



Understanding and Troubleshooting Cisco Catalyst 9800 Series Wireless Controllers

ciscopress.com

SIMONE ARENA
FRANCISCO SEDANO CRIPPA, CCIE® NO. 14859
NICOLAS DARCHIS, CCIE® NO. 25344
SUDHA KATGERI, CCIE® NO. 45857

FREE SAMPLE CHAPTER |



Understanding and Troubleshooting Cisco Catalyst 9800 Series Wireless Controllers

Simone Arena

Francisco Sedano Crippa, CCIE No. 14859

Nicolas Darchis, CCIE No. 25344

Sudha Katgeri, CCIE No. 45857

Cisco Press

Understanding and Troubleshooting Cisco Catalyst 9800 Series Wireless Controllers

Simone Arena

Francisco Sedano Crippa, CCIE #14859

Nicolas Darchis, CCIE #25344

Sudha Katgeri, CCIE # 45857

Copyright© 2023 Cisco Systems, Inc.

Published by:

Cisco Press

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearson.com/permissions.

No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ScoutAutomatedPrintCode

Library of Congress Control Number: 2022906015

ISBN-13: 978-0-13-749232-9

ISBN-10: 0-13-749232-4

Warning and Disclaimer

This book is about deploying and troubleshooting a wireless network with the next generation Catalyst 9800 Wireless Controller. It covers the software and hardware architecture, the design and deployment aspects and provides useful troubleshooting tools. 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 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 authors and are not necessarily those of Cisco Systems, Inc.

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: Arezou Gol

Director, ITP Product Management: Brett Bartow

Executive Editor: Nancy Davis

Managing Editor: Sandra Schroeder

Development Editor: Ellie C. Bru

Project Editor: Mandie Frank

Copy Editor: Chuck Hutchinson

Technical Editors: Shobhit, Flavio Correa

Editorial Assistant: Cindy Teeters

Designer: Chuti Prasertsith

Composition: codeMantra

Indexer: Timothy Wright

Proofreader: Barbara Mack



Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

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

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

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



CCDE, CCENT, Cisco Eos, Cisco HealthPresence, the Cisco logo, Cisco Lumin, Cisco Nexus, Cisco StadiumVision, Cisco TelePresence, Cisco WebEx, DCE, and Welcome to the Human Network are trademarks. Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks, and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCFP, CCNA, CCNP, CCSP, CCOVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0812R)

Pearson's Commitment to Diversity, Equity, and Inclusion

Pearson is dedicated to creating bias-free content that reflects the diversity of all learners. We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, socioeconomic status, ability, age, sexual orientation, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where

- Everyone has an equitable and lifelong opportunity to succeed through learning
- Our educational products and services are inclusive and represent the rich diversity of learners
- Our educational content accurately reflects the histories and experiences of the learners we serve
- Our educational content prompts deeper discussions with learners and motivates them to expand their own learning (and worldview)

While we work hard to present unbiased content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.

Please contact us with concerns about any potential bias at <https://www.pearson.com/report-bias.html>.

Figure

Figure 3-12, Figure 12-22, Figure 12-34,
Figure 13-16, Figure 13-19, Figure 13-20,
Figure A-1, Figure A-3, Figure A-5,
Figure A-11, Figure A-12, Figure A-16
Figure A-25, Figure A-28

Figure 6-2-Figure 6-4, Figure 9-5,
Figure 6-11, Figure 6-12

Figure 7-4

Figure 7-5

Figure 7-6

Figure 11-2

Figure 12-11-Figure 12-13, Figure 12-20,
Figure 12-21, Figure 12-24, Figure 13-7,
Figure 13-8

Figure 12-23, Figure 12-26, Figure 12-27,
Figure 12-30, Figure 12-31, Figure 12-32

Figure 13-3, Figure 13-10,
Figure A-17-Figure A-20, Figure A-29,
Figure A-30

Figure 13-26-Figure 13-30

Figure A-6-Figure A-10,
Figure A-13-Figure A-15,
Figure A-23, Figure A-24,
Figure A-26, Figure A-27,
Figure A-31, Figure A-32, Figure A-37

Credit/Attribution

Microsoft Corporation

Wireshark

nostal6ie/Shutterstock

Terry Leung/Pearson Education Asia
Limited

Monkey Business Images/Shutterstock

Bluepixel Technologies

Internet Society

Postman, Inc

GitHub, Inc

Grafana Labs

Apple, Inc

About the Authors

Simone Arena is a principal technical marketing engineer (TME) within the Cisco Enterprise Networking & Cloud group and is primarily focused on enterprise network architecture and on all things related to wireless and mobility. Simone is based in Italy and is a Cisco veteran, having joined Cisco in 1999. Throughout the years, Simone has covered multiple roles at Cisco, starting as a software engineer working with Catalyst switching platforms, to consulting system engineer in the field, to TME within different teams (Enterprise Solution Engineering, Wireless Business Unit, Enterprise Networking and Cloud, and now Networking Experiences Group). Today Simone is the lead TME architect for Catalyst Wireless, and his time is split between helping customers and partners design the best solution that fits their needs and engineering and product management, trying to evolve and improve the products and solutions. Simone is a Distinguished Speaker at Cisco Live and has spoken at Cisco Live events all over the world for several years. Besides wireless and networking, Simone has two passions: his family, with his two daughters Viola and Anita; and Fiorentina, the best soccer team in the world...no question. In his spare time, Simone enjoys listening to music, especially through his new tube amplifier (simply awesome!).

Francisco Sedano Crippa, CCIE No. 14859, joined Cisco in 2006. After some years at TAC supporting voice solutions and as a system engineer working with service providers, he moved to the development side, where he worked on routing, datacenter and, during the past 10 years, as a technical leader on the Wireless Controller development team, focused in serviceability, location services, programmability, and cloud. He's a Cisco Live speaker and is passionate about DevOps and automation, and he is now working on architecting next-generation cloud-based lab services. When not working, he spends his time building a full-size Boeing 737 simulator in his basement and enjoying his other passion: his daughter, Scarlett, and son, Marco, and his wife, Isabel.

Nicolas Darchis, CCIE Wireless No. 25344, joined the Wireless and AAA Cisco TAC team in Belgium in 2007, where his main focus was troubleshooting wireless networks, wireless management tools, and security products. Since 2016, Nicolas has been working as a technical leader for wireless at the same technical assistance center in Brussels; he has shifted a big part of his focus to improving product serviceability of new and upcoming products, as well as new software releases. He is also a major contributor to online documentation of Cisco wireless products and has participated in many of the wireless "Ask the Expert" sessions run by the Cisco support community. Nicolas has been a CCIE Wireless No. 25344 since 2009 and, more recently, he has achieved CWNE No. 208.

Sudha Katgeri, CCIE No. 45857, is a technical leader in services for Enterprise Wireless and has been with Cisco since 2006. Besides supporting customer escalations, Sudha collaborates with Customer Experience (CX), Enterprise Networking (ENB) Escalation, Engineering, and Product Management to improve product quality and serviceability in the next-generation Catalyst wireless stack. Sudha has a CCIE in Wireless (#45857) and is an author and contributor to Wireless TAC Innovation Tools like Wireless Config Converter, CLI Analyzer, and several documents on cisco.com.

About the Technical Reviewers

Shobhit is a principal engineer in Enterprise Wireless Engineering at Cisco Systems. He has over 15 years of experience working on Enterprise Wireless LANs and Mobility, across a range of products, and has been involved with the Catalyst 9800 wireless LAN controllers since they were conceptualized. He has extensively worked on the architecture, design, and implementation of the software, which runs on Catalyst 9800 WLCs, and he continues to be passionately involved with Catalyst 9800 adoption.

Shobhit lives in Bangalore, India, with his wife, Shweta, and their eight-year-old daughter, Snehi.

Flavio Correa, CCIE Wireless No. 38913, joined Cisco in 2008 and is based in Brazil. He is a lead technical solutions architect for Latin America with more than 20 years of experience designing and deploying wireless technologies. He holds a bachelor's degree in electrical and electronics engineering from Mackenzie University and an MBA in data science from the University of São Paulo.

Dedications

I would like to dedicate this book to my father; I know he is watching me from up there, and he would be very proud. I also want to thank my family (all of them, all members of my big Italian family!) for pushing me to be a better man every day and cheering for me when I need it. Final mention to my two dogs, Barney and Mia, and the two cats, Harry and Ginny, that keep me company in my long hours in the office.

—*Simone Arena*

I would like to first dedicate this book to my parents. When I started my studies, I often went with them to buy books to prepare for CCIE, and we dreamed about writing one someday. It has been possible, thanks to all their support. I'm also lucky to have the best friends ever, especially Abel, Oscar, and Jaci. We shared so many long study nights and some unforgettable memories. And my wife, Isabel, who has supported me in every (usually crazy) idea I have. And, my son, Marco, and daughter, Scarlett. Without them, the book would have probably been published earlier, but the world would be less awesome.

—*Francisco Sedano Crippa*

I would like to dedicate this book to my family, who have supported me on every step of this long journey: Caroline, my wife, and Maxime, Emeline, and Capucine, my children who accepted I spent time writing instead of playing with them. I would like to particularly thank all the people in Cisco Engineering who love the product they work on and are always happy to provide answers, help, and confirmation about certain behaviors and are always eager to hear feedback about how we could improve the product. Giving names would mean I probably would forget some, so I prefer to thank them privately myself and collectively here.

—*Nicolas Darchis*

I dedicate this book to my family, my parents for inspiring me to follow my dreams, and my husband and kids for their unwavering love and support, without which I could not have fulfilled my goals or been part of this book. I also want to thank my coauthors for their understanding and encouragement throughout this undertaking.

—*Sudha Katgeri*

Acknowledgments

Special thanks to our awesome technical reviewers: Shobhit and Flavio Correa. They both helped us tremendously throughout this book journey not only with their corrections and attention to details but also with their suggestions on the content, identifying missing use cases and hence improving the quality of the book.

We would like to thank our management leadership in Cisco for supporting us during the lengthy process of writing a book. Thank you to the EMEA and US CX leadership: Matthew Batson, Kathy Ferguson, Wes Moss; and EN Product Management leadership: Greg Dorai, Chandan Mehndiratta, and Muhammad Imman.

This book couldn't have been possible without the support of many people on the Cisco Press team. Thanks to Nancy Davis, executive editor; her enthusiasm and dedication were instrumental in getting the book done. Eleanor Bru, development editor, did an amazing job in the technical review cycle, and it has been an absolute pleasure working with you. Also, many thanks to the numerous unknown soldiers at Cisco Press working behind the scenes to make this book happen.

Contents at a Glance

	Introduction	xxvii
Chapter 1	Cisco C9800 Series	1
Chapter 2	Hardware and Software Architecture of the C9800	25
Chapter 3	C9800 Configuration Model	43
Chapter 4	C9800 Deployment and Installation	65
Chapter 5	Security	89
Chapter 6	Mobility and Client Roaming	159
Chapter 7	RF Deployment and Guidelines	195
Chapter 8	Multicast and Multicast Domain Name System (mDNS)	247
Chapter 9	Quality of Service (QoS)	285
Chapter 10	C9800 High Availability	323
Chapter 11	Cisco DNA Spaces Integration and IoT	361
Chapter 12	Network Programmability	393
Chapter 13	Model-Driven Telemetry	437
Chapter 14	Cisco DNA Center/Assurance Integration	469
Chapter 15	Backing Up, Restoring, and Upgrading Your C9800	493
Chapter 16	Troubleshooting	507
Appendix A	Setting Up a Development Environment	579
	Index	607

Contents

	Introduction	xxvii
Chapter 1	Cisco C9800 Series	1
	Why Cisco C9800?	2
	Intent-Based Networking (IBN)	3
	Flexible Software	4
	Flexible Hardware	5
	The Role of the Wireless Controller in a Cloud Era	7
	Managing the Cisco C9800	10
	Traditional Management Tools	11
	“On Box” Management	11
	Cisco Prime Infrastructure	16
	Cisco DNA Center	19
	C9800 Prerequisites for Cisco DNA Center	20
	CI/CD Tools	21
	Licensing	21
	Cisco Next-Generation Wireless Stack	22
	Summary	23
	References	23
Chapter 2	Hardware and Software Architecture of the C9800	25
	General CAPWAP Split MAC Architecture	25
	The Controller Control Plane Architecture Elasticity	27
	IOS-XE Software Architecture	27
	WNCd: The Heart of the Wireless Controller Control Plane	28
	Other Wireless Processes	31
	Wireless Client State Machine	31
	One Dataplane to Rule Them All (or Three at the Maximum)	35
	Hardware Overview	38
	C9800-40 and C9800-80	38
	C9800-L	40
	C9800-CL	41
	Summary	42

Chapter 3 C9800 Configuration Model 43

C9800 New Configuration Model	43
What Does My AireOS AP Group Migrate To?	46
What About FlexConnect?	47
Cisco C9800 Series Profile and Tag Considerations	48
Assigning Tags	48
Moving APs Between Wireless Controllers and Preserving Tags	54
Roaming Between Policy Tags	55
Designing with Site Tags in Mind (Local Mode APs)	57
Designing with Site Tags in Mind (FlexConnect Mode APs)	63
Summary	64
References	64

Chapter 4 C9800 Deployment and Installation 65

C9800 Deployment Models	65
C9800 for Private Cloud	65
C9800 Physical Appliance	66
C9800 Virtual Appliance	70
Embedded Wireless Controller on Catalyst AP and Switch	74
C9800 for Public Cloud	75
Setting Up Your First Catalyst Wireless Network	79
C9800 Initial Setup	80
Access Point Join	83
Configuring WLAN and Connecting a Client	85
Summary	87
References	87

Chapter 5 Security 89

Network Security Fundamentals	89
Access Control Lists (ACLs)	89
Defining ACLs	90
Applying ACLs	91
Applying Wireless ACLs on the WLC	93
FlexConnect ACLs on the AP	94
The Case of Downloadable ACLs (DACLS)	95
URL Filters (a.k.a. DNS-Based ACLs)	96
Certificates and Trustpoints	97
A Case for Trustpoints	98

How to Add a Certificate on the Controller	98
AAA	103
RADIUS	104
RADIUS Attributes	105
RADIUS Sequence Example	106
RADIUS Change of Authorization (CoA)	107
RADIUS Configuration and Load Balancing	108
Configuring RADIUS Servers	108
Configuring RADIUS Server Groups	108
RADIUS Server Fallback	110
RADIUS Load Balancing	111
RADIUS Accounting	111
AAA Methods	112
Local EAP	113
TACACS+	114
LDAP	116
Wireless Security Fundamentals	116
Wired Equivalent Privacy (WEP)	116
Wi-Fi Protected Access (WPA)	116
802.1X for WPA Enterprise	119
802.1X Components	119
EAP	120
EAP Methods	121
WPA3 Enterprise	124
Preshared Key for WPA Personal	124
WPA3 SAE	125
MPSK	126
Identity PSK (iPSK)	127
MAC Filtering	127
Enhanced Open	128
Securing the Air	128
WPA2 Personal	129
WPA3 SAE	130
WPA2 with iPSK	132
Enhanced Open	138
(Local) Web Authentication	140

Central Web Authentication	143
Web Authentication Best Practices	145
HTTPS Redirection	145
Captive Portal Bypass	146
Web Authentication Takeaways	147
Rogue Detection and WIPS	148
Securing Your Access Points	148
AP Authorization	148
AP 802.1X Authentication	149
Securing the AP Join Process Using Locally Significant Certificates	150
Securing Your Wireless Controller	151
Securing Administrator Access	151
Using TACACS+	151
Using RADIUS	153
Guest Users	153
The Lobby Ambassador Type of User	153
NETCONF	153
Granularity of WebUI Access	154
Connect to the WebUI Using Certificates	154
Securing Traffic	154
Encrypted Traffic Analytics	154
Cisco Umbrella	155
Cisco Secure Development Lifecycle (CSDL)	157
Summary	157
References	157

Chapter 6 Mobility and Client Roaming 159

802.11 Roaming	160
Full-Auth Roaming (or Slow Roam)	161
Fast Secure Roaming	163
PMKID Caching (Sticky Key Caching)	164
OKC	165
CCKM	167
Fast Transition (802.11r)	169
Roaming Optimizations	177
802.11k	177
802.11v BSS Transition	179

Types of Client Roaming	181
Intra-Controller Roaming	181
Intra-WNCd Roaming (Same Site Tag, Same Policy Profile)	181
Inter-WNCd Roam (Different Site Tags, Same Policy Profile)	182
Intra-WLC Roam (Same Site Tag, Different Policy Profile)	183
Inter-Controller Roaming	185
Layer 2 Roaming	185
Layer 3 Roaming	185
Static IP Client Mobility	187
Auto-Anchor Mobility (Guest Tunnel)	187
Configuring Secure Mobility Tunneling on a C9800	188
C9800 to AireOS Inter-Release Controller Mobility (IRCM)	191
Summary	192
References	193

Chapter 7 RF Deployment and Guidelines 195

Radio Resources Management (RRM) Concepts and Components	195
Antennas and Signal Propagation	195
Countries and Domains	197
Challenging RF Environments	199
Metal-Heavy Areas	200
High-Density Crowd Areas	200
Shielded Doors and Sudden Turns	201
Uneven Ceilings	201
Atriums	202
Radio Resources Management (RRM)	203
Data Collection	203
RF Grouping	206
RF Grouping Modes	207
TPC	208
TPC Overview	208
TPC Minimum and Maximum	209
Coverage Hole Detection	210
DCA	211
Overlapping Basic Service Set (BSS)	213
Cloud-Based RRM	215
RF Profiles	215
Spectrum Intelligence and CleanAir	219

Configuring CleanAir	222
Monitoring the Spectrum Live	222
Interferer Location Tracking	223
Monitoring the RF Space	224
Advanced RF Features	224
Band Select	225
Aggressive Client Load Balancing	226
Off-Channel Scanning Defer	227
Airtime Fairness (ATF)	228
Wi-Fi 6 Features	228
OFDMA	229
Multi User–Multiple Input Multiple Output (MU-MIMO)	229
Target Wake Time (TWT)	229
BSS Coloring	231
Channel Width	232
Dynamic Frequency Selection (DFS)	232
DFS Overview	233
DFS in the C9800	234
Flexible Radio Assignment (FRA)	235
Tri-radio	236
Wireless Intrusion Prevention System (WIPS) and Rogue Detection	238
Rogue AP Detection and Classification	238
Detecting a Rogue Access Point	238
Classifying Rogue Access Points	240
Understanding the Danger of a Rogue Access Point	241
Containing Rogue Access Points	241
Adaptive WIPS	244
Client Exclusion	245
Summary	246
References	246
Chapter 8 Multicast and Multicast Domain Name System (mDNS)	247
Wireless Multicast	250
Multicast Packet Flow in Wireless	250
Multicast in a Centralized Wireless Deployment	250
Multicast in Flex	251
Multicast in Fabric	251

How to Configure Multicast on the C9800	251
IGMP and MLD on the C9800	253
CAPWAP Multicast	254
Multicast over Unicast (MoU)	254
Multicast over Multicast (MoM)	256
802.11 Multicast	259
Wireless Broadcast and Non-IP Multicast	260
Multicast in Client Roaming Scenarios	262
Media Stream Feature	263
Cell Planning	264
Components of VideoStream	264
How to Configure Media Stream	267
mDNS	272
mDNS Bridging	273
mDNS Gateway	274
How to Configure mDNS Gateway	274
mDNS Gateway on WLAN	276
mDNS Service Policy on Policy Profile	277
mDNS Service Policy	277
mDNS Service Policy on VLAN SVI	280
mDNS Service Policy via AAA Override	281
mDNS-the AP	281
mDNS Gateway in FlexConnect Deployment	282
mDNS Gateway with Guest Anchor	283
Summary	283
References	283
Chapter 9 Quality of Service (QoS)	285
Wi-Fi Quality of Service (QoS)	286
Wi-Fi (802.11) QoS Fundamentals	287
QoS Design	289
UP and DSCP Mapping	290
DSCP to UP Mapping	295
Wireless Call Admission Control (CAC)	298
Implementing Wireless QoS on the C9800	300
QoS Policy Targets	300
Modular QoS CLI	301
Trust DSCP Model	302

Designing and Deploying Catalyst C9800 QoS	304
QoS Deployment Workflow	304
Auto QoS	310
QoS Profiles (a.k.a. Metal QoS Profiles)	313
Application Visibility and Control (AVC)	316
Deployment Verification and Restrictions	319
Fastlane+ (Plus)	319
Best Practices	320
Summary	322
References	322
Chapter 10 C9800 High Availability	323
SSO Redundancy	324
Prerequisites	325
Ports and Interfaces	327
Redundancy Management Interface (RMI)	327
Redundancy Port (RP)	328
Uplink Ports	330
Console Port	330
Out-of-Band Management/Service Port (SP)	330
RP+RMI Supported Topologies	331
Building an RP+RMI HA Pair	331
Configuration	332
Active-Standby Election Process	335
HA Sync	335
HA Formation in Action	336
SSO Switchover	338
System and Network Error Handling	339
Monitoring HA	344
Monitoring an HA Pair via the CLI	344
Monitoring an HA Pair via the GUI	347
Monitoring an HA Pair via SNMP	348
Monitoring an HA Pair via Programmatic Interfaces	348
RP Only to RP+RMI HA Migration	349
HA Teardown	349
SSO Deployment: Impact on Features	350
Mobility (Mobility MAC)	350

Link Aggregation Group (LAG)	351
Multi-Chassis LAG	352
N+1 Redundancy	352
N+1 HA Configuration	353
Configuration on the AP Join Profile	354
CAPWAP Timers	355
Preserving AP-to-Tag Mapping across N+1 Failovers	356
Licensing with N+1	357
N+1 vs. SSO High Availability	357
HA in EWC-AP Deployment	358
HA in EWC-SW Deployment	359
Summary	359
References	360
Chapter 11 Cisco DNA Spaces Integration and IoT	361
Value-Added Wireless Services	361
Location Tracking	361
Accuracy	362
Location Update Frequency	363
Presence	364
The Impact of Privacy MAC Addresses	364
Location Deployment Guidelines	364
Other Technologies	365
Analytics	366
Guest Services	366
Bluetooth and IoT	366
BLE	367
IoT	368
Bluetooth Location Tracking	371
Connected Mobile Experiences (CMX)	372
Cisco DNA Spaces	372
Deployment Modes	374
Direct Connection	374
Cisco DNA Spaces Connector	375
CMX Tethering	379
Specific Service Examples	379
OpenRoaming	379
What Problem Is OpenRoaming Trying to Solve?	379

OpenRoaming Architecture	379
OpenRoaming Configuration	381
Captive Portal	386
Advantages of a Portal on Cisco DNA Spaces	388
Proximity	389
BLE Gateway on Cisco DNA Spaces	389
Summary	392
References	392

Chapter 12 Network Programmability 393

What Is Network Programmability?	393
Why Is Network Programmability Needed?	393
Is Network Programmability a New Concept?	396
Orchestration of the Entire Network	396
Configuration Repeatability	396
Idempotency	397
Imperative vs. Declarative Models	397
Infrastructure as Code (IaC)	400
Network Programmability in the C9800	401
Data Models	402
YANG Data Models	403
Encoding Formats	406
XML	406
JSON	407
Protobuf	408
Protocols	408
NETCONF	409
NETCONF Capabilities	409
NETCONF Layers	409
RESTCONF	411
HTTP Methods	411
HTTP Return Codes	411
gNMI/gRPC	412
Tools to Examine YANG Models	412
pyang	412
Using pyang in a Docker Container	413
YANG Suite	414

How to Examine Data Using NETCONF and YANG Suite	419
How to Examine Data Using RESTCONF and POSTMAN	421
Enabling RESTCONF	422
RESTCONF URIs	422
Root	422
Resource	423
Data Model	423
Searching Data	425
Updating the Configuration	426
Python and Network Programmability	429
Assigning Tags to APs Based on Serial Number	429
Program Structure	431
Summary	436
References	436
Chapter 13 Model-Driven Telemetry	437
What Is Model-Driven Telemetry?	437
How to Enable Model-Driven Telemetry	438
NETCONF	439
RESTCONF	439
gNMI	440
Operational Data and KPIs	441
Polling vs. Subscribing	447
Telemetry Streams	448
Yang-notif-native Stream	448
Yang-push Stream	448
How to Identify Subtrees in YANG Models	449
Dial-out vs. Dial-in	450
Dial-out	450
Dial-in	451
Creating Dial-in Subscriptions	453
Tools	460
YANG Suite	460
TIG (Telegraf, Influx, Grafana)	461
Creating a Dashboard	463
Summary	467
References	467

Chapter 14 Cisco DNA Center/Assurance Integration 469

Introduction	469
Cisco DNA Center Assurance Architecture	471
Managing the C9800 with Cisco DNA Center	472
Client 360	473
AP 360	475
Network Services Analytics	477
Device Analytics	479
Apple Analytics	479
Samsung Analytics	481
Intel Analytics	481
Intelligent Capture	483
Cisco Active Sensor	488
Sensor Provisioning and Onboarding	489
Test Suites	489
Troubleshooting the Assurance Application	491
Summary	492
References	492

Chapter 15 Backing Up, Restoring, and Upgrading Your C9800 493

Saving and Restoring the Configuration for Disaster Recovery	493
Saving the Configuration Changes	494
Backing Up the Configuration and Restoring It	494
Backing Up Everything for Restoring on Another Controller	495
The Advantage of Backing Up Using the WebUI	496
The Case of Configuration Encryption	496
Backing Up Using Cisco Prime Infrastructure	497
Backing Up Using Cisco DNA Center	498
Running IOS-XE in Install or Bundle Mode	500
Bundle Mode	500
Install Mode	500
Upgrading (and Downgrading) the Controller Safely	501
Standard Upgrade	501
AP Predownload	503
Efficient Upgrade	505

Rolling AP Upgrade (for N+1)	505
In-Service Software Upgrade (ISSU)	506
Summary	506
References	506
Chapter 16 Troubleshooting	507
Control Plane Tracing	509
Syslog	509
Binary Tracing	511
Always-On Tracing	515
Per-Process Debugging	520
Radioactive Tracing	521
Embedded Packet Capture (EPC)	525
Packet Tracer	531
Troubleshooting Dashboard	536
Core Dump and System Report	536
Debug Bundle	539
Ping and Trace Route	539
Other On-the-Box Tools on the C9800 GUI	540
AireOS Config Translator	540
Command-Line Interface	541
File Manager	542
Walk-Me Integrated with the C9800 GUI	542
Configuration Validator	543
Offline Tools for the C9800	545
Wireless Configuration Convertor	545
Wireless Config Analyzer	546
Wireless Debug Analyzer	547
Log Advisor	548
Health and KPI Monitoring	548
Dashboard	549
Hardware Monitoring	550
Smart Licensing	557
Direct Connect	557
On-Premises SSM or CSLU	558
Airgap	558

AP Health Monitoring 559

Client Health Monitoring 563

CPU Monitoring 570

Memory Monitoring 575

Data Plane Monitoring 576

Summary 577

References 578

Appendix A Setting Up a Development Environment 579

Index 607

Reader Services

Icons Used in This Book



Building



Router



Switch



Phone



Laptop



Database



Layer 3 Switch



Cloud



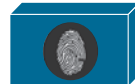
Terminal



Server



Wireless LAN Controller



ISE



Access Point



Wireless Connection

Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in Cisco's 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).
- *Italics* indicate arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets [] indicate optional elements.
- Braces { } indicate a required choice.
- Braces within brackets [{ }] indicate a required choice within an optional element.

Introduction

Wireless networks are continuously evolving: the technology is changing at a very fast pace from Wi-Fi 4/5 to Wi-Fi 6/6E and to Wi-Fi 7, which is not too far away on the horizon. The applications and the services enabled on the Wi-Fi network are evolving as well, becoming more and more business critical with stringent requirements on performance and latency. The number of devices that needs to be supported is increasing, and high-density deployments are becoming the norm.

To meet these fast-changing requirements, you need a wireless network that can quickly adapt, that is secure and can perform and scale to higher standards, and that can be easily and programmatically managed. This is the reason behind the introduction of the next-generation wireless controller, the Catalyst 9800 (C9800), which is a critical element of the new Catalyst wireless stack.

The C9800 is based on a new modern and secure operating system, Cisco IOS-XE, and it's built from the ground up for Cisco intent-based networks to deliver on the next wave of wireless innovations.

Goals and Methods

The goal of this book is to educate wireless professionals on how to design, deploy, and manage a Catalyst wireless network built using the new Catalyst 9800 wireless LAN controller, providing practical tips and recommendations.

Who Should Read This Book?

This book is for all readers who are passionate about wireless and want to learn how to design, deploy, and manage a Cisco Catalyst wireless network. The book has been written with three groups of readers in mind:

Technical Decision Makers (TDM): The book familiarizes TDMs with a deep technical view of the Catalyst 9800 wireless controller and helps you in addressing your specific business needs.

Network architects: The book provides technical details on the software and hardware architecture of the Catalyst wireless solution so that you can better understand how to design your own network and implement the features and functionalities you need.

Network operators and IT professionals: The book provides useful tools and tips for setting up, configuring, and monitoring and troubleshooting the network, making your work as a network operator easier.

How This Book Is Organized

The book layout follows the lifecycle of the design and deployment of a Catalyst wireless network using a C9800 wireless controller and hence is divided into three parts:

1. Designing and bringing up the network: Day Zero
2. Configuring and deploying: Day One
3. Monitoring and troubleshooting: Day Two/Day N

Most of the theory is laid out in advance so you can understand the concepts before you see the practical aspects of each topic. We recommend you go through the chapters sequentially to get the full benefit of the book.

Book Structure

In part one, you find the following chapters:

Chapter 1, Cisco C9800 Series: This chapter introduces the Catalyst 9800 wireless controller, its benefits, and its main characteristics. The chapter clarifies the reasons for a new wireless controller for intent-based networking and the role it plays in a cloud-based management network. The chapter also describes the flexible options to manage the C9800, the licensing model, and how this new wireless controller fits into the next-generation wireless stack.

Chapter 2, Hardware and Software Architecture of the C9800: This chapter covers the software architecture of the Catalyst 9800, the reasons behind using a unified IOS-XE for all Cisco enterprise platforms, and the benefits of a scalable architecture.

Part two has the following chapters:

Chapter 3, C9800 Configuration Model: This chapter introduces the new configuration model available for the Catalyst 9800 wireless controller. Important design considerations and best practices are illustrated to get the best performance and stability out of your C9800-based wireless network.

Chapter 4, C9800 Deployment and Installation: This chapter illustrates the different deployment modes supported for the private and public cloud and describes how to bring up your C9800 to an operational state. It then suggests different methods to easily configure your first WLAN on a Catalyst wireless network.

