Programmability Configuration Guide, Cisco IOS XE 17.16.x

First Published: 2024-12-11

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

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

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned 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. (1721R)

© 2024 Cisco Systems, Inc. All rights reserved.

Preface

This preface describes the conventions of this document and information on how to obtain other documentation. It also provides information on what's new in Cisco product documentation.

- Document Conventions , on page iii
- Related Documentation, on page v
- Obtaining Documentation and Submitting a Service Request, on page v

Document Conventions

This document uses the following conventions:

Convention	Description
^ or Ctrl	Both the ^ symbol and Ctrl represent the Control (Ctrl) key on a keyboard. For example, the key combination **^D** or **Ctrl-D** means that you hold down the Control key while you press the D key. (Keys are indicated in capital letters but are not case sensitive.)
bold font	Commands and keywords and user-entered text appear in **bold** font.
Italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
`Courier` font	Terminal sessions and information the system displays appear in `courier` font.
`Bold Courier` font	**`Bold Courier`** font indicates text that the user must enter.
[x]	Elements in square brackets are optional.
...	An ellipsis (three consecutive nonbolded periods without spaces) after a syntax element indicates that the element can be repeated.
|	A vertical line, called a pipe, indicates a choice within a set of keywords or arguments.
[x | y]	Optional alternative keywords are grouped in brackets and separated by vertical bars.

Convention	Description
{x | y}	Required alternative keywords are grouped in braces and separated by vertical bars.
[x {y | z}]	Nested set of square brackets or braces indicate optional or required choices within optional or required elements. Braces and a vertical bar within square brackets indicate a required choice within an optional element.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Reader Alert Conventions

This document may use the following conventions for reader alerts:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

Tip Means *the following information will help you solve a problem.*

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Timesaver Means *the described action saves time.* You can save time by performing the action described in the paragraph.

Take note of the following general safety warnings:

Warning IMPORTANT SAFETY INSTRUCTIONS

Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Read the installation instructions before using, installing, or connecting the system to the power source. Use the statement number at the beginning of each warning statement to locate its translation in the translated safety warnings for this device.

SAVE THESE INSTRUCTIONS

Related Documentation

Note Before installing or upgrading the device, refer to the device release notes.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.

CONTENTS

Full Cisco Trademarks with Software License ?

PREFACE

Preface iii

Document Conventions iii

Related Documentation v

Obtaining Documentation and Submitting a Service Request v

CHAPTER 1

New and Changed Information 1

New and Changed Feature Information 1

PART I

Provisioning 37

CHAPTER 2

Zero-Touch Provisioning 39

Restrictions for Zero-Touch Provisioning 39

Information About Zero-Touch Provisioning 39

Zero-Touch Provisioning Overview 39

DHCP Server Configuration for Zero-Touch Provisioning 40

DHCPv6 Support 41

Secure ZTP 41

Secure ZTP Workflow 42

Secure ZTP Transport Protocol 43

DHCP Options for Secure ZTP 44

Bootstrapping Servers 44

Bootstrapping Data 45

Image Update Support 46

Progress Reporting 47

Sample Zero-Touch Provisioning Configurations 48
Sample DHCP Server Configuration on a Management Port Using TFTP Copy 48
Sample DHCP Server Configuration on a Management Port Using HTTP Copy 48
Sample DHCP Server Configuration on an In-Band Port Using TFTP Copy 48
Sample DHCP Server Configuration on an In-Band Port Using HTTP Copy 49
Sample DHCP Server Configuration on a Linux Ubuntu Device 49
Sample DHCPv6 Server Configuration on a Management Port Using TFTP Copy 50
Sample Python Provisioning Script 50
Boot Log for Cisco 4000 Series Integrated Services Routers 51
Boot Log for Cisco Catalyst 9000 Series Switches 52
Additional References for Zero-Touch Provisioning 76
Feature Information for Zero-Touch Provisioning 77

CHAPTER 3 iPXE 85
Information About iPXE 85
About iPXE 85
iPXE Overview 86
IPv6 iPXE Network Boot 88
IPv6 Address Assignment in Rommon Mode 90
Supported ROMMON Variables 91
iPXE-Supported DHCP Options 91
DHCPv6 Unique Identifiers 93
How to Configure iPXE 93
Configuring iPXE 93
Configuring Device Boot 94
Configuration Examples for iPXE 95
Example: iPXE Configuration 95
Sample iPXE Boot Logs 96
Sample DHCPv6 Server Configuration for iPXE 96
Troubleshooting Tips for iPXE 98
Additional References for iPXE 99
Feature Information for iPXE 99

