubernetes #
net_config:
    node_subnet: 10.1.0.1/16    # Subnet to use for nodes
    pod_subnet: 10.2.0.1/16     # Subnet to use for Kubernetes Pods
    extern_dynamic: 10.3.0.1/24 # Subnet to use for dynamic external IPs
    node_svc_subnet: 10.5.0.1/24 # Subnet to use for service graph
    kubeapi_vlan: 4001         # The VLAN used by the physdom for nodes
    service_vlan: 4003         # The VLAN used by LoadBalancer services
    infra_vlan: 4093           # The VLAN used by ACI infra
# # Configuration for container registry
# Update if a custom container registry has been setup
# registry:
    image_prefix: noiro       # e.g: registry.example.com/noiro
    # image_pull_secret: secret_name     # (if needed)

**Note** The APIC administrator must not modify the Cisco ACI bridge domain configuration that is pushed by the acc-provisioning tool.
Note Make sure to remove the following line from the `net_config` section:

```
extern_static: 10.4.0.1/24    # Subnet to use for static external IPs
```

This subnet is not used for OpenShift.

**Step 3.** Edit the sample configuration file with the relevant values for each of the subnets, VLANs, and so on, as appropriate to your planning, and then save the file.

**Step 4.** Provision the Cisco ACI fabric by using the following command:

```
acc-provision -f openshift-<version> -c aci-containers-config.yaml -o aci-containers.yaml \
-a -u [apic username] -p [apic password]
```

This command generates the file `aci-containers.yaml`, which you use after installing OpenShift. It also creates the files `user-[system id].key` and `user-[system id].crt`, which contain the certificate that is used to access the Cisco APIC. Save these files in case you change the configuration later and want to avoid disrupting a running cluster because of a key change.

Note The file `aci-containers.yaml` is security sensitive. It contains keys necessary for connecting to the Cisco APIC administration API.

Note Currently, the provisioning tool supports only the installation of a single OpenShift cluster on a single or multi-pod ACI fabric. However, you can run the tool as often as needed to install multiple OpenShift clusters. A single ACI installation can support more than one OpenShift cluster.

**Step 5.** (Optional) Configure advanced optional parameters to adjust to custom parameters other than the ACI default values or base provisioning assumptions. For example, if your VMM’s multicast address for the fabric is different from 225.1.2.3, you can configure it by adding the following:

```
aci_config:
  vmm_domain:
    mcast_fabric: 225.1.2.3
```

If you are using VLAN encapsulation, you can specify the VLAN pool for it, as follows:

```
aci_config:
  vmm_domain:
    encap_type: vlan
```
Preparing the OpenShift Nodes

After you provision Cisco ACI, you prepare networking for the OpenShift nodes by following this procedure:

**Step 1.** Configure your uplink interface with or without NIC bonding, depending on how your AAEP is configured. Set the MTU on this interface to 1600.

**Step 2.** Create a subinterface on your uplink interface on your infrastructure VLAN. Configure this subinterface to obtain an IP address by using DHCP. Set the MTU on this interface to 1600.

**Step 3.** Configure a static route for the multicast subnet 224.0.0.0/4 through the uplink interface that is used for VXLAN traffic.

**Step 4.** Create a subinterface (for example, kubeapi_vlan) on the uplink interface on your node VLAN in the configuration file. Configure an IP address on this interface in your node subnet. Then set this interface and the corresponding node subnet router as the default route for the node.

**Note** Many OpenShift installer tools look specifically for the default route to choose interfaces for API server traffic and other traffic. It's possible to install with the default route on another interface. To do this, you set up static routes into this interface and override your installer configuration. However, we recommend setting up the default route through the node uplink.
Step 5. Create the /etc/dhcp/dhclient-eth0.4093.conf file with the following content, inserting the MAC address of the Ethernet interface for each server on the first line of the file:

```
send dhcp-client-identifier 01:<mac-address of infra VLAN interface>;
request subnet-mask, domain-name, domain-name-servers, host-name;
send host-name <server-host-name>;

option rfc3442-classless-static-routes code 121 = array of unsigned integer 8;
option ms-classless-static-routes code 249 = array of unsigned integer 8;
option wpad code 252 = string;

also request rfc3442-classless-static-routes;
also request ms-classless-static-routes;
also request static-routes;
also request wpad;
also request ntp-servers;
```

**Note** If you have a single interface, you could name the file just dhclient.conf and not include the interface name, as in dhclient-eth0.4093.conf.

The network interface on the infrastructure VLAN requests a DHCP address from the Cisco APIC infrastructure network for OpFlex communication. The server must have a dhclient configuration for this interface to receive all the correct DHCP options with the lease.

**Note** The infrastructure VLAN interface in your environment may be a basic Linux-level subinterface, such as eth0.4093.

Step 6. If you have a separate management interface for the node being configured, configure any static routes required to access your management network on the management interface.

Step 7. Ensure that OVS is not running on the node.

Here is an example of the interface configuration (in /etc/network/interfaces):

```
# Management network interface (not connected to ACI)
# /etc/sysconfig/network-scripts/ifcfg-eth0
NAME=eth0
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
TYPE=Ethernet
```
IPADDR=192.168.66.17
NETMASK=255.255.255.0
PEERDNS=no
DNS1=192.168.66.1

# /etc/sysconfig/network-scripts/route-eth0
ADDRESS0=10.0.0.0
NETMASK0=255.0.0.0
GATEWAY0=192.168.66.1

# Interface connected to ACI
# /etc/sysconfig/network-scripts/ifcfg-eth1
NAME=eth1
DEVICE=eth1
ONBOOT=yes
BOOTPROTO=none
TYPE=Ethernet
IMTU=1600

# ACI Infra VLAN
# /etc/sysconfig/network-scripts/ifcfg-4093
VLAN=yes
TYPE=Vlan
PHYSDEV=eth1
VLAN_ID=4093
REORDER_HDR=yes
BOOTPROTO=dhcp
DEFRROUTE=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=4093
DEVICE=eth1.4093
ONBOOT=yes
MTU=1600

# /etc/sysconfig/network-scripts/route-4093
ADDRESS0=224.0.0.0
NETMASK0=240.0.0.0
METRIC0=1000

# Node Vlan
# /etc/sysconfig/network-scripts/ifcfg-node-vlan-4001
VLAN=yes
TYPE=Vlan
PHYSDEV=eth1
Installing OpenShift and Cisco ACI Containers

After you provision Cisco ACI and prepare the OpenShift nodes, you can install OpenShift and ACI containers. You can use any installation method appropriate to your environment. We recommend using this procedure to install the OpenShift and Cisco ACI containers.

When installing OpenShift, ensure that the API server is bound to the IP addresses on the node subnet and not to management or other IP addresses. Issues with node routing table configuration, API server advertisement addresses, and proxies are the most common problems during installation. If you have problems, therefore, check these issues first.

The procedure for installing OpenShift and Cisco ACI containers is as follows:

**Step 1.** Install OpenShift by using the following command:

```bash
git clone https://github.com/noironetworks/openshift-ansible/tree/release-3.9
git checkout release-3.9
```

Follow the installation procedure provided at https://docs.openshift.com/container-platform/3.9/install_config/install/advanced_install.html. Also consider the configuration overrides listed at https://github.com/noironetworks/openshift-ansible/tree/release-3.9/roles/aci.

**Step 2.** Install the CNI plug-in by using the following command:

```bash
oc apply -f aci-containers.yaml
```

**Note** You can use this command wherever you have oc set up—generally from an OpenShift master node. The command installs the following:

- ACI containers host agent and OpFlex agent in a daemon set called aci-containers-host
- Open vSwitch in a daemon set called aci-containers-openvswitch
- ACI containers controller in a deployment called aci-containers-controller
- Other required configurations, including service accounts, roles, and security context
Updating the OpenShift Router to Use the ACI Fabric

To update the OpenShift router to use the ACI fabric, follow these steps:

**Step 1.** Remove the old router by entering the commands such as the following:

```bash
oc delete svc router
oc delete dc router
```

**Step 2.** Create the container networking router by entering a command such as the following:

```bash
oc adm router --service-account=router --host-network=false
```

**Step 3.** Expose the router service externally by entering a command such as the following:

```bash
oc patch svc router -p '{"spec":{"type": "LoadBalancer"}}'
```

Verifying the OpenShift Integration

After you have performed the steps described in the preceding sections, you can verify the integration in the Cisco APIC GUI. The integration creates a tenant, three EPGs, and a VMM domain. The procedure to do this is as follows:

**Step 1.** Log in to the Cisco APIC.

**Step 2.** Go to Tenants > `tenant_name`, where `tenant_name` is the name you specified in the configuration file that you edited and used in installing OpenShift and the ACI containers.

**Step 3.** In the tenant navigation pane, expand the following: `tenant_name` > Application Profiles > `application_profile_name` > Application EPGs. You should see three folders inside the Application EPGs folder:

- **kube-default**: The default EPG for containers that are otherwise not mapped to any specific EPG.
- **kube-nodes**: The EPG for the OpenShift nodes.
- **kube-system**: The EPG for the kube-system OpenShift namespace. This typically contains the kube-dns pods, which provide DNS services for a OpenShift cluster.

**Step 4.** In the tenant navigation pane, expand the Networking and Bridge Domains folders, and you should see two bridge domains:

- **node-bd**: The bridge domain used by the node EPG
- **pod-bd**: The bridge domain used by all pods
Step 5. If you deploy OpenShift with a load balancer, go to Tenants > common, expand L4-L7 Services, and perform the following steps:

- Open the L4-L7 Service Graph Templates folder; you should see a template for OpenShift.
- Open the L4-L7 Devices folder; you should see a device for OpenShift.
- Open the Deployed Graph Instances folder; you should see an instance for OpenShift.

Step 6. Go to VM Networking > Inventory, and in the Inventory navigation pane, expand the OpenShift folder. You should see a VMM domain, with the name you provided in the configuration file, and in that domain you should see folders called Nodes and Namespaces.