Chapter 5, Security: This chapter covers all the security aspects of the Catalyst 9800 controller, focusing on AAA operations, the use of access control lists (ACLs) to restrict client traffic or to protect the controller management plane, and rogue detection and the Wireless Intrusion Prevention System (WIPS).

Chapter 6, Mobility and Client Roaming: This chapter explains in detail the concept of seamless client roaming in an enterprise deployment. It also describes optimizations such as Fast Secure Roaming, 802.11k, 802.11v, and 802.11r-FT. Deploying mobility and related

roaming optimizations on the C9800, and in co-existence with existing AireOS WLCs, is key to enabling a successful adoption of the C9800.

Chapter 7, RF Deployment and Guidelines: This chapter covers basic antenna concepts and radio resource management (RRM) features and functionalities and provides the tuning and recommendation details you need to have your wireless network running like clockwork.

Chapter 8, Multicast and Multicast Domain Name System (mDNS): This chapter describes the configuration and optimizations available on the Catalyst 9800 wireless controller to deliver broadcast, multicast, and mDNS traffic effectively and efficiently.

Chapter 9, Quality of Service (QoS): The C9800 is based on IOS-XE and leverages the Cisco Modular QoS CLI (MQC) to define and implement an end-to-end architecture for QoS. The chapter describes the design and deployment of the C9800 “trust DSCP” QoS model and provides best practices.

Chapter 10, C9800 High Availability: This chapter describes high availability capabilities of the Catalyst 9800 wireless controller. The C9800 provides stateful switchover (SSO) to allow subsecond failover for access points (APs) and clients. The chapter delves into the design and deployment of the network to leverage redundancy features for the C9800.

Part three features the following chapters:

Chapter 11, Cisco DNA Spaces Integration and IoT: This chapter covers at a high level the value-added services that the wireless network can provide, mainly via Cisco DNA Spaces. The chapter explains the integration of DNAs with the Catalyst 9800 and APs through the Cisco DNA Spaces connector, the protocols involved, and the solutions offered.

Chapter 12, Network Programmability: This chapter describes network programmability concepts and the protocols used in modern network programmability, such as NETCONF and RESTCONF. It describes the YANG data models used in the Cisco 9800. It also presents examples illustrating how Python can be used to query the Cisco 9800 operational model and how to create site tags using RESTCONF.

Chapter 13, Model-Driven Telemetry: This chapter describes how you can benefit from model-driven telemetry on the Catalyst 9800 wireless controller and how to integrate the rich data models available with open-source tools.

Chapter 14, Cisco DNA Center/Assurance Integration: This chapter gives an overview of how Cisco DNA Center Assurance works and how it can help you in troubleshooting your Catalyst wireless networks. The chapter focuses on the Catalyst 9800 integration with Cisco DNA Center.

Chapter 15, Backing Up, Restoring, and Upgrading Your C9800: This chapter covers different operational aspects of the C9800, including how to save the configuration on the controller, how to back it up to an external location, and how to restore the configurations in case of problems on the device or in case of complete device replacement.

Chapter 16, Troubleshooting: This chapter explains the various monitoring, debugging, tracing, and packet-capturing features that are native to the C9800 and introduces tools that are available on-the-box and offline that help with data collection and analysis to determine the root cause of problem scenarios.

Appendix A, Setting Up a Development Environment: In this extra hands-on appendix, you learn how to install and use the tools needed to work with C9800 programmability. This appendix offers an overview of a modern development environment, including Git and container-based environments.

Security

Everyone typically agrees that wireless always had a reputation of being less secure than cables. This belief is due to the inherent nature of an open medium where an attacker can place attacks without any physical contact and without cutting into cables, as well as the ever-increasing mobile workforce, which makes identifying and tracking connected devices more difficult. Security has always been important and has become even more important over wireless due to the open nature of the medium. With the ever-increasing power of brute-force cloud resources and continuously discovered weaknesses in existing security protocols, security will continue to increase in importance.

Cisco Identity Services Engine (ISE) is the platform of choice for identity and access control and policy management. It enables organizations to enforce wireless access controls by implementing 802.1X, MAC authentication, web authentication, and administrative user authentication. It also has many other features in terms of visibility and monitoring, but these topics are covered in other books. This chapter covers all of the security aspects, one by one, even if each section references others because security is very closely integrated into all these areas.

Network Security Fundamentals

Wireless security uses many concepts and services familiar to the wired security world. As a matter of fact, a good part of the security infrastructure sits on the wired network. It is therefore essential to have a good grasp of concepts like access control lists, authentication servers, and certificates.

Access Control Lists (ACLs)

Access control lists are an important part of the security of a platform. Conceptually, ACLs allow you to match specific traffic and take specific actions on that traffic. The traffic is matched by giving a network address and adding a wildcard mask, which is a kind of inverted subnet mask. For example, an ACL covering 192.168.1.0 with mask 0.0.0.255 affects all IP addresses from 192.168.1.0 to 192.168.1.255. The binary 1s of the

wildcard mask define which bits can vary (“don’t care” bits), and the 0s define which bits have to statically match.

An access control entry (ACE) is a single statement in this ACL; it covers a network address, a wildcard mask combination, and a permit or deny statement. An ACL is made up of one or more ACEs; because an ACE can cover only a contiguous range of addresses covered by a wildcard mask, the ACL can combine several ACEs to deny or permit unrelated address ranges (and sometimes port combination too). The order in which the ACEs are defined matters because it is the order in which the network device compares the traffic and takes the corresponding action as soon as one ACE matches. The rest of the ACEs are not even considered as soon as a match is found.

An ACL defines a specific set of traffic to be matched and does not take any action on that traffic until you apply it somewhere. Where the ACL is applied is the key and determines exactly what action is taken. Too often, network administrators link ACLs with dropping or allowing traffic, and although these are important use cases, they are not the only ones. ACLs can define where to apply specify quality of service (QoS) actions in the traffic; ACLs also can define the traffic to intercept for a web portal and many other things.

ACLs use computing power intensively in the way they work because traffic evaluated against the ACL has to be evaluated against each statement of the ACL. If the statement does not match, the Catalyst 9800 moves to the next statement until a match is found. If the end of the ACL is reached and no statement matched, a default action is taken, which is typically to deny the traffic, but the action depends on the type of ACL. Luckily, the 9800 appliances are hardware-accelerated (as discussed in Chapter 2, “Hardware and Software Architecture of the C9800”). ACLs have little impact overall on the platform performance, but it is always good to consider writing the most efficient ACLs with as few lines as possible to meet the security criteria.

Defining ACLs

IOS-XE inherits several types of ACLs that can be configured, but from a practical viewpoint, extended ACLs are the only ones you should be concerned about. They are like standard ACLs but allow you to identify the traffic more granularly. There is no reason to define a standard ACL on a Catalyst 9800 controller because the extended ACLs supersede them in every way.

An exception to the previous statement is the fact that the Catalyst 9800 supports Cisco TrustSec and Scalable Group ACLs (SGACLs), also known as role-based ACLs, and is part of the TrustSec solution as well as the SD-Access solution.

The command-line interface (CLI) allows you to define ACLs by name or by an identifying number. It’s much easier to use names in general:

```
c9800-CL(config)#ip access-list extended ?
<100-199>    Extended IP access-list number
<2000-2699> Extended IP access-list number (expanded range)
WORD        Access-list name
```

The structure of a CLI ACL statement or ACE is as follows:

```
<sequence number> [permit/deny] <protocol> <address or any> eq <port number> <subnet> <wildcard>
```

For example:

```
1 permit tcp any eq www 192.168.1.0 0.0.0.255
```

The sequence number allows you to specify where in the ACL order of ACEs to insert the ACE. It is usually a best practice to define your statements with the sequence 10, 20, 30, 40, and so on. This way, you can “insert” a statement later without deleting half of the ACL by using sequence 5 or 15, for example, in the statement you want to insert between two existing statements.

The WebUI allows you to write a complete ACL much more easily by going to the **Configuration > Security > ACL** page, as shown in Figure 5-1. You can then see a list of protocols to pick from and make changes to an existing ACL much more conveniently (rather than having to use the “no” form of a statement to remove it and reconfigure it differently afterward).

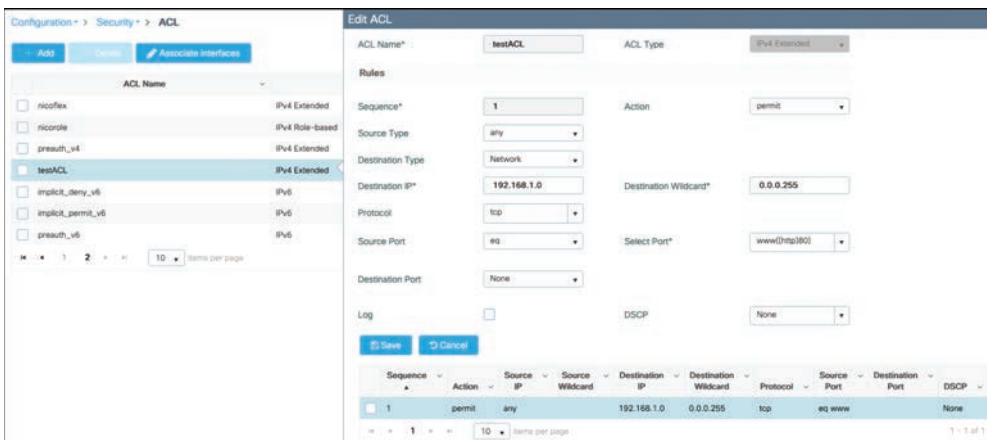


Figure 5-1 Creating an ACL on the Catalyst 9800

At the end of an ACL statement, you can add the **log** keyword; this has the effect that a syslog message will be thrown when that specific statement is matched. This option is extremely verbose on the controller terminal and will typically overwhelm it. Considering the syslog is thrown on the controller, it also only works with centrally switched ACLs and does not work when the ACL is applied by the FlexConnect AP.

Applying ACLs

An ACL might be called by different names depending on where it is applied and its function. In all cases, the ACL is defined the same way on the controller (as an extended

IP ACL). There are several ways of applying an ACL, but two stand out and are named differently because they have a different impact on the traffic:

- **Security ACL:** This name refers to an ACL in its most popular understanding, that is, defining what traffic is allowed through the device and which type of traffic is blocked and dropped.
- **Punt ACL or redirect ACL:** This name refers to an ACL that specifies which traffic is sent to the CPU (instead of its normal expected handling by the data plane) for further processing. A good example is the central web authentication (CWA) redirect ACL, which defines which traffic is intercepted and redirected to the web login portal. The ACL does not define any traffic to be dropped or allowed through but simply what follows the regular processing or forwarding rules and what is sent to the CPU for interception. A redirect ACL has an invisible last statement that is an implicit deny. This implicit deny is applied as a security access list entry (and therefore drops traffic that isn't explicitly allowed through or sent to the CPU).

You may have heard of DNS ACLs (or URL filters) or downloadable ACLs; we touch on those later. There are other ways of applying an ACL, but they typically refer to a specific feature (for example, an ACL in a QoS policy specifies traffic where this QoS policy will apply). On top of the “how,” there is also the “where.” A security ACL permits or drops traffic, as we have already stated, but where the ACL is applied also plays a role, so let's clarify that point.

A security ACL can be applied as follows:

- **On an SVI interface:** The ACL is evaluated only against the traffic that is routed through that interface. Using an SVI as a default gateway for a wireless client is not supported by the Catalyst 9800 at the time of this writing, so this only leaves the scenario where you have SVIs for management purposes and the ACLs applied on those SVIs apply to traffic destined to those interfaces.

```
myc9800-CL(config)#interface Vlan<number>
myc9800-CL(config-if)#ip access-group myACL in/out
```

- **On a physical interface of the controller:** The ACL is evaluated against all traffic that passes through that interface. Along with applying ACLs on the SVI, this is your other option for restricting traffic hitting the Catalyst 9800 management plane.

```
myc9800-CL(config)#interface GigabitEthernet1
myc9800-CL(config-if)#ip access-group myACL in/out
```

- **In a wireless policy profile or WLAN:** This vague category encompasses several places where you can configure an ACL that will be applied to the wireless client traffic both in the case of central switching or local switching of traffic. Such ACLs are supported only in the inbound direction.
- **On the AP itself:** In the case of FlexConnect local switching, the ACL is still configured and applied from the policy profile on the controller, but there is an extra step

of downloading this ACL to the AP through the Flex profile. ACLs must be downloaded to the AP before they can be applied at all. As an exception, fabric mode APs (in the case of software-defined access) also use Flex ACLs even though the AP is not operating in Flex mode.

Applying Wireless ACLs on the WLC

When editing a policy profile, you can specify an IPv4 and a separate IPv6 ACL (see Figure 5-2) as WLAN ACL. This is the most straightforward way of applying an ACL to all traffic to and from clients that are fully authenticated (that is, in the RUN state).

Figure 5-2 Access policies of the policy profile on the C9800

The explicit mention of clients in the RUN state is deliberate. Are there cases where you want to apply ACLs to clients that have not hit the RUN state yet? Apart from getting an IP address, clients are not supposed to be sending traffic before[el]except in the case of a web authentication WLAN. In that scenario, they can send and receive the traffic necessary to submit their credentials on the web login page before being in the final RUN state of the client state machine (for more details, see the “Wireless Security Fundamentals” section of this chapter). In some cases, you might want to allow some traffic before it is authenticated. For example, the external portal should be allowed, or you might want to also allow specific internal resources to be consulted before being authenticated. This requires a preauthentication ACL. In the WLAN edit section, illustrated by Figure 5-3, you can see a preauthentication ACL section (again doubled between IPv4 and IPv6) that allows you to define traffic to be allowed before the RUN state. A preauthentication ACL should mostly contain permit statements. Whatever does not match a statement in the preauthentication ACL automatically matches a default deny because the client is in a WEBAUTH_REQD state where all traffic is blocked apart from HTTP (that is intercepted).

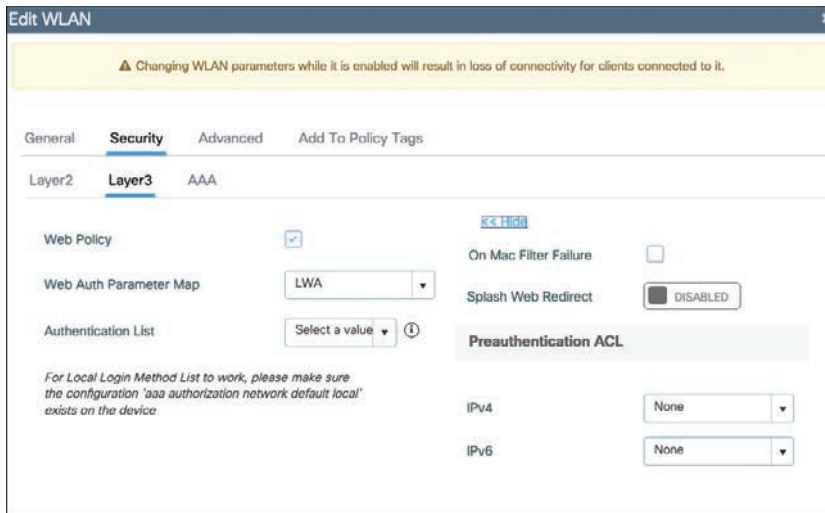


Figure 5-3 Layer3 WLAN security settings where the preauthentication ACL can be configured

These examples cover the static definition of ACLs. The Catalyst 9800 wireless LAN controller (WLC) supports the dynamic assignment of ACLs from a RADIUS server through the use of the Airespace-ACL-Name. This is covered with an example in the later “RADIUS” section. In this case, it requires enabling AAA override on the policy profile.

FlexConnect ACLs on the AP

If your AP is running as FlexConnect and locally switching traffic, there is no client traffic hitting the WLC data plane at all, and therefore, the WLC cannot physically apply any policing to the client traffic. This means the AP has to do the job itself. Defining an ACL for use on a FlexConnect AP is the exact same thing as covered before: you define a regular ACL on the Catalyst 9800. The specificity is that you have an extra step of predownloading it to the AP. The term *predownload* is more accurate than *apply* because the latter term implies that traffic might be matched against it already. But there are basically three different steps in the configuration for a local switching scenario:

- You define the ACL on the WLC.
- You predownload it on the AP(s).
- You apply the ACL somewhere in the configuration.

The predownload step is simple. Navigate to **Configuration > Tags and Profiles > Flex** and select the **Policy ACL** tab of a given Flex profile (see Figure 5-4). There, selecting **Add** means predownloading an ACL to the AP. You are able to add any ACL that was configured on the controller, and when you click **Save**, the ACL statements

are downloaded on the AP and visible in a **Show ip access-list** on the AP itself (as displayed in Example 5-1).



Figure 5-4 The Flex profile Policy ACL tab allows you to download ACLs to the APs

Example 5-1 Output of `show ip access-list` on an AP

```
9120-AP#show ip access-list
Extended IP access list nicoflex
 1 deny icmp any 1.1.1.1 0.0.0.0
 2 deny icmp any 2.2.2.2 0.0.0.0
 3 permit icmp any any
 4 deny icmp any 3.3.3.3 0.0.0.0
```

On that page, you select the **Central Web Auth** check box if your ACL is meant to be a redirect/punt ACL for use in a central web auth scenario. Its effect is to invert the permit and deny statements because the redirect ACL has inverted meaning on the AP operating system versus IOS-XE. (The preauth URL filter is covered in the next section.)

When the provisioning of ACLs on the APs is done, if the ACL is called out in the WLAN or policy profile, as demonstrated before, it is properly applied given that the ACL is now defined on the AP as well.

The Case of Downloadable ACLs (DACLs)

A downloadable ACL is an access list that is not predefined on the network device but returned dynamically, including all its statements, to the device via RADIUS during a network authentication and authorization. This type of ACL is popular with wired 802.1X deployments because it's much easier to centralize all the ACL definitions on the RADIUS servers than it is to make sure they are deployed on all network switches. The RADIUS server returns an authorization result containing a specific AV-pair mentioning the DACL name. The network device is then expected to start a new RADIUS request of a special type to download all the statements from that ACL. Therefore, the DACL download shows up as a separate successful authentication after the client authentication in the RADIUS logs. Wireless deployments have less need for DACL because they use a centralized controller most of the time. The 9800 controller benefits from the DACL

implementation inside IOS-XE but is not adapted to the wireless workflow yet, and therefore, at the time of this writing in IOS-XE 17.7.1, DACLs are not supported on the Catalyst 9800. There is a good chance that support will come in future releases though, so keep an eye on the document titled *List of IOS-XE Wireless Features per Release* for it or release notes of each version.

URL Filters (a.k.a. DNS-Based ACLs)

Allowing access based on IP addresses is fine when covering your internal network, but what if you want to allow or restrict access to public resources on the Internet? IP addresses can change at any time without you knowing, and maintaining an IP-based access list covering Internet resources is just not practical. This is where URL filters come in. They work like ACLs in the sense that they permit or deny access, but their statements include URLs instead of IP addresses.

URL filters don't require you to have DNS configured on your wireless controller because they snoop the wireless client DNS traffic in real time. When traffic is centrally switched through the WLC, it is the controller that does this DNS snooping, whereas the AP can do the job for fabric or locally switched WLANs.

Before using an app on a smartphone or before visiting a website on a browser, the client device typically does a DNS lookup to get an IP address resolution of the app or website domain and then sends traffic to the specific IP addresses returned. Those IPs can be load-balanced and can change based on which DNS server you ask or your geographic location. When the wireless client is dynamically sending this DNS request, the WLC or the AP snoops and takes a peek at the DNS response. It looks at the first A record in the DNS reply and notes the IP address linked to it if the URL is a match to anything configured in the URL filter on the controller. It then adds this IP address dynamically in an IP cache table for this wireless client. It is therefore some kind of IP-based access list that is specific to clients and changing in real time based on the DNS requests made by the clients.

There are two types of URL filters: standard and enhanced. Standard URL filters can be applied before client authentication (preauth) or after a successful client authentication (postauth). Preauth filters are extremely useful in the case of external web authentication to allow access to the external login page, as well as potentially some internal websites before authentication takes place. Postauth, they can work to block specific websites or allow only very specific Internet websites while all the rest is blocked by default, but this type of URL filtering postauth is better handled by using Cisco DNS Layer Security (formerly known as Umbrella) for a lot more flexibility. The standard URL filters apply the same action (permit or deny) for the whole list of URLs: it's either all permit or all deny. Standard URL filters work on both local mode APs and FlexConnect APs.

Enhanced URL filters allow specification of a different action (deny or permit) for each URL inside the list and have per-URL hit counters (see Figure 5-5). They are supported only on FlexConnect APs in local switching (or fabric APs).

URL	Preference	Action	Validity	Invalidated URL
<input type="checkbox"/> meraki.com	2	PERMIT	VALID	0
<input type="checkbox"/> *.cisco.com	1	DENY	VALID	0

Figure 5-5 URL filter configuration

In both types of URL filters, you are allowed to use a wildcard subdomain such as *.cisco.com. URL filters are standalone but always applied along with an IP-based ACL. As of release 17.6, a maximum of 20 URLs is supported in a given URL filter. Considering that one URL can resolve to multiple IP addresses, it is interesting to note that only up to 40 resolved IP addresses can be tracked for each client. Another important limitation to note is that only DNS A records are tracked by URL filters. The WLCs or APs do not track the resolved IP address of a URL if the DNS answer uses a CNAME alias record.

Certificates and Trustpoints

Explaining public key infrastructure (PKI) in detail is beyond the scope of this book. However, certificates are a very important part of any network device and especially for the Catalyst 9800. Certificate-based authentication is a method to identify a user, device, or machine before it can be granted access to a network. A wireless network, comprising a wireless LAN controller (hereafter referred to as WLC), access points (APs), and clients, commonly uses certificate-based authentication to validate the identities of peer devices when participating in services such as AP join, device management access, and web authentication. Each service can use different sets of client and server certificates.

But how do devices get their digital identities?

To begin with, each participating device (controller, access point, or client) has its own device certificate and a certificate authority (CA) certificate that validates its authenticity. A closer look at the certificates available on the Catalyst 9800 controller shows the following types:

- **Cisco-installed manufacturer installed certificate (MIC):** On physical appliances (Catalyst 9800-40, Catalyst 9800-80, Catalyst 9800-L), these are, by default, factory installed and widely known as the Cisco-installed MIC or Secure Unique Device Identifier (SUDI) device certificate. In addition, controllers and access points have a Cisco manufacturing certificate authority (CA) certificate that is used to sign and validate device certificates.
- **Wireless LAN controller self-signed certificate for virtual controller:** The Catalyst 9800-CL (the virtual instance of the controller) does not come with any manufacturing certificate. In the absence of an identity certificate, it relies on the self-signed certificate that has to be generated by the Day 0 wizard or manually using a script

and validated by the local Cisco IOS certificate authority (which is a self-signed local CA and is not the same as the manufacturing Cisco CA certificate). This acts as the Catalyst 9800-CL's local identity certificate and is used for AP joins, mobility connections, and Network Mobility Services Protocol-Connected Mobile Experience (NMSP-CMX) connections.

- **IOS-XE device self-signed certificate:** The default self-signed certificate is auto-generated during the controller's initial startup if any HTTPS, SSH, or NETCONF service is configured on the controller.

The listed default certificates provide an easy and out-of-the-box method of early trust between peer devices. However, if you want to provide better security, you can consider using third-party validated certificates, including locally significant certificates (LSCs).

Third-party certificates require a PKI framework that enables encryption of public keys and digital certificates. Along with different authentication protocols, the PKI model works with certificate authorities, root certificates, and asymmetric key encryption to ensure that the digital certificates are securely exchanged over encrypted tunnels during a client and server exchange.

On Catalyst 9800 controllers, these digital certificates are configured and held in containers called *trustpoints* and used when the devices initiate a secure communication with other network devices or network clients. A trustpoint is one of the most important configuration entities for a PKI client. A trustpoint includes the identity certificate of the CA that signed the device certificate, CA-specific trustpoint configuration parameters, and an association with an enrolled identity certificate.

Trustpoints provide a mapping between the identity certificate and the application or service that needs the certificate. For example, for the SSL/HTTPS server functionality, the **ip http secure-trustpoint <trustpoint name>** command tells the controller what identity certificate to present to an SSL client. Depending on your requirement, you can configure many trustpoints.

A Case for Trustpoints

Identity validation using certificates spans across a range of functions and protocols in the Catalyst 9800 wireless environment. Certificates are primarily used for authentication when an access point joins the controller using CAPWAP with DTLS, for web administration and web authentication using HTTP with TLS, and for local EAP authentication. Certificates are also used when the controller communicates with Cisco Connected Mobile Experience (CMX), Cisco Digital Network Architecture Center (DNA Center), and Digital Network Architecture Spaces (DNA Spaces). Some of these exchanges require additional configuration, whereas others do not require any action from your side.

How to Add a Certificate on the Controller

To add a certificate on the controller, there are basically two things you can do: either start a certificate request on the WLC (as shown in Figure 5-6) or import a ready-for-use certificate to the WLC (as shown in Figure 5-7), get it signed, and install the resulting certificate. You

can do various activities outside of the Catalyst 9800 device to end up with the same result. A certificate has to contain a private key and be linked to a given certificate authority.

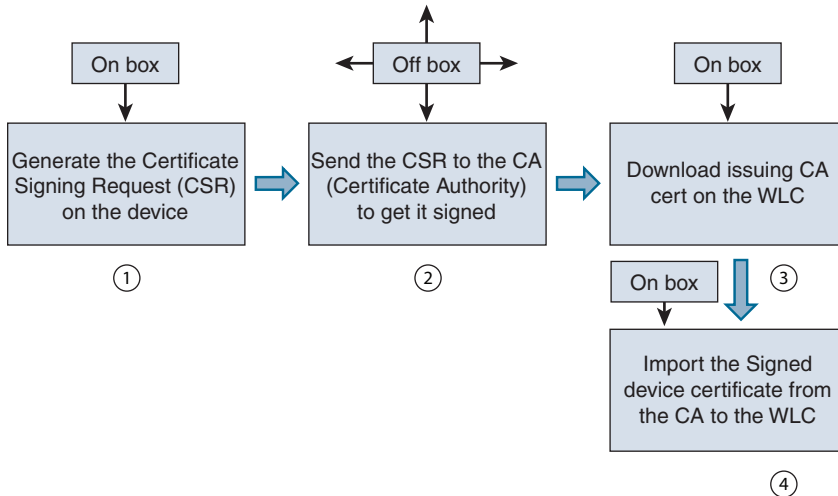


Figure 5-6 Adding a certificate by generating the question on the controller

Generating the certificate on the WLC itself is the most secure because the private key never leaves the device. However, it does not allow for configuring the SAN field at this time. The workflow is shown here and can mostly happen in the **Configuration > Security > PKI Management > Add Certificate** page of the WebUI:

- Step 1.** Go to the **Key Pair Generation** tab of the PKI Management page, as depicted in Figure 5-7. Click **Add**, enter a key name, choose **RSA**, choose a Modulus of 4096, make sure it's exportable, and then click **Generate**.

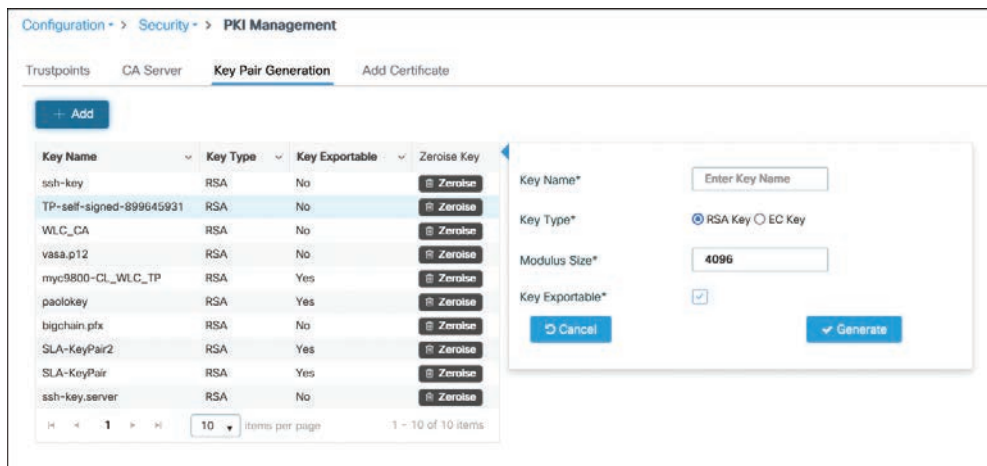


Figure 5-7 Creating a key pair for use with a certificate

- Step 2.** To generate the CSR on the WLC, go to **Generate Certificate Signing Request**, as shown in Figure 5-8, and enter the desired fields that will appear on your certificate after it is signed.

The screenshot shows the Cisco WebUI interface for PKI Management. The breadcrumb trail is Configuration > Security > PKI Management. The main navigation tabs are Trustpoints, CA Server, Key Pair Generation, and Add Certificate. The 'Add Certificate' tab is active, showing a list of options: Generate CSR, Authenticate Root CA, Import Device Certificate, and Import PKCS12 Certificate. The 'Generate Certificate Signing Request' section is expanded, displaying a form with the following fields:

- Certificate Name*: Enter Certificate Name
- Key Name*: Search or Select (with a dropdown arrow and a plus icon)
- Country Code: [Text Input]
- State: [Text Input]
- Location: [Text Input]
- Organizational Unit: [Text Input]
- Organisation: [Text Input]
- Domain Name: [Text Input]

A blue 'Generate' button is located at the bottom right of the form.

Figure 5-8 Generate a certificate signing request from the WebUI

- Step 3.** Get the certificate signed by the certificate authority of your choice. The details of this step vary greatly depending on which certificate authority you choose. Many publicly trusted CAs have a website where you can easily get your certificate signed.
- Step 4.** Go to the **Authenticate Root CA** section of the **PKI Management** page, as depicted in Figure 5-9. Enter a trustpoint name (choose the name you want to use for your certificate) and paste the content of the PEM file.
- Step 5.** Go to the **Import Device Certificate** section of the **PKI Management** page, as depicted in Figure 5-10. Enter the same trustpoint name as in step 3 and paste the content of the PEM file of the device signed certificate you received from your CA.