PART II Shells and Scripting 101

CHAPTER 4 Guest Shell 103

Restrictions for Guest Shell 103

Information About the Guest Shell 104

Guest Shell Overview 104

Guest Shell Software Requirements 104

Guest Shell Security 105

Hardware Requirements for the Guest Shell 105

Guest Shell Storage Requirements 106

Enabling and Running the Guest Shell 106

Disabling and Destroying the Guest Shell 107

Accessing Guest Shell on a Device 107

Accessing Guest Shell Through the Management Port 107

Day Zero Guest Shell Provisioning Using Front-Panel Port or Fiber Uplink 108

Stacking with Guest Shell 108

Cisco IOx Overview 108

IOx Tracing and Logging Overview 109

IOXMAN Structure 109

NETCONF Access from Guest Shell 110

Logging and Tracing System Flow 111

Logging and Tracing of Messages 112

How to Enable the Guest Shell 114

Managing IOx 114

Managing the Guest Shell 115

Managing the Guest Shell Using Application Hosting 117

Configuring the AppGigabitEthernet Interface for Guest Shell 118

Enabling Guest Shell on the Management Interface 120

Enabling and Disabling NETCONF Access from Guest Shell 121

Accessing the Python Interpreter 122

Configuration Examples for the Guest Shell 123

Example: Managing the Guest Shell 123

Sample VirtualPortGroup Configuration 125

Example: Configuring the AppGigabitEthernet Interface for Guest Shell 126

Example: Enabling Guest Shell on the Management Interface 126

Example: Guest Shell Usage 127
Example: Guest Shell Networking Configuration 127
Sample DNS Configuration for Guest Shell 127
Example: Configuring Proxy Environment Variables 128
Example: Configuring Yum and PIP for Proxy Settings 128
Additional References for Guest Shell 129
Feature Information for Guest Shell 129

CHAPTER 5 **Python API** 133
About Python 133
Cisco Python Module 133
Cisco Python Module to Execute IOS CLI Commands 135
Python Scripts Overview 137
Interactive Python Prompt 137
Python Script 137
Supported Python Versions 139
Updating the Cisco CLI Python Module 140
Additional References for Python API 141
Feature Information for Python API 141

CHAPTER 6 **EEM Python Module** 143
Prerequisites for the EEM Python Module 143
Information About EEM Python Module 143
Python Scripting in EEM 143
EEM Python Package 143
Python-Supported EEM Actions 144
EEM Variables 145
EEM CLI Library Command Extensions 145
How to Configure the EEM Python Policy 146
Registering a Python Policy 146
Running Python Scripts as Part of EEM Applet Actions 148
Adding a Python Script in an EEM Applet 149
Additional References EEM Python Module 151
Feature Information for EEM Python Module 152

PART III **Model-Driven Programmability** 155

CHAPTER 7 **NETCONF Protocol** 157

Information About the NETCONF Protocol 157

Introduction to Data Models: Programmatic and Standards-Based Configuration 157

NETCONF 158

Restrictions for the NETCONF Protocol 158

YANG Model Version 1.1 158

NETCONF RESTCONF IPv6 Support 160

Converting IOS Commands to XML 160

NETCONF Global Session Lock 173

NETCONF Kill Session 174

NETCONF-YANG SSH Server Support 174

Named Method List 175

Candidate Configuration Support 176

NETCONF Operations on Candidate 176

Confirmed Candidate Configuration Commit 178

Candidate Support Configuration 179

Side-Effect Synchronization of the Configuration Database 179

How to Configure the NETCONF Protocol 180

Providing Privilege Access to Use NETCONF 180

Configuring NETCONF-YANG 182

Configuring NETCONF Options 183

Configuring SNMP 183

Configuring the SSH Server to Perform RSA-Based User Authentication 184

Configuring a Named Method List 185

Verifying the NETCONF Protocol Configuration Through the CLI 186

Example: Named Method List 189

Displaying NETCONF-YANG Diagnostics Through RPCs 189

Additional References for NETCONF Protocol 192

Feature Information for the NETCONF Protocol 194

CHAPTER 8 **RESTCONF Protocol** 205

Prerequisites for the RESTCONF Protocol 205
Restrictions for the RESTCONF Protocol 205
Information About the RESTCONF Protocol 206
Overview of RESTCONF 206
HTTPs Methods 206
RESTCONF Root Resource 206
Displaying Version Information 207
RESTCONF API Resource 209
Methods 209
RESTCONF YANG-Patch Support 209
NETCONF RESTCONF IPv6 Support 213
Converting IOS Commands to XML 213
How to Configure the RESTCONF Protocol 226
Authentication of NETCONF/RESTCONF Using AAA 226
Enabling Cisco IOS HTTP Services for RESTCONF 228
Verifying RESTCONF Configuration 229
Configuration Examples for the RESTCONF Protocol 231
Example: Configuring the RESTCONF Protocol 231
Additional References for the RESTCONF Protocol 234
Feature Information for the RESTCONF Protocol 235