VMM Integration with ACI at Multiple Locations

In a single ACI fabric with a single APIC cluster located at a single site or stretched between multiple sites using transit leaf, multi-pod, or remote leaf design options, individual VMM integration can be leveraged using the same policy model in any of the locations where the ACI fabric is stretched. This is because a single control and data plane has been stretched between multiple data center locations. In a dual ACI fabric or multi-site environments, separate APIC clusters are deployed in each location and, therefore, a separate VMM domain is created for each site.

Multi-Site

In order to integrate VMM domains into a Cisco ACI multi-site architecture, as mentioned earlier, you need to create separate VMM domains at each site because the sites have separate APIC clusters. Those VMM domains can then be exposed to the ACI multi-site policy manager in order to be associated to the EPGs defined at each site.

Two deployment models are possible:

- Multiple VMMs can be used across separate sites, each paired with the local APIC cluster.
- A single VMM can be used to manage hypervisors deployed across sites and paired with the different local APIC clusters.

The next two sections provide more information about these models.

Multiple Virtual Machine Managers Across Sites

In a multi-site deployment, multiple VMMs are commonly deployed in separate sites to manage the local clusters of hypervisors. Figure 6-13 shows this scenario.
VMM Integration with ACI at Multiple Locations

Multiple VMM Domains Across Multiple Sites

Figure 6-13
The VMM at each site manages the local hosts and peers with the local APIC domain to create a local VMM domain. This model is supported by all the VMM options supported by Cisco ACI: VMware vCenter Server, Microsoft SCVMM, and OpenStack controller.

The configuration of the VMM domains is performed at the local APIC level. The created VMM domains can then be imported into the Cisco ACI multi-site policy manager and associated with the EPG specified in the centrally created templates. If, for example, EPG 1 is created at the multi-site level, it can then be associated with VMM domain DC 1 and with VMM domain DC 2 before the policy is pushed to Sites 1 and 2 for local implementation.

The creation of separate VMM domains across sites usually restricts the mobility of virtual machines across sites to cold migration scenarios. However, in specific designs using VMware vSphere 6.0 and later, you can perform hot migration between clusters of hypervisors managed by separate vCenter servers. Figure 6-14 and the list that follows demonstrate and describe the steps required to create such a configuration.

**Note** At this writing, vCenter Server Release 6.0 or later is the only VMM option that allows live migration across separate Cisco ACI fabrics. With other VMMs (such as vCenter releases earlier than 6.0, SCVMM, and OpenStack deployments), if you want to perform live migration, you must deploy the VMMs in a single Cisco ACI fabric (single pod or multi-pod). Please check Cisco.com for the latest updates.

![Figure 6-14](image)

*Figure 6-14  Live Migrations Across VMM Domains with vCenter 6.0 and Later*
Step 1. Create a VMM domain in each fabric by peering the local vCenter server and the APIC. This peering results in the creation of local vSphere distributed switches (VDS 1 at Site 1 and VDS 2 at Site 2) in the ESXi clusters.

Step 2. Expose the created VMM domains to the Cisco ACI multi-site policy manager.

Step 3. Define a new Web EPG in a template associated with both Sites 1 and 2. The EPG is mapped to a corresponding Web bridge domain, which must be configured as stretched with flooding across sites enabled. At each site, the EPG then is associated with the previously created local VMM domain.

Step 4. Push the template policy Sites 1 and 2.

Step 5. Create the EPGs in each fabric, and because they are associated with VMM domains, each APIC communicates with the local vCenter server, which pushes an associated Web port group to each VDS.

Step 6. Connect the Web virtual machines to the newly created Web port groups. At this point, live migration can be performed across sites.

Single Virtual Machine Manager Across Sites

Figure 6-15 depicts the scenario in which a single VMM domain is used across sites.

In this scenario, a VMM is deployed in Site 1 but manages a cluster of hypervisors deployed within the same fabric and also in separate fabrics. Note that this configuration still leads to the creation of different VMM domains in each fabric, and different VDSs are pushed to the ESXi hosts that are locally deployed. This scenario essentially raises the same issues as discussed in the previous section about the support for cold and hot migration of virtual machines across fabrics.

Remote Leaf

ACI fabric allows for integration with multiple VMM domains. With this integration, the APIC pushes the ACI policy configuration—such as networking, telemetry monitoring, and troubleshooting—to switches based on the locations of virtual instances. The APIC can push the ACI policy in the same way as a local leaf. A single VMM domain can be created for compute resources connected to both the ACI main DC pod and remote leaf switches. VMM/APIC integration is also used to push a VDS to hosts managed by the VMM and to dynamically create port groups as a result of the creation of EPGs and their association to the VMM domain. This allows you to enable mobility ("live" or "cold") for virtual endpoints across different compute hypervisors.

Note It is worth noting that mobility for virtual endpoints can also be supported if a VMM domain is not created (that is, if VMs are treated as physical resources).
Figure 6.15 Single VMM Domain Managing VMs Across Multiple Sites
Virtual instances in the same EPG or Layer 2 domain (VLAN) can be behind the local leaf as well as the remote leaf. When a virtual instance moves from the remote leaf to the local leaf or vice versa, the APIC detects the leaf switches where virtual instances are moved and pushes the associated policies to new leaves. All VMM and container domain integration supported for local leaves is supported for remote leaves as well.

Figure 6-16 shows the process of vMotion with the ACI fabric.

![Figure 6-16 vMotion Between Remote Leaf to ACI Fabric in the Main Data Center](image)

The following events happen during a vMotion event:

**Step 1.** The VM has IP address 10.10.10.100 and is part of the Web EPG and the Web bridge domain with subnet 10.10.10.1/24. When the VM comes up, the ACI fabric programs the encapsulation VLAN (vlan-100) and the switch virtual interface (SVI), which is the default gateway of the VM on the leaf switches where the VM is connected. The APIC pushes the contract and other associated policies based on the location of the VM.

**Step 2.** When the VM moves from a remote leaf to a local leaf, the ACI detects the location of the VM through the VMM integration.

**Step 3.** Depending on the EPG-specific configuration, the APIC may need to push the ACI policy on the leaf for successful VM mobility, or a policy may already exist on the destination leaf.
Integrating the virtual compute platform into ACI extends the policy model down and provides deep visibility into the virtualization layer. As discussed in this chapter, due to the open architecture of Cisco ACI, any hypervisor or container-based platform vendor—such as VMware, Microsoft, OpenStack, Kubernetes, or OpenShift—can be integrated into ACI.

In a single ACI fabric located at a single site or stretched between multiple sites using transit leaf, multi-pod, or remote leaf, individual VMM integration can be leveraged using the same policy model in any of the locations where the ACI fabric is stretched because a single control and data plane has been stretched between multiple data center locations. In a dual ACI fabric or multi-site environment, separate APIC clusters are deployed in each location; therefore, a separate VMM domain is created for each site.

**Note** There are no Key Topics or Review Questions for this chapter.
AAA (authentication, authorization, accounting) policies, 36
configuring, 533–551
Cisco ISE configuration, 542–547
Cisco Secure ACS configuration, 533–542
steps in, 547–549
verifying configuration, 550–551
AAEP (attachable access entity profile), 56–57, 186–187
access policies, 597–600
access policies, 597–600
access ports, 201–206
configuring, 202–203
interface policies, 204–205
policy groups, 596
switch profiles, 205–206
access SPAN, configuring, 567–570
accessing
Bash shell, 74
iBash CLI, 78
NX-OS-style CLI, 68
VSH shell, 82
VSH_LC shell, 83
ACI (Application Centric Infrastructure)
Clos topology, 612
domains, 597–600
interface policies, 596–597
interface profiles, 595–596
NetFlow configuration with, 580–582
overview, 600–601
switch policies, 595–596
switch profiles, 595–596
VLAN pools, 597–600
accessing
Bash shell, 74
iBash CLI, 78
NX-OS-style CLI, 68
VSH shell, 82
VSH_LC shell, 83
ACI (Application Centric Infrastructure)
Clos topology, 612
configuration management, 7
control plane protocols, 15–17
data plane protocols, 17–18
design philosophies, 6
explained, 3–4, 8
goals and objectives, 6
hardware specifications, 8–14
   APIC (ACI controllers), 12–14
   Nexus 9000 Series, 9–12
key concepts, 14–15
security, 6–7
three-tier network infrastructure
   versus, 4–6
version compatibility for NAE
   (Network Assurance Engine), 440
ACI fabric design
   logical design, 149–180
      container-as-a-service (CaaS), 149–165
      vBrick Digital Media Engine (DME), 175–180
      vendor-based ERP (enterprise resource planning), 165–175
physical design, 85–148
   multi-pod, 97–116
   multi-site, 116–130
   remote leaf, 131–142
   remote leaf and multi-pod integration, 143–148
   single-fabric versus multiple-fabric, 87–97
ACI fabric switch CLIs (command-line interfaces), 78–84
iBash CLI, 78–81
VSH shell, 81–82
VSH _LC shell, 83–84
ACI policy model, 31–63
   benefits of, 37
   characteristics of, 32–33
   logical constructs, 37–38
      application profile objects, 40
      bridge domain objects, 43–46
      contract objects, 46–50
      EPG objects, 41–43
      outside network policies, 51–52
      subnet objects, 43–44
      tenant objects, 38–39
      VRF objects, 39–40
      vzAny managed objects, 50–51
MIT (management information tree), 33–37
physical constructs, 52
   access policies, 52
   default policies, 58–60
   global policies, 55–57
   interface policies, 54–55
   managed object relationships, 57–58
   policy resolution, 57–58
   switch policies, 53
tags, 58
troubleshooting with, 60–63
for VMM domains, 250
ACLQOS (Access Control List and Quality of Service), 604
ACLs (access control lists), 46
ADC
   PBR service graph with, 324
   service graph with, 316–317
ADM (Application Dependency Mapping), 466
administration domains (ADs), 90
ADs (administration domains), 90
advertising subnets, 651–656
aliases, 48–49
  in iBash CLI, 81
Ansible, 372–392
  ACI support, 375–378
  APIC authentication, 382–384
    explained, 372–375
    installing, 378–381
    use cases, 384–392
Anycast Gateway, 3
anycast services
  gateways, 647–649
    in multi-pod design, 334–338
API Inspector, 351–353
APIC (ACI controllers), 12–14
  authentication in Ansible, 382–384
  bond interfaces, viewing, 730
CLIs (command-line interfaces),
  68–78
    Bash shell, 74–78
      NX-OS-style, 68–74
cluster deployment, 113–116
cluster troubleshooting, 795–798
initial setup, 593–595, 728
LLDP frames, viewing, 730–731
mapping multiple SCVMMs to, 262
monitoring, 475–482
  purpose of, 14–15
    in three-site stretch fabric, 97
troubleshooting clusters, 727–734
  in two-site stretch fabric, 97
  wiring errors, 732–733
Application Centric Infrastructure. See ACI (Application Centric Infrastructure)
Application Dependency Mapping (ADM), 466
application deployment workflow
  policy-based model, 3
  traditional, 1–3
application profiles, 20–21, 40,
  384–388
application-centric design, 6
  logical topology for, 606
applications
  connectivity
    standalone compute application
      connectivity example,
        433–435
    virtual compute application
      connectivity example, 435
monitoring, 499–505
  external network connectivity,
    502–504
  PBR service graph, 504–505
  traffic status, 499–502
    in Tetration, 465–467
ARP flooding, 686
  in multi-pod design, 700
  in multi-site design, 703–705
  in remote leaf design, 707–710
ARP optimization, 690
ARP replies in Layer 2 known unicast, 688
ASIC interface, 744–745
assurance groups, 444–447
Atomic Counter tab (Visibility & Troubleshooting tool), 782–784
attachable access entity profile (AAEP), 56–57, 186–187
  access policies, 597–600
  best practices, 207
audit log entries