Configuration - > Security - > PKI Management

Trustpoints CA Server Key Pair Generation **Add Certificate**

- **Generate CSR**
 - Input certificate attributes and send generated CSR to CA
- **Authenticate Root CA**
 - Copy and paste the root certificate of CA received in .pem format that signed the CSR
- **Import Device Certificate**
 - Copy and paste the certificate signed by the CA
- **Import PKCS12 Certificate**
 - Signed certificate can be received in pkcs12 format from the CA
 - Use this section to load the signed certificate directly

> Generate Certificate Signing Request

▼ **Authenticate Root CA**

Trustpoint* ▼

Root CA Certificate (.pem)*

Figure 5-9 *Authenticating the CA that issued your device certificate*

- **Generate CSR**
 - Input certificate attributes and send generated CSR to CA
- **Authenticate Root CA**
 - Copy and paste the root certificate of CA received in .pem format that signed the CSR
- **Import Device Certificate**
 - Copy and paste the certificate signed by the CA
- **Import PKCS12 Certificate**
 - Signed certificate can be received in pkcs12 format from the CA
 - Use this section to load the signed certificate directly

> Generate Certificate Signing Request

> Authenticate Root CA

▼ **Import Device Certificate**

Trustpoint* ▼

Trustpoint label is required

Signed Certificate (.pem)*

Figure 5-10 *Adding the device signed certificate received from the CA*

Generating the CSR outside of the 9800 is a possibility (depicted in Figure 5-11). In this case, you typically have to bundle your private key along with your certificate in a PKCS12 file format and download everything at one time to the WLC:

1. The first step would be to use a tool such as OpenSSL to generate your CSR as well as a private key. Some public CAs also offer a web page where you can easily generate a key and CSR for them to sign as well.
2. Have your CA sign your certificate.

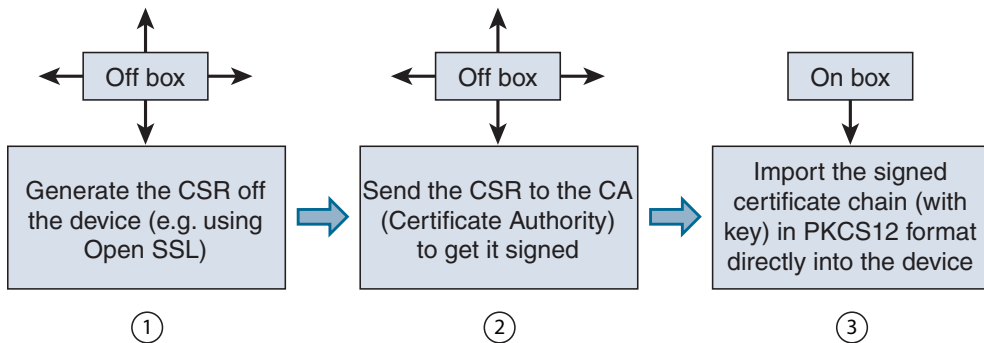


Figure 5-11 Adding a certificate when the private key was generated outside of the 9800

3. Obtain the PKCS12 formatted file containing the certificate chain (your device certificate and the CA chain), along with your private key. If your CA does not provide this information, you can combine the pieces you received together with OpenSSL to obtain the PKCS12 file. As depicted in Figure 5-12, go to the **Import PKCS12 Certificate** section of the **PKI Management** page in the WebUI and enter the file location and private key password.

When you're importing PKCS12 files, it is possible to import a complete chain if your PKCS12 contains a chain of certificates up to a root CA (if the CA that signed your device certificate is not a root CA but an intermediate CA, for example). Be aware that the Catalyst 9800 controller does not send the whole chain of certificates when a client connects to the web interface, and therefore, the client should have most of the chain imported rather than only the top root CA. The **9800 Web UI PKI Management** page contains a guided workflow for certificate import to simplify all these steps. The exact workflow is explained in great detail in the guide titled *Configuring Trustpoints on Cisco Catalyst 9800 Series Controllers*. Commands and some tips are also covered in *Generate CSR for Third-Party Certificates and Download Chained Certificates to Catalyst 9800 Wireless Controllers*.

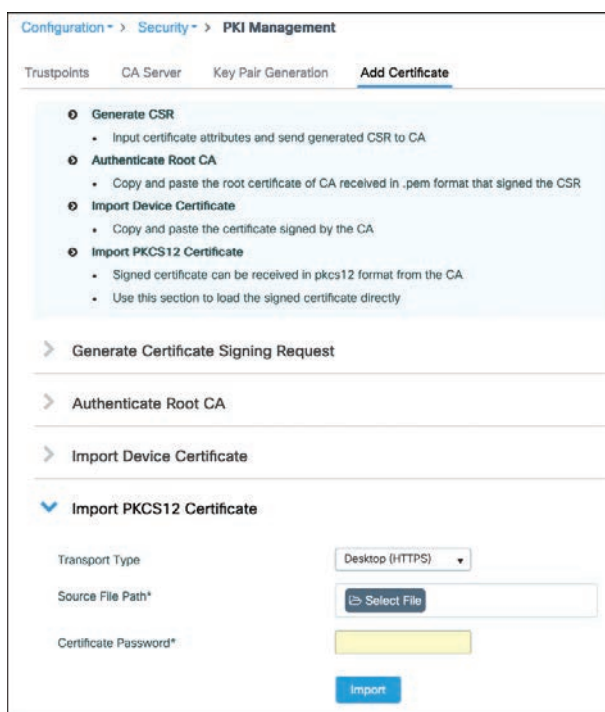


Figure 5-12 Adding a certificate chain in PKCS12 format to the WLC

AAA

Network authentication, authorization, and accounting (AAA) provide the means of getting answers to the identity of a person or device requesting access, what resources are being accessed, if this person has specific rights toward that resource, and logging activities of the person or device during a session. The term AAA is used in a relaxed manner to depict any part of the access control process, but each letter stands for a specific phase of this flow:

- **Authentication:** This term is the most used, even to depict the whole flow. Technically, the authentication phase consists of verifying the identity of a device or a person. This is done by requesting user credentials, which can be a username and password pair or even something else such as a certificate.
- **Authorization:** This phase consists of checking the privilege or authorization level the client has. The client may ask for permission to access every single resource uniquely, or this can happen in a one-time operation where the authentication server returns the list of privileges and accesses that the client can benefit from, and it's then up to the network access server to apply those privilege levels to the user.

- **Accounting:** This often-optional phase happens after the authentication is complete and allows you to track the activity of the user or device on the network. It is also useful to know when the session ends, that is, when the device is no longer connected to the network.

RADIUS

The Remote Authentication Dial-In User Service (RADIUS) is a protocol used to control network access. It is quite old and, as its name implies, dates from the dial-up era, which explains why some of the terms and attributes sound quite outdated and unrelated to wireless, but it still does the job and is the main protocol used for wireless enterprise class security. RADIUS operates in a client/server model where the client is the wireless controller and the server is the RADIUS authentication server (ISE in a Cisco environment). The exact terms are

- **Network access server (NAS) or network access device (NAD):** This device is responsible for passing the user information to the RADIUS server and acting on the response that is returned.
- **RADIUS server:** This device is responsible for processing the access request and returning all required parameters for the NAS to provide access to the user.

RADIUS works over UDP on port 1812 for authentication and authorization and on UDP port 1813 for accounting.

There are four authentication packet types:

- **Access-request:** The NAS sends this type of packet to provide the server with information required to proceed with the authentication. It is basically used for any RADIUS packet sent to the RADIUS server.
- **Access-challenge:** This is the type used by the RADIUS server to send data to the NAS to proceed with the authentication. You can guess that these first two types are repeated a certain number of times (depending on the authentication type and details), and the value is in the attributes and data they carry rather than the message type. The request and challenge RADIUS packets carry the EAP authentication frame in one of the attributes (called EAP payload).
- **Access-accept:** The RADIUS server sends this packet type to tell the NAS the authentication has been successful. The access-accept contains all the necessary attribute fields to provide the authorization details to the NAS. This is a specificity of the RADIUS protocol where authorization is not a separate phase but is embedded as a one-time operation on the final authentication packet.

- **Access-reject:** The RADIUS server uses this packet type to notify the NAS that the authentication failed. It rarely contains attribute fields other than the username in the case of 802.1X.

There are two extra packet types for accounting:

- **Accounting-request:** The NAS sends these packets to deliver information about the session to the server.
- **Accounting-response:** These packets work as a kind of acknowledgment that the server has received and processed the accounting request successfully.

RADIUS Attributes

We have already hinted at the fact that the authentication server can return attributes that involve privilege levels and specific information about accessing or not accessing specific resources. RADIUS attribute-value pairs (AVPs), often called *AV-pairs*, are a major part of the success of RADIUS as a protocol. They are used to exchange information between the NAS device and the authentication server. They can provide extra information about the request being made and provide various parameters for the network device to enforce. They contain a type, a length, and a value that defines the attribute format and how to read it (shown in Figure 5-13) and are extremely flexible, which is the reason RADIUS was able to stay relevant and survive so many technology changes. Because the field is an octet, up to 255 attributes can be used. Some of them are predefined and well known, whereas others leave room for vendors to communicate vendor-specific information by encapsulating their own extended attributes inside it.

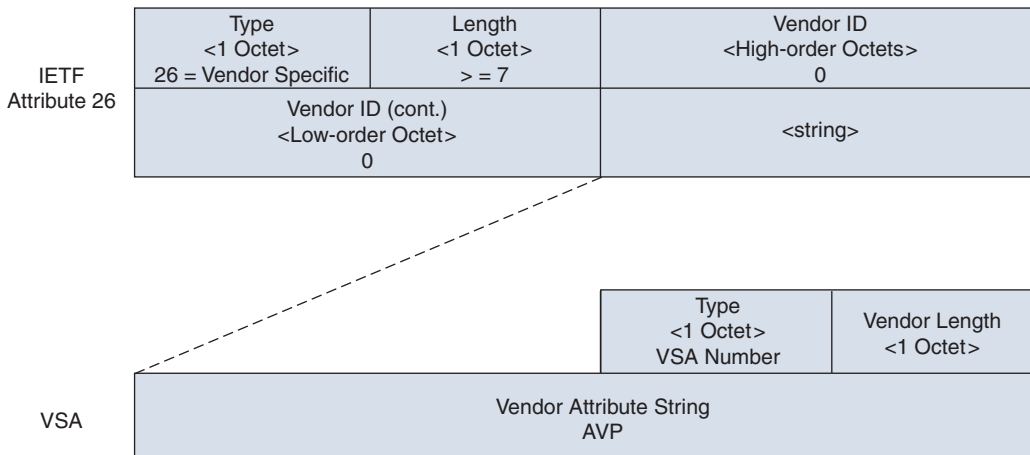


Figure 5-13 RADIUS AV-pair format

RADIUS Sequence Example

Now that we've covered the protocol, let's examine, at a high level, the operational sequence that shows how all the components work together during an 802.1X authentication. The EAP exchange is illustrated in Figure 5-14.

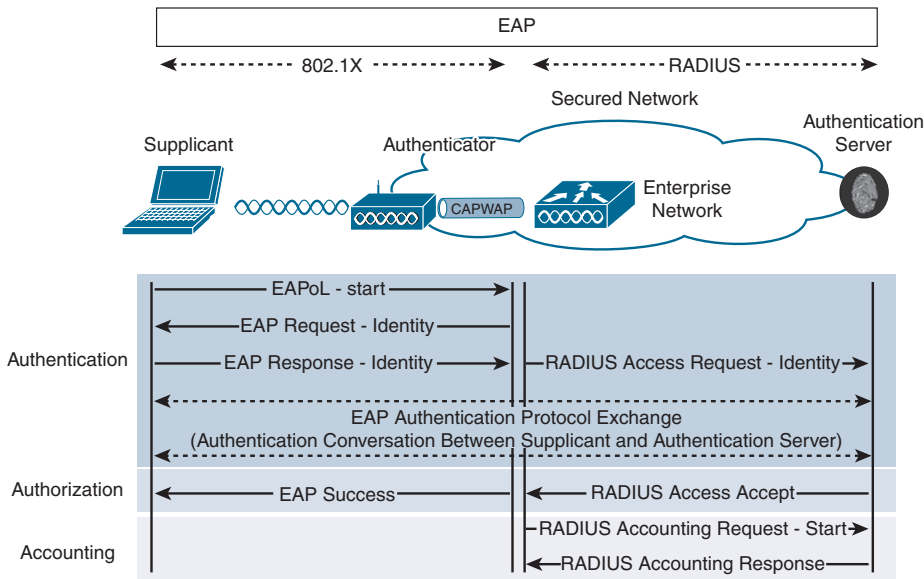


Figure 5-14 Workflow of an EAP authentication between a supplicant and authentication server

1. The supplicant can initiate the authentication process by sending an EAPoL-Start message to the authenticator, but this is optional. Both switches and access points know when a new client is connecting and can be the initiator of the authentication.
2. The authenticator (the WLC or the AP) then sends an EAP identity request frame asking for the client's identity. The WLC is the authenticator from the point of view of initiating the RADIUS traffic in the case of central authentication, but the AP can be the authenticator in the case of FlexConnect with local authentication. When the WLC is the authenticator, it is the one originating the EAP messages sent to the client, and AP is just transmitting those messages over the air. That is part of the split MAC model.
3. The supplicant submits its identity for the authenticator to forward to the authentication server. First of all, this identity does not have to be the real username (and never a certificate). Because many EAP methods use a TLS tunnel, the identity of the first authentication method is not so useful, and many clients send **anonymous** as the identity. This is called the *outer identity* (in the case of a tunnel EAP method like PEAP). This outer identity is the only thing the authenticator (that is, the WLC) is able to use to refer to the user. It is not able to read the inner identity sent inside the encrypted tunnel because it does not have the keys to decrypt this tunnel and is merely forwarding data between the supplicant and the authentication server. The WLC can sometimes learn the supplicant

identity if the authentication server reveals it in RADIUS attributes like it sometimes does by appending the username RADIUS attribute with the access-accept. This identity is carried to the authentication server through RADIUS encapsulation, and the authenticator appends a number of RADIUS AVPs to give extra detail about the authentication, such as the AP MAC address (as Called-station-id), the client MAC address (as calling-station-id), the SSID details, and so on.

4. The rest of the authentication is defined by the specific EAP method in use. The choice of the method is determined by the authentication server proposing EAP methods one by one to the client until the client acknowledges the use of a method. So, the RADIUS server suggests a method, and the supplicant is free to accept or request another one (without knowing what will be offered next).
5. The EAP authentication goes on, and the authenticator forwards each frame/packet back and forth between the supplicant and the authentication server. When the authentication is successful, the server sends a RADIUS access-accept message that may contain extra RADIUS AVPs to define the type of authorization granted (a privilege level, a VLAN assigned, an access list to apply, a session timeout, and so on). The authenticator relays this as an EAP success message.
6. On the supplicant side, WPA keys typically are exchanged then. On the infrastructure side, the authenticator starts sending accounting packets with informative details about the session. Depending on the configuration, the authenticator can keep sending regular accounting packets during the session (which is useful for tracking bandwidth consumption).

RADIUS Change of Authorization (CoA)

In a standard RADIUS interaction, the network device is the one initiating all the communications. Those communications are either a new authentication request or some accounting data. There is no means for the authentication server to spontaneously send data to an NAS or to terminate a current session (that is, a client that has been authorized before to access the network). RFC 5176 adds this flexibility to the RADIUS protocol by defining enhancements that allow the authentication server to dynamically modify the authorization of a user session via Change of Authorization (CoA) messages.

CoA messages are transported over UDP using 3799 as the destination port; however, Cisco devices use port 1700 for this task. The NAS replies with a CoA acknowledgment if it can successfully change the authorization for the user session or a nonacknowledgment if it is unsuccessful. Here are a few examples of what the authentication server can do:

- Terminating a user session
- Requesting the NAS to reauthenticate the user/device from scratch

Practical examples vary a lot:

- The ISE can decide to reauthenticate a client after new profiling data is made available (the ISE obtains new DHCP packets or intercepts new HTTP packets that refine the profiling information), and this potentially results in a different policy applied for that client.

- The ISE can decide to terminate a specific session if a security threat is automatically or manually detected.
- The ISE uses CoA in regular workflows such as central web authentication (this topic is covered in the section titled, “*Central Web Authentication*”).

To do so, the authentication server has to attach several attributes to the CoA request, such as the accounting session ID, the audit session ID (a Cisco vendor-specific attribute), and a calling session ID, which is basically the host MAC address. With this information, the NAS is able to execute on the authentication server request.

CoA is not enabled by default on many Cisco NAS or WLCs, and you need to specifically configure that you will tolerate the authentication server sending CoA messages that affect user sessions.

RADIUS Configuration and Load Balancing

Authentication of the wireless client involves defining a RADIUS server, optionally creating a server group, defining AAA methods that use the server you created, and calling these methods from the WLAN configuration, for example (or someplace else if talking about a form of authentication other than wireless client authentication).

Configuring RADIUS Servers

To configure a RADIUS server, you can use the WebUI in **Configuration > Security > AAA**, as depicted in Figure 5-15. Under the CLI, you would enter the following:

```
radius server <servername>
  address ipv4 8.60.0.252 auth-port 1812 acct-port 1813
  key <key>
```

You can choose the server name, and it is only for reference. The address can be configured with IPv4 or IPv6 or a hostname (that requires configuring a DNS server for the 9800 to use). It is advised you enter the key in clear text. The encrypted options are mostly used in CLI when you are pasting the key that is already encrypted from another configuration file. The timeout defines the number of seconds the 9800 will wait before declaring the RADIUS packet lost (and send retries), and the retry count is the number of times a RADIUS request will be retried until declared over.

Configuring RADIUS Server Groups

If you are planning to use several RADIUS servers for redundancy or load balancing, creating a server group makes things easier. It basically consists of naming a specific list of servers. The RADIUS servers need to be created on the Catalyst 9800 configuration first before you can assign them to a server group. It is easily defined in the WebUI under **Configuration > Security > AAA**, where you have a **Server Groups** tab, as shown in Figure 5-16, or in the CLI:

```
Aaa group server radius <group name>
  Server name <server1>
  Server name <server2>
```

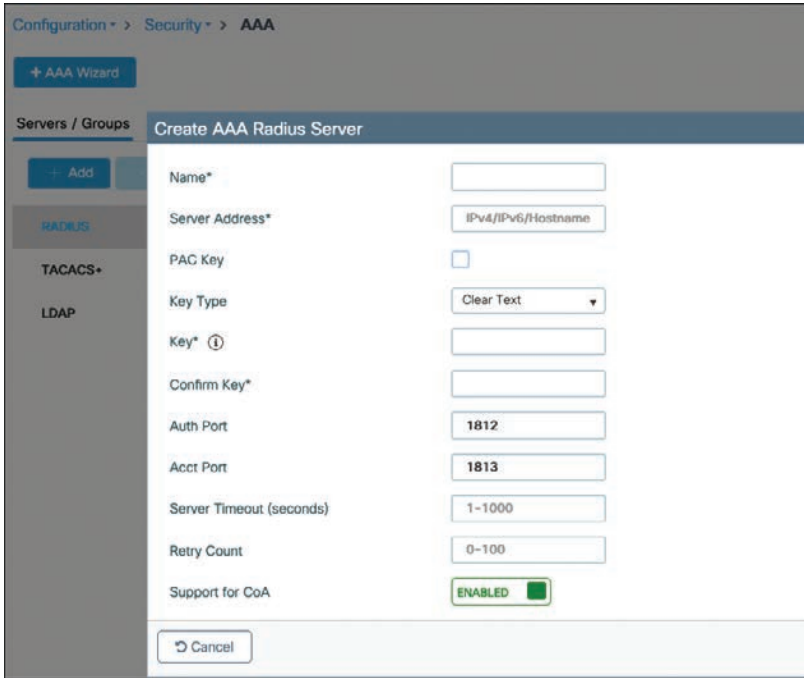


Figure 5-15 RADIUS server creation on the C9800 WebUI

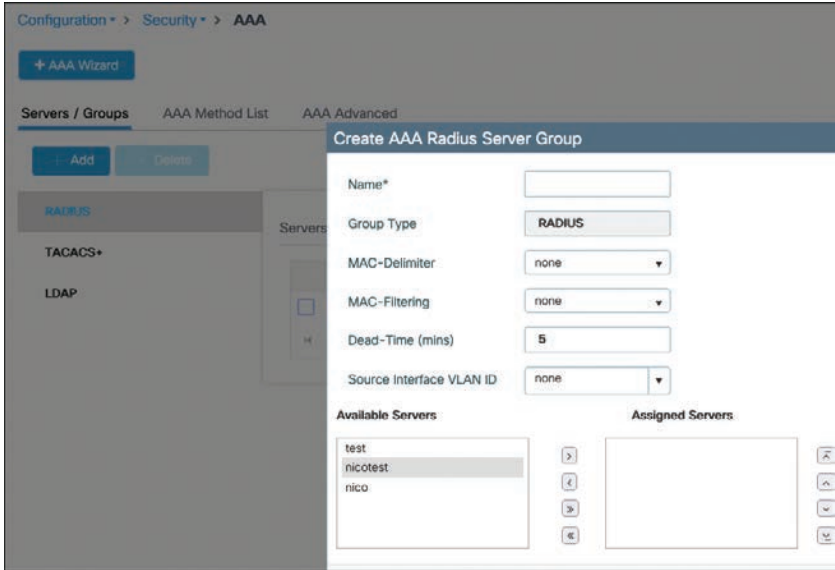


Figure 5-16 RADIUS server group configuration

The server group is also a specific place to configure extra RADIUS settings:

- In the case of MAC filtering authentication, for example, you can define what the MAC address delimiter will be (hyphen, colon, and so on) because different RADIUS servers might expect different formats.
- Similarly, the MAC filtering attribute allows you to define what value will be used in the password field. Some RADIUS servers expect the MAC address to be repeated there, whereas others expect the RADIUS shared secret to be used.
- The dead-time is a critical setting to configure how long a RADIUS server will be dead before it can be attempted again. If you don't configure this setting, the RADIUS server is instantly marked back alive after being declared dead, and you basically never fail over. Although newer software versions configure 5 minutes by default, you may still have unconfigured dead time if you are upgrading from an earlier release and kept your configuration. A RADIUS server is considered dead when it does not reply after a configurable number of retries and a configurable timeout to RADIUS requests (configuration illustrated in Figure 5-15) sent by the C9800.
- The source VLAN ID allows you to overwrite the typical interface. The 9800 uses its routing table to determine the interface or SVI to use to send RADIUS traffic to the server. You can override this globally with the `ip radius source-interface <int>` command or on a per AAA server group basis with this source VLAN ID. This is similar to the **RADIUS interface overwrite** feature in AireOS controllers except that, because the 9800 does not have dynamic interface, this setting is on a per group basis and allows you to choose a specific interface to source RADIUS traffic for each group.

The order in which you define the servers in the group is the order in which they are used. It is not possible to change this order after creation unless you delete and re-create the server group with the servers in another order.

Any setting or timer that is set under the specific RADIUS server configuration supersedes the same setting configured under the AAA server group, if present.

The global setting `radius-server dead-criteria time <seconds> tries <number>` defines the criteria to declare a RADIUS server dead. Basically, when the 9800 does not get a reply to a RADIUS request after the configured server timeout, it retries sending that packet the number of times configured in the retry settings. If there is still no reply, that specific RADIUS request is dropped on the 9800, but the RADIUS server is not marked dead until the dead-criteria is met (in both time and number of tries of different RADIUS requests). The dead time configured in the RADIUS server group supersedes the globally configured setting if there is one. More specific configuration, in general, supersedes more global configuration if they represent the same setting.

RADIUS Server Fallback

We have explained the conditions under which the controller moves from the primary RADIUS server in the list to the next one and so on. Let's bring a bit of clarity on when the controller falls back to using the primary RADIUS server.

The equivalent of the passive fallback setting in AireOS is achieved by using the **dead-time** command previously mentioned. This means that when you enter **radius-server deadtime <minutes>**, this command defines the number of minutes for which the RADIUS server is considered dead, and therefore, the WLC uses the next RADIUS servers in the list. After this time has passed, the WLC automatically starts using the primary RADIUS server again with the next authentication that it has to perform.

If you add the command **automate-tester username <dummy username> probe-on** to the RADIUS server configuration section, test RADIUS authentications (using the dummy username you entered) are sent to the RADIUS server only when it is marked dead to see if it is back alive. If you configure **automate-tester username <dummy user> idle-time <minutes>**, the controller sends the test authentication every “idle-time” period even when the server is alive (which can be useful to detect whether it goes dead when there are no authentications ongoing). The automate tester considers the server to be alive if it receives any reply from the server; the tester does not need to receive a successful authentication result (especially because no password was configured). Just make sure the RADIUS server does not ignore such a plaintext PAP authentication, which can sometimes be the case with a default configuration.

Somewhat similarly, but in a purely manual fashion, you can test an authentication against a RADIUS server or group with the exec mode command **test aaa group radius <user> <password> new-code**. This command also sends a PAP authentication request to the RADIUS server list with the given username and password.

RADIUS Load Balancing

Load balancing helps in handling authentications during a server failure but can also be used to balance the load between several servers and make sure no RADIUS servers get overwhelmed with a storm of authentication requests. By default, configuring several servers in a radius server group allows for redundancy, but only one server is used at a time; the controller uses the next RADIUS only if the current server does not respond and gets declared dead. You can also configure several RADIUS server groups in your AAA authentication methods, but the effect is the same: the second RADIUS server group is used only if the first one is completely dead.

If you add the **load-balance method least-outstanding** command to the RADIUS server group (or globally with **radius-server load-balance method least-outstanding**), the WLC rotates RADIUS servers even when they are alive. For a new batch of RADIUS requests, it checks which server has the least number of transactions and uses it.

RADIUS Accounting

A similar configuration can be done to add a RADIUS accounting server. Periodic interim accounting records can be sent to the accounting server (otherwise, a START is sent when the session starts and a STOP when the client is deleted) if you enable **aaa accounting update [newinfo] [periodic] number** in global configuration mode.

AAA Methods

The command **aaa new-model** is required to configure anything pertaining to AAA that is described here, and it unlocks many other commands. You can create various types of AAA methods from there:

- **Authentication dot1x:** The **aaa authentication dot1x <method name> <aaa server group>** command allows you to define the authentication server(s) for 802.1X authentication for both wireless clients or mesh APs (if applicable).
- **Authentication login:** The **aaa authentication login <method> <server group>** command allows you to define methods for logging in the device (GUI or CLI).
- **Authorization network:** The **aaa authorization network <method name> <server group>** command is a kind of multipurpose method. Its main use is to allow the override of the AAA attribute, that is, to accept and apply RADIUS or TACACS attributes received from the server during an authorization result. For example, it is required for the controller to apply a dynamic RADIUS VLAN assignment or session timeout or ACL. This method type is also used for MAC address authorization (client or APs).
- **Authorization credential-download:** The **aaa authorization credential-download <method name> <server group>** command allows you to define the server(s) that will be used for verifying credentials of an authentication happening locally. It is used in the case of the local EAP (when the 9800 acts as RADIUS server) or LDAP authentication.
- **Authorization exec:** The **aaa authorization exec <name> <server group>** command is used to determine which server is used to authorize users in starting an EXEC shell on the device.
- **Accounting identity:** The **aaa accounting identity <name> start-stop <aaa server group>** command allows you to define RADIUS accounting servers that will receive session statistics for network users.

When doing a local authentication (as in the case of local EAP) on the 9800, you need to define which methods are used with

```
9800 (config) #aaa local authentication <method> authorization <method>
```

Only named AAA methods have been mentioned so far. Every method also has the “default” keyword that can replace the name. A named method has no effect until you call it from a specific part of the configuration. The default method is automatically applied for the said use case. For example, if you configure the **aaa authentication login default** method, you impact the way users authenticate when accessing the 9800 via the CLI (unless you configured a named method in the VTY line configuration already). They are sometimes required (for example, the NETCONF authentication cannot be configured with named methods at the time of this writing and will always use the default one), but in general you should try not to use them unless you know what you are doing exactly.

AAA methods can point to RADIUS server groups or TACACS server groups (except for dot1x) but can also use the “local” destination. An important point is that if you select local first and then add a server group, the method first checks in the local database and

consults the external server if the user is not found locally. However, the opposite is not true: when defining a server group first and then local as fallback, the controller checks only in the local database if the server group is completely down (and not if the user is not found in the server group because the controller has no means of knowing why an authentication has failed or if the user was not found or has an invalid password).

When you're using a local database of users, it is possible to define a set of AAA attributes for them. Defining these attributes could be useful if you are doing local EAP—that is, local authentication of 802.1X users on the WLC, and you want to return a specific VLAN or ACL. It can also be leveraged when you're doing local MAC address authentication. You can define an AAA attribute list this way:

```
9800 (config) #Aaa attribute list user1_list
9800 (config-attr-list) #Attribute type ssid "test-ssid"
```

You can then map this list to the user in the local database:

```
9800 (config) #User-name user1
9800 (config-user-name) #Aaa attribute list user1_list
```

Or if you have MAC address users, you can use the following:

```
9800 (config) #User-name <MAC> mac aaa attribute list <attribute-list-name>
9800 (config-user-name) #Aaa attribute list user1_list
```

You need to set the **aaa authorization network** method to point to local to accept these attributes. Don't forget to enable AAA override on the policy profile as well.

Possible and popular attributes that you can use are

- **SSID:** You can restrict the SSID that the user can connect to.
- **VLAN:** You can assign a VLAN depending on the username.
- **QoS:** You can assign QoS policies specifically for the user.
- **Session-timeout:** You can override the WLAN session timeout on a per-user basis.
- **ACL:** This attribute helps assigns a predefined ACL to the user.

Local EAP

Often confusingly shortened as LEAP (which is a very old and insecure EAP type), *local EAP* refers to the capability of the C9800 to act as a RADIUS server for wireless clients only. It supports PEAP, EAP-TLS, and EAP-FAST (let's not even mention *LEAP* anymore, which still shows up as an option in the configuration). It is meant for use in limited-scale deployments or as backup when the RADIUS server is not reachable (in branches where you have a controller, for example). Local EAP is explained in detail in a configuration example on Cisco.com and referenced at the end of this chapter, but let's still look at the basic concepts around it.

Local EAP requires you to

- Configure a local user, or “network” type of user, with **username <username> password 0 <password>**, for example.
- Configure a dot1x authentication method pointing locally.
- Configure a credential download authorization method pointing locally.
- Define the methods name under **aaa local authentication**.
- Configure an EAP profile where you define the EAP types supported as well as point to a trustpoint for the 9800 to use a specific certificate.
- Define the local EAP authentication profile in the WLAN settings.