CHAPTER 9 **NETCONF and RESTCONF Service-Level ACLs 241**
Information About NETCONF and RESTCONF Service-Level ACLs 241
Overview of NETCONF and RESTCONF Service-Level ACLs 241
How to Configure NETCONF and RESTCONF Service-Level ACLs 241
Configuring an ACL for a NETCONF-YANG Session 241
Configuring an ACL for a RESTCONF Session 243
Configuration Examples for NETCONF and RESTCONF Service-Level ACLs 244
Example: Configuring an ACL for a NETCONF Session 244
Example: Configuring an ACL for a RESTCONF Session 245
Additional References for NETCONF and RESTCONF Service-Level ACLs 245
Feature Information for NETCONF and RESTCONF Service-Level ACLs 246

CHAPTER 10 **gNMI Protocol 247**

Restrictions for gNMI Protocol 247
Information About the gNMI Protocol 248
About GNMI 248
JSON IETF Encoding for YANG Data Trees 248
Proto Encoding 249
gNMI GET Request 251
gNMI SetRequest 254
gNMI Namespace 255
gNMI Wildcards 257
gNMI Configuration Persistence 260
gNMI Username and Password Authentication 260
gNMI Error Messages 260
How to Enable the gNMI Protocol 260
Creating Certs with OpenSSL on Linux 261
Installing Certs on a Device Through the CLI 261
Enabling gNMI in Insecure Mode 262
Enabling gNMI in Secure Mode 263
Connecting the gNMI Client 265
Configuration Examples for the gNMI Protocol 266
Example: Enabling gNMI in Insecure Mode 266
Example: Enabling gNMI in Secure Mode 266
Additional References for the gNMI Protocol 266
Feature Information for the gNMI Protocol 267

CHAPTER 11 **gRPC Network Operations Interface** 275
Information About the gRPC Network Operations Interface 275
gNOI Protocol 275
Certificate Management Service 275
Install RPC 276
Rotate RPC 278
Revoke RPC 279
GetCertificate RPC 280
CanGenerateCSR RPC 281
Mutual Authentication 282

Bootstrapping with Certificate Service 282
OS Installation Service 282
OS Install RPC 284
OS Activate RPC 285
OS Verify RPC 289
GNOI Factory-Reset Services 289
gNOI Factory-Reset Error Messages 289
Additional References for the gRPC Network Operations Interface 290
Feature Information for the gRPC Network Operations Interface 291

CHAPTER 12 gNMI Dial-Out Using the gRPC Tunnel Service 295
gNMI Dial-Out Using gRPC Tunnel Service 295
Information About gNMI Dial-Out Using gRPC Tunnel Service 296
Traditional gRPC Connection 296
gRPC Tunnel 296
Connecting to GNMIB Using the gRPC Tunnel 297
How to Configure gNMI Dial-Out Using gRPC Tunnel Service 297
Configuring and Enabling a Target 297
Configuring a gRPC Tunnel 298
Verifying the gNMI Dial-Out Using gRPC Tunnel Service Configuration 300
Feature Information for gNMI Dial-Out Using gRPC Tunnel Service 301

CHAPTER 13 Model Based AAA 303
Model Based AAA 303
Prerequisites for Model Based AAA 303
Initial Operation 303
Group Membership 304
NACM Privilege Level Dependencies 305
NACM Configuration Management and Persistance 305
Resetting the NACM Configuration 305
Sample NACM Configuration 305
Additional References for Model Based AAA 309
Feature Information for Model-Based AAA 309