Events and Audits tab (Visibility & Troubleshooting tool), 779–780
filtering
  with iCurl, 725
  with MOQuery, 723–724
in syslog, 420–426
troubleshooting with, 720
viewing, 70
authentication of APICs in Ansible, 382–384
auto-completing commands in NX-OS–style CLI (command-line interface), 70
automation
  with Ansible, 372–392
    ACI support, 375–378
    APIC authentication, 382–384
    explained, 372–375
    installing, 378–381
    use cases, 384–392
benefits of, 344–345
configuration examples, 345–349
orchestration versus, 343–344
with REST API tools, 351
  API Inspector, 351–353
  MOQuery, 357–364
  Object (Save As) method, 353–355
  use cases, 364–372
  Visore, 355–358
with UCS Director, 392–401
  explained, 392–393
  tasks and workflows, 393–395
  use cases, 395–401
availability zones (AZs), 90
AVS (Cisco Application Virtual Switch)
  guidelines and limitations, 257–258
  prerequisites, 257
  verifying, 259–260
AZs (availability zones), 90
B
bash -c command, 75
bash command, 74
Bash shell, 74–78
  accessing, 74
  executing from NX-OS CLI, 75–76
  navigating file system, 76–77
  scripts in, 77
  viewing Linux routing table, 75
  viewing previous commands, 78
BD_EXT_VLANs, 628
BD_VLANs, 628
BDs (bridge domains), 24–25, 43–46
  creating, 384–388
  endpoint learning
    in Layer 2–only bridge domain, 627–635
    in Layer 3–enabled bridge domain, 635–640
  extending outside ACI fabric, 216–218
  logical topology, 605–606
  with subnets, 527–529
best practices
  for access policies, 206–207
  for multi-site design, 129–130
BGP (Border Gateway Protocol), 220–221
  configuring, 15
    for container-as-a-service (CaaS), 154–155
    route reflectors, configuring, 221–222

blade chassis servers
  connectivity, 208
    in OpenStack integration, 270

bond interfaces, viewing, 730
Border Gateway Protocol. See BGP (Border Gateway Protocol)

border leafs, 15, 25, 26, 51–52, 219
  access policies, 520–522
  network visibility, 428–430
bridge domains. See BDs (bridge domains)
buffer drops, 737
  ingress/egress packets, 774
    viewing, 742

C

CaaS (container-as-a-service), 149–165
Calico CNI for container-as-a-service (CaaS), 149–165
channels, 92
Cisco ISE (Identity Service Engine), configuring, 542–547
Cisco Secure ACS (Access Control System), configuring, 533–542
classnames in MOQuery, 722
CLIs (command-line interfaces)
  ACI fabric switch CLIs, 78–84
  iBash CLI, 78–81
  VSH shell, 81–82
  VSH_LC shell, 83–84
  APIC CLIs, 68–78
    Bash shell, 74–78
    NX-OS–style, 68–74
Clos topology, 612
cloud computing, 3
cluster minority state, 114
CNI (Container Network Interface), 149–165
  command syntax in iBash CLI, 79
  command-line interfaces. See CLIs (command-line interfaces)
Common tenant, 35
compute connectivity, 207–209
conge concrete objects, 32
configuration management, 7
configuration mode, entering, 72
configurations, restoring, 72–73
configuring
  AAA (authentication, authorization, accounting) policies, 533–551
    Cisco ISE configuration, 542–547
    Cisco Secure ACS configuration, 533–542
    steps in, 547–549
    verifying configuration, 550–551
access policies, best practices, 206–207
access ports, 202–203
BGP, 15
BGP route reflectors, 221–222
configuring Cisco ISE (Identity Service Engine), 542–547
Cisco Secure ACS (Access Control System), 533–542
external routed networks, 222–226
INB (in-band) management, 517–533
BDs (bridge domains) with subnets, 527–529
border leaf access policies, 520–522
external management EPG, 524–527
external routed networks, 522–526
leaf interface access policies, 518–520
management contracts, 517–518
node management EPG, 529
static management addresses, 530
verifying configuration, 515–517
IPNs (inter-pod networks), 234–237, 388–392
NAE (Network Assurance Engine), 440–450
NetFlow, 577–586
with access policies, 580–582
steps in, 577–579
with tenant policies, 582–585
verifying configuration, 585–586
OOB (out-of-band) management, 509–517
external management entity EPG, 513–515
management contracts, 510–513
node management EPG, 513–514
static management addresses, 510
verifying configuration, 515–517
PBR service graph, 325–326
port channels, 198
route peering, 345–347
service graph, 307–311
SNMP (Simple Network Management Protocol), 556–566
steps in, 556–562
verifying configuration, 562–566
SPAN (Switched Port Analyzer), 566–577
access SPAN, 567–570
fabric SPAN, 571–573
tenant SPAN, 572–574
verifying configuration, 576–577
in Visibility & Troubleshooting tool, 575–576
syslog, 551–556
steps in, 551–555
verifying configuration, 555–556
Tetration, 455–461
email alerts, 463
VMM (Virtual Machine Manager) domains, 601–602
VPC, 192–193, 347–349
vSwitch policies, 602
connectivity
applications
  external network connectivity, 502–504
  standalone compute application connectivity example, 433–435
  virtual compute application connectivity example, 435
compute, 207–209
end hosts, 185–207
  AAEP (attachable access entity profile), 186–187
  access policy best practices, 206–207
  access ports, 201–206
domains, 186
  interface policies, 188–191
  port channels, 197–201
switch policies, 187–188
troubleshooting, 751–759, 807–812
VLAN pools, 186
VPC (Virtual Port Channel), 191–197
external Layer 2, troubleshooting, 812–813
external Layer 3, troubleshooting, 814–821
L4/L7 service devices, 210–213
  firewalls, 211–212
  load balancers, 212–213
leaf nodes, troubleshooting, 821–826
networks, 213–241
  external bridged networks, 213–218
  external routed networks, 218–227
  for multi-pod/multi-site design, 228–241
storage, 209–210
troubleshooting, 242–245
Container Network Interface (CNI), 149–165
container-as-a-service (CaaS), 149–165
containers, installing
  for Kubernetes, 279
  for OpenShift, 290
contexts, 19
Contract Drops tab (Visibility & Troubleshooting tool), 777
contracts, 22–24, 46–50
  Contract Drops tab (Visibility & Troubleshooting tool), 777
  Contracts tab (Visibility & Troubleshooting tool), 777–779
creating, 384–388, 510–513, 517–518
directionality, 804–807
inheritance, 49
labels, filters, aliases, 48–49
logical topology, 605–606
policy enforcement with, 303–306
preferred groups, 49–50
QoS values, 671
troubleshooting, 759–765, 801–807
Contracts tab (Visibility & Troubleshooting tool), 777–779
control plane protocols, 15–17
  inter-site, 117
  remote leaf, 138–140
control plane TEP (CP-TEP), 675

COOP (Council of Oracle Protocol), 15, 95, 632–634

CoS (class of service)
  preservation, 672–674
  values, 670

counters, 737–743, 782–784

CP-TEP (control plane TEP), 675

CPU packet captures, 743–748

CPU utilization, monitoring
  on APICs, 475–477
  on leafs/spines, 482–485

CRC errors
  health scores with, 414–415
  viewing, 741, 742–743

current working path, viewing, 73

custom QoS policies, 671

Deployment Immediacy option
  (EPGs), 256

design. See ACI fabric design

df command, 76

DHCP Relay support in IPNs (inter-pod networks), 107–109
diagnosing. See troubleshooting

Digital Media Engine (DME), 175–180
discovery
  leaf discovery troubleshooting use case, 792–795
  in remote leaf design, 136–139
disk utilization, monitoring on APICs, 477–478

DME (Digital Media Engine), 175–180
dMZs (demilitarized zones), 211
domains
  access policies, 597–600
  best practices, 206
  defined, 186
  types of, 56, 186–187

VMM (Virtual Machine Manager), 56, 187, 250–252
  AAEP association, 252
  components, 250
  configuring, 601–602
  EPG association, 253–256
  policy model, 250
  publishing EPGs to with VMware, 258–259
  troubleshooting, 826–832
  VLAN pool association, 252