TACACS+

Terminal Access Controller Access Control System (TACACS) is a protocol from 1984 that got enhanced by Cisco under the name TACACS+ in 1993. TACACS+ has core differences with RADIUS, despite the fact both are authentication protocols covering all three As of the AAA acronym. Here are a few of the core differences of TACACS+ compared to RADIUS:

- TACACS+ uses TCP instead of UDP. TCP is reliable and offers a connection-oriented transaction system. TCP covers the retransmission problems, whereas with RADIUS you need to specify timers and handle retransmissions in the application layer itself. TCP also allows you to know if the server is still dead or simply ignoring the actual authentication request for some reason.
- RADIUS only encrypts the password in the access-request packet while all the rest of the RADIUS packet is unencrypted. TACACS+ encrypts the entire body of the packet but leaves a standard TACACS+ header. This protects the username as well as other network attributes present in the packet.
- RADIUS combines the authorization phase with the authentication phase by returning all the required privileges and attributes in the access-accept payload. TACACS+ separates all the phases, and the authentication phase is fully separate and only identifies the user without covering access privileges. The NAS can then immediately do an authorization request to request access to a specific resource. The TACACS+ server can reply to this authorization request without reauthenticating the client. This allows you to force the NAS to request authorization for every specific resource and therefore allows much more granularity.

The separation of the authorization phase, combined with the increased number of packets of the whole process (as seen in Figure 5-17), makes TACACS+ a protocol better suited for network device management than end user authentication (like 802.1X).

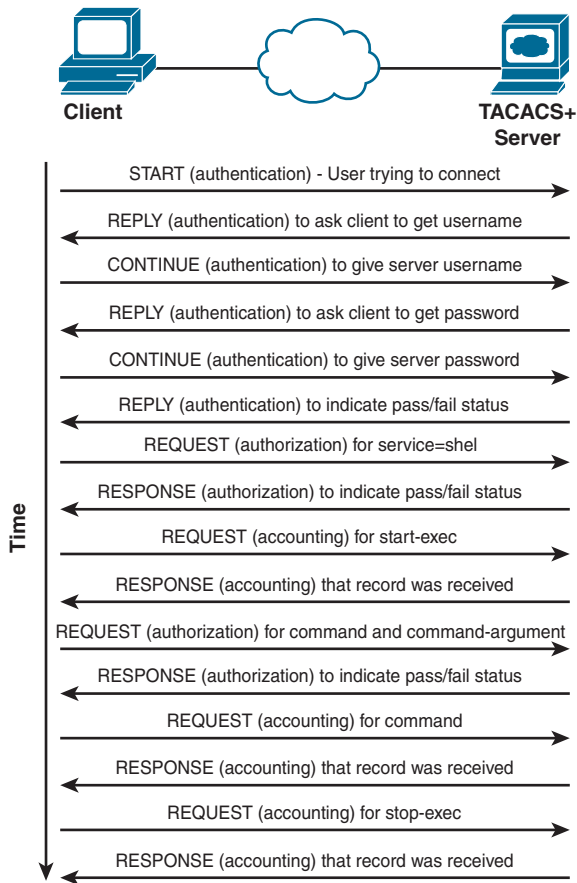


Figure 5-17 TACACS workflow

As a matter of fact, TACACS+ does not support encapsulating an 802.1X authentication, which means RADIUS is your one and only option to transport 802.1X on your wireless network. To secure and control access to your network devices, such as the wireless controller, on the other hand, you have the choice between RADIUS and TACACS+. Using RADIUS would seem like a logical option because this means installing only one type of authentication server (although ISE does both RADIUS and TACACS+) and using a single protocol in your network. However, the drawback of using RADIUS for network device management is that you can return only a one-time authorization set of attributes, which typically are an IOS privilege level, and possibly some specific attributes (like defining the user as a lobby admin and not a regular admin user). TACACS+, because it allows you to force an authorization for every specific access, allows you to define granular CLIs or web pages that can be accessed. The NAS then asks for authorization for every single command typed on the CLI, and the authentication server accepts or denies each of them. TACACS+ command authorization obviously can add latency if you are going to paste many commands, but it allows for very granular security where specific users are allowed only a very specific subset of commands.

LDAP

Lightweight Directory Access Protocol has remained popular over the years, although it is not a very secure end-to-end authentication solution. LDAP is a network protocol for accessing user directories. It does not take care of the authorization part of things (no network access level defined or custom attributes returned) but simply validates user credentials. It can be used as a back-end user database for web authentication or for local EAP. There are many limitations around LDAP (limited use cases, insecurity of the passwords, impossibility to use it for PEAP-MSCHAPV2 authentication, and so on), so this book does not cover it. There is a document about it on cisco.com if you seek to learn more; check the “References” section at the end of this chapter for the link.

As you have read in the previous sections, the AAA override check box allows the WLC to honor the attributes received from the RADIUS server and that are attached to the RADIUS access-accept packet. Some of the attributes don't have any particular prerequisites and can be honored on the spot, such as the session timeout (it overrides the session timeout configured in the policy profile). On the other hand, some attributes, such as ACL or VLAN assignment, require the VLAN or ACL to be preconfigured on the WLC for the WLC to simply have to apply the setting. In FlexConnect, those settings need to exist on the AP itself, where they have to be applied. This means that you need to precreate specific VLANs and/or ACLs on the APs before being able to return them via the AAA server. This can be done in the Flex profile where you can add policy ACLs (which download the ACL to the AP and not necessarily apply it yet because it is dynamically applied), and VLANs in the VLAN-ACL mapping (where you do not necessarily have to assign an ACL but can just add VLANs).

Wireless Security Fundamentals

Next, we cover in enough detail all the security settings that can be applied on an SSID so that you can feel comfortable understanding what you are configuring on a Catalyst 9800 WLAN.

Wired Equivalent Privacy (WEP)

Wired Equivalent Privacy is not supported by the Catalyst 9800. WEP is not considered security and has been deprecated by the IEEE for many years. History books are available to learn more about WEP.

Wi-Fi Protected Access (WPA)

Wi-Fi Protected Access is the Wi-Fi Alliance certification based on the 802.11i amendment to the standard. The first version was released as a stopgap for devices that did not have the hardware to implement the final 802.11i specifications, and therefore, WPA1 implemented the Temporal Key Integrity Protocol (TKIP), which was based on the same RC4 algorithm used by WEP. WPA2 uses only the Advanced Encryption Standard (AES) algorithm, which is still considered secure to this day.

WPA2 originally combines AES with Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP), which is, to make things simple, the way that the AES encryption algorithm is applied to the data.

WPA uses the existing legacy open system authentication, which basically consists of the client and AP sending each other empty successful authentication frames before being able to send the association request. Although this approach seems weird, the reason is that the only other authentication frame type at the time referred to share key WEP authentication and was very unsecure. Therefore, the core of the WPA exchange happens after the association. The association request and response are the opportunity for both the client and the AP to exchange their security capabilities. The Robust Security Network (RSN) Information Element (IE) is there to specify which authentication and encryption methods and ciphers can be used by both parties.

WPA comes in two flavors (aside from the WPA1 prestandard certification and WPA2 final certification): Personal and Enterprise. The encryption of the traffic between the wireless client and the AP is unique (that is, it's different for every session) and securely created from the user authentication phase. It is important to note that only data frames are encrypted by WPA, and all the management frames are sent in clear text. WPA Personal is the most popular flavor out there due to the simplicity to deploy it: the wireless network (or SSID or WLAN) uses a preshared key that is known by all authorized clients and that serves as authentication (if you know it, you must be in on the secret and therefore allowed to join the network) as well as a seed for the unique encryption key. This means that every wireless client, in reality, uses a different key for sending and receiving traffic to and from the AP, but all those keys are generated from the same master secret. WPA Enterprise requires the use of a RADIUS server and relies on the 802.1X/EAP authentication framework (which typically works based on username/password pairs but can also accommodate certificates, tokens, smartcards, and the like) to generate the key seed. The benefits of WPA Enterprise are its use of different credentials for each user, and the fact that each user encryption key is based on those unique credentials rather than shared credentials. The 802.1X/EAP can come with stronger security during the authentication phase, but that largely depends on the EAP method chosen and the way it is implemented. The cost of WPA Enterprise includes setting up a RADIUS server (which can be more or less of a hassle depending on what you choose) and the management of unique credentials for every client (although they could technically still use identical credentials if you choose that for simplicity of management). IoT devices (printers, scanners, and so on) typically implement WPA Personal only for the simplicity of implementation and do not allow WPA Enterprise to be configured.

The security of both WPA flavors lies mainly in the credentials chosen. Even with the most secure EAP method for WPA Enterprise, if a user password is **P@sswOrd** (even with that @ and 0 included), the most basic tools can guess or brute-force it very quickly. The same goes for PSK: if you choose a complex preshared key that is long enough, your network is very secure to brute-forcing, and the main weakness resides in the fact that all clients share this same key. Therefore, someone could leak it to an attacker, and all the clients are then compromised instead of just one. There are multiple proprietary implementations of PSK, which allow for having more than one PSK (up to having a different

key for every device if you wanted to). The methods supported by the Catalyst 9800 are covered in the section “Preshared Key for WPA Personal.”

Protected Management Frames (PMF) is a standard amendment, compatible with WPA2 but ratified much later, that encrypts unicast management frames and provides protection against forgery to broadcast management frames as well. Thanks to PMF, it is not possible anymore for an attacker to impersonate the AP in an existing client-to-AP-established encrypted relationship. Probes and association frames, for example, are still in clear text because they happen before a key handshake can occur.

The encryption key is derived from the seeding material through what is often called the WPA *four-way handshake*. In reality, several keys are created during this exchange because there exists a separate key for broadcast traffic called the Groupwise Transient Key (GTK) and a few others that are beyond the scope of this chapter. This four-way exchange is always the last step before traffic can be sent (typically the DHCP DISCOVER would be the first traffic sent by the client) in an encrypted manner on the network. This four-way handshake has to happen between every client and every AP the clients connect to: because the encryption key is unique to the client-AP session, a different key has to be negotiated when the client roams to another AP. This handshake is a ping-pong between the AP and the client exchanging nonces (that is, random and arbitrary numbers used to see the cryptographic communication) to derive the same encryption key without transmitting that key over the network and considering both parties know a shared secret (either the PSK or the result of the 802.1X/EAP authentication that occurred just before). Those messages are called EAPoL Message 1 to 4 or EAPoL M1, M2, M3, or M4.

Those four messages are enough to exchange all the needed session keys between the AP and the client. Because the broadcast traffic is encrypted by a separate group key that is identical between every device connected to the same AP, that group key may need to be rotated regularly or when clients join or leave the WLAN. The AP can sometimes send an M5 message to which clients reply with an M6 message to derive a new group key for broadcast traffic.

WPA3 recently became the latest cool thing in the domain of wireless security and is covered later. It was created to improve on the weaknesses of WPA2. Despite the fact that WPA2 has still not been fully cracked yet to this day, it still showed vulnerabilities that required improvements. The main issues of WPA2 are as follows:

- The passphrase/password is used for both authentication and encryption.
- It is susceptible to offline attacks if the four-way handshake is captured. The attacker can try a dictionary of keys or passphrases until they are able to verify message 3 and 4 with it. With the public cloud offering a lot of compute power at a rental cost, this is making dictionary or brute-force attacks possible for anyone, even on a budget.
- There is no forward secrecy, which means if you know the password or passphrase, you are able to derive all the session keys for all past, present, and future sessions.
- By default, only the data frames are encrypted, which means that all the control and management frames are in clear text, and this basically opens the door for imperson-

ation and for attackers to pretend to be someone else and disconnect other clients remotely. PMF was released to take care of this issue but didn't get much traction from a Wi-Fi device manufacturer's standpoint and so was not widely implemented.

802.1X for WPA Enterprise

802.1X-2010 (yes the *X* is capital) is a protocol framework for Layer 2 access control. This implies that the network device controls the user identity before providing an IP address to it and permits or denies network connectivity based on the identity of the user or device requesting access. Built around a wired port concept, it still translates perfectly as a virtual port concept for wireless users. 802.1X allows you to authenticate wired and wireless devices similarly, as shown in Figure 5-18. The controlled port blocks all data except for 802.1X-related authentication frames (called EAPoL frames) until authentication has completed successfully. At that point, the port forwards all or some traffic only (based on the authentication result). Over wireless, each association between a client and access point generates a unique pair of 802.1X virtual ports. The only difference with 802.1X over wireless is that the authentication results in the generation of cryptographic keys that are used for traffic encryption (which is not necessarily the case over a wired connection).

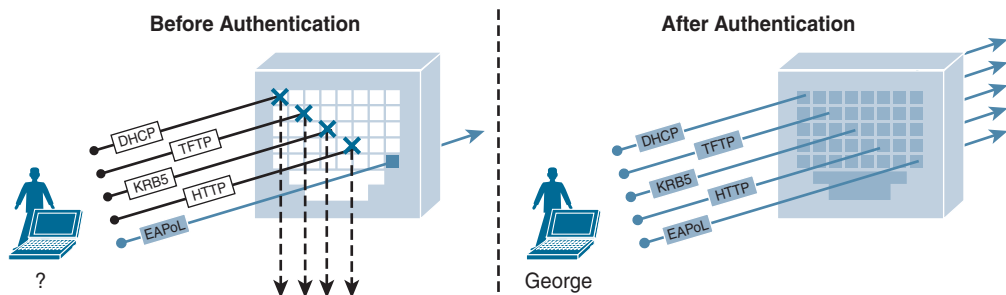


Figure 5-18 802.1X port-based authentication system

802.1X Components

As specified in the standard, and as shown in the Figure 5-19, a few terms depict the various roles of devices taking part in an 802.1X authentication:

- **The supplicant:** The client requesting access to a network resource and submitting credentials for authentication.
- **The authenticator:** A network device controlling the access to the network and facilitating the authentication process by relaying the credentials of the supplicant to the authentication server. This typically is a switch or, in this case, a WLC.
- **The authentication server:** A server responsible for validating the credentials sent by the supplicant and determining the network access level to grant to the user. In this case, this is the ISE server.

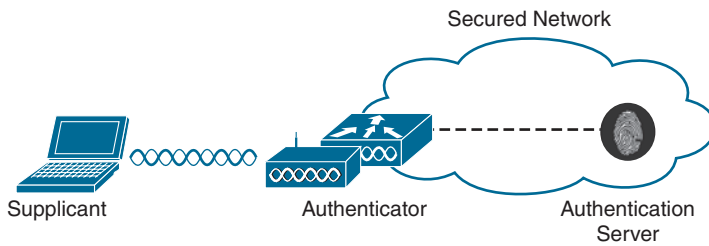


Figure 5-19 802.1X components

EAP

The Extensible Authentication Protocol (EAP) is at the heart of the 802.1X authentication process and provides a flexible framework and protocol, which supports multiple methods that are negotiated between the client and the authentication server and used to mutually authenticate them. EAP messages between the supplicant and the authenticator run directly over the data link using EAP over LAN (EAPoL) encapsulation (as shown in Figure 5-20). The authenticator can then extract the EAP payload and forward it to the authentication server over the wired network, typically inside a RADIUS packet. There is a clear demarcation between the EAP exchange over EAPoL between the client and the authenticator, which uses no addressing because the link is point to point and is supposed to be the only traffic allowed, and the transport of this EAP exchange over the IP network to the authentication server, which can be much further away on the network.

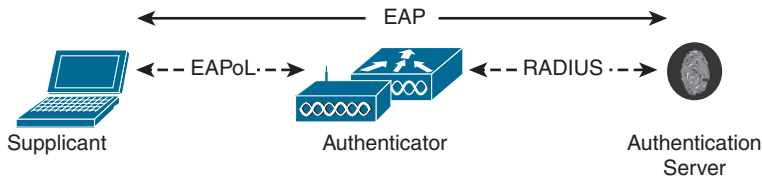


Figure 5-20 EAP authentication workflow

EAP is a request-response protocol consisting of only four EAP type packets identified in the “code” field:

- **Request:** This packet is sent by the authenticator to the supplicant and uses the type field to indicate what is requested (for example, identity, or EAP method to use).
- **Response:** This packet is sent by the supplicant in response to a valid request from the authenticator.
- **Success:** This packet is sent by the authenticator to signify a successful completion of an authentication.
- **Failure:** This packet is used by the authenticator if the authentication ends in a failure.

EAPoL frames also have several types and can be an EAPoL START (to signify one party wants to start a new authentication) or an EAPoL key (which transports a key).

EAP Methods

EAP is a framework protocol allowing for many different authentication methods (each having a unique ID) that define how credentials are exchanged. Although many exist, only a handful are used for authentication over wireless, and they are covered here.

- PEAP (Protected EAP):** As depicted in Figure 5-21, this method relies on server-side TLS authentication to create an encrypted tunnel using a valid certificate that the authentication server sends to the supplicant at the beginning of the handshake. Just like during HTTPS web browsing, a TLS tunnel is built, and the server side always has to provide a certificate to validate its identity. The client doesn't have to provide anything. Historically, many operating systems allowed the supplicant to not enforce validation of the server certificate, which means you could run a bogus certificate on your ISE server without issues, but more and more operating systems started to enforce this validation with no option to disable it manually. This means it is becoming a requirement to have a certificate on your ISE server that the client has in its trust store. Some clients allow you to manually import the ISE certificate in real time, whereas others require you to preload it. When this TLS tunnel is established, a new EAP negotiation using a new/another EAP method takes place inside the tunnel.

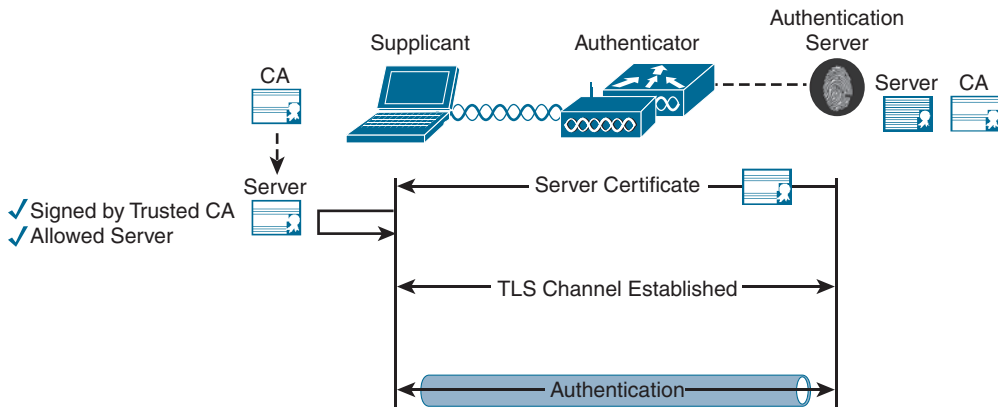


Figure 5-21 PEAP authentication high-level workflow

Therefore, the goal of PEAP is only to provide an encrypted tunnel to securely use another (even less secure normally) EAP method inside it. That new inner EAP method is typically EAP-GTC, MS-CHAPv2, or EAP-TLS.

- PEAP with inner EAP-GTC:** This EAP method was originally designed for one-time password authentication or token authentication. This means sending the password in clear text (but it's inside the PEAP original TLS tunnel, so it's not visible to an attacker/eavesdropper). This method is required if you are using a

third-party user credentials database like LDAP or such, as is typically the case when using one-time passwords. The external user database needs to get the password in clear text to verify it because it is not able to verify the MS-CHAPv2 encrypted version of it.

- **PEAP with inner EAP-MS-CHAPv2:** This is the most popular EAP method so far among Windows PCs and smartphones. EAP-MS-CHAPv2 is a way to exchange a username and password in encrypted form (it's encrypted twice because CHAP already encrypts the password and it's inside the PEAP TLS tunnel as well).
- **PEAP with inner EAP-TLS:** This is maybe the most secure method because another TLS authentication happens inside the encrypted TLS tunnel. This inner TLS authentication, this time, requires both server- and client-side certificate authentication. This means typically running an Enterprise PKI (something ISE can do for you with BYOD workflows) and issuing your own certificates to your clients. The client certificate is encrypted to any attacker/eavesdropper, so it is impossible for anyone other than ISE to know the certificate fields such as the username. Considering it involves two TLS handshakes, this method is quite slow if the client has to reauthenticate fully every time it roams to another access point. Fast roaming is advised if you want to deploy it.
- **EAP-FAST (EAP-Flexible Authentication via Secure Tunneling):** Depicted in Figure 5-22, developed by Cisco, and published as IETF RFC 4851, EAP-FAST enables mutual authentication by using a shared secret, called the Protected Access Credential (PAC), to establish a TLS tunnel that secures the exchange of user authentication messages. It has similarities to PEAP in this regard, but the PAC is at the core of EAP-FAST and allows for more deployment options. The PAC can be manually distributed to the client out-of-band if you want to ensure the most secure deployment. EAP-FAST has three phases. Phase 0 is required only if you opted out of manually distributing the PAC out of band. During that phase, the PAC is provisioned in-band on the supplicant using either an anonymous TLS handshake (using the Diffie-Hellman protocol) or an authenticated tunnel using the server certificate. Concretely, this means you can avoid using a valid and trusted certificate on the authentication server in case you are doing manual PAC out-of-band provisioning or if you provision in-band but decided to use an anonymous TLS tunnel. Phase 1 is the TLS tunnel establishment using the PAC, which takes care of mutual authentication (if both sides have the PAC, they must know each other). Finally in phase 2, user authentication credentials are passed within the TLS tunnel created in phase 1. Either EAP-MS-CHAPv2 or EAP-TLS is available as the inner method, just like for PEAP. EAP-FAST also allows for proprietary features such as EAP-chaining where two identities (the device itself and the user on it) can be provided in the same authentication flow. The security of EAP-FAST depends greatly on your deployment options and the inner method used. Most people use it with anonymous in-band provisioning, which typically requires a first failed authentication to deliver the PAC before authentications can succeed after the client has installed the PAC correctly.

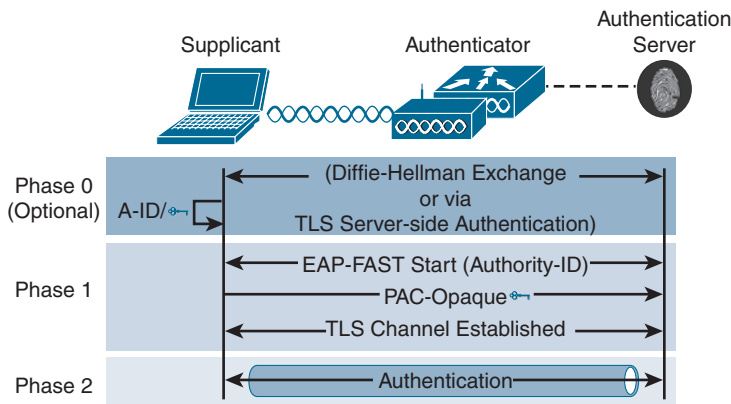


Figure 5-22 EAP-FAST high-level overview

- EAP-TLS:** Depicted in Figure 5-23 and described in IETF RFC 5216, this method was covered previously as an inner method to PEAP or EAP-FAST. It is also available as a standalone method. A TLS tunnel is then built based on both a server-side as well as a client-side certificate. The tunnel itself is the authentication, and no other inner method occurs. It is therefore possible for an eavesdropper to read certificate details (which is not necessarily a major security concern) during the authentication. For an EAP-TLS authentication to succeed, the client device should have the root CA that issued the authentication server certificate chain in its trusted root CA store, and the authentication server should similarly have the root CA on top of the client certification chain in its trusted root CA store. This way, both parties can authenticate the other party certificate as issued by a trusted chain of CAs. It is interesting to note that there is technically no username in EAP-TLS authentication: a certificate is provided, and it's either trusted by the other side or it is not. The certificate is not issued to a username but to specific fields such as Common Name (CN), Country (C), and Organization (OU). To be able to identify the certificate uniquely to return a specific set of privilege, the authentication server can be configured to consider any of these certificate fields (or a combination of fields) as a username.

The validity of a certificate in an 802.1X context consists of verifying whether the certificate is still in its validity period and if it is issued by a trusted root CA on top of the chain and optionally if the certificate hasn't been revoked in a certificate revocation list (CRL). Some authentication servers can proceed in making a binary comparison of the certificate; that is, they compare bit by bit that the certificate presented is the same one as the one stored in the authentication server database, but that is uncommon. Being issued by a trusted source is typically enough. There is no verification of the name on the certificate because users don't really map to domain names like web servers do. The client also can hardly verify the authentication server DNS FQDN because 802.1X is a Layer 2 authentication method happening before any sort of IP connectivity. The client only relies on having the trusted CA installed in its store as well as the validity period. On its side, the ISE can be configured to read one field of the certificate (typically the CN) to obtain a username out of the certificate if that can be useful for matching authorization policies.

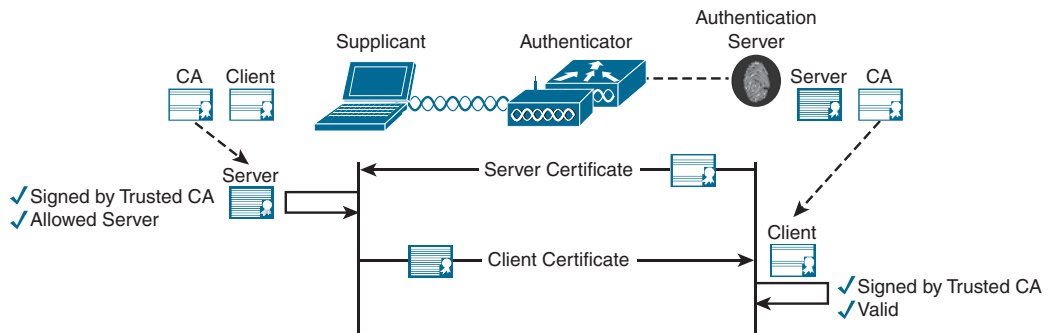


Figure 5-23 *EAP-TLS high-level overview*

WPA3 Enterprise

WPA3 also comes with a revision of the Enterprise flavor of WPA. There is no major advertised change, but the main thing is that WPA is also an interoperability certification, and this certification was revised to take care of many previous flaws and attack vectors around its implementation. More notably, WPA3 Enterprise brings a 192-bit cryptographic strength aimed at sensitive government and finance sectors, while the 128-bit security remains the default. The list of supported and allowed cryptographic protocols is revised and updated in WPA3, and a more restrictive option called SuiteB brings a uniform list of very secure protocols for the most demanding customers. PMF is now a requirement for WPA3, which means that clients supporting WPA3 support PMF, and on the administrator side, it is impossible to configure a WPA3 SSID without PMF being enabled.

A transition mode exists to allow WPA3 clients to use PMF while WPA2 clients can still connect. WPA transition mode basically downgrades the security of your SSID to a regular WPA2 SSID though.

Fast Transition (FT), the standard for fast and secure roaming (covered in more detail in the next chapter, along with client mobility), is not supported with 192-bit WPA3 Enterprise at this time; this would need a new AKM for 192 bits + FT and would need clients to support it, so it's an industrywide limitation. Also, Catalyst 9115 and 9120 access points do not support WPA3 SuiteB GCMP256 ciphers (AES256 is okay).

Preshared Key for WPA Personal

When you're configuring WPA Personal with a preshared key, what you typically think of as "the key" is actually the passphrase. It is not what is used to encrypt the traffic and not even what is used to generate the actual encryption key. Details are beyond the scope of this book, but the passphrase is converted to a 256-bit key, called a Pairwise Master Key (PMK), using a standard conversion method. That 256-bit key is used to generate the actual final encryption key. This allows you to have a variable-length passphrase (imagine having to remember a 64-character passphrase, for example), which is easier to remember. The requirements for the passphrase are that it must contain between 8 and 63 printable ASCII characters. Eight characters is really not a lot and

should definitely not be used in production because it is a reasonable length to brute-force by using dictionaries of commonly used passphrases. Using a longer passphrase (including fancier characters) randomizes the 256-bit final key better and protects you from brute-force attacks.

One of the inherent weaknesses of the PSK authentication and key management protocol is that even if the encryption/session key is unique for each client, these keys are derived from the same seeding material (the passphrase). In practice, this means that if an attacker starts to listen to over-the-air traffic and knows the passphrase, it is not able to decrypt the traffic because it cannot derive the same session key. However, if the attacker is listening when a given client is associating and the attacker hears the four-way handshake, the attacker is able to derive the same session key as the client because they know the passphrase.

WPA3 SAE

WPA3 Personal is also called SAE for Simultaneous Authentication of Equals and brings much more than just a change of name. Contrary to its predecessors (WPA1 and 2), SAE does not rely on the open authentication mechanism but adds a new SAE authentication frame type, as shown in Figure 5-24. This adds two frames to the overall frame exchange count because there are four SAE authentication frames exchanged in total (two back and forth between the client and the AP), then the association and the typical four-way handshake.

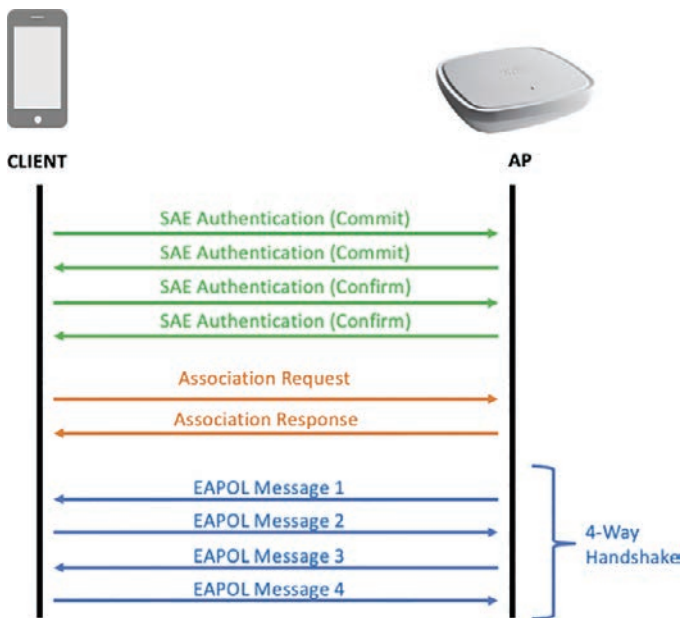


Figure 5-24 WPA3 SAE workflow

The SAE authentication frame exchange consists of four frames because it is a kind of key exchange by itself. Finishing the SAE initial four-frame exchange results in deriving a Pairwise Master Key (PMK) already. The concept of SAE is to protect the actual four-way key exchange by having it encrypted thanks to an SAE exchange done first, which allows you to exchange anonymously session keys. This process is similar in principle to the PEAP concept of building a secure tunnel to exchange credentials more securely. The effect of this is that even if an attacker knows the passphrase (which is still the weakest attack vector to get on to your network), the attacker is not able to decrypt the traffic from any wireless client, and that is a strong advantage over WPA2. In summary, SAE strongly enhances privacy while authentication security stays related to the strength of your passphrase and how you share it with your users. The SAE handshake also prevents the attacker from doing offline brute-force attacks on a captured handshake because that attacker has to be online and try one key at a time, which drastically reduces the speed of brute-forcing (and allows you to detect the attack).

WPA3 SAE requires PMF to be enabled (as for all WPA3 deployment types) and does not support Fast Transition (FT) at the time of this writing.