CHAPTER 14 **Model-Driven Telemetry** 311

Model-Driven Telemetry 311

Prerequisites for Model-Driven Telemetry 311

Restrictions for Model-Driven Telemetry 314

Information About Model-Driven Telemetry 315

Model-Driven Telemetry Overview 315

Telemetry Roles 316

Subscription Overview 316

Subscription Monitoring 341

Streams 343

Transport Protocol 349

High Availability in Telemetry 351

Pubd Restartability 352

Sample Model-Driven Telemetry RPCs 352

Managing Configured Subscriptions 352

Receiving a Response Code 355

Receiving Subscription Push Updates for NETCONF Dial-In 355

Retrieving Subscription Details 356

Configuring Named Protocol Receiver Using the CLI 358

Subscription Configuration Using Named Receivers Using CLI 359

Additional References for Model-Driven Telemetry 360

Feature Information for Model-Driven Telemetry 361

CHAPTER 15 **In-Service Model Update** 383

Restrictions for In-Service Model Update 383

Information About In-Service Model Update 383

Overview of In-Service Model Updates 383

Compatibility of In-Service Model Update Packages 383

Update Package Naming Conventions 384

Installing the Update Package 384

Deactivating the Update Package 385

Rollback of the Update Package 385

How to Manage In-Service Model Update 386

Managing the Update Package 386
Configuration Examples for In-Service Model Updates 387
Example: Managing an Update Package 387
Feature Information for In-Service Model Update 391

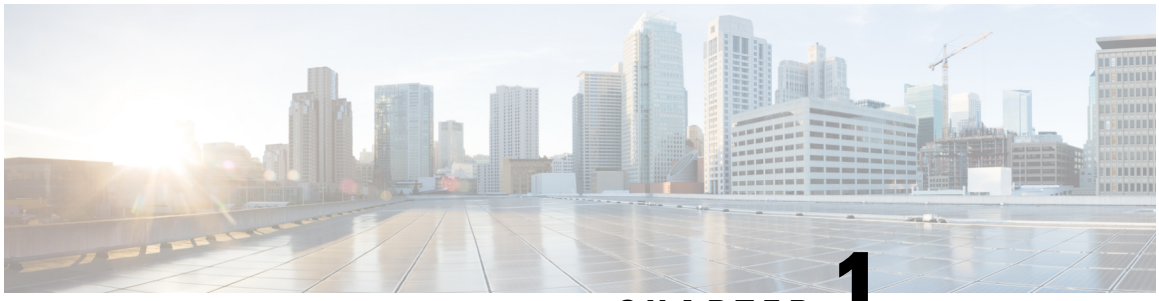
PART IV Application Hosting 393

CHAPTER 16 Application Hosting 395
Prerequisites for Application Hosting 395
Restrictions for Application Hosting 395
Information About Application Hosting 396
Need for Application Hosting 396
Cisco IOx Overview 396
Application Hosting Overview 397
Application Hosting on Front-Panel Trunk and VLAN Ports 398
Application Hosting on Cisco Catalyst 9300 Series Switches 398
Front-Panel App Hosting on Cisco Catalyst 9300X Series Switches 398
High Availability on Cisco Catalyst 9300X Series Switches 400
Application Hosting on Cisco Catalyst 9400 Series Switches 401
Application Hosting on Cisco Catalyst 9410 Series Switches 402
Application Hosting on Cisco Catalyst 9500 Series Switches 403
Application Hosting on Cisco Catalyst 9600 Series Switches 403
Autotransfer and Auto-Install of Apps from Internal Flash to SSD 403
Native Docker Container: Application Auto-Restart 404
Application Auto-Restart Scenarios 404
Application Auto-Restart on Cisco Catalyst 9300 Series Switches 405
Supported Network Types 406
Virtual Network Interface Card 407
ERSPAN Support on the AppGigabitEthernet Port 407
Multicast Routing on the AppGigabitEthernet Interface 407
How to Configure Application Hosting 408
Enabling Cisco IOx 408
Configuring Application Hosting on Front-Panel VLAN Ports 409
Configuring Application Hosting on Front-Panel Trunk Ports 411

Starting an Application in Configuration Mode 412
Lifecycle of an Application 413
Configuring Docker Run Time Options 414
Configuring a Static IP Address in a Container 415
Configuring Application Hosting on the Management Port 417
Manually Configuring the IP Address for an Application 419
Overriding App Resource Configuration 419
Configuring ERSPAN Support on the AppGigabitEthernet Port 420
Configuring an ERSPAN Source Session 421
Configuring the AppGigabitEthernet Interface for ERSPAN 422
Enabling Multicast Routing on the AppGigabitEthernet Interface 424
Verifying the Application-Hosting Configuration 426
Verifying the Application-Hosting Configuration 429
Configuration Examples for Application Hosting 432
Example: Enabling Cisco IOx 432
Example: Configuring Application Hosting on Front-Panel VLAN Ports 432
Example: Configuring Application Hosting on Front-Panel Trunk Ports 433
Example: Installing an Application from disk0: 433
Example: Starting an Application 434
Example: Lifecycle for an Application 434
Example: Configuring Docker Run Time Options 434
Example: Configuring a Static IP Address in a Container 434
Example: Configuring Application Hosting on the Management Port 435
Example: Overriding App Resource Configuration 435
Example: Configuring ERSPAN Support on an AppGigabitEthernet Port 435
Example: Configuring an ERSPAN Source Session 435
Examples: Configuring ERSPAN Through an AppGigabitEthernet Interface 435
Example: Enabling Multicast Routing on the AppGigabitEthernet Interface 436
Additional References 436
Feature Information for Application Hosting 437