VPC, defining, 193–194
Dot1q support in IPNs (inter-pod networks), 109

downloading software agents (Tetration), 456

drops
  Contract Drops tab (Visibility & Troubleshooting tool), 777
  Drop/Stats tab (Visibility & Troubleshooting tool), 773–777
  in NIR, 788
  storm control, 774–775
  types of, 737–738

Drop/Stats tab (Visibility & Troubleshooting tool), 773–777

du command, 76

DWDM (dense wavelength-division multiplexing), 92

dynamic routing, 651–656

dynamic tunnels, 679

domain, 186

dmother policies, 188–191

end host connectivity, 185–207
  AAEP (attachable access entity profile), 186–187
  access policy best practices, 206–207
  access ports, 201–206

domains, 186

downloading software agents (Tetration), 456

EoMPLS (Ethernet over MPLS)
  pseudowire, 92–97

EIGRP (Enhanced Interior Gateway Routing Protocol), 220

ELAM (Embedded Logic Analyzer Module), 765–768

ELTMC (Ethernet Lif Table Manager Client), 604

email alerts, configuring in Tetrathon, 463

enabling syslog, 464

epend point learning, 626–645
  fabric glean process, 640–641
  in Layer 2-only bridge domain, 627–635
  in Layer 3-enabled bridge domain, 635–640
  remote learning, 641–645

Endpoint Manager Client (EPMC), 604

log files, 752–755

Endpoint Manager (EPM), 603

endpoint mobility, 645–647

Endpoint Tracker, 752–755

Enhanced Endpoint Tracker (EPT) app, 756–757

Enhanced Interior Gateway Routing Protocol (EIGRP), 220

enterprise resource planning (ERP), vendor-based, 165–175

Enhanced MPLS (Ethernet over MPLS)
  pseudowire, 92–97
EPGs (endpoint groups)
application profiles, 20–21
connecting VMs to port groups, 259
contracts, 22–24, 46–50
creating, 384–388
explained, 21–23
extending outside ACI fabric, 213–216
external management entity EPGs, creating, 513–515
external management EPGs, creating, 524–527
L3Out flags, 668–669
logical topology, 605–606
MOs (managed objects), 41–43
node management EPGs
  choosing, 513–514
  configuring, 529
out-of-band, 799–801
policy enforcement, 661–663
publishing to VMM domain with VMware, 258–259
QoS values, 671
shadow EPGs, 306–307
shared services, 664–668, 695–698
VMM association, 253–256
vzAny managed objects, 50–51
EPM (Endpoint Manager), 603
EPMC (Endpoint Manager Client), 604
  log files, 752–755
EPT (Enhanced Endpoint Tracker) app, 756–757
ERP (enterprise resource planning), vendor-based, 165–175
error drops, 737, 774
E-TEP (external TEP), 675
Ethernet Lif Table Manager, 603
Ethernet Lif Table Manager Client (ELTMC), 604
Ethernet networks, limitations of, 611–612
Ethernet over MPLS (EoMPLS) pseudowire, 92–97
events
  Events and Audits tab (Visibility & Troubleshooting tool), 779–780
  in syslog, 420–426
  troubleshooting with, 720–722
Events and Audits tab (Visibility & Troubleshooting tool), 779–780
examples
  0.0.0.0/0 learned dynamically from peer via OSPF, 656
  abbreviations and command syntax in iBash CLI, 79
  accessing APIC NX-OS CLI by using SSH, 68
  accessing Bash shell, 74
  accessing iBash CLI, 78
  accessing VSH shell, 82
  accessing VSH LC shell, 83
  actrlRule contract configuration on ACI leaf, 762
  actrlRule contract object on APIC, 760
  actrlRule stats information for PCTags 49154 and 16387, 762
  adding more specific prefix to external EPG for development, 821
aliases in iBash CLI, 81
all PTEP address in fabric, 643
Ansible playbook to configure signature-based authentication in ACI, 382–383
Ansible playbook using signature-based authentication in ACI, 383–384
API messages for leaf node registration and OOB address, 364
ARP request/reply validation through iStack, 747
auto-completing commands, 70
BGP RT and route map/prefix to leak routes, 668
border leaf 201 back to active state, 825
Calico manifest files for BGP peering with ACI leafs, 157
checking Ansible and Python software version, 379
checking audit log entries between two times, 724, 725
checking audit log entries for specific date and time, 723
checking BGP process in VRF instance to determine RD and export/import list, 657
checking contract programming on leaf 101 between EPG web and EPG app, 662
checking COOP state for endpoint by using MAC address, 633–634
checking COOP state on spine for IP endpoint, 639
checking details of tunnel39 interface, 856
checking external network connectivity from ACI leafs via route peering using OSPF, 61–62
checking interface unicast packets received by using REST API, 498
checking interface unicast packets transmitted by using REST API, 499
checking multicast status on all routers in network topology, 857–859
checking PIM status on BD dme-bd, 856
checking routing table of APIC shard leader, 828–829
checking Secure SSH connection to Ansible managed nodes, 379
checking shard leader IP connectivity to vCenter using ping, 828
checking spine 201 for GIPo route with external interfaces, 838
checking spine BGP neighborships in overlay-1 VRF instance, 657
checking the details of tunnel9 interface, 854
class ID assigned to route with zoning rules, 665–666
configuration example for traditional NX-OS IP access list to count packets, 782
configuring Phantom RP, 107
configuring PIM and Auto-RP settings on data center core router (NX-OS), 179
configuring PIM and RP settings on WAN core router (IOS XR), 179–180
confirming nonresponsive subnet, 450
connecting to APIC with SSH, 515, 530
contract and filter verification for PCTag 15 on leaf 101, 820
contract and filter verification for PCTag 16389 on leaf 101, 821
contract deployment on leaf 101 for traffic destined to JumpBox EPG, 805
contract deployment on leaf 101 for traffic sourced from JumpBox EPG, 805
CoPP filter verification on leaf CLI, 747–748
copying SSH key to Ansible managed node, 380–381
current working path, 73
data plane policer drops, 776
displaying VTEP address and PTEP address of nodes advertising them, 643–644
DME IP address is learned on leaf 1001, 848
DPP policy, 776
endpoint aging is enabled per IP, 639
endpoint and tunnel verification on border leaf 201, 825–826
endpoint MO on APIC, 634–635
endpoint with PCTag for policy enforcement, 663
ensuring APIC 1 can ping all APICs in cluster, 734
entering configuration mode, 72
executing Ansible playbook, 381
executing Bash commands from NX-OS CLI, 75–76
executing commands and redirecting to file, 82, 83
fabric command, 71–72
fabric node out-of-service output through the REST API, 493
FCS errors ingressing on leaf, 774
finding BD class and filtering on name with ARP flooding enabled using MOQuery, 362–364
finding classes with MOQuery, 359–360
finding DN with MOQuery, 360
finding EPG class and filtering on name with MOQuery, 360–362
FTag 11 for leaf 101, 625
FTag programming on spine 201, 620–621
FTags on leaf switch, 621–622
generating SSH key on Ansible control node, 380
GIPo mroute for BD in IS-IS, 624
host A MAC address not learned on leaf 103, 812
identifying IP address moves with EPMC log files, 755
identifying MAC moves between non-VPC pair using EPMC log files, 754
identifying MAC moves with VPC pair with EPMC log files, 753
initiating SSH sessions, 73
interface information for eth-1/5 viewed from iBash, 740–741
inventory INI file, 375
inventory YAML file, 375
IP prefix verification within ACLQOS for given VRF instance on leaf 101, 818
IP routing table for overlay-1 VRF instance and IS-IS neighborships on Leaf 101, 735–736
IP routing table for overlay-1 VRF instance and IS-IS neighborships on Spine 201, 736
iPing failing to the destination, 824
IPN1 variable file (ipn1-var.yml), 391
IPN2 variable file (ipn2-var.yml), 391
IPv4 next hop reachable through spines, 618
IPv4 routing table built through IS-IS, 617
IS-IS neighbors for leaf 101, 617
JSON format in ACI, 353
JSON script to configure leaf node registration and OOB address, 367–368
JSON script to configure leaf node registration and OOB address using variables, 368–369
L3Out VLAN deployed on leaf 101, 653
Layer 2 endpoint verification on leaves 101 and 102, 808–809
Layer 3 interface for L3Out, 653
leaked route from border leaves 101 and 102, 667
listing command options, 79
listing show commands, 69–70
monitoring aggregate amount of traffic flow to specific application tier, 500–502
monitoring CPU utilization on leaves and spines, 483–485
monitoring external network connectivity by using REST API, 503–504
monitoring fan status, 488–489
monitoring leaf and spine fabric membership by using REST API, 492–493
monitoring memory utilization on leaves and spines, 485–486
monitoring PBR status by using REST API, 504–505
monitoring power supply unit status, 487–488
monitoring status of leaf and spine interfaces, 497–498
monitoring supervisor module, line card, and fabric module status, 489–491
MOQuery command-line help, 358–359
mroute validation on IPN 1, 838
multicast receiver IP reachability to multicast source and RP, 849–850
multicast routing table on leaf 1001, 856
multicast routing table on RP, 851–852
multicast RP and PIM status along with multicast routing table on ACI border leaf 202, 853–854
multicast RP and PIM status along with multicast routing table on data center core router, 852–853
multicast RP and PIM status along with multicast routing table on leaf 1001, 855
multicast RP and PIM status on RP, 850–851
multiple IP addresses learned against a single MAC address, 638
naming schemes, 772
navigating file system, 76–77
no BGP neighbors exist on leaf 302, 840
OSPF neighbor verification on leaf 201, 823
PI VLAN configured for EPG VLAN2, 629
ping text to TEP address of leaf 103 fails, 735
ping to APIC 2 from APIC 1 fails, 796
platform counters for eth-1/5 on Leaf 101, 738–739
playbook YAML file, 374
pulling fabric node status by using REST API, 494–496
remote endpoint learned for host B on leaf 101, 815
remote learned IP endpoint, 642
REST API call to monitor APIC cluster state, 481–482
REST query to check faults, 409–411
restoring configurations, 72–73
retrieving information about APIC CPU and memory usage, 475–477
retrieving information about APIC disk utilization, 477–478
retrieving information about APIC interface status, 479–481
route map to redistribute external routes into OSPF, 655
route on compute leaf is now readable via border leafs in pod 1, 841
route peering configuration using OSPF in NX-OS, 346
routes marked as private to VRF assigned the default VRF instance tag, 655
routing table for route to host B on leaf 101, 815
routing table of leaf 103 with pervasive route pointing to spine proxy, 803
RP address reachability from leaf 1001 sourcing from DME IP subnet gateway, 849
sample Ansible playbook to create tenant in ACI, 376
sample Ansible playbook using aci_rest module to assign OOB address in ACI, 377–378
sample configuration of data center core routers filtering CaaS subnets via eBGP, 162
sample configuration of data center core routers peering with ACI border leaf using OSPF, 161
scripts in Bash shell, 77
setting page size and number to collect objects, 726
show cli list command, 82, 84
show faults subcommands, 409
show switch command, 74
SPAN session on leaf 102 showing different IP address than leaf 101, 750–751
SPAN session verification on leaf 101, 749–750
storm control drop stats per class, 775
storm control drops on leaf interface, 775
system ID of leaf 101, 616–617
tcpdump on kpm_inb showing no LACP packets being received, 835
TEP address of leaf 101, 616
tunnel interface, 643
use case 1 Ansible inventory file, 388
use case 1 Ansible playbook, 384–387
use case 2 Ansible inventory file, 391
use case 2 Ansible playbook, 388–390
use case 2 IPN router baseline configuration file (NX-OS), 391–392
verifying AAA configuration, 551
verifying active bond interface on APIC 1, 792
verifying INB configuration, 531–532
verifying interface status through SNMP IFMIB value, 565
verifying JSON script to configure leaf node registration and OOB address, 370–372
verifying LLDP and Infra VLAN deployment on leaf 101, 796–797
verifying LLDP frames from connected leaf on APIC 1, 793–794
verifying MAC address is learned on eth1/6 in PI VLAN, 629–630
verifying NetFlow configuration, 585–586
verifying OOB as preferred management method for APIC, 533
verifying OOB configuration, 515–517
verifying route maps for redistribution on VRF instance for OSPF, 654
verifying Secure SSH connection to Ansible managed node, 381
verifying SNMP configuration on APIC, 562
verifying SNMP configuration on leaf/spine, 563–564
verifying SNMP read query functionality through tcpdump utility on a leaf, 566
verifying SNMP trap functionality through tcpdump utility on a leaf, 564–565
verifying SPAN configuration, 576–577
verifying syslog configuration on APIC, 555
verifying syslog functionality through tcpdump utility on a leaf, 556
verifying TACACS+ server reachability through nginx logs, 551
viewing APIC bond interfaces from CLI, 730
viewing audit log entries, 70
viewing BGP route state on leaf 103 for VRF instance, 658–659
viewing buffer drops via platform counters, 742
viewing Cisco_APIC_SCVMM_Service log file, 262–263
viewing CRC errors globally with MOQuery, 742–743
viewing CRC errors via iBash, 741
viewing CRC errors via platform counters, 741
viewing ELTMC hardware programming and mapping from EPG to BD, 628
viewing endpoint information triggered by VPC sync from leaf 101 to leaf 102, 631
viewing fabric node status with acidiag fnvread, 734–735
viewing interface rate via iBash, 742
viewing IP endpoint from EPMC software on line card of a leaf, 637
viewing Linux routing table, 75
viewing LLDP frames sent from APIC, 730–731
viewing LLDP objects on leaf for all interfaces with wiring issues, 731–732
viewing MAC endpoint from EPMC software on line card of a leaf, 630
viewing multi-site unicast and date plane TEP address on spine, 680
viewing multi-site VRF instance and sclass translations on spine, 682
viewing packets on taho00 by using tcpdump2, 745
viewing packets on taho00 using knet_parser, 746
viewing packets that met a policy permit rule, 765
viewing packets that matched a policy deny rule, 764
viewing PI VLAN and BD mapped to EPG VLAN 2, 629
viewing previous commands, 78
viewing redistribution route map from BGP to OSPF and corresponding prefix list, 660
viewing routing table on leaf 103 for VRF instance, 658
viewing total policy TCAM entries on cloud-scale platform, 763
viewing VLAN name (EPG) mapping to encap VLAN, 628
viewing VMM domain configuration and checking shard leader, 827–828
VLAN and anycast gateway deployment on leaf 1006, 823
VLAN and endpoint verification for EPG WebServers on leaf 103, 811
VLAN deployment on leafs 101 and 102, 808
VLAN verification on leaf 101, 833
VNID rewrite information for leaked routes, 666
VPC configuration in NX-OS, 347
watch command, 80
wiring issue raised on eth1/3 of leaf 101, 797
zoning rule between globally unique PCTag and locally significant PCTag, 665