MPSK

MPSK (Multiple PreShared Key) is an option you can configure in the WLAN settings (as shown in Figure 5-25); it allows you to configure up to five preshared keys for a given WLAN. This allows you to rotate keys or to use specific keys for specific devices. Keep in mind, however, that there is no authentication: anyone can connect using any of the keys entered.

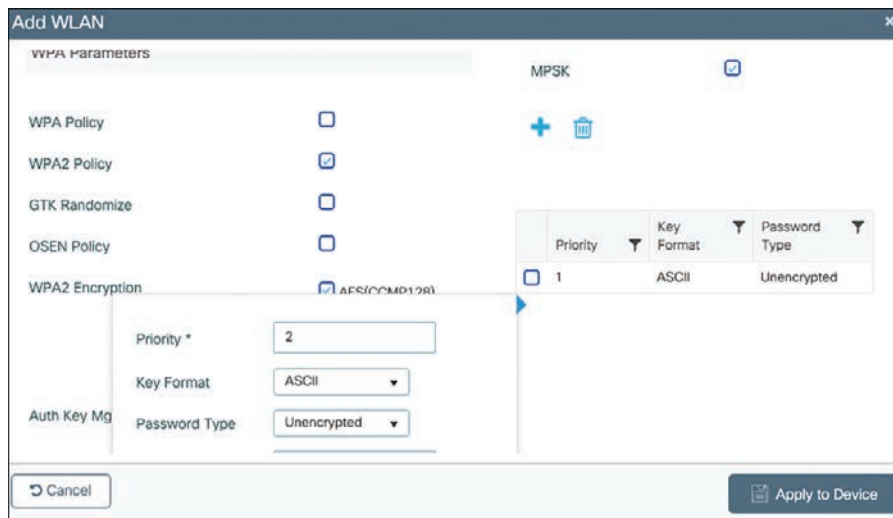


Figure 5-25 MPSK configuration in WLAN settings

Due to the way SAE works, MPSK is not supported with WPA3.

Identity PSK (iPSK)

Identity PSK is a standard PSK SSID where the administrator configures MAC filtering to have a pretext to reach out to a RADIUS server (you don't necessarily want to validate the MAC addresses). The RADIUS server can then return, through specific attributes, a (possibly) unique key for use with the wireless client. It is assumed the wireless client knows the right key to use. Although many people think this means they have to add wireless device MAC addresses in the RADIUS server and return specific PSKs depending on the MAC used for authentication, this is not the only possibility. As a matter of fact, the RADIUS server can use any RADIUS attribute in its authorization policy to determine which key to return: the location of the AP/WLC, the type of client device used (through profiling), or any creative combination based on the RADIUS attributes present in the MAC filtering authentication request. An example is shown later, but first let's look at MAC filtering itself and RADIUS concepts.

MAC Filtering

If you configure MAC filtering (also sometimes called MAC bypass) on a WLAN, the controller basically verifies the client MAC address authorization after the client sends the association request frame and before sending the association response. The wireless controller can either check locally in its local database or send the request to an external authentication server. The association response status code sent back to the client then depends on the authentication result: the association is rejected if the MAC address is not authorized to access the network.

There are several concerns with this system:

- It is not a security measure. A MAC address can be spoofed, and MAC addresses can be easily discovered by eavesdropping because they are always in clear text, and therefore it cannot be considered as a proper security measure.
- The fact that more and more devices use a randomized MAC address to connect to networks makes this measure completely unpractical and unmanageable. The last remaining use case could be IoT devices that do not perform MAC randomization or devices where the administrator manually configured the wireless adapter to not change the MAC address. But again, it cannot be considered secure because a MAC can be spoofed.
- It can be used as a pretext to trigger a RADIUS authentication to the external authentication server, as is the case with central web authentication (covered later).
- The MAC filtering authentication happens between the association request and the association response, which means that if the wireless controller has to verify the MAC address with an external RADIUS server, this can add a great deal of latency, and the association response sent back to the client can be delayed. There is no standard recommended maximum latency, but after more than 50 ms, most clients do not wait for an association response anymore and try to connect to other APs, possibly on other channels. MAC filtering has to be implemented with a low-latency nearby MAC address database.

Note At the time of this writing, the only MAC address format tolerated in the WLC local database for authorizing clients and APs via MAC filtering is aaaabbbbcccc (without any separators).

Enhanced Open

Not part of the WPA3 specification but released at the same time, Enhanced Open brings more privacy to open networks by encrypting all the guest traffic without requiring any key to be shared. Guest networks are here to stay, and although some administrators configured a preshared key on their guest WLAN, it is clear that this approach is neither secure nor practical: you have to share the key with your guests one way or another, and this often leads to anyone being able to figure out what the key is. This means that anyone can get on your network without having to type in anything. Enhanced Open does not bring any security in the authentication sense of the word; anyone can still onboard the network without any form of verification of identity. However, based on Opportunistic Wireless Encryption (OWE), the client and AP securely derive an encryption session key pretty much out of thin air (that is, without having a shared secret known beforehand from both parties).

Once connected, their traffic is private, meaning no attacker is able to decrypt their traffic (offline dictionary attacks are rendered impossible). Enhanced Open also requires PMF, which means it is not possible anymore for an attacker to remotely deauthenticate anyone else or impersonate certain management frames. Interestingly, Enhanced Open even beats WPA2-PSK from a privacy standpoint because when the attacker knows the WPA2-PSK, they can decode everyone's traffic, which is not the case with Enhanced Open, because every client uses a different master key that differs at each session.

Enhanced Open comes with a great transition mode because client support for Enhanced Open is still limited at this time. When using Enhanced Open transition mode, the controller creates two SSIDs in fact. One is broadcasted and is a regular open SSID, and the second is hidden (that is, its name is not present in beacons) and uses Enhanced Open as security. The regular open SSID beacons contain an information element advertising the presence of a hidden Enhanced Open BSS. This behavior is standardized, which means it is not a Cisco hack, and any client supporting Enhanced Open is aware of this technique. The result is that the legacy clients not supporting Enhanced Open simply see a regular open guest SSID and connect to it without seeing anything different (and do not benefit from any improved privacy, of course), whereas clients supporting Enhanced Open automatically connect to the Enhanced Open SSID even when the user selected the regular one on their device.

Securing the Air

Many configuration examples are available on cisco.com, showing very specific configuration. Here, we highlight the usual security combinations and the configuration logic, not the detailed steps.

WPA2 Personal

When adding a WLAN, as depicted in Figure 5-26, simply choose WPA+WPA2 as the Layer 2 security mode. Fast Transition is possible to configure with a PSK but does not bring a lot of advantages to the table.

The screenshot shows the 'Add WLAN' configuration window with the 'Security' tab selected. Under the 'Layer2' sub-tab, the following settings are visible:

- Layer 2 Security Mode: WPA + WPA2
- MAC Filtering:
- Protected Management Frame:
- PMF: Disabled
- WPA Parameters:
- Lobby Admin Access:
- Fast Transition: Disabled
- Over the DS:
- Reassociation Timeout: 20
- MPSK Configuration:


Figure 5-26 WPA 2 PSK SSID settings

The standard configuration, as shown in Figures 5-27 and 5-28, is PSK as AKM with AES128. GCMP is a protocol that allows for fast encryption of very high throughputs of data but is not required at typical Wi-Fi 6 speeds yet, and 256 bits of encryption is typically used in conjunction with WPA3 (although not necessarily, because not all clients support it).

The screenshot shows the 'Add WLAN' configuration window with the 'Security' tab selected. Under the 'Layer2' sub-tab, the following settings are visible:

- WPA Policy:
- WPA2 Policy:
- GTK Randomize:
- OSEN Policy:
- WPA2 Encryption:
 - AES(CCMP128)
 - CCMP256
 - GCMP128
 - GCMP256
- Auth Key Mgmt:
 - 802.1x
 - PSK
 - Easy-PSK

Figure 5-27 WPA2 PSK Layer2 security settings



PSK Format: ASCII

PSK Type: Unencrypted

Pre-Shared Key*:

Figure 5-28 WPA2 PSK SSID key configuration

Enter the key as unencrypted if you are typing it manually. You would choose an AES key if the passphrase you were pasting is already AES-encrypted (if you were copying from an existing encrypted configuration file, for example).

Figure 5-29 shows the Robust Security Network (RSN) information element (a field from the beacon frame of the AP) of the WPA2 PSK SSID illustrated in Figure 5-26.

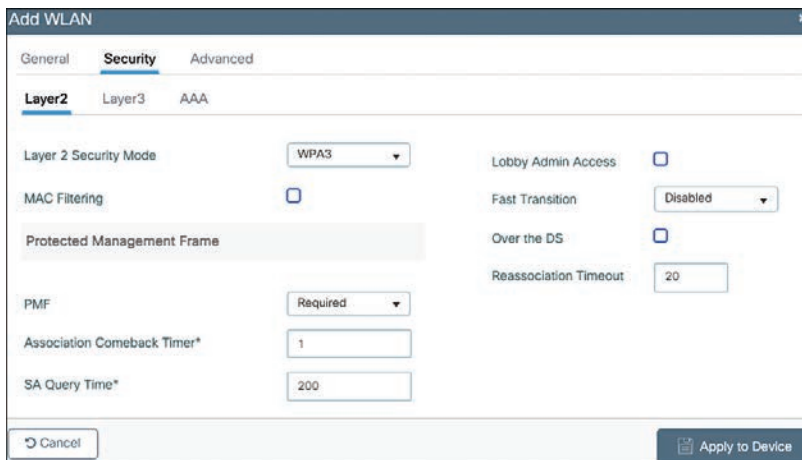
```

▼ Tag: RSN Information
  Tag Number: RSN Information (48)
  Tag length: 20
  RSN Version: 1
  ▼ Group Cipher Suite: 00:0f:ac (Ieee 802.11) AES (CCM)
    Group Cipher Suite OUI: 00:0f:ac (Ieee 802.11)
    Group Cipher Suite type: AES (CCM) (4)
    Pairwise Cipher Suite Count: 1
    ▼ Pairwise Cipher Suite List 00:0f:ac (Ieee 802.11) AES (CCM)
      ► Pairwise Cipher Suite: 00:0f:ac (Ieee 802.11) AES (CCM)
      Auth Key Management (AKM) Suite Count: 1
      ▼ Auth Key Management (AKM) List 00:0f:ac (Ieee 802.11) PSK
        ► Auth Key Management (AKM) Suite: 00:0f:ac (Ieee 802.11) PSK
      ► RSN Capabilities: 0x0028
  
```

Figure 5-29 RSN information element of a beacon frame of a WPA PSK SSID

WPA3 SAE

For configuring SAE, select **WPA3** (starting with 176 IOS-XE) or **WPA2+WPA3** as the security mode, as shown in Figure 5-30. As a reminder, FT is not supported with SAE at this time.



Add WLAN

General Security Advanced

Layer2 Layer3 AAA

Layer 2 Security Mode: WPA3

MAC Filtering:

Protected Management Frame:

PMF: Required

Association Comeback Timer*: 1

SA Query Time*: 200

Lobby Admin Access:

Fast Transition: Disabled

Over the DS:

Reassociation Timeout: 20

Cancel Apply to Device

Figure 5-30 SAE WLAN example configuration

Choose **AES-128** and **SAE** for the AKM, as shown in Figure 5-31. WPA3 offers the same 128-bit encryption as was used with WPA2 but allows you to turn it up to 256 bits and offers the choice between the same CCMP mode of operation or to use Galois Counter Mode Protocol (GCMP). GCMP was meant to be used with 802.11ac wave 2 because it is better suited for very high throughput encryption, but it did not get traction in the field back then, especially on the client side. Many WPA3-certified devices now support 256-bit AES and GCMP.

WPA3 Policy	<input checked="" type="checkbox"/>
WPA2/WPA3 Encryption	<input checked="" type="checkbox"/> AES(CCMP128)
	<input type="checkbox"/> CCMP256
	<input type="checkbox"/> GCMP128
	<input type="checkbox"/> GCMP256
Auth Key Mgmt	<input type="checkbox"/> 802.1x
	<input type="checkbox"/> CCKM
	<input checked="" type="checkbox"/> SAE
	<input type="checkbox"/> OWE
	<input type="checkbox"/> FT + 802.1x
	<input type="checkbox"/> 802.1x-SHA256

Figure 5-31 *Ciphers option example for SAE*

Enter your preshared key unencrypted, as depicted in Figure 5-32, and you should be good to go.


Anti Clogging Threshold*	<input type="text" value="1500"/>
Max Retries*	<input type="text" value="5"/>
Retransmit Timeout*	<input type="text" value="400"/>
PSK Format	<input type="text" value="ASCII"/>
PSK Type	<input type="text" value="Unencrypted"/>
Pre-Shared Key*	<input type="text" value="....."/> 

Figure 5-32 *PSK configuration for SAE*

If you take a wireless capture and look at the beacon RSN IE, you can see SAE advertised as the authentication key management method (see Figure 5-33).

```

▼ Tag: RSN Information
  Tag Number: RSN Information (48)
  Tag length: 26
  RSN Version: 1
  ▼ Group Cipher Suite: 00:0f:ac (Ieee 802.11) AES (CCM)
    Group Cipher Suite OUI: 00:0f:ac (Ieee 802.11)
    Group Cipher Suite type: AES (CCM) (4)
    Pairwise Cipher Suite Count: 1
  ▼ Pairwise Cipher Suite List 00:0f:ac (Ieee 802.11) AES (CCM)
    ► Pairwise Cipher Suite: 00:0f:ac (Ieee 802.11) AES (CCM)
    Auth Key Management (AKM) Suite Count: 1
  ▼ Auth Key Management (AKM) List 00:0f:ac (Ieee 802.11) SAE (SHA256)
    ► Auth Key Management (AKM) Suite: 00:0f:ac (Ieee 802.11) SAE (SHA256)
  ► RSN Capabilities: 0x00e8
  PMKID Count: 0
  PMKID List
  ▼ Group Management Cipher Suite: 00:0f:ac (Ieee 802.11) BIP (128)
    Group Management Cipher Suite OUI: 00:0f:ac (Ieee 802.11)
    Group Management Cipher Suite type: BIP (128) (6)

```

Figure 5-33 SAE advertised in the SSID beacon RSN IE

WPA2 with iPSK

The concept of Identity PSK is to use a different WPA preshared key depending on the identity of the client connecting. This differs from MPSK, where several keys are available, and anyone can use any key. In this method, you rely on the AAA server to return a specific key, depending on the authentication result. iPSK is covered in detail in *Configure Catalyst 9800 WLC iPSK With Cisco ISE* on cisco.com, but let's tackle the important points and use the opportunity to explain the concepts of RADIUS attributes. On the WLC side, you need to create a WPA PSK WLAN and add MAC filtering, as illustrated in Figure 5-34. The MAC filtering is basically an excuse to reach out to the AAA server and initiate a RADIUS conversation; the intent is not necessarily to authorize the MAC address.

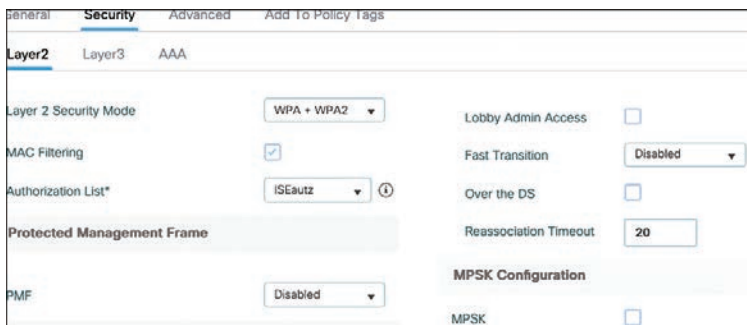


Figure 5-34 iPSK SSID security settings part 1

As you can see in Figure 5-35, you still need to configure a PSK that is the default key, used when the AAA server returns only a permit-access and not specifying any extra key.

WPA2 Encryption	<input checked="" type="checkbox"/> AES(CCMP128)
	<input type="checkbox"/> CCMP256
	<input type="checkbox"/> GCMP128
	<input type="checkbox"/> GCMP256
Auth Key Mgmt	<input type="checkbox"/> 802.1x
	<input checked="" type="checkbox"/> PSK
	<input type="checkbox"/> Easy-PSK
	<input type="checkbox"/> CCKM
	<input type="checkbox"/> FT + 802.1x
	<input type="checkbox"/> FT + PSK
	<input type="checkbox"/> 802.1x-SHA256
	<input type="checkbox"/> PSK-SHA256
PSK Format	ASCII
PSK Type	Unencrypted
Pre-Shared Key*	*****

Figure 5-35 *iPSK SSID security settings part 2*

The AAA method used in MAC filtering is a “aaa authorization network” type. The policy profile used also needs to allow AAA override for the WLC to accept RADIUS attributes returned by the AAA server.

On the Cisco ISE side, you have many options. You can enter specific devices’ MAC addresses in the endpoint database so that MAC addresses are authenticated. You could also configure your authentication policy to proceed and continue with the authorization despite the authentication failing if the user is not found. Figure 5-36 illustrates an example of authorization rules you could configure. The first rule returns an access-accept along with the PSK “Cisco123” that the client has to use. The second rule illustrates the fact that you do not necessarily have to enter specific MAC addresses in your RADIUS server database but can use any of the RADIUS attributes present in the authentication request.

<input checked="" type="checkbox"/>	Specificdevice - PSKCisco123	Radius-Calling-Station-ID EQUALS E8-7F-95-53-20-12	PSK-Cisco123
<input checked="" type="checkbox"/>	Specificarea-PSKIoT	Radius-Called-Station-ID CONTAINS FactoryArea	PSKWireless123

Figure 5-36 *ISE authorization policies for IPSK*

This example uses the called-station-id and relies on the fact that the 9800 is configured to specify the AP locations in that attribute. You can configure it under **Configuration > Security > AAA > AAA Advanced > Global Config > Show Advanced Settings**, as shown in Figure 5-37. The same page also allows you to specify the MAC address format that will be handy if you are using a third-party RADIUS server such as FreeRadius. You can therefore define a specific key to use in a specific area of the network, regardless of who the client is. It could also be an option to write a rule based on the device group the client MAC belongs

to (either through static group membership in the ISE endpoint database or through profiling dynamic rules); your creativity and the existing RADIUS attributes set are the limit.

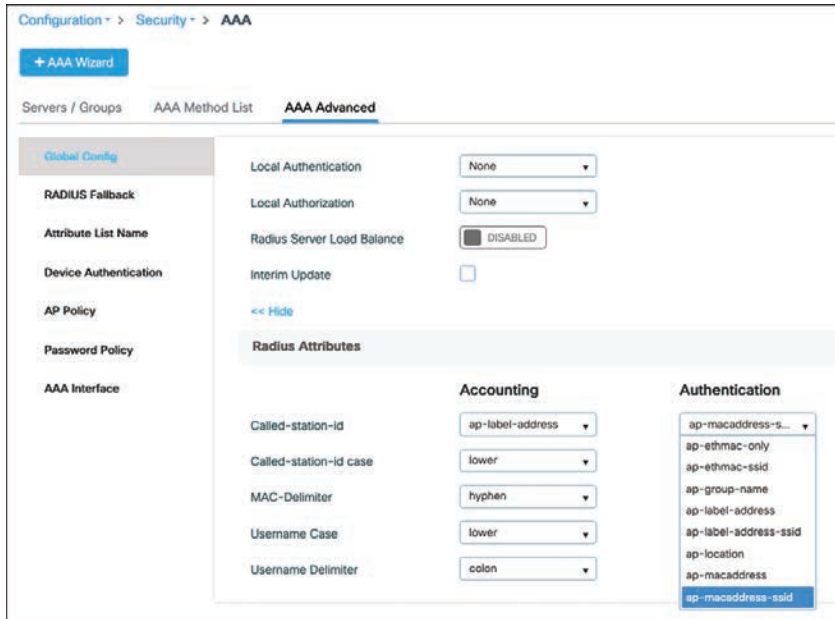


Figure 5-37 Customization of the called-station-id RADIUS field

The custom authorization results are straightforward and only need two attributes that are Cisco proprietary and called cisco-av-pair (they are shown in Figure 5-38). The first attribute is cisco-av-pair, and the value is **psk=<actual key you want to use>**, whereas the other is an identical cisco-av-pair and its value is **ascii** because you probably want to use your WPA key as an ASCII key.

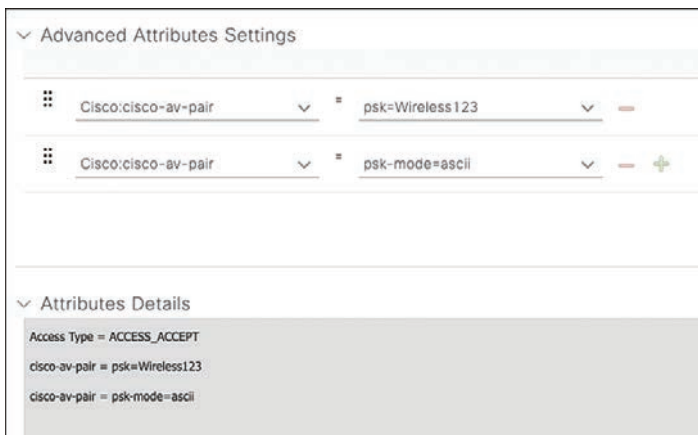


Figure 5-38 ISE authorization result

iPSK works because the MAC filtering is an authentication that happens at the 802.11 association phase. The AP or WLC authenticates the MAC address before replying with an association response, and therefore, this is a time-sensitive process where the client can give up on waiting for this association response and move to another channel if your RADIUS server is not quick enough.

Let's take a look at the RADIUS exchange from the RadioActive Trace output taken from the Catalyst 9800. The exchange is simple: the controller sends an access-request mentioning the MAC address of the client as both username and password. The RADIUS server is expected to reply with access-accept or access-reject. Example 5-2 shows the request, as sent by the 9800 for an iPSK SSID.

Example 5-2 Always-On Logs Corresponding to a RADIUS Access-Request Packet Being Sent

```
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Send Access-Request to
192.168.1.99:1812 id 0/59, len 384
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: authenticator 9b 9e 12
1c 3b d9 d2 b3 - 53 4d f5 f0 2b 63 ae 1c
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: User-Name [1]
14 "e87f95532012"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: User-Password [2]
18 *
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Service-Type [6]
6 Call Check [10]
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco [26]
31
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
25 "service-type=Call Check"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Framed-MTU [12]
6 1485
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Message-
Authenticator[80] 18 ...
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: EAP-Key-Name [102]
2 *
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco [26]
49
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
43 "audit-session-id=8501A8C0000004ECC162224"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco [26]
18
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
12 "method=mab"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco [26]
30
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
24 "client-iif-id=50334480"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco [26]
17
```

```

wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
11 "vlan-id=1"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: NAS-IP-Address [4]
6 192.168.1.133
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: NAS-Port-Id [87]
17 "capwap_9000001b"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: NAS-Port-Type [61]
6 802.11 wireless [19]
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: NAS-Port [5]
6 116
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco [26]
28
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
22 "cisco-wlan-ssid=iPSK"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco [26]
30
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
24 "wlan-profile-name=iPSK"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Called-Station-Id
[30] 24 "d4-ad-bd-a2-8f-20:iPSK"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Calling-Station-Id
[31] 19 "e8-7f-95-53-20-12"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Airespace
[26] 12
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Airespace-WLAN-ID [1]
6 5
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Nas-Identifier
[32] 7 "Katar"

```

Note a couple of interesting RADIUS attributes:

- **User-name:** In this case, the client didn't initiate any authentication and therefore does not provide any username. It's the 9800 that decides to fill the User-name field with the MAC address using a specific format.
- **User-password:** This is one of the few fields that is encrypted with the RADIUS shared secret.
- **Service-type:** Although the naming of the possible options doesn't match current usages (they go back to the origin of RADIUS for dial-up communications), they depict whether the authentication is a MAC authentication or an 802.1X authentication, for example ("Call check" in the case of MAC, "Framed" in case of 802.1X).
- **Framed MTU:** The controller expresses the MTU it can honor for RADIUS. It is interesting to note that the RADIUS server is not required to follow this value, and Cisco ISE doesn't follow it; therefore, fragmentation of RADIUS packets can occur.
- **Message authenticator:** This field is encrypted using the shared secret, to make sure the WLC is allowed to communicate with the RADIUS server.

- **Cisco AVPair** : A couple of Cisco AVPairs mention some additional and optional information such as the SSID name, the WLAN profile name, the VLAN, or the fact that it's a MAC address filtering authentication ("method=mab").
- **Audit-session-id** : It is created by the WLC and serves as session ID (for example, if you want to terminate the session via CoA from ISE) even when accounting is not in use.
- **Airespace-WLAN-ID** : The Catalyst 9800 keeps using the same Airespace proprietary attributes introduced by the former generation of WLCs, which specifies the WLAN ID the client is connecting to.
- **Calling-station-id** : This attribute is always the MAC address of the client.
- **Called-station-id** : This attribute is, by default, the MAC address of the AP but can be customized to append other values. Here, the SSID name is appended to it.

All these attributes can be used to create authentication and authorization rules in the ISE, which means it is possible to authenticate the client based on where it is trying to connect, on which SSID it is trying to connect, its MAC address, and so on.

After this request, the RADIUS server replies with an access-accept, authorizing the client on the network and also providing the PSK that this client should be using, as shown in Example 5-3.

Example 5-3 Always-On Logs Corresponding to a RADIUS Access-Accept Packet Being Received

```
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Received from id
1812/59 192.168.1.99:0, Access-Accept, len 185
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: authenticator 2c d3 14
54 c5 e4 f2 7d - e1 4a b3 ee 80 ba 3f 2f
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: User-Name [1]
19 "E8-7F-95-53-20-12"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Class
[25] 50 ...
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Message-
Authenticator[80] 18 ...
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco
[26] 23
wncd_x_R0-0{1}: [radius] [25734]: (info): *
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
17
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco
[26] 22
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
16 "psk-mode=ascii"
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Vendor, Cisco
[26] 33
wncd_x_R0-0{1}: [radius] [25734]: (info): RADIUS: Cisco AVpair [1]
27 "profile-name=Apple-Device"
```


iPSK becomes a bit trickier to use for clients such as smartphones that now typically use a private MAC address by default (that is subject to change) but stays relevant for IoT devices that typically have a fixed MAC address because privacy is not their concern.

iPSK is supported in FlexConnect local switching mode. The P2P blocking action configuration drop-down in the WLAN advanced settings allows for an “allow private group” intriguing setting. This, when combined with iPSK, allows users using the same PSK to talk to each other but not talk to users using a different PSK.

Enhanced Open

Let’s cover a typical use case where you, as administrator, want to configure Enhanced Open but still allow older clients to be able to connect to the guest SSID. Figure 5-39 shows the final result: one WLAN is configured for WPA3 Enhanced Open (called OWE), and the other is a fully open SSID. Only the fully open SSID called “Open” has its name broadcasted in the beacons while Enhanced Open is hidden.

<input type="checkbox"/>		EnhancedOpen		3	EnhancedOpen	[WPA3][OWE][AES]
<input type="checkbox"/>		Open		4	Open	[open]

Figure 5-39 A pair of Open/Enhanced Open SSIDs for Enhanced Open Transition mode

For this example, create the first SSID, call it **EnhancedOpen** (in reality, choose a name that is relevant for your company), and make sure it is hidden by disabling **Broadcast SSID**, as depicted in Figure 5-40.

General	Security	Advanced	Add To Policy Tags
Profile Name*	EnhancedOpen		
SSID*	EnhancedOpen		
WLAN ID*	3		
Status	ENABLED		
Broadcast SSID	DISABLED		
	Radio Policy ⓘ		
	5 GHz	ENABLED	
	2.4 GHz	ENABLED	
	802.11b/g Policy (2.4 GHz)	802.11b/g	

Figure 5-40 Enhanced Open SSID general settings

Configure the security settings to be WPA3 with OWE as AKM and choose a cipher (AES 128 is a good conservative choice for a guest SSID), as shown in Figures 5-41 and 5-42. The last field asks you for the transition mode WLAN ID. This basically asks you for the “other SSID in the pair.” So, when configuring the Enhanced Open SSID, enter the WLAN ID of the open SSID (4 in this example).

General	Security	Advanced	Add To Policy Tags
Layer2 Layer3 AAA			
Layer 2 Security Mode	WPA3 ▼		
MAC Filtering	<input type="checkbox"/>		
Protected Management Frame			
PMF	Required ▼		
Association Comeback Timer*	1		
SA Query Time*	200		

Figure 5-41 Enhanced Open security settings part 1

WPA3 Policy	<input checked="" type="checkbox"/>
WPA2/WPA3 Encryption	<input checked="" type="checkbox"/> AES(CCMP128) <input type="checkbox"/> CCMP256 <input type="checkbox"/> GCMP128 <input type="checkbox"/> GCMP256
Auth Key Mgmt	<input type="checkbox"/> 802.1x <input type="checkbox"/> CCKM <input type="checkbox"/> SAE <input checked="" type="checkbox"/> OWE <input type="checkbox"/> FT + 802.1x <input type="checkbox"/> 802.1x-SHA256
Transition Mode WLAN ID	4

Figure 5-42 Enhanced Open SSID security settings part 2

Proceed with the creation of the regular open SSID, which is broadcasted and with no particular security settings other than enabling OWE transition mode and specifying the WLAN ID of the Enhanced Open SSID, as shown in Figure 5-43.

General	Security	Advanced	Add To Policy Tags
Layer2 Layer3 AAA			
Layer 2 Security Mode	None	Lobby Admin Access	<input type="checkbox"/>
MAC Filtering	<input type="checkbox"/>	Fast Transition	Disabled
OWE Transition Mode	<input checked="" type="checkbox"/>	Over the DS	<input type="checkbox"/>
Transition Mode WLAN ID*	3	Reassociation Timeout	20

Figure 5-43 Regular open SSID linked with transition mode to an Enhanced Open SSID

When you're done, if you look at the scan list of SSIDs for a wireless client present in the area where the WLANs are active, no matter what it supports, it shows only the Open SSID (because it is the only one broadcasted). If your client does not support Enhanced Open, it connects to the Open SSID, and it is business as usual. If your client supports Enhanced Open, even if you click the "Open" SSID in the list of heard SSIDs, in reality you are connected to the secure SSID. You can see from the **Monitor > Clients** page in the WebUI that the client is, in fact, using encryption, as depicted in Figure 5-44.

WLAN Profile Name	EnhancedOpen
Wireless LAN Network Name (SSID)	EnhancedOpen
Client Entry Create Time	21 seconds
Policy Type	WPA3
Encryption Cipher	CCMP (AES)
Authentication Key Management	OWE

Figure 5-44 Security details of a client connected to an Enhanced Open SSID

(Local) Web Authentication

Cisco.com offers a great guide titled *Configuring Web-based Authentication on Cisco Catalyst 9800 Series Controllers* covering the various types of web authentication available on the Catalyst 9800 as well as recommendations and limitations.

Web authentication is a Layer 3 security policy (contrary to central web authentication, which is covered in the next section) and is sometimes called local web authentication because the ACL and portal URL are locally configured on the WLC. It does not mean that the portal cannot be hosted on an external server but only that the configuration of these details has to appear in the WLC configuration. The naming convention

can quickly become confusing around this feature; therefore, it is better and easier to fully refer to each component. The authentication part of the web authentication can be hosted locally on the controller or distributed to an LDAP database or a RADIUS server. The portal page can also be hosted on the WLC (and even customized) or on an external web server (that only has a web server role and does not authenticate the user). For example, saying you need to configure “Local web authentication with a local user database and an external login page” is easier to understand (although much longer to articulate) than saying you are configuring external web authentication. Figure 5-45 depicts the workflow of web authentication, although many different examples would be needed to cover all the possibilities, depending on where each part is hosted, but the concepts stay the same.

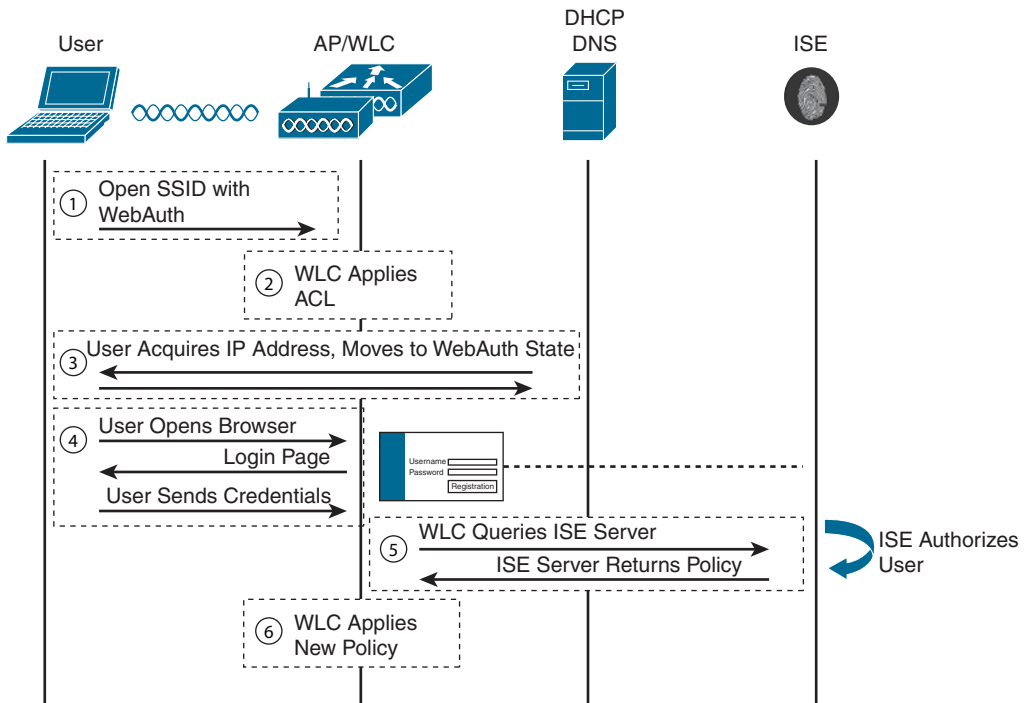


Figure 5-45 Local web authentication workflow

Web authentication can be configured on top of an existing preshared key or even 802.1X SSID. What defines a local web authentication SSID is the fact that the Layer 3 security tab of the WLAN configuration shows one type of web authentication configured. When a client connects to a web authentication SSID, a hidden preauthentication ACL is applied to it; it blocks all traffic except for DHCP and DNS while HTTP traffic is intercepted and redirected for web authentication. The wireless client can obtain an IP address (step 3) and detect the web portal by sending HTTP packets (step 4). The WLC spoofs whatever IP address was the destination of this HTTP packet and replies back with a redirection to the login page (whether it is hosted on the WLC or not). The login page

can potentially be external and is only an HTML repository. The authentication process is completely separate, and the WLC either authenticates the user credentials locally or sends them via LDAP or RADIUS to an external server for validation. When the authentication is complete, the WLC moves the client to the RUN state, and the preauthentication ACL is removed.

A very focused reader wonders how the WLC can know the user credentials if they are entered in a web page that is not necessarily hosted on it. The controller has to use a trick for this. When the page is hosted on the WLC, the controller uses its virtual IP (a nonroutable IP like 192.0.2.1 typically) to serve it. The user credentials are then submitted to it, and this is how the WLC knows the credentials. A similar system is used even if the page is hosted externally: the web redirection sends the client first to the virtual IP anyway, which then sends the user again to the external login page while it adds arguments to the URL, such as the location of the virtual IP. Even when the page is hosted externally, the user still submits its credentials to the virtual IP thanks to this trick so that the WLC can receive the user credentials and authenticate them. If you are to write your own login page, refer to the HTML example provided in the Catalyst 9800 wireless controller download section on cisco.com to see how to have the credentials properly submitted to the WLC.

When we talk about web authentication, it is important to talk about the certificate validation. For a good web authentication experience, it is crucial to install a trusted certificate on the controller and apply it for use with web authentication. There are multiple criteria for a browser to validate a certificate; the main ones are

- The certificate must be within its validity period.
- The certificate must have a Common Name (CN) field that corresponds to the URL visited.
- The certificate must be issued by trusted certificate authorities (CAs).

The second criterion is also related to the fact that you typically cannot issue certificates to IP addresses. You must configure a virtual hostname linked to the virtual IP you are using. Instead of being redirected to the virtual IP, the clients are then redirected to the hostname entered. That's the hostname you need to issue your web authentication portal certificate to. It is also expected that the DNS used by the wireless client redirects requests to the virtual hostname toward the virtual IP.

The third one does not necessarily mean that you need to have a public third-party-signed certificate, but if you use an enterprise PKI, you need to make sure that the root CA is installed on every wireless client (which can be challenging in the case of a guest SSID).

The catch is that if you are hosting your login page on an external web server, the client is presented with two certificates during the login experience: the WLC virtual IP certificate and the external login page certificate.

There are other slight variations of web authentication:

- **Webconsent:** Sometimes also called *web passthrough*, this variation removes the authentication phase. The page typically displays only the terms and conditions and asks the user to agree before letting them through on the network. The page can optionally ask the user to leave an email address, but no validation is performed. This type is used by many external portals relying on social network login credentials because the controller is not able to perform the authentication of the credentials. It is then the login page that is expected to perform the credentials validation, but in the back end, the controller has no means of verifying it.
- **Web authentication on MAC filter failure:** This variation is a MAC filtering SSID where the client is put in a web authentication pending state if the MAC authentication fails and is fully allowed on the network if it is successful.

Central Web Authentication

Central web authentication (CWA) is named this way because the configuration of the web authentication details (such as the ACL and the redirect URL) are configured centrally on the RADIUS servers. This means that the SSID on the WLC is configured with MAC filtering enabled (to have a pretext to send RADIUS requests to the ISE server) and optionally a preshared key or Enhanced Open. The ISE server then replies with an access-accept, along with a redirect URL and redirect ACL, as depicted in step 3 in Figure 5-46. The redirect ACL is a punt ACL that needs to be predefined on the WLC (or the AP in the case of FlexConnect local switching). The ISE server just returns the name of the ACL and not its definition. The redirect ACL defines traffic (matching “deny” statements because it denies redirection for it) that is allowed through on the data plane and traffic (matching “permit” statements) that is sent to the control plane toward the CPU for further processing (that is, web interception and redirection in this case). The ACL has implicit (that is, invisible) statements allowing DHCP and DNS traffic toward all IPs, just like the case with LWA. It also ends with an implicit deny statement, which is acting as security ACL implicit deny (which means it drops traffic). This means that the last invisible statement of the redirect ACL drops all the traffic that does not match any previous punt statement. An example of a redirect ACL would be (considering 10.48.39.28 is the ISE IP address):

```
ip access-list extended REDIRECT
deny ip any host 10.48.39.28
deny ip host 10.48.39.28 any
permit tcp any any eq 80
```

The first two statements allow traffic toward ISE to not be redirected and follow regular processing. The last statement sends all HTTP traffic toward the CPU for URL redirection, and an implicit security ACL ACE “deny” is present afterward to drop all the other types of traffic.

The redirect URL simply defines toward which URL the WLC redirects the guest clients.

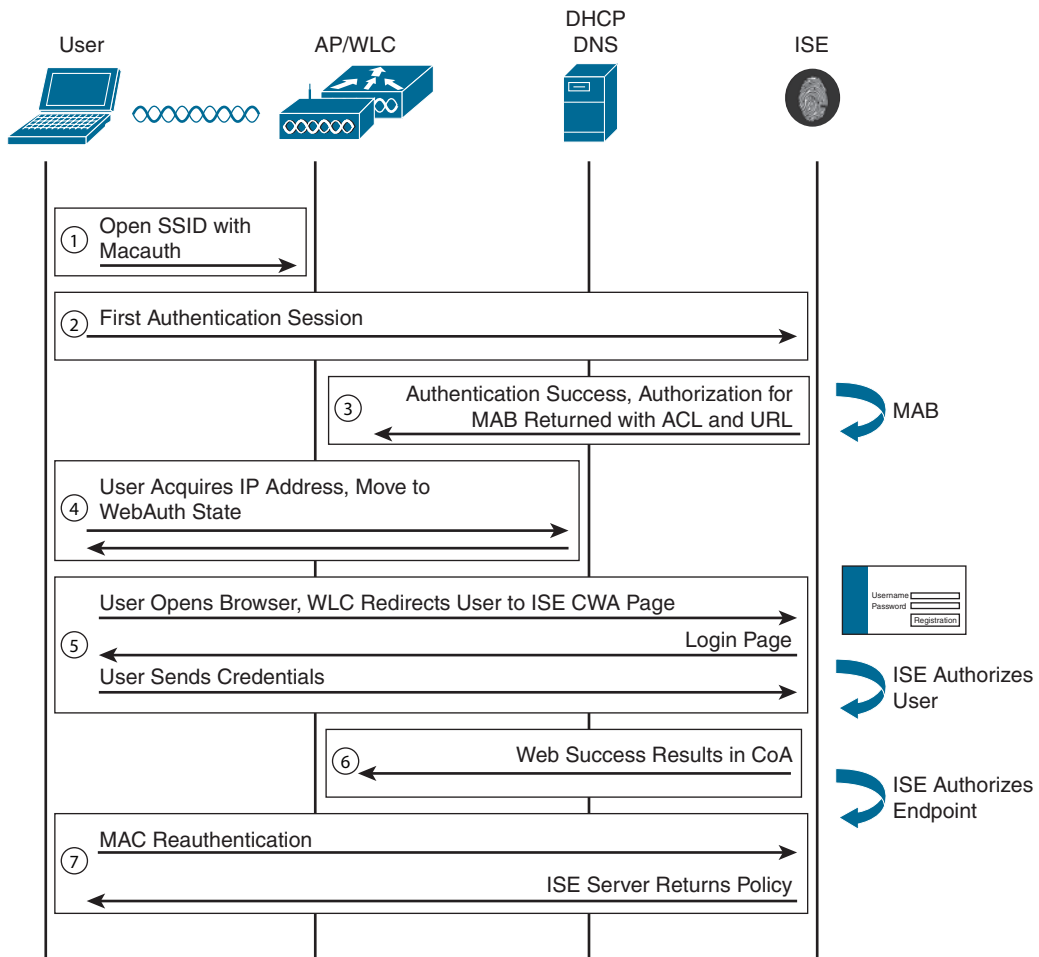


Figure 5-46 Central Web Authentication workflow

The Wi-Fi client can then obtain an IP address (DHCP and DNS traffic are allowed by default, HTTP traffic is intercepted for redirection, and the rest is dropped). All modern operating systems have a portal detection mechanism that works by sending HTTP probe packets to specific test URLs. If the device obtains the requested page, the device must have Internet access. If the device does not obtain the page, that's where you get the famous "limited or no connectivity" sign or any of its variations on your device. However, if the HTTP request gets a reply that is a redirection, it means the device is behind a captive portal. The behavior then varies based on operating systems and even from version to version, but overall, the operating system either shows a pop-up that the user needs to log in to get Internet access or automatically opens a browser window showing the login page. This means that you probably do not have to open your web browser manually to test for a login portal.

The fifth step of the workflow includes all the login details (depending on the type of portal you choose, you can self-register or enter an authorization code) on the portal page(s) until the user has submitted their credentials and received the success message. Something unique about CWA is that after a successful login, the user is basically reauthenticated.

In step 6, the ISE sends a RADIUS Change of Authorization message to the WLC, containing the user session ID (from the audit-session-id attribute, it does not require accounting necessarily) and a reauthentication action. The WLC then restarts all the client state machines back to square one.

Step 7 is then a repetition of step 2 (but with a twist): the WLC sends a MAC authentication request to the ISE, which still simply contains the MAC address of the wireless client to the ISE. However, the ISE remembers that this client just logged in and appends an internal attribute containing the client username entered on the portal page to the authentication workflow. The client authentication can then hit authorization rules that are based on the client user identity and return any other authorization attribute: a VLAN (even if it is not recommended to change the VLAN of a client after it already obtained an IP address), a session timeout, an ACL, an SGT, and so on. This “trick” is done because the portal authentication happens via HTTPS, and the WLC knows nothing from the credentials that the user typed in the login page; therefore, no RADIUS authentication can happen at that stage containing the portal page username. This way, it is still possible to return various authorization results depending on the client login page identity, and they are applied during the Layer 2 authentication phase. The key is that the second MAC authentication taking place returns an access-accept that can contain various authorization attributes but should not contain any redirect URL/ACL again; otherwise, the portal page is shown again to the user in a loop.

One benefit of the CWA workflow is that the WLC Virtual IP is not used, and therefore, the WLC does not have to present its certificate to the user. Only the ISE login portal page gives its certificate to the client, so this reduces some of the administrative burden. Another benefit is that there is a load balancing of the portal pages taking place naturally: the redirect URL points to the portal page of the specific ISE policy node against which the client authenticated previously. This means that your portal load balancing is, in fact, following your RADIUS load-balancing strategy.

Web Authentication Best Practices

There are many mistakes when it comes to web authentication that lead to a poor user experience or simply a nonworking wireless network. Let’s look at the most common points.

HTTPS Redirection

HTTPS redirection is bad and should never be configured. When doing an HTTP redirection, the wireless client triggers the TCP handshake (on port 80) with whatever website it requested, but in reality, the WLC is intercepting this traffic, spoofing the website IP, and doing the TCP handshake. The client then sends its HTTP GET to request whatever page it intended to, and the WLC replies with a redirection to the login page. Intercepting HTTP sessions is not too hard for the controller, and there are no obstacles to task.

When doing an HTTPS redirection, the wireless client still triggers the TCP handshake (on port 443), and the controller still has an easy time spoofing the website IP address and completing the TCP handshake. But before the client can send a GET request, the TLS handshake of HTTPS must happen, and the server side (the controller here) is supposed to send a certificate to prove its identity. Even if the controller has a valid certificate installed, this certificate is issued to the controller hostname and definitely not to whatever website the client was requesting. This is the whole point of HTTPS—being able to verify the identity of the site you are visiting—and therefore, any other certificate presented depicts a session hijacking (which web authentication is, in fact).

There are no good solutions to this situation because this is the whole point of the protocol and why it is secure in the first place. The controller sends its certificate to complete the TLS handshake. The client browser always throws at least an error on this, and most of the time it is even impossible to ignore or avoid. Some browsers do not even show the page or do not show any error at all. This is why HTTPS redirection is a broken concept to begin with. On top of that, simply enabling it is extremely resource intensive for the controller.

The rationale behind this feature comes from people noticing that the Internet is now mostly HTTPS, and therefore, when a user opens up a browser, there is a good chance that an HTTPS URL is entered or that the home page is HTTPS, even if the user does not type **HTTPS** in the address bar. Therefore, for a time, this was indeed an issue, and the only way to have people notice there is a portal page was to enable HTTPS redirection. However, as explained in the previous section on CWA, most operating systems today have a portal detection mechanism relying on HTTP probes. Even for operating systems not benefiting from this feature, web browsers themselves implement this detection too and send an HTTP probe when opened to warn the user whether a captive portal is in place. There are therefore little to no reasons to still need to have HTTPS redirection enabled.

Some networks have configured all their laptops to use a web proxy to access the Internet or even an intranet, which means the clients never send their packets on ports 80 or 443 but typically on a higher port used by the web proxy. This issue is once again solved by relying on the operating system HTTP probe that is sent on port 80 no matter what the proxy configuration in the browser is.

Captive Portal Bypass

Captive portal bypass is a feature that administrators often enable out of a trial-and-error process when something is not working right with web authentication. It is important to understand its origin and purpose. Apple was the first company to implement the captive portal detection system on its company phones. What is now considered a great feature started with a few concerns because the captive portal page is not opened in the favorite or default browser but in a pseudo-browser called Apple Captive Network Assistant. The problem was that this pseudo-browser did not support dynamic pages very well and would display a blank page on many bring-your-own-device (BYOD) portals. The captive portal bypass feature on the WLC then made the WLC spoof an “HTTP 200 OK” reply from the captive detection URL, pushing the client to believe it had full Internet connectivity and therefore not show any pop-up at all. The user was then expected to open

their browser and get redirected to the captive portal BYOD page displayed on this full-blown browser. The problem, as mentioned previously, is that nowadays most websites are HTTPS, so this workflow would typically not work at all. But this is not a concern because both the Apple CNA pseudo-browser solved some of its limitations, and the ISE BYOD portal page got adapted to fully display on those pseudo-browsers. Android now also uses a pseudo-browser, and no problems are reported with the vast majority of captive login pages. It is therefore recommended to disable captive portal bypass in the webauth parameter map, as shown in Figure 5-47, and to fully rely on the captive portal detection system. One limitation of those pseudo-browsers, still, is their inability to do the client provisioning flow from ISE.

The screenshot shows the 'Edit Web Auth Parameter' configuration window. The 'General' tab is selected. The 'Captive Bypass Portal' checkbox is unchecked and highlighted with a red box. Other settings include: Parameter-map name: global; Banner Type: None; Maximum HTTP connections: 100; Init-State Timeout(secs): 120; Type: webauth; Virtual IPv4 Address: 192.0.2.1; Trustpoint: myc9800-CL_WL...; Watch List Expiry Timeout(secs): 600. Buttons for 'Cancel' and 'Update & Apply' are visible at the bottom.

Figure 5-47 Do not enable Captive Bypass Portal unless you really have a good reason

Web Authentication Takeaways

It is important to summarize the key points of web authentication because they will save you a lot of headaches when deploying your configuration:

- Do not enable HTTPS redirection; it's pointless.
- Do not enable captive portal bypass.
- Make sure you have valid certificates on your portal page (always) and on the WLC (if you are not using CWA).

- You need a working DNS. Really.
- Rely on the client devices' captive portal detection pop-up.
- Configure both an IPv4 and an IPv6 virtual IP; there's little reason not to.

RFC 8910 is also something to watch out for because it is a new standard for detecting captive portals. Whether it becomes popular will depend on its client adoption rate.

Rogue Detection and WIPS

Rogue detection, WIPS, and more generally protecting against wireless attacks are covered in Chapter 7, "RF Deployment and Guidelines."

Securing Your Access Points

Securing your wireless networks doesn't mean much if your APs are not secured as well. If anyone can join an access point to your network and start broadcasting your WLANs in unwanted places, you have a problem. If your wired network simply has strict security guidelines, you need to set some policies in place for the wireless network to fit in with the existing policies.

AP Authorization

If any AP plugged to your network is allowed to join your WLC and broadcast the configured SSIDs, that configuration is considered quite insecure. Even if you don't consider this a concern because somehow your wired network is physically very secure, you may have several WLCs in your network and may still look for a way to make sure which WLC an AP can join and which WLC it can't. Some people think DHCP Option 43 covers this requirement, but Option 43 only allows the AP to learn about WLC IP addresses. If the AP learns another WLC IP address (via broadcast or via the mobility group), it tries to discover it too. Enabling AP authorization on a WLC means that globally, no AP is allowed to join until its MAC address is verified by the WLC. MAC addresses have to be entered in the **Device Authentication** page under the **AAA Advanced** section in the *aabbccddeeff* format, that is, without any separator.

The **AP Policy** page under **AAA Advanced** allows you to enable the authorization of APs against MAC addresses globally and to pick the AAA method that authenticates them (you can use a list on the WLC itself or use a AAA server). The AAA method you need to select refers to the command **aaa authorization credential-download**.

An option to authorize APs against their serial number has also been available since IOS-XE 17.3.2.

Here's a summary of the CLI commands required:

```
9800# config t
9800(config)# aaa new-model
```

```

9800(config)# aaa authorization credential-download <method name>
local
9800(config)# ap auth-list authorize-mac
9800(config)# ap auth-list method-list <method name>
9800(config)# username <aaaabbbbcccc> mac

```

This topic is covered in more detail in the configuration example titled *Catalyst 9800 Wireless Controllers AP Authorization List* on cisco.com.

Whether or not you have configured AP authorization, any AP that is in Bridge mode has to be added to the device authentication list. This explains the confusion people experience when they order an outdoor AP like a 1562 or a 9124, receive it as a local mode AP, and have no problems joining it to the WLC. Then, when converting the AP to Bridge mode or Flex+Bridge mode, the AP does not join anymore. If checking the wireless join statistics (from the widget in the WLC home page that tells you some APs are disjoined), you see they are stuck in “AP auth pending” status.

On top of having their Ethernet MAC address verified when they join, the Bridge mode APs need to authenticate themselves either through PSK or through 802.1X to the controller. The reason is that anyone could be joining a Mesh mode AP to your backhaul and joining your WLC, so some form of authorization is always required.

The configuration example titled *Configuring Mesh on Catalyst 9800 Wireless LAN Controllers* is a good resource to learn more about configuring Bridge mode APs.

AP 802.1X Authentication

A secure wired network often means implementing security at the switchport level. Having good physical security on your premises is one thing, but even your own employees could be plugging unwanted devices in the network. And if you think about it, if the only thing it takes to get access to sensitive resources or a management VLAN is to find the right switchport and connect to it physically, you can understand that this is vastly unsecure by today’s standards. Many networks implement 802.1X authentication on most wired ports accessible to access devices. There are often exceptions for server ports or IoT devices that cannot perform 802.1X authentication, but that’s another story. Considering how easy it is to climb on a chair and remove an AP from the ceiling and use its network cable, securing the AP switchport is definitely good sense. Enabling 802.1X on the switchports where APs are connected means that when the cable gets plugged in, the only traffic that is allowed to pass is the EAP authentication protocol until the authentication succeeds, and only then can the AP get an IP address and send some traffic. Not much is required to configure on the wireless to achieve this: the main thing required is to configure credentials on the access points. You obviously need to configure those credentials somewhere in the user database that your RADIUS server uses and need to configure that RADIUS server appropriately from an authorization policy standpoint.

You can configure 802.1X credentials for the access points in the AP join profile under the **Management > Credentials** tab (not globally anymore, as was the case in AireOS controllers). You simply have to set a username and password, and the AP tries to trigger an EAP-FAST authentication with those credentials when connected to the network. There is no harm in configuring the 802.1X credentials even if the AP is connected to a non-802.1X switchport. Now the problem is that this only works for APs that join the WLC once to get this configuration provisioned on the AP. That can work if your APs are staged or provisioned on an unsecured port before being placed in the secure network. Alternatively, you can enter the credentials directly on the AP console if it is plugged directly to an 802.1X-enabled port. The command is `capwap ap dot1x username <user> password <password>`.

The 802.1X is typically used on access switchports to authenticate hosts, but an access point can also be in FlexConnect mode and therefore connected on a trunk port. Having the AP perform 802.1X authentication on a trunk port is also supported, provided that the switchport authentication is set to multihost mode. First of all, this allows several MAC addresses to be seen on the switchport (802.1X normally limits every switchport to communicate with only one MAC), and only the first MAC address (logically the AP) is authenticated. All the extra MAC addresses learned afterward are applied the same authorization result as the AP.

Using 802.1X to authenticate access points is also a great way of assigning a specific VLAN dynamically to the AP. This means you do not need to have static VLAN configurations on your ports, and depending on the username authenticating, the right VLAN (a user VLAN or the AP management VLAN) is dynamically assigned. Another way to achieve a similar goal is to use smartport macros that detect whether an AP is connected via CDP learning and is able to apply more switchport configuration lines. However, the CDP detection mechanism is probably a bit less secure than a full-blown 802.1X authentication.

One important point for AP 802.1X to work on trunk ports is that if you configure periodic reauthentication on your switchport, you need to make sure to configure **Termination action** (RADIUS attribute 29) to be **Radius-request** (a value of 1). If left to the default, it means that the AP loses network connectivity for the time of the reauthentication. When you set this attribute, the AP is able to keep sending traffic during the reauthentication until the reauthentication completes successfully.

Securing the AP Join Process Using Locally Significant Certificates

The concept of configuring LSC is simple: you decide to issue certificates yourself to each and every access point you own, and the WLC allows only APs with such a certificate to join, therefore securely preventing any new APs from joining without your authorization. LSC requires owning an enterprise PKI that automatically issues a certificate to each access point. This certificate distribution happens during a provisioning phase that you define where any AP that joins is automatically provided with a custom certificate signed by your enterprise CA. When you determine that provisioning phase to be over, only APs with a valid enterprise-signed certificate are allowed to join. This certificate

supersedes the typical Cisco-manufactured MIC certificate (which isn't deleted but stays as backup) for establishing the DTLS connection.

This solution is much more secure than relying on a MAC address as well as much more scalable, but it requires an enterprise PKI supporting Simple Certificate Enrollment Protocol (SCEP). This procedure is covered in the document titled *Configure SCEP for Locally Significant Certificate Provisioning on 9800 WLC* on cisco.com.

Until 17.5.1, there was a limitation where the enterprise CA used for SCEP had to be a root CA, but intermediate CAs are now supported as well.

Securing Your Wireless Controller

Network devices must be secured because they are a target of choice for attackers. When an attacker can access one network device, that attacker can typically easily access more network resources. The controller is an excellent target because it manages the configuration of the whole wireless infrastructure. Let's review a couple of ways to protect unintended administrative access to the C9800.

Securing Administrator Access

One of the first measures to secure your wireless controller is to restrict administrator access to it. Leveraging local users stored on the controller can work only for small teams. In large companies, where more administrators have access to the controller and even just for simplicity of password management (having your credentials stored securely in one place is better than having them all over the network on each device, especially if you have to change your password), it becomes critical to use RADIUS or TACACS to authenticate users accessing your device as a network administrator.

Using TACACS+

A configuration example titled *Configure RADIUS and TACACS+ for GUI and CLI Authentication on 9800 Wireless LAN controllers* is available on cisco.com to better illustrate the process of using TACACS+. Let's cover it from a conceptual perspective nonetheless.

The first step is to configure the TACACS+ server on the WLC. You can easily do this in the **Configuration > Security > AAA > Servers/Groups** page. You need to specify an IP address and a shared secret.

The second step is to configure AAA methods. You need an **aaa authentication login <methodname> group <tacacs server group>** and an **aaa authorization exec <methodname> group <tacacs server group> method**. The login method takes care of authentication, and the exec authorization method is required to allow the user to enter any command (and basically do anything on the device). Because these are named methods, they do not have any effect until they are called somewhere else in the configuration.

To use the TACACS+ server for CLI authentication, add your method in the VTY configuration (it is also configurable under **Administration > Management>HTTP/HTTPS/Netconf/VTY** in the WebUI after 17.6):

```
9800(config)#line vty 0 15
9800(config-line)#login authentication <aaa login method name>
9800(config-line)#authorization exec <aaa exec method name>
```

To use it for WebUI:

```
9800(config)#ip http authentication aaa login-authentication <aaa
login method name>
9800(config)#ip http authentication aaa exec-authorization <aaa exec
method name>
```

You can also configure this from the WebUI in the **Administration > HTTP/Netconf** page, as shown in Figure 5-48.

Figure 5-48 VTY configuration for CLI access in the HTTP/Netconf web page

For the settings to take effect, you might need to restart the web server by typing **no ip http server** followed by **ip http server**.

Some extra configuration is required on ISE (refer to the configuration example for more), but in a nutshell, you need to return two things on top of a successful authorization result:

- A TACACS+ shell profile allows you to set some attributes, and the most important for this situation is the default privilege level. This is the privilege level at which the user ends up after connecting. Although it ranges from 0 to 15, there are no differences by default between all the values ranging from 1 to 14. A privilege level of 0 allows you only some basic commands, and a privilege level of 15 allows full access.
- On top of a TACACS profile, you can also assign a command set from ISE, which is a list of CLI commands that the user is allowed to run. For each command, the WLC asks for authorization to run the specific commands. You can use wildcards or allow all but a subset of commands if you want.

Using RADIUS

Restricting WLC administrator access through RADIUS follows the exact same concept as with TACACS+. Obviously, you define a RADIUS server in lieu of a TACACS server, but the AAA methods and other commands stay the same. One behavioral difference with RADIUS is that it does not ask for authorization every time a CLI is entered: no shell profiles and command sets can be used. Instead, you need to edit your RADIUS authorization result and return a `cisco-av-pair` with the value `shell:priv-lvl=15`.

Guest Users

Guest users are created by the administrator or the lobby ambassador, which we discuss in more detail next. They have no device administration access (SSH or GUI). They are only able to connect to the network and access the Internet. They can also be temporary and be deleted after a set timer.

The Lobby Ambassador Type of User

There is an extra type of administrative user who is not simply defined by the privilege level. The lobby ambassador is a type of user who can be created on the WLC (either on WebUI or CLI) and whose only purpose is to create guest users. When you log in as a lobby ambassador, your only privilege is to create guests, and you get a special WebUI dedicated to this activity. In previous releases, it was tricky to authenticate lobby ambassadors via TACACS because it required configuring a command on the WLC for each ambassador username. After release 17.5, the only thing you require from ISE is to return the mandatory `user-type` attribute with value `lobby-admin` as a custom attribute in the shell profile. Without this, your lobby ambassador gets full administrator privilege on the controller, which you probably want to avoid. This process is covered in detail in the configuration example titled *Configure 9800 WLC Lobby Ambassador with RADIUS and TACACS+ Authentication*.