excluded EPGs, 50

executing commands
Bash command from NX-OS CLI, 75–76
in NX-OS–style CLI (command-line interface), 71–72
and redirecting to file, 82, 83

extending
BDs (bridge domains) outside ACI fabric, 216–218
EPGs (endpoint groups) outside ACI fabric, 213–216
OpFlex to compute node, 264

external bridged domains, 56, 187
extending outside ACI fabric, 216–218
external bridged networks, 25–26
  network connectivity to, 213–218
external IPv4 proxy, 675
external IPv6 proxy, 675
external Layer 2 connectivity, troubleshooting, 812–813
external Layer 3 connectivity, troubleshooting, 814–821
external Layer 3 routing protocols, 220–221
external MAC proxy, 675
external management entity EPGs, creating, 513–515
external management EPGs, creating, 524–527
external monitoring tools, 430–473
  Network Assurance Engine, 437–453
    building blocks, 437–438
    configuring, 440–450
    installing, 439–440
    subnet reachability example, 450–453
    use cases, 438–439
  Network Insights suite, 430–435
    standalone compute application connectivity example, 433–435
    tools in, 431–433
    virtual compute application connectivity example, 435
Tetration, 453–473
  applications, 465–467
  cluster reboots, 469
  cluster shutdowns, 469–470
  code upgrades, 467–468
  email alerts, 463
  enabling syslog, 464
  hardware agents, 455
  installing and configuring, 455–461
  patch upgrades, 467–469
  scopes, 465
  software agents, 455
  TAN (Tetration Alerts Notification) agent, 461–463
  workload security agent, 470–473
external network connectivity, monitoring, 502–504
external orchestrators, 466–467
external routed domains, 56, 187
external routed networks, 25–26
  configuring, 222–226
  for INB (in-band) management, 522–526
  learning routes, 656–659
  network connectivity to, 218–227
  policy enforcement, 695
external TEP (E-TEP), 675

F

fabric command, 71–72
Fabric Extender (FEX) connectivity, 207–208
fabric glean process, 640–641
fabric nodes
  troubleshooting, 734–737
  viewing status, 734–735
fabric policies, 36. See also access policies
fabric SPAN, configuring, 571–573
fan status, monitoring on leafs/spines, 488–489
faults
Faults tab (Visibility & Troubleshooting tool), 772–773
monitoring with, 407–411
in syslog, 420–426
troubleshooting with, 718–719
Faults tab (Visibility & Troubleshooting tool), 772–773
FD_VLANs, 628
FEX (Fabric Extender) connectivity, 207–208
Fibre Channel domains, 56, 187
file system, navigating, 76–77
filtering audit log entries
with iCurl, 725
with MOQuery, 723–724
filters, 48–49, 384–388, 510–513
firewalls
connectivity, 211–212
as default gateways, 312–313
in multi-pod design, 332–333
not default gateways, 312–314
PBR service graph with, 324–325
route peering with, 314–315
service graph with, 316
troubleshooting, 801–804
forward drops, 737, 775–776, 788
forwarding. See packet forwarding FTags, 618–625
G
GET BULK command, 417
GET command, 417
GET method, 349, 473
GET NEXT command, 417
GIPo (Group IP outer), mapping to FTags, 623
glean process, 640–641
global policies, 55–57
GOLF connections, 227
H
HAL (Hardware Abstraction Layer), 604
hardware agents (Tetration), 455
installing, 456–459
hardware specifications, 8–14, 603–605
APIC (ACI controllers), 12–14
for NAE (Network Assurance Engine), 439
Nexus 9000 Series, 9–12
for remote leaf design, 134
health scores, monitoring with, 411–415
history command, 78
host files, 374–375
hybrid clouds, 165
hybrid mode (service graph), 302
I
iBash CLI (command-line interface), 78–81
abbreviations and command syntax, 79
accessing, 78
aliases, 81
listing command options, 79
watch command, 80
iCurl, 724–726
idempotent, 349, 473–474
imperative model, 7
importing policies, 127–128
INB (in-band) management, configuring, 517–533
BDs (bridge domains) with subnets, 527–529
border leaf access policies, 520–522
external management EPG, 524–527
external routed networks, 522–526
leaf interface access policies, 518–520
management contracts, 517–518
node management EPG, 529
static management addresses, 530
verifying configuration, 530–533
included EPGs, 49
INFORM command, 417
Infra tenant, 36
ingress error drop packets, 774
ingress forward drop packets, 775–776
ingress load balancer drop packets, 776–777
ingress/egress buffer drop packets, 774
inheritance of contracts, 49
initiating SSH sessions, 73
inner headers, defined, 615
inside networks, 211
installing
Ansible, 378–381
containers
for Kubernetes, 279
for OpenShift, 290
NAE (Network Assurance Engine), 439–440
Tetration, 455–461
hardware agents, 456–459
software agents, 459–461
TAN agent, 461–463
interface errors. See CRC errors
interface policies, 54–55, 188–191
for access ports, 204–205
for leaf access, 518–520
policy groups, 596–597
for port channels, 199–200
for VPC, 195–196
interface policy groups, 188
interface profiles, 189–191
interface rate, viewing, 742
interface selectors, 189
interface status, monitoring on leafs/spines, 496–499
interfaces, monitoring on APICs, 478–481
internal monitoring tools, 415–430
NetFlow, 426–430
SNMP, 415–420
commands, 417
interface failures example, 418–420
MIB and TRAPs support in ACI, 417–418
syslog, 420–426
critical messages, 422–423
IPN failure example, 423–426
leaf membership failure example, 423–424
message structure, 420–421
severity levels, 421–422
inter-site control plane, 117
inter-site data plane, 118
inter-VRF traffic in remote leaf design, 142
intrusion prevention system (IPS), service graph with, 319
inventory files, 374–375
IPNs (inter-pod networks), 98, 104–113
    configuring, 234–237, 388–392
    connectivity in, 228–234
    DHCP Relay support, 107–109
    Dot1q support, 109
    failure example, 423–426
    MTU support, 109
    multicast support, 104–107
    packet forwarding, 674–679
    QoS support, 111–113, 672–674
    remote leaf connectivity, 237–241
IPS (intrusion prevention system), service graph with, 319
IS-IS protocol, 17
    TEP addressing and, 615–618
iStack, 745, 747
iTraceroute, 672, 780–782
iVXLAN. See VXLAN