NETCONF

NETCONF is what Cisco DNA Center mostly uses to configure the Catalyst 9800 and a protocol you can use yourself as well, with custom scripts or third-party tools even. At the time of this writing, NETCONF security is ruled only by the default login authentication and exec authorization methods. This means that you need to set `aaa authentication login default local` and `aaa authorization exec default local` to have it working. It is also possible to point these methods to a RADIUS or TACACS server for authentication if you want. A common catch is that the management stations (Prime Infrastructure or Cisco DNA Center) tend to overwrite these config lines and point them back to local, so you may want to have a backup administrative user on the WLC to keep accessing it in case of trouble. The NETCONF implementation of the 9800 WLC uses SSH over port 830 and currently does not allow you to manually select ciphers or the certificate used for authentication although this may change in the future.

Granularity of WebUI Access

The WebUI is an interface that uses a hybrid data model in the background. For some pages, it actually opens a VTY line and enters CLI commands (which are authorized for each of them if you are using TACACS, for example), whereas some other pages use a TDL programmability model and do not use a CLI back end at all. This means that the WebUI is affected by changes you make to the CLI authorization method. At this time, there is no way to restrict a WebUI user from using certain pages and not others: a user either has privilege level 15 and has full access to the WebUI or has privilege level 1 and has access only to the dashboard and some monitoring subpages.

Connect to the WebUI Using Certificates

In the HTTP(s) administration page on the WLC, you can enable Personal Identity Validation. You then have to point to a trustpoint. This feature restricts the WebUI connection to only devices that can present a client certificate that will be trusted according to the designed trustpoint on the WLC. HTTPS websites always require the server to present its certificate, but optionally the client can do the same. This means you could get rid of credentials and require a smart card or certificate to be able to access the 9800 WebUI.

Securing Traffic

Another key to securing your controller is to make sure it is not receiving unwanted traffic and that only the right people connecting from the right locations can access it. By default, your wireless clients are not allowed to access the WLC command line or WebUI. To lift that restriction, you need to enable **Management Via Wireless** under **Configuration > Wireless > Wireless Global**.

A Catalyst 9800 controller can be managed, using any of its SVI IP addresses. It is recommended that you minimize the number of SVIs you configure on the 9800 for this reason, unless you have specific needs for SVI (in the case of mDNS proxy, for example, SVIs in wireless client VLANs are required). You can restrict access by applying ACLs on the 9800 SVIs.

Catalyst 9800 wireless controller appliances provide you with a service port to manage the WLC. The specificity of this SP is to be really out-of-band; that is, the port is physically connected to the WLC CPU and separated from the other data ports and the data plane. From a software perspective, IOS-XE places this port in a hard-coded VRF called mgmt-intf. You cannot change this VRF configuration on the service port. You can add routing for the service port, but it has to be part of the mgmt-intf VRF.

Encrypted Traffic Analytics

The fact that more and more network protocols are encrypted means it is harder and harder to analyze the traffic going through the network. Encrypted Traffic Analytics (ETA) is a solution from Cisco (its flow is illustrated in Figure 5-49) where network devices all report information from the network traffic to a Cisco Secure Network Analytics (formerly Stealthwatch) appliance that runs a machine learning algorithm to be able to iden-

tify threats and attacks, not based on the traffic content but on the traffic patterns. The Catalyst 9800 can be part of this solution, leveraging the Flexible Netflow configuration directly on the C9800 (not on the FlexConnect APs themselves). You can find more information in the *Encrypted Traffic Analytics White Paper* and the configuration guide.

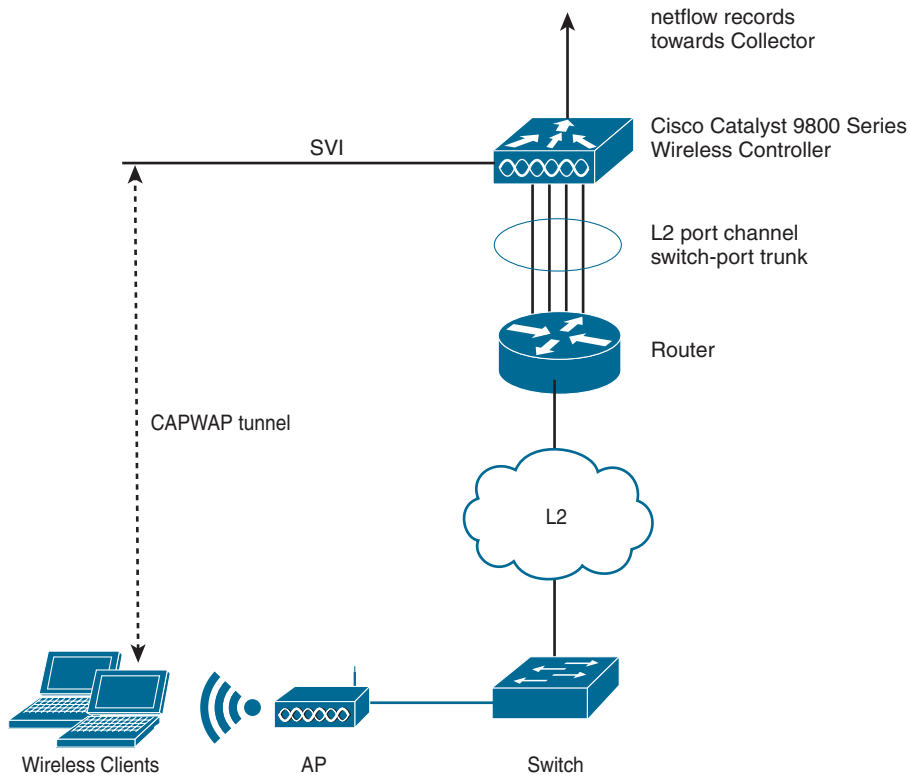


Figure 5-49 *ETA workflow topology*

Cisco Umbrella

Cisco Umbrella is a cloud DNS service. The first question that may come to your mind is “What is the relation with wireless?” Well, many administrators try to secure network access by restricting the content that wireless users can access. This can be done with AVC if you want to target specific applications, but websites are more complicated because you would need to do URL filtering. Solutions exist in the shape of web proxies, firewalls, or content engines, but those indeed have little integration with wireless. First of all, Cisco Umbrella is a very fast DNS server that helps guarantee a very low latency of DNS resolutions and accelerates web browsing. On top of that, because it is cloud managed, it is constantly updated with the latest safety against malware and helps you to not resolve malicious domain names. Simply configuring a client to use Umbrella as a DNS server is enough to provide some protection. On top of that,

Umbrella offers an administrator dashboard that allows you to restrict what URLs can be resolved in your network. Without personally choosing to block website X or Y, you can decide to ban all adult-content websites or all websites promoting violence, for example. You can also, of course, add specific sites to be allowed or blocked, as well as add personal URLs that you want resolved (as if it were your private DNS server). Rather than relying on antivirus software analyzing the threats downloaded to your clients, Umbrella prevents your clients from accessing the malicious content, an approach that is much more efficient and secure.

How does Umbrella relate to wireless though? The integration allows you to do two things:

- Make sure that your clients use Umbrella as a DNS server.
- Allow you to mark the DNS traffic coming from your network toward Umbrella to use specific DNS policies on Umbrella. For example, you may want to allow more websites on your secure employee SSID but block a lot more things on your guest SSID (such as illegal software, movies, and VoD websites).

You must first register your controller to the Umbrella cloud by going to the **Configuration > Security > Threat Defense > DNS Layer Security/Umbrella** page (as shown in Figure 5-50), where you can create a global umbrella parameter-map in your configuration. You can obtain the token from the Umbrella cloud account and enter it on the controller. You can then enable DNS packet encryption (which means the WLC encrypts the packets it sends to Umbrella) and create parameter maps. Those are just names that you are able to apply to a policy profile to identify requests on the Umbrella cloud site.

Configuration > Security > Threat Defense > umbrella

Registration Token* [Click here to get your Token](#)

Organization ID

Allowed Domains

Enable DNS packets encryption

Umbrella Parameter Map

Figure 5-50 Umbrella global configuration on the 9800

On the policy profile advanced settings section, you can choose one of the parameter maps you created to segregate DNS policies on the Umbrella cloud. By default, the controller injects the Umbrella DNS IP address into DHCP option 6 (which points to the DNS servers) when the clients receive their IP address so that you don't even have to configure the Umbrella IPs in your DHCP pools. The other settings depicted in Figure 5-51 allow you to also inject the DHCP option 6 setting in Flex APs when the traffic is locally

switched and to force the DNS traffic redirection. This means that even if the client then manually configures a custom DNS server, the WLC intercepts the DNS queries anyway, sends them to OpenDNS, and spoofs the original DNS IP back so that the client doesn't notice that it's another DNS server that replied.

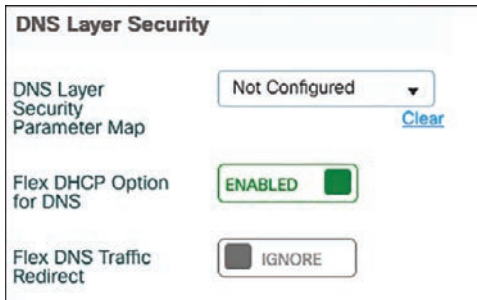


Figure 5-51 DNS Layer Security (formerly called Umbrella) section of a policy profile.

Cisco Secure Development Lifecycle (CSDL)

Cisco Secure Development Lifecycle is at the heart of new Cisco development, and the Catalyst 9800 controller fully embraces the CSDL concepts. These requirements are broad and range from code validation to vulnerability and penetration testing, to default settings on the devices or the use of third-party modules in the code. Security is a never-ending journey, and the Catalyst 9800 controller keeps implementing stricter and stricter security guidelines and principles with every release.

Summary

This chapter covers the security aspects of the Catalyst 9800 controller. The biggest aspect revolves around AAA, which is either locally handled by the controller or delegated to an external RADIUS or TACACS server. Security also consists of ACLs to restrict the traffic that clients can pass or to protect the controller management plane from undesired access. Encrypted Traffic Analytics, rogue detection and WIPS, and Cisco Umbrella are other security components that can help secure your overall solution. Security happens at every layer of the OSI model and is an all-encompassing topic.

References

Trustpoints on 9800: <https://www.cisco.com/c/en/us/td/docs/wireless/controller/9800/config-guide/trustpoints/b-configuring-trustpoints-on-cisco-catalyst-9800-series-controllers/m-overview-of-trustpoints-on-catalyst-9800.html>

Generate and download CSR on Catalyst 9800: <https://www.cisco.com/c/en/us/support/docs/wireless/catalyst-9800-series-wireless-controllers/213917-generate-csr-for-third-party-certificate.html>

Configuring WLC with LDAP for 802.1X and web-auth: <https://www.cisco.com/c/en/us/support/docs/wireless/catalyst-9800-series-wireless-controllers/216744-configuring-catalyst-9800-wlc-with-ldap.html>

Local EAP configuration on 9800: <https://www.cisco.com/c/en/us/support/docs/wireless/catalyst-9800-series-wireless-controllers/215026-local-eap-authentication-on-catalyst-980.html>

List of IOS-XE wireless features per release: <https://www.cisco.com/c/en/us/support/docs/wireless/catalyst-9800-series-wireless-controllers/214855-ios-xe-wireless-feature-list-per-release.html>

Wireless controller AP authorization list: <https://www.cisco.com/c/en/us/support/docs/wireless/catalyst-9800-series-wireless-controllers/213916-catalyst-9800-wireless-controllers-ap-au.html>

Configure SCEP for LSC certificate on 9800 WLC: <https://www.cisco.com/c/en/us/support/docs/wireless-mobility/wireless-lan-management/215557-configure-scep-for-locally-significant-c.html>

Configure RADIUS and TACACS+ for GUI and CLI authentication on 9800 wireless LAN controllers: <https://www.cisco.com/c/en/us/support/docs/wireless/catalyst-9800-series-wireless-controllers/214490-configure-radius-and-tacacs-for-gui-and.html>

Configure Lobby ambassador with RADIUS and TACACS authentication: <https://www.cisco.com/c/en/us/support/docs/wireless/catalyst-9800-series-wireless-controllers/215552-9800-wlc-lobby-ambassador-with-radius-an.html>

Cisco CSDL: https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/cisco-secure-development-lifecycle.pdf

Index

Numerics

- 11k assisted roaming, 218–221
- 802.1X, 183–184. *See also* EAP (Extensible Authentication Protocol)
 - client join state machine processing, 51–53
 - components, 184–185
 - EAP, 185–189
- 802.11, 298–299, 301–302. *See also* wireless networks
- 802.11ax, Fastlane+, 319–320
- 802.11v BSS Transition, 221–222

A

- AAA (authentication, authorization, and accounting), 103–104. *See also* authentication
 - attributes, 113
 - LDAP (Lightweight Directory Access Protocol), 116
 - methods, 112–114
 - RADIUS (Remote Authentication Dial-In User Service)
 - accounting*, 111
 - AVPs (attribute-value pairs)*, 105
 - CoA (Change of Authorization)*, 107–108
 - EAP exchange*, 106–107
 - iPSK (Identity PSK) and*, 132–138
 - load balancing*, 111
 - packet types*, 104–105
 - securing administrator access*, 153
 - server configuration*, 108–109
 - server fallback*, 110–111
 - server group configuration*, 108–110
 - service policies and, 281
 - TACACS+ (Terminal Access Controller Access Control System), 114–115, 151–152
- aaa accounting identity command, 112
- aaa authentication dot1x command, 112
- aaa authentication login command, 112
- aaa authorization credential-download command, 112, 148–149
- aaa authorization exec command, 112
- aaa authorization network command, 112
- aaa new-model command, 112, 439
- accounting, RADIUS (Remote Authentication Dial-In User Service), 111
- accuracy, location tracking, 362–363
- ACE (access control entry), 90
- ACLs (access control lists), 89–90
 - ACE (access control entry), 90
 - defining, 90–91
 - DNS-based, 96–97
 - downloadable, 95–96
 - FlexConnect, 94–95
 - log keyword, 91

- preauthentication, 93–94
- redirect, 143–144
- wildcard mask, 89–90
- wireless, applying on the WLC, 93–94
- adaptive WIPS, 244–245**
- administrator access**
 - lobby ambassador, 153
 - securing
 - using RADIUS, 153*
 - using TACACS+, 151–152*
- advanced RF features**
 - aggressive load balancing, 226
 - ATF (air time fairness), 228
 - Band Select, 225–226
 - off-channel scanning defer, 227
- AES (Advanced Encryption Standard) algorithm, 116–117**
- aggressive load balancing, 226**
- AIFSN (Arbitrated Interframe Space Number), 287–288**
- AireOS, 43**
 - C9800 configuration model and, 46–47
 - FlexConnect, 45, 47–48
 - IRCM (inter-release controller mobility), 191–192
- AireOS Config Translator, 540–541**
- analytics**
 - device, 479
 - Apple, 479–480*
 - Samsung, 481*
 - Intel, 481–483
 - location tracking, 366
 - network services, 477–479
- antennas**
 - gain, 196
 - self-identifying, 197
- ap country command, 199**
- ap tag persistency enable command, 51**
- Apple**
 - analytics, 479–480
 - Bonjour, 272
 - Fastlane+, 319–320
- applying, ACLs (access control lists), 91–93**
 - FlexConnect, 94–95
 - wireless, 93–94
- APs, 160. *See also* roaming; RRM (radio resources management)**
 - authorization, 148–149
 - CAPWAP split MAC architecture, 25–27
 - challenging RF environments, 199–200
 - atriums, 202*
 - high-density crowd areas, 200*
 - metal-heavy areas, 200*
 - shielded doors and sudden turns, 201*
 - uneven ceilings, 201–202*
 - channel width, 232
 - CleanAir, 221–224
 - country configuration, 197–199
 - coverage hole detection, 210–211
 - data collection, 203–206
 - DCA (Dynamic Channel Assignment), 211–213
 - DFS (Dynamic Frequency Selection), 232–235
 - FlexConnect ACLs, applying, 94–95
 - FRA (Flexible Radio Assignment), 235–236
 - health monitoring, 559–563
 - Hyperlocation module, 362
 - image predownload, 503–505
 - ISSU (In-Service Software Upgrade), 506
 - join process
 - static configuration, 83*
 - using PnP, 84*
 - local mode, 57–63
 - LSC (locally significant certificates), 150–151
 - master mode, 75
 - mDNS, 281–282
 - moving between wireless controllers, 54
 - off-channel scanning defer, 227
 - power levels, 197
 - RF neighborhoods, 203
 - roaming, 61, 160–161, 181
 - 11k assisted, 218–221*
 - 802.11 authentication, 197*
 - 802.11v BSS Transition, 221–222*
 - client access policies, 161*
 - fast, 163–176*
 - inter-controller, 185–188*

- intra-controller*, 181–184
 - L2 authentication*, 161
 - between policy tags*, 55–57
 - RF scanning*, 160
 - slow*, 161–163
 - rogue, 31, 238, 241
 - classifying*, 240–241
 - containing*, 241–243
 - detecting*, 238–240
 - rolling upgrade, 505–506
 - RSSI (received signal strength indicator), 160
 - sticky clients, 210–211
 - tags, assigning, 48–54
 - tri-radio operations, 236–237
 - txpower, 196–197
 - architecture**
 - Cisco DNA Center, 471–472
 - IOS-XE
 - binOS*, 27
 - control plane*, 37
 - data plane*, 35–37
 - wireless client state machine*, 31–35
 - wireless processes*, 31
 - WNCd (Wireless Network Control Daemon)*, 28–31
 - OpenRoaming, 379–381
 - split MAC, 25–27
 - ARP proxy**, 30
 - ASIC (application-specific integrated circuit)**, 5
 - assigning, tags**, 48–54
 - ATF (air time fairness)**, 228
 - attributes, AAA**, 113
 - authentication**
 - 148.1X, 183–184
 - components*, 184–185
 - EAP*, 185–189
 - FT, 171–172
 - L2, 161
 - RADIUS (Remote Authentication Dial-In User Service), 104
 - AVPs (attribute-value pairs)*, 105
 - CoA (Change of Authorization)*, 107–108
 - EAP exchange*, 106–107
 - iPSK (Identity PSK) and*, 132–138
 - load balancing*, 111
 - packet types*, 104–105
 - server configuration*, 108–109
 - server fallback*, 110–111
 - server group configuration*, 108–110
 - TACACS+ (Terminal Access Controller Access Control System), 114–115
 - web, 140–143, 147–148
 - captivate portal bypass and*, 146–147
 - central*, 143–145
 - HTTPS redirection and*, 145–146
 - authorization, AP**, 148–149
 - auto QoS**, 310–313
 - auto-anchoring**, 187–188
 - automate-tester username command**, 111
 - automation**, 3. *See also* IBN (intent-based networking)
 - declarative model, 399–400
 - imperative model, 397–399
 - AVC (Application Visibility and Control)**, 316–319
 - AVPs (attribute-value pairs)**, 105
-
- ## B
- backing up**
 - using Cisco DNA Center, 498–499
 - using Cisco Prime Infrastructure, 497
 - using the WebUI, 496–497
 - bandwidth**, 285
 - best practices**
 - Catalyst 9800-CL, private cloud deployment, 72–73
 - QoS, 320–322
 - binary tracing**, 511
 - always-on, 515–520
 - decoding, 513–515
 - rotation, 511
 - severity levels, 512–513
 - BinOS**, 27, 520–521
 - BLE (Bluetooth Low Energy)**, 367–368, 389–392. *See also* IoT (Internet of Things)

Bluetooth, 365, 366–367, 371–372. *See also* IoT (Internet of Things)
 Bonjour, 272. *See also* mDNS (multicast DNS)
 bootstrap configuration, 66–67
 BSS (Basic Service Set)
 coloring, 231–232
 overlapping, 213–214
 bss-transition disassociation-imminent command, 1614

C

C9800-CL, 10. *See also* Catalyst 9800 series wireless controllers
 CAC (call admission control), 265–266, 298–299
 Captive Portal, 386–389
 captive portal bypass, web authentication and, 146–147
 CAPWAP (Control and Provisioning of Wireless Access Points)
 multicast, 254
 MoM mode, 256–258
 MoU mode, 254–255
 split MAC architecture, 25–27
 capwap fallback command, 355
 Catalyst 9800 series wireless controllers.
 See also configuration model
 11k assisted roaming, configuring, 220–221
 802.11v BSS Transition, configuring, 222
 ACLs, writing, 91
 adaptive WIPS, 244–245
 adding a certificate, 98–102
 AP join
 static configuration, 83
 using PnP, 84
 ARP proxy, 30
 ATF (air time fairness), configuring, 228
 AVC (Application Visibility and Control), 316–319
 Band Select settings, 225–226
 CAPWAP split MAC architecture, 25–27
 Cisco DNA Center and, 472–473
 Cisco RF ASIC, 6–7
 CleanAir, configuring, 222
 client delete reason codes, 567–568
 cloud networking and, 9–10
 configuration database, 402
 control plane tracing, 508
 country configuration, 197–199
 Day Zero setup, 80–82
 DFS (Dynamic Frequency Selection) settings, 232–235
 FRA (Flexible Radio Assignment), configuring, 235–236
 IGMP support, 253–254
 IOS-XE, 4–5
 IRCM (inter-release controller mobility), 191–192
 licensing, 21–22
 load balancing settings, 226
 management LEDs, 555–556
 management options
 “on box” management, 11–16
 PI (Cisco Prime Infrastructure), 16–18
 traditional management tools, 11
 mDNS bridging and gateway, configuring, 276
 Media Stream, configuring, 267–272
 MLD support, 253–254
 MoM (multicast over multicast), configuring, 257–258
 monitoring the RF space, 224
 multicast, configuring, 251–253
 network programmability, 401
 off-channel scanning defer, 227
 operational database, 402
 prerequisites for DNA Center, 20–21
 private cloud deployment, 66
 bootstrap configuration, 66–67
 PnP protocol, 67–69
 resetting to factory defaults, 70
 SP (service port), 69–70
 QoS, 304
 auto, 310–313
 best practices, 320–322
 deployment verification and restrictions, 319
 policy workflow, 304–310
 profiles, 313–316

- RF profiles, configuring, 215–219
- RRM (radio resources management)
 - data collection settings*, 203–206
 - DCA settings*, 212–213
 - manual channel configuration*, 213–214
 - RF group settings*, 206
 - RF grouping modes*, 207–208
 - TPC (Transmit Power Control)*, 208–210
- secure mobility tunneling, configuring, 188–191
- upgrading
 - AP predownload*, 503–505
 - efficient upgrade*, 505
 - ISSU (In-Service Software Upgrade)*, 505–506
 - rolling upgrade*, 505–506
 - standard upgrade*, 501–503
- Web interface, 12
- WebUI, 154
- Wi-Fi 6 settings, 230
- WLAN configuration, 85–87
- Catalyst 9800–40, 38–40
- Catalyst 9800–80, 38–40
- Catalyst 9800-CL, 41
 - deployment modes, 77–78
 - interface mappings, 71–72
 - private cloud deployment, 70–71
 - best practices*, 72–73
 - hypervisor security settings*, 72
 - PnP protocol*, 73–74
 - public cloud deployment, 77–79
- Catalyst 9800-L, 1, 40
- CCKM (Cisco Centralized Key Management), fast roaming, 167–168
- CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol), 116–117
- CD (continuous development), 21
- cell planning, 264. *See also* Media Stream
- certificates, 97–98
 - adding on the controller, 98–102
 - locally significant, 150–151
 - third-party, 98
 - trustpoints, 98
 - web authentication and, 142
- challenging RF environments, 199–200**
 - atriums, 202
 - high-density crowd areas, 200
 - metal-heavy areas, 200
 - shielded doors and sudden turns, 201
 - uneven ceilings, 201–202
- channel width, 232**
- CI (continuous integration), 21**
- Cisco Active Sensor, 488–489**
 - sensor provisioning and onboarding, 489
 - test suites, 489–490
- Cisco DNA Center/DNA Center Assurance, 19–20, 470. *See also* Cisco Active Sensor**
 - analytics
 - Apple*, 479–480
 - device*, 479
 - Intel*, 481–483
 - network services*, 477–479
 - Samsung*, 481
 - AP 360 page, 475–477
 - architecture, 471–472
 - backing up with, 498–499
 - C9800 prerequisites for, 20–21
 - client health dashboard, 473–475
 - Intelligent Capture, 483–487
 - managing the C9800, 472–473
 - Rogue Management application, 245
 - telemetry and data processing flows, 470
 - troubleshooting, 491–492
- Cisco DNA Spaces, 23, 372**
 - BLE gateway, 389–392
 - Captive Portal, 386–389
 - CMX tethering, 379
 - deployment modes
 - Cisco DNA Spaces Connector*, 375–377
 - direct connection*, 374–375
 - licenses, 373–374
 - OpenRoaming, 379
 - architecture*, 379–381
 - configuration*, 381–386
 - proximity engine, 389
- Cisco Prime Infrastructure, backing up with, 497**

- Cisco RF ASIC, 6–7
- Cisco Secure Development Lifecycle, 157
- Cisco Umbrella, 155–157
- classifying rogue APs, 240–241
- CleanAir, 220, 221
 - configuring, 222
 - Interferer Location Tracking, 223–224
 - monitoring the spectrum live, 222–223
- clear platform condition all command, 525
- CLI, 11–12, 344–347, 541. *See also* commands
- client
 - health monitoring, 563–570
 - mobility, 159. *See also* roaming
- cloud networking
 - RRM (radio resources management), 215
 - wireless controllers, 7–10
- CMX (Connected Mobile Experiences), 372, 379
- CoA (Change of Authorization), 107–108
- code. *See also* Python, infrastructure as, 400–401
- commands
 - aaa accounting identity, 112
 - aaa authentication dot1x, 112
 - aaa authentication login, 112
 - aaa authorization credential-download, 112, 148–149
 - aaa authorization exec, 112
 - aaa authorization network, 112
 - aaa new-model, 112, 439
 - ap country, 199
 - ap tag persistency enable, 51
 - automate-tester username, 111
 - bss-transition disassociation-imminent, 1614
 - capwap fallback, 355
 - clear platform condition all, 525
 - config-key password-encrypt, 497
 - copy bootflash, 494
 - copy run bootflash, 494
 - crypto pki trustpool import url, 375
 - deadtime, 111
 - debug wireless, 524
 - factory-reset, 70
 - gnxi server, 440–441
 - ip name-server, 375
 - ip nbar custom, 318–319
 - license smart factory reset, 559
 - redundancy force-switchover, 338–339
 - request platform software trace archive, 513
 - sh dot11 qos, 297–298
 - show ap tag summary, 49–50
 - show capwap summary, 533
 - show chassis rmi, 344
 - show cloud connector connection detail, 370–371
 - show cloud connector key access, 371
 - show controllers, 312
 - show gnxi state, 441
 - show install summary, 502–503
 - show interface, 330
 - show iox applications, 391
 - show logging, 509–510, 515–516
 - show monitor capture, 530
 - show platform hardware chassis active qfp feature packet-trace summary, 532
 - show platform hardware chassis active qfp feature wireless punt statistics, 574–575
 - show platform hardware chassis active qfp infrastructure punt policer summary, 572
 - show platform hardware slot, 328
 - show platform software punt-policer, 572–573
 - show platform software yang-management process, 439–440
 - show policy-map, 312
 - show proc cpu plat sorted, 571
 - show processes cpu history, 570–571
 - show processes cpu platform sorted, 570–571
 - show redundancy config-sync failures, 337
 - show run, 312, 370
 - show telemetry ietf subscription all, 370
 - show telemetry ietf subscription configured, 491
 - show telemetry internal connection, 491
 - show wireless stats mobility, 191
 - show wlan summary, 428

- test aaa group radius, 111
 - wireless client vlan-persistent, 57
 - wireless config validate, 54
 - write memory, 334–335
 - config-key password-encrypt command, 497
 - configuration database, 402
 - configuration files, encryption, 496–497
 - configuration model, 43. *See also* data models
 - AireOS and, 46–47
 - FlexConnect and, 47–48
 - flexibility, 45–46
 - profiles, 43–44, 46
 - tags, 44–46
 - assigning*, 48–54
 - custom site*, 57–63
 - moving APs between wireless controllers*, 54
 - persistency*, 51–52
 - policy*, 45, 55–57
 - RF*, 45
 - site*, 45, 63–64
 - Configuration Validator, 543–545
 - congestion avoidance, 287–288
 - containing rogue APs, 241–243
 - control plane tracing, 509
 - copy bootflash command, 494
 - copy run bootflash command, 494
 - Core Dump and System Report, 536–538
 - country configuration, 197–199
 - coverage hole detection, 210–211
 - CPP (Cisco Packet Processor) dataplane, 35–36
 - CPU, monitoring, 570–575
 - crypto pki trustpool import url command, 375
 - CSMA/CA (Carrier Sense Multiple Access/ Collision Avoidance), 287
 - custom site tag, 59–63
 - CWA (central web authentication), 143–145
- ## D
-
- DACLs (downloadable ACLs), 95–96
 - data models, 403
 - encoding formats, 406
 - JSON*, 407–408
 - Protobuf*, 408
 - XML*, 406–407
 - examining data
 - using NETCONF*, 419–420
 - using Postman*, 424–425
 - using pyang*, 412–414
 - using RESTCONF*, 421–423
 - using YANG Suite*, 414–418, 419–420
 - OpenConfig, 403
 - protocols
 - gNMI/gRPC*, 412
 - NETCONF*, 409–411
 - RESTCONF*, 411–412, 426–428
 - searching data, 425–426, 455–456
 - YANG, 403
 - modules*, 404–405
 - types*, 405–406
 - data plane, 508–509
 - IOS-XE, 35–37
 - monitoring, 576–577
 - datastore, 441
 - Day Zero setup, 80–82, 394–395
 - dBm (decibels per milliwatt), 195–196
 - DCA (Dynamic Channel Assignment), 211–213
 - deadtime command, 111
 - Debug Bundle, 539
 - debug wireless command, 524
 - debugging, 508, 520–521
 - declarative model, 399–400
 - detecting rogue APs, 238–240
 - DFS (Dynamic Frequency Selection), 232–235
 - dial-in subscriptions, 451–454
 - dial-out subscriptions, 450–451, 460–461
 - disaster recovery, 493. *See also* backing up
 - backing up everything for restoring on another controller, 495–496
 - backing up the configuration and restoring it, 494–495
 - backing up using the WebUI, 496
 - saving the configuration changes, 494

DNS-based ACLs, 96–97

Docker

- running pyang, 413–414
- running YANG Suite, 414–416

domains

- regulatory, 197
- roaming, 61
- ROW (Rest of the World), 199

DSCP (Differentiated Service Code Point), 292–293

- classes, 293–295
- trust model, 302–304
- to UP mapping, 295–298

DTLS (Datagram Transport Layer Security), 26–27

E

EAP (Extensible Authentication Protocol), 120

- methods, 121–124
- packets, 120–121

EAP exchange, 106–107. *See also* RADIUS (Remote Authentication Dial-In User Service)

EDCA (Enhanced Distribution Channel Access), 287–288

ED-RRM (event-driven RRM), 212

efficient upgrade, 505

EIRP, 196

Encrypted Traffic Analytics, 39, 154–155

encryption, configuration file, 496–497

Enhanced Open, 128, 138–140

enhanced URL filters, 96–97

EPC (Embedded Packet Capture), 525–531

error handling, SSO (stateful switchover) redundancy, 339–344. *See also* troubleshooting

Ethernet, 195

EWC-AP (Cisco Embedded Wireless Controller on Catalyst Access Points), 74–75, 358–359

EWC-SW (Embedded Wireless Controller on Catalyst 9k switches), HA (high availability), 359

exclusion, 245–246

F

fabric deployments, wireless multicast, 251

factory-reset command, 70

fast roaming, 63–64, 163–164

CCKM, 167–168

FT, 169

association request/response,
170–171

authentication, 171–172

beacon and probe responses,
169–170

over-the-air, 172–173

over-the-DS, 173–176

OKC, 164

PMKID caching, 164

Fastlane+, 319–320

File Manager, 542

FlexConnect, 45, 47–48

ACLs (access control lists), 94–95

fast secure roaming support, 164

mDNS and, 282

multicast and, 251

site tags, 63–64

flexibility, 5, 45–46

FRA (Flexible Radio Assignment), 235–236

FT (Fast Transition), 124, 169

association request/response, 170–171

authentication, 171–172

beacon and probe responses, 169–170

over-the-air, 172–173

over-the-DS, 173–176

full authentication roam. *See* slow roaming

G

gain, 196

gNMI/gRPC, 412, 440–441

gnxi server command, 440–441

GPS (Global Positioning System), 361–362.
See also location tracking

guest

services, 366

tunneling, 1628

users, 153

GUI

- Configuration Validator, 543–545
- Dashboard, 549–550
- HA pair, monitoring, 347–348
- Troubleshooting Dashboard, 536. *See also* tools; troubleshooting
 - AireOS Config Translator*, 540–541
 - CLI, 541
 - Core Dump and System Report*, 536–538
 - Debug Bundle*, 539
 - Ping and Trace Route*, 539–540
- Walk-Me integration, 542–543

Guided Assistance tool, 13**H****HA (high availability), 323–324**

- in EWC-AP deployment, 358–359
- in EWC-SW deployment, 359
- LAG (Link Aggregation Group), 351
 - monitoring, 344
 - via programmatic interfaces*, 348–349
 - via SNMP*, 348
 - via the CLI*, 344–347
 - via the GUI*, 347–348
- multi-chassis LAG (Link Aggregation Group), 352
- N+1 redundancy, 352–353
 - CAPWAP timers*, 355–356
 - configuration*, 353–354
 - configuration on the AP join profile*, 354–355
 - licensing*, 357
 - preserving AP-to-tag mapping across failovers*, 356–357
 - SSO and*, 357–358
- RP only to RP+RMI migration, 349
- RP+RMI HA pair, 331–332
 - active-standby election process*, 335
 - configuration*, 332–335
 - HA formation*, 336–338
 - HA sync*, 335–336

- SSO (stateful switchover) redundancy, 324–325
 - console port*, 330
 - impact on features*, 350
 - mobility MAC*, 350–351
 - out-of-band management/service port*, 330
 - peer requirements*, 325–327
 - RMI (Redundancy Management Interface)*, 327–328
 - RP (redundancy port)*, 328–330
 - RP+RMI supported topologies*, 331
 - switchover*, 338–339
 - system and network error handling*, 339–344
 - uplink ports*, 330
- teardown, 349–350

hardware

- flexible, 5
- monitoring, 550–557

HTTPS redirection, web authentication and, 145–146**I****IaaS (Infrastructure as a Service), 76–77****IaC (Infrastructure as Code), 400–401****IBN (intent-based networking), 3–4. *See also* DNA Center**

- IOS-XE, 4–5

idempotency, 397**Identity PSK (iPSK), 127****IGMP (Internet Group Management Protocol), 248, 253–254****imperative model, 397–399****Intel, analytics, 481–483****Intelligent Capture, 483–487****inter-controller roaming, 185**

- auto-anchoring, 187–188
- Layer 2, 185
- Layer 3, 185–187
- static IP client mobility, 187

inter-WNCd roam, 182–183**intra-controller roaming, 181**

- inter-WNCd, 182–183

- intra-WLC, 183–184
- intra-WNCd, 181–182
- IOSd process, 27, 330
- IOS-XE, 4–5, 508. *See also* CLI
 - Bundle mode, 500
 - Install mode, 500–501
 - SANET library, 29–30, 31–32
 - SISF (Switch Integrated Security Features) library, 30
 - software architecture
 - binOS*, 27
 - control plane*, 37
 - data plane*, 35–37
 - IOSd process*, 27
 - wireless client state machine*, 31–35
 - wireless processes*, 31
 - WNCd (Wireless Network Control Daemon)*, 28–31, 57
- IoT (Internet of Things), 3, 368–371
- ip name-server command, 375
- ip nbar custom command, 318–319
- iPSK (Identity PSK), WPA2 with, 132–138
- IRCM (inter-release controller mobility), 191–192
- ISE (Cisco Identity Services Engine), 89
- ISSU (In-Service Software Upgrade), 1, 28, 506

J-K

- jitter, 285
- JSON (JavaScript Object Notation), 407–408
- Kay, A., 5, 25
- KPI (key performance indicator), 446

L

- LAG (Link Aggregation Group), 351, 352
- latency, 285
- Layer 2 roaming, 185
- Layer 3 roaming, 185–187
- LDAP (Lightweight Directory Access Protocol), 116
- legacy WLCs, 1620

- license smart factory reset command, 559
- licensing, 21–22
 - Cisco DNA Spaces, 373–374
 - N+1, 357
 - Smart, 557
 - using an airgap network*, 558–559
 - using direct connection*, 557–558
 - using on-premises SSM or CSLU*, 558
- load balancing
 - aggressive, 226
 - RADIUS (Remote Authentication Dial-In User Service), 111
- lobby ambassador, 153
- local EAP, 113–114
- local web authentication. *See* web authentication
- location tracking, 361–362
 - accuracy, 362–363
 - analytics, 366
 - Bluetooth, 371–372
 - Bluetooth and, 365
 - deployment guidelines, 364–365
 - location update frequency, 363–364
 - presence, 364
 - privacy MAC addresses, 364
 - UWB (Ultra-Wide Band), 365
- Log Advisor, 548
- logging, 508, 509–510. *See also* tracing
- LSC (locally significant certificates), 150–151

M

- MAC filtering, 127–128, 135
- macro cells, 235
- management options
 - “on box” management, 11–16
 - PI (Cisco Prime Infrastructure), 16–18
 - traditional management tools, 11
- master service list, 277
- mDNS (multicast DNS), 272–273
 - AP, 281–282
 - bridging, 273

- in FlexConnect deployments, 282
 - gateway, 274
 - configuring*, 274–276
 - with guest anchor*, 283
 - service policy, 277–280
 - via AAA override*, 281
 - on VLAN SVI*, 280–281
 - on WLAN, 276
 - MDT (model-driven telemetry)**, 437. *See also*
 - data models**
 - gNMI and, 440–441
 - NETCONF, enabling, 439
 - polling, 447
 - RESTCONF, enabling, 439–440
 - roles, 438
 - RPCs, 444–446
 - subscriptions, 447, 454–455, 457–458
 - dial-in*, 451–454
 - dial-out*, 450–451
 - periodic*, 454
 - telemetry streams, 448
 - Yang-notif-native*, 448
 - Yang-push*, 448–450
 - TIG (Telegraf, Influx, Grafana) and, 461–467
 - YANG models, 442–444
 - Media Stream**, 263
 - cell planning, 264
 - components, 264–267
 - configuring, 267–272
 - packet flow, 267
 - memory, monitoring, 575–576
 - Metal QoS policies, 299, 313–316
 - MIC (manufacturer installed certificate), 26–27, 97
 - micro cells, 235
 - MLD (Multicast Listener Discovery), 253–254
 - mobility, 159
 - auto-anchor, 187–188
 - domain, 1624
 - inter-release controller, 191–192
 - MAC, 350–351
 - secure, configuring, 188–191
 - mobility groups, 55
 - MobilityD (Mobility Daemon), 31
 - monitoring
 - AP health, 559–563
 - client health, 563–570
 - CPU, 570–575
 - HA pair
 - via programmatic interfaces*, 348–349
 - via SNMP*, 348
 - via the CLI*, 344–347
 - via the GUI*, 347–348
 - hardware, 550–557
 - memory, 575–576
 - tools, 548–549
 - MPSK (Multiple PreShared Key)**, 126
 - MQC (Modular QoS CLI)**, 301–302. *See also* QoS (quality of service)
 - multicast, 247–249. *See also* mDNS (multicast DNS)
 - in client roaming scenarios, 262–263
 - non-IP, 260–261
 - wireless
 - 802.11, 298–299
 - CAPWAP, 254–258
 - configuring on the C9800*, 251–253
 - in fabric deployments*, 251
 - in FlexConnect deployments*, 251
 - Media Stream*, 263, 264–272
 - packet flow*, 250–251
 - multi-chassis LAG (Link Aggregation Group), 352
 - MU-MIMO, 229
-
- ## N
-
- N+1 redundancy**, 352–353
 - configuration, 352–353
 - on the AP*, 354–355
 - CAPWAP timers*, 355–356
 - licensing, 357
 - preserving AP-to-tag mapping across failovers, 356–357
 - vs. SSO, 357–358

NBAR (Network-Based Application Recognition), 316–319

NDP (Neighbor Discovery Protocol), 203–206

NETCONF, 153, 406, 457–458, 459

- capabilities, 409
- examining data using, 419–420
- HA pair, monitoring, 348–349
- layers, 409–410
 - content*, 410
 - messages*, 411
 - operations*, 410
 - secure transport*, 411
- MDT (model-driven telemetry) and, 439
- notification, 458–459

network programmability, 393. *See also* data models

- automation and, 397–400
- in the C9800, 401
- configuration repeatability and, 396–397
- Day Zero and, 394–395
- definition, 396
- IaC (Infrastructure as Code), 400–401
- idempotency, 397
- need for, 393–395
- orchestration and, 396
- Python, 429
 - assigning tags to APs based on serial number*, 429–431
 - c9800.py file*, 433–435
 - change_ap_tag.py file*, 431–432
 - readinventory.py code*, 435–436

next-generation wireless stack, 22–23

NMSP (Network Mobility Service Protocol), 372

non-IP multicast, 260–261

O

OFDMA (orthogonal frequency-division multiple access), 229

off-channel scanning defer, 227

OKC (Opportunistic Key Caching), fast roaming and, 164

“on box” management, 11–16

OpenConfig data model, 403

OpenRoaming, 379

- architecture, 379–381
- configuration, 381–386

operational database, 402, 437

orchestration, network programmability and, 396

overlapping BSS (Basic Service Set), 213–214

P

PaaS (Platform as a Service), 76

Packet Capture tool, 14–15

Packet Tracer, 531–536

packets

- capturing, 525–531
- EAP, 120–121
- RADIUS, 104–105
- tracing, 531–536

PEAP (Protected EAP), 121–122

periodic subscriptions, 454

per-process debugging, 520–521

PI (Cisco Prime Infrastructure), 16–18

Ping and Trace Route tool, 539–540

PMF (Protected Management Frames), 118, 124, 128

PMKID (Pairwise Master Key ID) caching, fast roaming and, 164

PnP protocol, 67–69

- AP join process, 84
- C9800-CL and, 73–74

policing, 308

policy(ies)

- client access, 161
- exclusion, 245–246
- mDNS service, 277–280
- Metal QoS, 299, 313–316
- MQC, 301–302
- QoS, 304–310
- Smart Licensing, 557
 - using an airgap network*, 558–559
 - using direct connection*, 557
 - using on-premises SSM or CSLU*, 558

- tag, 45
 - assigning*, 48–54
 - roaming*, 55–57
- targets, 300–301
- polling, 447
- Postman, examining data using, 424–425
- preauthentication ACLs, 93–94
- predownloading ACLs, 94–95
- presence, 364
- private cloud deployment, 7–8, 65–66
 - Catalyst 9800 physical appliance, 66
 - bootstrap configuration*, 66–67
 - PnP protocol*, 67–69
 - resetting to factory defaults*, 70
 - SP (service port)*, 69–70
 - Catalyst 9800-CL, 70–71
 - best practices*, 72–73
 - hypervisor security settings*, 72
 - interface mappings*, 71–72
 - PnP protocol*, 73–74
- probing, 364
- processes, 31, 508
 - IOSd, 27, 330
 - WNCd (Wireless Network Control Daemon), 28–32, 57
 - WNCMgrd, 29
- profiles, 43–44, 46
 - QoS, 313–316
 - Radio, 218–219
 - RF, configuring, 215–219
- programmatic interfaces. *See also* network programmability, HA pair, monitoring, 348–349
- Protobuf, 408
- protocols
 - gNMI/gRPC, 412, 440–441
 - NETCONF, 409, 439, 457–459, 459
 - capabilities*, 409
 - layers*, 409–411
 - Postman, examining data using, 424–425
 - RESTCONF, 411, 439–440
 - examining data using*, 421–423
 - HTTP methods*, 411
 - HTTP return codes*, 411–412

- updating the configuration using*, 426–428
 - URIs*, 422–423
- public cloud deployment, 7–8, 75
 - advantages of, 77
 - Catalyst 9800-CL, 77–79
 - service models, 75–76
 - IaaS (Infrastructure as a Service)*, 76–77
 - PaaS (Platform as a Service)*, 76
 - SaaS (Software as a Service)*, 76
- pyang, 412–414
- Python
 - assigning tags to APs based on serial number
 - c9800.py file*, 433–435
 - change_ap_tag.py file*, 431–432
 - readinventory.py code*, 435–436
 - network programmability and, 429
 - subscriptions and, 457–458

Q

- QFP (Quantum Flow Processor), 35–36, 38–39
- QoS (quality of service), 285–286, 304
 - auto, 310–313
 - best practices, 320–322
 - congestion avoidance, 287–288
 - CSMA/CA, 287
 - deployment verification and restrictions, 319
 - design, 289–290
 - DSCP, 292–293
 - classes*, 293–295
 - trust model*, 302–304
 - to UP mapping*, 295–298
 - labels, 290
 - policy targets, 300–301
 - policy workflow, 304–310
 - profiles, 313–316
 - UP (User Priority), 290–292
 - wireless CAC, 298–299

R

RA (radioactive tracing), 521–525

Radio profiles, 218–219

RADIUS (Remote Authentication Dial-In User Service), 104

accounting, 111

AVPs (attribute-value pairs), 105

CoA (Change of Authorization), 107–108

EAP exchange, 106–107

iPSK (Identity PSK) and, 132–138

load balancing, 111

local EAP, 113–114

packet types, 104–105

securing administrator access, 153

server configuration, 108–109

server fallback, 110–111

server group configuration, 108–110

TACACS+ and, 114

redirect ACL, 143–144

redundancy force-switchover command, 338–339

regulatory domains, 197

request platform software trace archive command, 513

RESTCONF, 411

examining data using, 421–423

HTTP methods, 411

HTTP return codes, 411–412

MDT (model-driven telemetry) and, 439–440

updating the configuration using, 426–428

URIs, 422–423

RF

advanced features

aggressive load balancing, 226

ATF (air time fairness), 228

Band Select, 225–226

off-channel scanning defer, 227

grouping, 206, 207–208

monitoring, 224

neighborhoods, 203

profiles, configuring, 215–219

scanning, 160

tags, 45, 48–54

RFC (request for comments)

4594, 289

5176, 107

RMI (Redundancy Management Interface), 327–328

roaming, 61, 159, 160–161, 181. *See also* mobility

802.11 authentication, 197

159k assisted, 218–221

1642.11v BSS Transition, 221–222

client access policies, 161

domain, 61

fast, 63–64, 163–164

CCKM, 167–168

FT, 169–176

OKC, 165–167

PMKID caching, 164

inter-controller, 185

auto-anchoring, 187–188

Layer 2, 185

Layer 3, 185–187

static IP client mobility, 187

intra-controller, 181

inter-WNCd, 182–183

intra-WLC, 183–184

intra-WNCd, 181–182

L2 authentication, 161

multicast and, 262–263

between policy tags, 55–57

RF scanning, 160

slow, 56, 161–163

rogue APs, 31, 238, 241

classifying, 240–241

containing, 241–243

detecting, 238–240

rolling upgrade, 505–506

-ROW domain, 199

RP (redundancy port), 328–330

migration to RP+RMI, 349

RP+RMI HA pair, 331–332

active-standby election process, 335

- configuration, 332–335
- HA formation, 336–338
- HA sync, 335–336
- RPCs (remote procedure calls), 409, 420, 444–446
- RRC (Resource Reservation Control), 265–266
- RRM (radio resources management), 195, 203, 210–211
 - antennas
 - challenging RF environments*, 199–202
 - gain*, 196
 - self-identifying*, 197
 - cloud-based, 215
 - data collection, 203–206
 - dBm (decibels per milliwatt), 195–196
 - DCA (Dynamic Channel Assignment), 211–213
 - event-driven, 212
 - overlapping BSS (Basic Service Set), 213–214
 - RF grouping, 206, 207–208
 - TPC (Transmit Power Control), 208–209
 - coverage hole detection*, 210–211
 - minimum and maximum*, 209–210
 - txpower, 196–197
- RRMgrd (Radio Resource Manager Daemon), 31
- RSSI (received signal strength indicator), 160
- RxSoP (Receiver Start of Packet), 217–218

S

- SaaS (Software as a Service), 76
- SAE (Simultaneous Authentication of Equals), 125–126, 130–132
- Samsung devices, analytics, 481
- schema, 406
- SDA (software-defined access), 1, 75
- SDN (software-defined networking), 3
- secure mobility, configuring, 188–191
- security, 89. *See also* AAA (authentication, authorization, and accounting)
- AAA (authentication, authorization, and accounting), 103–104
- ACLs (access control lists), 89–90
 - ACE (access control entry)*, 90
 - applying*, 91–93
 - defining*, 90–91
 - DNS-based*, 96–97
 - downloadable*, 95–96
 - FlexConnect*, 94–95
 - log keyword*, 91
 - preauthentication*, 93–94
 - wildcard mask*, 89–90
 - wireless*, 93–94
- administrator access
 - lobby ambassador*, 153
 - RADIUS (Remote Authentication Dial-In User Service)*, 153
 - TACACS+ (Terminal Access Controller Access Control System) and*, 151–152
- authentication
 - RADIUS (Remote Authentication Dial-In User Service)*, 104–111
 - TACACS+ (Terminal Access Controller Access Control System)*, 114–115
- certificates, 97–98
 - adding on the controller*, 98–102
 - third-party*, 98
- guest users, 153
- NETCONF, 153
- traffic, 154–155
- trustpoints, 98
- web authentication, 140–145
- wireless. *See also* wireless networks
 - AP authorization*, 148–149
 - 148.1X*, 183–189
 - Enhanced Open*, 128, 138–140
 - MAC filtering*, 127–128
 - Umbrella*, 155–157
 - WEP (Wired Equivalent Privacy)*, 116
 - WPA (Wi-Fi Protected Access)*, 116–119
 - WPA2 Personal*, 129–130

- WPA2 with iPSK, 132–138
- WPA3 Enterprise, 124
- WPA3 Personal, 124–126
- WPA3 SAE, 130–132
- self-signed certificate, 97–98
- server groups, RADIUS, 108–110
- service models, 75–76
 - IaaS (Infrastructure as a Service), 76–77
 - PaaS (Platform as a Service), 76
 - SaaS (Software as a Service), 76
- service policy, mDNS, 277–281
 - via AAA override, 281
 - on VLAN SVI, 280–281
- services, 361
 - guest, 366
 - location tracking, 361–362
 - accuracy, 362–363
 - analytics, 366
 - Bluetooth and, 365, 371–372
 - deployment guidelines, 364–365
 - location update frequency, 363–364
 - presence, 364
 - privacy MAC addresses, 364
 - UWB (Ultra-Wide Band), 365
 - OpenRoaming, 379
 - architecture, 379–381
 - configuration, 381–386
- sh dot11 qos command, 297–298
- shaping, 308
- shielded doors, antenna reception and, 201
- show ap tag summary command, 49–50
- show capwap summary command, 533
- show chassis rmi command, 344
- show cloud connector connection detail command, 370–371
- show cloud connector key access command, 371
- show controllers command, 312
- show gnxi state command, 441
- show install summary command, 502–503
- show interface command, 330
- show iox applications command, 391
- show logging command, 509–510, 515–516
- show monitor capture command, 530
- show platform hardware chassis active qfp feature packet-trace summary command, 532
- show platform hardware chassis active qfp feature wireless punt statistics command, 574–575
- show platform hardware chassis active qfp infrastructure punt policer summary command, 572
- show platform hardware slot command, 328
- show platform software punt-policer command, 572–573
- show platform software yang-management process command, 439–440
- show policy-map command, 312
- show proc cpu plat sorted command, 571
- show processes cpu history command, 570–571
- show processes cpu platform sorted command, 570–571
- show redundancy config-sync failures command, 337
- show run command, 312, 370
- show telemetry ietf subscription all command, 370
- show telemetry ietf subscription configured command, 491
- show telemetry internal connection command, 491
- show wireless stats mobility command, 191
- show wlan summary command, 428
- SIA (self-identifying antennas), 197
- SISF (Switch Integrated Security Features) library, 30
- site tags, 45
 - assigning, 48–54
 - custom, 59–63
 - FlexConnect, 47–48, 63–64
 - local mode APs, 57–63
- SKC (Sticky Key Caching), fast roaming and, 164
- slow roaming, 56, 161–163
- Smart Licensing Using Policies, 21–22, 557
 - advantages, 1–2
 - using an airgap network, 558–559
 - using direct connection, 557–558
- SNMP (Simple Network Management Protocol), 11, 348

spectrum intelligence, 219–220

split MAC architecture, 25–27

SSH (Secure Shell), monitoring the health of standby WLC, 345–347

SSIDs, 161

SSO (stateful switchover) redundancy, 324–325

- console port, 330
- impact on features, 350
- mobility MAC, 350–351
- vs. N+1 redundancy, 357–358
- out-of-band management/service port, 330
- peer requirements, 325–327
- RMI (Redundancy Management Interface), 327–328
- RP (redundancy port), 328–330, 349
- RP+RMI HA pair, 331–332
 - active-standby election process*, 335
 - configuration*, 332–335
 - HA formation*, 336–338
 - HA sync*, 335–336
 - supported topologies*, 331
- switchover, 338–339
- system and network error handling, 339–344
- uplink ports, 330

standard upgrade, 501–503

steel, antenna reception and, 200

sticky clients, 210–211

subscriptions, 454–455, 457–458

- dial-in, 451–454
- dial-out, 450–451
- periodic, 454
- YANG Suite and, 460–461

syslog, 509–510

T

TACACS+ (Terminal Access Controller Access Control System), 114–115, 151–152

tags, 44–46

- assigning, 48–54
- moving APs between wireless controllers, 54

- persistency, 51–52
- policy, 45, 55–57
- RF tag, 45
- site, 45
 - FlexConnect*, 47–48, 63–64
 - local mode APs*, 57–63

telemetry, 437, 448. *See also* MDT (model-driven telemetry)

test aaa group radius command, 111

third-party certificates, 98

throughput, 38

TIG (Telegraf, Influx, Grafana), 461–467

TKIP (Temporal Key Integrity Protocol), 116

tools, 548–549

- AireOS Config Translator, 540–541
- CLI, 541
- Configuration Validator, 543–545
- Core Dump and System Report, 536–538
- Debug Bundle, 539
- EPC (Embedded Packet Capture), 525–531
- File Manager, 542
- Log Advisor, 548
- Packet Tracer, 531–536
- Ping and Trace Route, 539–540
- pyang, 412–414
- TIG (Telegraf, Influx, Grafana), 461–467
- Walk-Me, 542–543
- Wireless Config Analyzer, 546–547
- Wireless Configuration Converter, 545–546
- Wireless Debug Analyzer, 547–548
- YANG Suite, 414–418, 460–461

TPC (Transmit Power Control), 208–209, 234

- coverage hole detection, 210–211
- minimum and maximum, 209–210

tracing

- binary, 511
 - always-on*, 515–520
 - decoding*, 513–515
 - rotation*, 511
 - severity levels*, 512–513
- packet, 531–536
- radioactive, 521–525

traffic. *See also* QoS

- multicast, 247–249
- policing, 308
- security, 154–155
- shaping, 308
- unicast, 247

tri-radio operations, 236–237

troubleshooting. *See also* logging; monitoring; tracing

- Cisco DNA Center Assurance, 491–492
- SSO (stateful switchover) redundancy, system and network error handling, 339–344

tools

- AireOS Config Translator*, 540–541
- CLI*, 541
- Configuration Validator*, 543–545
- Core Dump and System Report*, 536–538
- Debug Bundle*, 539
- EPC (Embedded Packet Capture)*, 525–531
- File Manager*, 542
- Log Advisor*, 548
- Packet Tracer*, 531–536
- Ping and Trace Route*, 539–540
- Walk-Me*, 542–543
- Wireless Config Analyzer*, 546–547
- Wireless Configuration Converter*, 545–546
- Wireless Debug Analyzer*, 547–548

trust DSCP model, 302–304

trustpoints, 98

TSDB (time-series database), 462

TSPEC (Traffic Specification), 298–299

tunneling, 26

- guest, 1628
- secure mobility, 188–191

TWT (target wake time), 229–230

TXOP (transmit opportunity), 287–288

txpower, 196–197

U

Umbrella, 155–157

UP (User Priority), 290–292

upgrading C9800

- AP predownload, 503–505
- efficient upgrade, 505
- ISSU (In-Service Software Upgrade), 506
- standard upgrade, 501–503

URL filters, 96–97. *See also* ACLs (access control lists)

UWB (Ultra-Wide Band), 365

V

VideoStream, 263

- cell planning, 264
- components, 264–267
- configuring, 267–272
- packet flow, 267

VLAN(s), AAA override, 55–56

VM (virtual machine), 70–71

W

Walk Me configuration tool, 86–87, 542–543

web authentication, 140–143

- captive portal bypass and, 146–147
- central, 143–145
- HTTPS redirection and, 145–146

WebUI, 154, 496

WEP (Wired Equivalent Privacy), 116

Wi-Fi

- CAC (call admission control), 298–299
- CleanAir, 220, 221–223
- deployments, EWC-AP, 74–75
- QoS, 304. *See also* QoS (quality of service)
 - auto*, 310–313
 - best practices*, 320–322
 - congestion avoidance*, 287–288
 - CSMA/CA, 287
 - deployment verification and restrictions*, 319
 - design*, 289–290
 - DSCP (Differentiated Service Code Point)*, 292–295
 - DSCP to UP mapping*, 295–298
 - labels*, 290

- policy targets*, 300–301
- policy workflow*, 304–310
- profiles*, 313–316
- trust DSCP model*, 302–304
- UP (User Priority)*, 290–292
- spectrum intelligence, 219–220
- version 6, 228–229
 - BSS coloring*, 231–232
 - channel width*, 232
 - MU-MIMO*, 229
 - OFDMA*, 229
 - TWT*, 229–230
- wildcard mask, 89–90
- WIPS (wireless intrusion prevention system), 238
 - adaptive, 244–245
- wireless ACLs, applying on the WLC, 93–94
- wireless client state machine, 31–35
- wireless client vlan-persistent command, 57
- Wireless Config Analyzer, 546–547
- wireless config validate command, 54
- Wireless Configuration Converter, 545–546
- wireless controllers, cloud networking and, 7–10
- Wireless Debug Analyzer, 547–548
- wireless multicast
 - 802.11, 298–299
 - CAPWAP, 254
 - MoM mode*, 256–258
 - MoU mode*, 254–255
 - in client roaming scenarios, 262–263
 - configuring on the C9800, 251–253
 - in fabric deployments, 251
 - in FlexConnect deployments, 251
 - mDNS, 272–273
 - AP*, 281–282
 - bridging*, 273
 - in FlexConnect deployments*, 282
 - gateway*, 274–276, 283
 - service policy*, 277–281
 - on WLAN*, 276
 - Media Stream, 263
 - components*, 264–267
 - configuring*, 267–272
 - non-IP, 260–261
 - packet flow, 250–251
- wireless networks. *See also* APs; roaming
 - AP authorization, 148–149
 - broadcast, 260–261
 - 148.1X, 183–184
 - components*, 184–185
 - EAP*, 185–189
 - cell planning, 264. *See also* Media Stream
 - client mobility, 159
 - Enhanced Open, 128, 138–140
 - MAC filtering, 127–128
 - roaming, 160–161, 181
 - 802.11 authentication*, 197
 - 159k assisted*, 218–221
 - 1642.11v BSS Transition*, 221–222
 - client access policies*, 161
 - fast*, 163–176
 - inter-controller*, 185–188
 - intra-controller*, 181–184
 - L2 authentication*, 161
 - RF scanning*, 160
 - slow*, 161–163
 - trust boundary, 302–304
 - Umbrella, 155–157
 - web authentication, 140–143, 147–148
 - captive portal bypass and*, 146–147
 - central*, 143–145
 - HTTPS redirection and*, 145–146
 - WEP (Wired Equivalent Privacy), 116
 - WPA (Wi-Fi Protected Access), 116–119
 - WPA2 Personal, 129–130
 - WPA2 with iPSK, 132–138
 - WPA3 Enterprise, 124
 - WPA3 Personal
 - preshared key*, 124–125
 - SAE (Simultaneous Authentication of Equals)*, 125–126
 - WPA3 SAE, 130–132
- wireless setup wizard, 80–82
- WLAN wizard, 85–87
- WMM (Wi-Fi Multimedia), 287–288
- WNCd (Wireless Network Control Daemon), 28–32, 57

WPA (Wi-Fi Protected Access), 116–119

WPA2 Personal, 129–130, 132–138

WPA3 Enterprise, 124

WPA3 Personal

 preshared key, 124–125

 SAE (Simultaneous Authentication of
 Equals), 125–126, 130–132

write memory command, 334–335

X-Y-Z

**XML (Extensible Markup Language),
406–407**

YANG, 396, 403. *See also* data models

 datastore, 441

 examining data, 442

using NETCONF, 419–420

using pyang, 412–414

using RESTCONF, 421–423

using YANG Suite, 414–418

 modules, 404–405

 operational models, 442–443, 446

 searching data, 425–426

YANG Suite, 414–418, 460–461

Yang-notif-native stream, 448

Yang-push stream, 448–450