J

JSON format, 353
JSON script errors, troubleshooting, 844–846

K

knet_parser, 746
Kubernetes integration, 272–280
    installing containers, 279
    planning, 272–273
preparing nodes, 277–279
prerequisites, 273–274
provisioning ACI for, 274–277
verifying, 280

L

L3Out
    border leaf access policies, 520–522
    dynamic routing, 651–656
    flags, 668–669
    INB management external routed networks, 522–526
    logical topology, 606–608
    policy enforcement, 693
    troubleshooting, 839–841

L4/L7 services
    deployment, troubleshooting, 832–836
    device connectivity, 210–213
        firewalls, 211–212
        load balancers, 212–213
    in multi-pod design, 332–338
    in multi-site design, 338–340
    in OpenStack integration, 270
PBR (policy-based redirect), 322–331
    configuring, 325–326
    design considerations, 323
    design scenarios, 324–325
    service node health check, 326–328
    troubleshooting, 328–331
service graph, 300–319
    configuring, 307–311
    contracts, 303–306
    design and deployment options, 312–319
integration use cases, 302–303
modes, 301–302
shadow EPGs, 306–307
troubleshooting, 834–836
service insertion, 299–300
labels, 48–49
lambdas, 92
Latency tab (Visibility & Troubleshooting tool), 785
Layer 2 known unicast, 688
Layer 2 unknown unicast proxy, 690–693
Layer 2–only bridge domain, endpoint learning in, 627–635
Layer 3 proxy flow
in multi-pod design, 700–703
in multi-site design, 705
in remote leaf design, 710–713
Layer 3–enabled bridge domain, endpoint learning in, 635–640
leaf command, 68
leaf nodes
connectivity, troubleshooting, 821–826
discovery troubleshooting use case, 792–795
health scores, 413
interface access policies, 518–520
membership failure example, 423–424
monitoring, 482–499
  CPU utilization, 482–485
  fan status, 488–489
  interface status, 496–499
  membership status, 491–496
  memory utilization, 485–486
module status, 489–491
PSU (power supply unit) status, 486–488
registering, 364–372
leaf switches, purpose of, 14
Linux, installing Tetrion software agents, 459–460
Linux routing table, viewing, 75
listing
  available commands, 82, 84
  command options, 79
  show commands, 69–70
LLDP frames, viewing, 730–731
load balancer drops, 738, 776–777
load balancers
  connectivity, 212–213
  for container-as-a-service (CaaS), 151–154
  in multi-pod design, 332
local admin fallback user, logging in as, 550
log files (EPMC), 752–755
logical constructs, 37–38
  application profile objects, 40
  bridge domain objects, 43–46
  contract objects, 46–50
  EPG objects, 41–43
  outside network policies, 51–52
  subnet objects, 43–44
  tenant objects, 38–39
  VRF objects, 39–40
  vzAny managed objects, 50–51
logical design, 149–180
  container-as-a-service (CaaS), 149–165
vBrick Digital Media Engine (DME), 175–180
vendor-based ERP (enterprise resource planning), 165–175
logical interface profiles for container-as-a-service (CaaS), 155
logical objects, 32
logical topology
for application-centric design, 606
for L3Out, 606–608
for network-centric design, 605
with OpenStack, 265–266
loopback addresses in multi-pod design, 675

mapping
multiple SCVMMs to APIC, 262
OpenStack constructs, 266–267
SCVMM constructs, 261–262
maximum transmission unit (MTU) support in IPNs (inter-pod networks), 109
MDT (multicast distribution tree), 618–625
membership status, monitoring on leafs/spines, 491–496
memory utilization, monitoring on APICs, 475–477
on leafs/spines, 485–486
metadata services in OpenStack integration, 270, 271
Mgmt tenant, 36
Microsoft SCVMM integration, 260–263
mapping constructs, 261–262
mapping multiple SCVMMs to APIC, 262
topology, 260
verifying, 263
verifying OpFlex certificate deployment, 262–263
workflow, 261
migration scenarios for multi-site design, 124–128
minority state, 95
MIT (management information tree), 33–37
model-driven architecture, 31
modes
in NX-OS–style CLI (command-line interface), 69
for service graph, 301–302
module status, monitoring on leaves/spines, 489–491

monitoring

benefits of, 405–406
with external monitoring tools, 430–473
  Network Assurance Engine, 437–453
  Network Insights suite, 430–435
  Tetration, 453–473
with faults, 407–411
with health scores, 411–415
with internal monitoring tools, 415–430
  NetFlow, 426–430
  SNMP, 415–420
  syslog, 420–426
with REST API, 473–505
  APIC components, 475–482
  applications, 499–505
  health scores, 413
  leaves and spines, 482–499

MOs (managed objects)
application profiles, 40
BDs (bridge domains), 43–46
contracts, 46–50
EPGs (endpoint groups), 41–43
in MIT (management information tree), 33–37
outside network policies, 51–52
relationships, 57–58
subnets, 43–44
tags, 58
tenants, 38–39
updating tree, 634–635
VRF objects, 39–40
vzAny, 50–51
MSO (Multi-Site Orchestrator), 117, 120–122
  creating policies, 124–127
  importing policies, 127–128
MTU (maximum transmission unit) support in IPNs (inter-pod networks), 109
multicast distribution tree (MDT), 618–625
multicast issues, troubleshooting, 846–860
multicast support in IPNs (inter-pod networks), 104–107
multiple locations, VMM integration at, 292–297
  multi-site design, 292–295
  remote leaf design, 295–297
multiple service nodes, service graph with, 317–319
multiple-fabric design, single-fabric design versus, 87–97
multi-pod design, 97–116
  APIC cluster deployment, 113–116
  inter-pod connectivity, 104–113
  L4/L7 services in, 332–338
  network connectivity, 228–241
  packet forwarding, 674–679, 698–703
  physical topology for, 591
QoS support, 672–674
remote leaf connectivity, 239–241
remote leaf integration, 143–148
scalability of, 103
troubleshooting, 837–841
use cases, 100–103
**multi-site design, 116–130**
best practices, 129–130
L4/L7 services in, 338–340
migration scenarios, 124–128
MSO (Multi-Site Orchestrator), 120–122
network connectivity, 228–241
packet forwarding, 680–682, 703–705
physical topology for, 591–592
troubleshooting, 841–843
use cases, 122–124
VMM integration, 292–295
Multi-Site Orchestrator (MSO), 117, 120–122
creating policies, 124–127
importing policies, 127–128

**NetFlow, 426–430**
configuring, 577–586
*with access policies, 580–582*
*steps in, 577–579*
*with tenant policies, 582–585*
*verifying configuration, 585–586*

**network connectivity, 213–241**
external bridged networks, 213–218
external routed networks, 218–227
for multi-pod/multi-site design, 228–241
troubleshooting, 242–245

**Network Insights Advisor (NIA), 432–433**

**Network Insights for Resources (NIR), 431, 787–790**

**Network Insights suite, 430–435**
standalone compute application
connectivity example, 433–435
tools in, 431–433
virtual compute application
connectivity example, 435

**network-centric design, 6**
logical topology for, 605
Nexus 9000 Series, 9–12
Nexus 9300 Series, 11–12
Nexus 9500 Series, 9–10
NIA (Network Insights Advisor), 432–433
NIR (Network Insights for Resources), 431, 787–790
no keyword, 72–73
node management EPGs
choosing, 513–514
configuring, 529

**NAE (Network Assurance Engine), 437–453**
building blocks, 437–438
configuring, 440–450
installing, 439–440
subnet reachability example, 450–453
use cases, 438–439
naming schemes, 772

**NAT (Network Address Translation) in OpenStack integration, 269, 271**
navigating file system, 76–77
nullipotent, 349, 473–474

NX-OS–style CLI (command-line interface), 68–74
  accessing, 68
  auto-completing commands, 70
  current working path, 73
  entering configuration mode, 72
  executing Bash commands, 75–76
  executing commands, 71–72
  initiating SSH sessions, 73
  listing show commands, 69–70
  modes, 69
  restoring configurations, 72–73
  retrieving fabric node information, 74
  viewing audit log entries, 70

Object (Save As) method, 353–355
Object Store Browser, 355–358
objects, 31. See also MOs (managed objects)
  in MIT (management information tree), 33–37
  types of, 32
offline analysis with NAE (Network Assurance Engine), 447–450
One-Arm mode (service graph), 312
OOB (out-of-band) EPGs, troubleshooting, 799–801
OOB (out-of-band) management addresses, assigning, 364–372, 377–378
configuring, 509–517
  external management entity EPG, 513–515
management contracts, 510–513
node management EPG, 513–514
static management addresses, 510
verifying configuration, 515–517
Open Shortest Path First (OSPF), 220
OpenShift for container-as-a-service (CaaS), 149–165
OpenShift integration, 281–292
  installing containers, 290
  planning, 282–283
  preparing nodes, 287–290
  prerequisites, 283
  provisioning ACI for, 284–287
  updating router, 291
  verifying, 291–292
OpenStack integration, 263–271
  configuration examples, 271
  extending OpFlex to compute node, 264
  guidelines and limitations, 268–270
  logical topology, 265–266
  mapping constructs, 266–267
  physical architecture, 264
  prerequisites, 267–268
  software architecture, 265
  verifying, 270–271
OpFlex, 15
  extending to compute node, 264
  verifying certificate deployment, 262–263
optimized DHCP in OpenStack integration, 270, 271
orchestration
- automation versus, 343–344
- benefits of, 344–345
- configuration examples, 345–349
- with UCS Director, 392–401
  - explained, 392–393
  - tasks and workflows, 393–395
  - use cases, 395–401
- OSPF (Open Shortest Path First), 220
- outer headers, defined, 615
- out-of-band (OOB) EPGs, troubleshooting, 799–801
  - configuring, 509–517
    - external management entity EPG, 513–515
    - management contracts, 510–513
    - node management EPG, 513–514
    - static management addresses, 510
    - verifying configuration, 515–517
- outside network policies, 25–26, 51–52
- outside networks, 211
- overlay, defined, 615
- overlay networks, benefits of, 611–612

P

packet forwarding
- anycast gateways, 647–649
- ARP flooding, 686
- ARP optimization, 690
- endpoint learning, 626–645
  - fabric glean process, 640–641
  - in Layer 2–only bridge domain, 627–635
  - in Layer 3–enabled bridge domain, 635–640
  - remote learning, 641–645
- endpoint mobility, 645–647
- Layer 2 known unicast, 688
- Layer 2 unknown unicast proxy, 690–693
- Layer 3 policy enforcement
  - for external traffic, 695
to L3Out, 693
- in multi-pod design, 674–679, 698–703
- in multi-site design, 680–682, 703–705
- QoS support, 669–674
  - CoS preservation, 672–674
  - CoS values, 670
  - externally set markings, 671
- in remote leaf design, 141, 684, 707–713
- routing, 651–661
  - dynamic, 651–656
  - learning external routes, 656–659
  - transit, 659–661
- VPC (Virtual Port Channel), 649–651
- VXLAN, 17–18, 613–625
  - benefits of, 17
  - FTags and MDT, 618–625
  - IS-IS and TEP addressing, 615–618
operational overview, 613–615
policy enforcement, 661–663
purpose of, 14
shared services, 664–668, 695–698

packets, viewing
  with knet_parser, 746
  with tcpdump2, 745
paging, 726
PBR (policy-based redirect), 312, 322–331
  configuring, 325–326
  design considerations, 323
  design scenarios, 324–325
  service node health check, 326–328
  troubleshooting, 328–331
PBR (policy-based routing)
  monitoring service graph, 504–505
  for vendor-based ERP, 170–175
PCTags
  policy enforcement, 661–663
  shared services, 664–668
Phantom RP, 105–107
physical constructs, 52
  access policies, 52
  default policies, 58–60
  global policies, 55–57
  interface policies, 54–55
  managed object relationships, 57–58
  policy resolution, 57–58
  switch policies, 53
  tags, 58
physical design, 85–148
  multi-pod, 97–116
  multi-site, 116–130

with OpenStack, 264
remote leaf, 131–142
single-fabric versus multiple-fabric, 87–97
physical domains, 56, 187
physical topologies
  multi-pod design, 591
  multi-site design, 591–592
  remote leaf design, 592–593
  single-pod design, 589–590
physical tunnel endpoint (PTEP), defined, 615
physical-related issues,
  troubleshooting, 737–751
  with counters, 737–743
  with CPU packet captures, 743–748
  with SPAN sessions, 748–751
PIM Bidir (Platform Independent Multicast Bidirectional), 105
PIM Sparse Mode ASM (Any-Source Multicast), 846–860
planning
  Kubernetes integration, 272–273
  OpenShift integration, 282–283
platform-related issues,
  troubleshooting, 737–751
  with counters, 737–743
  with CPU packet captures, 743–748
  with SPAN sessions, 748–751
playbooks, 372–374
policies
  access policies, 52, 595–601
  best practices, 206
  creating, 124–127
  Deployment Immediacy option (EPGs), 256
enforcing
  with contracts, 303–306
  for external traffic, 695
to L3Out, 693
  with PCTags, 661–663
importing, 127–128
interface policies, 54, 188–191, 596–597
Resolution Immediacy option (EPGs), 255–256
switch policies, 53, 187–188, 595–596
vSwitch policies, 602
policy deny drops, verifying, 764–765
policy drops, 788
policy groups
  for access policies, 52
  for interface policies, 54, 596–597
  for switch policies, 53
Policy Manager, 600
policy model. See ACI policy model
policy resolution, 57–58
policy-based redirect (PBR), 312, 322–331
  configuring, 325–326
  design considerations, 323
  design scenarios, 324–325
  service node health check, 326–328
  troubleshooting, 328–331
policy-based routing (PBR)
  monitoring service graph, 504–505
  for vendor-based ERP, 170–175
port channels, 197–201
  configuring, 198
  interface policies, 199–200
policy groups, 596
  switch profiles, 200–201
port local SPAN sessions, 749
POST method, 349, 473
Postman Collection Runner tool, 369–370
power supply unit (PSU) status, monitoring on leafs/spines, 486–488
private subnets, 25, 44
proactive monitoring with health scores, 413–414
problem-solving. See troubleshooting
profiles
  for access policies, 52
  for interface policies, 54, 595–596
  for switch policies, 53, 595–596
promise theory, 7
provisioning ACI
  for Kubernetes integration, 274–277
  for OpenShift integration, 284–287
PSU (power supply unit) status, monitoring on leafs/spines, 486–488
PTEP (physical tunnel endpoint), defined, 615
public subnets, 25, 44
publishing EPGs to VMM domain with VMware, 258–259
PUT method, 349, 474
Q

QoS (Quality of Service) support, 669–674
  CoS preservation, 672–674
  CoS values, 670
externally set markings, 671
in IPNs (inter-pod networks), 111–113
for remote leaf design, 134–136
QoS drops, 788

Representational State Transfer. See REST API
Resolution Immediacy option (EPGs), 255–256
resolved objects, 32
RESPONSE command, 417
REST API
automation tools, 351
API Inspector, 351–353
MOQuery, 357–364
Object (Save As) method, 353–355
use cases, 364–372
Visore, 355–358
explained, 349–350, 473–475
monitoring with, 473–505
APIC components, 475–482
applications, 499–505
health scores, 413
leafs and spines, 482–499
restoring configurations, 72–73
RL-DP-TEP (remote leaf data-plane TEP), 684
RL-Mcast-TEP (remote leaf multicast TEP), 684
RL-Ucast-TEP (remote leaf unicast TEP), 684
RL-VPC-TEP (remote leaf VPC TEP), 684
RNs (relative names), 35
Rogue Endpoint Detection, 758–759
route command, 75
route controls for container-as-a-service (CaaS), 157–160
route peering
  configuring, 345–347
  with firewalls, 314–315
route reflectors, configuring, 221–222
Routed mode (service graph), 312
routing, 651–661
  dynamic, 651–656
  learning external routes, 656–659
  transit, 659–661
Scalability
  of multi-pod design, 103
  of OpenStack integration, 268–269
scopes in Tetration, 465
scripts in Bash shell, 77
SCVMM integration, 260–263
  mapping constructs, 261–262
  mapping multiple SCVMMs to APIC, 262
  topology, 260
  verifying, 263
  verifying OpFlex certificate
deployment, 262–263
  workflow, 261
security, whitelist policy model, 6–7
security policies. See contracts
server staging for container-as-a-service (CaaS), 163–164
server virtualization, 2–3
service chaining, 299–300
service graph, 300–319
  configuring, 307–311
  contracts, 303–306
  design and deployment options, 312–319
  integration use cases, 302–303
  modes, 301–302
PBR (policy-based redirect), 322–331
  configuring, 325–326
  design considerations, 323
  design scenarios, 324–325
  service node health check, 326–328
  troubleshooting, 328–331
shadow EPGs, 306–307
  troubleshooting, 834–836
service insertion, 299–300
SET command, 417
shadow EPGs, 306–307
shards, 12–13, 35, 113
shared services, 664–668, 695–698
shared subnets, 25, 44
show audits command, 70
show cli list command, 82, 84
show commands, listing, 69–70
show switch command, 74
signature-based authentication, 382
single-fabric design
  multiple-fabric design versus, 87–97
  remote leaf connectivity, 238–240
single-pod design, physical topology
  for, 589–590
SNMP (Simple Network Management Protocol), 415–420
  commands, 417
configuring, 556–566
   steps in, 556–562
   verifying configuration, 562–566
interface failures example, 418–420
MIB and TRAPs support in ACI, 417–418
software agents (Tetration), 455
downloading, 456
installing
   on Linux, 459–460
   on Windows, 460–461
software architecture with OpenStack, 265
software specifications, 603–605
SPAN (Switched Port Analyzer)
configuring, 566–577
   access SPAN, 567–570
   fabric SPAN, 571–573
   tenant SPAN, 572–574
   verifying configuration, 576–577
   in Visibility & Troubleshooting tool, 575–576
troubleshooting with, 748–751
SPAN tab (Visibility & Troubleshooting tool), 575–576, 786–787
spines
monitoring, 482–499
   CPU utilization, 482–485
   fan status, 488–489
   interface status, 496–499
   membership status, 491–496
   memory utilization, 485–486
   module status, 489–491

PSU (power supply unit) status, 486–488
   purpose of, 14
split-brain, 95
SSH sessions, initiating, 73
standalone rack-mount servers
   connectivity, 209
static management addresses,
   creating, 510, 530
static routes, 220
storage connectivity, 209–210
storm control, 774–775
stretched fabric, 97
subjects, creating, 384–388
subnets
   advertising, 651–656
   BDs (bridge domains) with, 527–529
   creating, 384–388
   reachability example, 450–453
   types of, 25, 43–44
switch policies, 53, 187–188, 595–596
switch policy groups, 187
switch profiles, 187–188, 595–596
   for access ports, 205–206
   for port channels, 200–201
   for VPC, 196–197
Switched Port Analyzer. See SPAN (Switched Port Analyzer)
syslog, 420–426
   configuring, 551–556
      steps in, 551–555
      verifying configuration, 555–556
   critical messages, 422–423
   enabling for Tetration, 464
IPN failure example, 423–426
leaf membership failure example, 423–424
message structure, 420–421
severity levels, 421–422

system requirements. See hardware specifications

T
TACACS+. See AAA (authentication, authorization, accounting) policies
tags, 58
TAN (Tetration Alerts Notification) agent, 461–463
tasks in UCS Director, 393–395
tcpdump
   SNMP verification, 564–565
   syslog verification, 556
tcpdump2, 745
tenant policies, NetFlow configuration with, 582–585
tenant SPAN sessions, 749
   configuring, 572–574
tenants
   explained, 18–19
   health scores, 413
   MOs (managed objects), 35–36, 38–39
   VRF instances in, 19

TEP addresses
   IS-IS and, 615–618
   in multi-pod design, 591
   in multi-site design, 592
   in remote leaf design, 593, 684
   in single-pod design, 590

Tetration, 453–473
   applications, 465–467
   cluster reboots, 469
   cluster shutdows, 469–470
   code upgrades, 467–468
   email alerts, 463
   enabling syslog, 464
   hardware agents, 455
   installing and configuring, 455–461
   patch upgrades, 467–469
   scopes, 465
   software agents, 455
   TAN (Tetration Alerts Notification) agent, 461–463
   workload security example, 470–473

Tetration Alerts Notification (TAN) agent, 461–463
three-site stretch fabric, APIC (ACI controllers) in, 97
three-tier network infrastructure, 2–3
   ACI (Application Centric Infrastructure) versus, 4–6

TL (transit leaf), 90
topology of ACI, 8
Traceroute tab (Visibility & Troubleshooting tool), 780–782
traffic status, monitoring, 499–502

transit leaf (TL), 90
transit routing, 659–661

Transparent mode (service graph), 312

TRAPS command, 417
troubleshooting
   with ACI policy model, 60–63
   APIC (ACI controllers) clusters, 727–734
with audit logs, 720
collection, 242–245
contracts, 759–765
with ELAM, 765–768
endpoint connectivity, 751–759
  with Endpoint Tracker, 752–755
  with EPMC log files, 752–755
  with EPT (Enhanced Endpoint Tracker) app, 756–757
Layer 2 traffic flow, 807–812
with Rogue Endpoint Detection, 758–759
with events, 720–722
fabric nodes, 734–737
with faults, 718–719
with iCurl, 724–726
with MOQuery, 722–724
with NIR (Network Insights Resources), 787–790
PBR (policy-based redirect), 328–331
physical- and platform-related issues, 737–751
  with counters, 737–743
  with CPU packet captures, 743–748
  with SPAN sessions, 748–751
use cases
  APIC clusters, 795–798
  contract directionality, 804–807
  end host connectivity, 807–812
  external Layer 2 connectivity, 812–813
  external Layer 3 connectivity, 814–821
  firewall traffic, 801–804
JSON script errors, 844–846
L4/L7 deployment, 832–836
leaf and spine connectivity, 821–826
leaf discovery, 792–795
multicast issues, 846–860
multi-pod design, 837–841
multi-site design, 841–843
out-of-band EPG, 799–801
VMM (Virtual Machine Manager) domains, 826–832
with Visibility & Troubleshooting Tool, 771–787
  Atomic Counter tab, 782–784
  Contract Drops tab, 777
  Contracts tab, 777–779
  Drop/Stats tab, 773–777
  Events and Audits tab, 779–780
  Faults tab, 772–773
  Latency tab, 785
  SPAN tab, 786–787
  Traceroute tab, 780–782
with Visore, 726
Two-Arm mode (service graph), 312
two-site stretch fabric, APIC (ACI controllers) in, 97

U

UCS B-Series blade chassis
  connectivity, 208
UCS Director, 392–401
  explained, 392–393
  tasks and workflows, 393–395
  use cases, 395–401
uFib (Unicast Forwarding Information Base), 604
underlay, 613–625
defined, 615
FTags and MDT, 618–625
IS-IS and TEP addressing, 615–618
unicast routing, 45
un-managed mode (service graph), 302
uRib (Unicast Routing Information Base), 600
URL format for REST API, 349–350, 474–475
use cases
for automation
with Ansible, 384–392
with REST API tools, 364–372
with UCS Director, 395–401
for multi-pod design, 100–103
for multi-site design, 122–124
for NAE (Network Assurance Engine), 438–439
for remote leaf design, 131–132
for troubleshooting
APIC clusters, 795–798
contract directionality, 804–807
end host connectivity, 807–812
external Layer 2 connectivity, 812–813
external Layer 3 connectivity, 814–821
firewall traffic, 801–804
JSON script errors, 844–846
L4/L7 deployment, 832–836
leaf and spine connectivity, 821–826
leaf discovery, 792–795
multicast issues, 846–860
multi-pod design, 837–841
multi-site design, 841–843
out-of-band EPG, 799–801
VMM (Virtual Machine Manager) domains, 826–832
user tenants, 35

vBrick Digital Media Engine (DME), 175–180
VDS (VMware vSphere Distributed Switch)
guidelines and limitations, 257–258
prerequisites, 257
verifying, 259–260
vendor-based ERP (enterprise resource planning), 165–175
viewing
audit log entries, 70
available commands, 82, 84
bond interfaces, 730
buffer drops, 742
CRC errors, 741, 742–743
current working path, 73
fabric node status, 734–735
health scores, 413
interface rate, 742
Linux routing table, 75
LLDP frames, 730–731
packets
  with `knet_parser`, 746
  with `tcpdump2`, 745
previous commands, 78
Virtual Port Channel (VPC), 191–197
  configuring, 192–193
  defining domains, 193–194
  interface policies, 195–196
  switch profiles, 196–197
virtual routing and forwarding (VRF) objects, 39–40
virtual switch status, verifying with VMware, 259
Visibility & Troubleshooting Tool, 771–787
  Atomic Counter tab, 782–784
  Contract Drops tab, 777
  Contracts tab, 777–779
  Drop/Stats tab, 773–777
  Events and Audits tab, 779–780
  Faults tab, 772–773
  Latency tab, 785
  SPAN tab, 575–576, 786–787
  Traceroute tab, 780–782
Visore, 355–358, 726
VLAN pools, 55, 186, 252, 597–600
VLANs, types of, 628
VMM (Virtual Machine Manager)
  domains, 56, 187, 250–252
    AAEP association, 252
    components, 250
    configuring, 601–602
    EPG association, 253–256
    policy model, 250
    publishing EPGs to with VMware, 258–259
  troubleshooting, 826–832
  VLAN pool association, 252
VMM (Virtual Machine Manager) integration, 249
  with Kubernetes, 272–280
    installing containers, 279
    planning, 272–273
    preparing nodes, 277–279
    prerequisites, 273–274
    provisioning ACI for, 274–277
    verifying, 280
  at multiple locations, 292–297
    multi-site design, 292–295
    remote leaf design, 295–297
  with OpenShift, 281–292
    installing containers, 290
    planning, 282–283
    preparing nodes, 287–290
    prerequisites, 283
    provisioning ACI for, 284–287
    updating router, 291
    verifying, 291–292
  with OpenStack, 263–271
    configuration examples, 271
    extending OpFlex to compute node, 264
    guidelines and limitations, 268–270
    logical topology, 265–266
    mapping constructs, 266–267
    physical architecture, 264
    prerequisites, 267–268
software architecture, 265
verifying, 270–271
with SCVMM, 260–263
mapping constructs, 261–262
mapping multiple SCVMMs to APIC, 262
topology, 260
verifying, 263
verifying OpFlex certificate deployment, 262–263
workflow, 261
with VMware, 257–260
connecting VMs to EPG port groups, 259
guidelines and limitations, 257–258
prerequisites, 257
publishing EPGs to VMM domain, 258–259
verifying, 259–260
workflow, 258
VMM (Virtual Machine Manager) policies, 36
VMs (virtual machines), connecting to EPG port groups, 259
VMware integration, 257–260
connecting VMs to EPG port groups, 259
guidelines and limitations, 257–258
prerequisites, 257
publishing EPGs to VMM domain, 258–259
verifying, 259–260
workflow, 258
vNIC status, verifying with VMware, 259
VPC (Virtual Port Channel), 191–197, 649–651
configuring, 192–193, 347–349
defining domains, 193–194
interface policies, 195–196
policy groups, 597
switch profiles, 196–197
VPC TEP (VPC tunnel endpoint), defined, 615
VRF (virtual routing and forwarding) objects, 39–40
VRF instances, 19
logical topology, 605–606
VSH shell, 81–82
accessing, 82
executing commands and redirecting to file, 82
show cli list command, 82
VSH_LC shell, 83–84
accessing, 83
executing commands and redirecting to file, 83
show cli list command, 84
vSwitch policies, configuring, 602
VTEP (VXLAN Tunnel Endpoint) addresses, 15, 17
VXLAN, 17–18, 613–625
benefits of, 17
FTags and MDT, 618–625
IS-IS and TEP addressing, 615–618
operational overview, 613–615
policy enforcement, 661–663
purpose of, 14
shared services, 664–668, 695–698
vzAny managed objects, 50–51

W

watch command, 80
where command, 73
whitelist policy model, 6–7

Windows, installing Tetration
software agents, 460–461
wiring errors, 732–733
workflows in UCS Director, 393–395

Y

YAML, 373–374