CHAPTER 17 **ThousandEyes Enterprise Agent** 441
Prerequisites for the ThousandEyes Enterprise Agent 441
Information About ThousandEyes Enterprise Agent 442

ThousandEyes Enterprise Agent Overview 442
Resources Required for the ThousandEyes Enterprise Agent 442
ThousandEyes Enterprise Agent Download 443
ThousandEyes BrowserBot 444
ThousandEyes Agent Upgrade and Downgrade 445
How to Install the ThousandEyes Enterprise Agent 445
Configuring AppHosting for the ThousandEyes Enterprise Agent 446
Configuring AppGigabitEthernet Interface for the ThousandEyes Enterprise Agent 448
Installing the ThousandEyes Enterprise Agent 449
Configuration Examples for ThousandEyes Enterprise Agent 450
Example: Installing ThousandEyes Enterprise Agent 450
Sample Configuration for ThousandEyes Enterprise Agent 451
Additional References 454
Feature Information for ThousandEyes Enterprise Agent 454



CHAPTER 1

New and Changed Information

This chapter provides release-specific information about all features.

- New and Changed Feature Information, on page 1

New and Changed Feature Information

This table summarizes the new and changed features, the supported platforms, and links to features.

Table 1: New and Changed Feature Information

Feature	Release & Platform
Provisioning	

Feature	Release & Platform
Zero-Touch Provisioning	

Feature	Release & Platform
	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500-High Performance Series Switches Cisco IOS XE Fuji 16.8.2 • Cisco ASR 1000 Series Aggregation Services Routers (ASR1004, ASR1006, ASR1006-X, ASR1009-X, ASR1013) Cisco IOS XE Gibraltar 16.12.1 • Cisco Catalyst 9200 Series Switches **Note** This feature is not supported on C9200L SKUs. • Cisco Catalyst 9300L SKUs • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers Cisco IOS XE Amsterdam 17.2.1 • Cisco Cloud Services Router 1000V Series • Cisco C1100 Terminal Services Gateway (Supported only on C1100TGX-1N24P32A) Cisco IOS XE Amsterdam 17.3.1 • Cisco Catalyst 8200 Series Edge Platforms

Feature	Release & Platform
	• Cisco Catalyst 8300 Series Edge Platforms Cisco IOS XE Cupertino 17.8.1 • Cisco Catalyst 9800-L Wireless Controllers
Secure Zero-Touch Provisioning	Cisco IOS XE Dublin 17.11.1 • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches
Zero Touch Provisioning: HTTP Download	Cisco IOS XE Fuji 16.8.1 • Cisco 4000 Series Integrated Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500-High Performance Series Switches
DHCPv6 Support for Zero-Touch Provisioning	Cisco IOS XE Fuji 16.9.1 • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Amsterdam 17.3.2a • Cisco Catalyst 9800-40 Series Wireless Controllers • Cisco Catalyst 9800-80 Series Wireless Controllers
Zero-Touch Provisioning Through YANG Models	Cisco IOS XE Cupertino 17.7.1 • This feature is supported on all platforms that support NETCONF-YANG.

Feature	Release & Platform
iPXE	Cisco IOS XE Denali 16.3.2 and Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3650 Series Switches Cisco IOS XE Everest 16.6.1 • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500-High Performance Series Switches Cisco IOS XE Fuji 16.9.2 • Cisco Catalyst 9200 Series Switches Cisco IOS XE Gibraltar 16.11.1 • Cisco Catalyst 9600 Series Switches
Shells and Scripting	

Feature	Release & Platform
Guest Shell	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) • Cisco Cloud Services Router 1000V Series Cisco IOS XE Fuji 16.8.1 • Cisco Catalyst 9500-High Performance Series Switches Cisco IOS XE Fuji 16.9.1 • Cisco 1000 Series Integrated Services Routers Cisco IOS XE Gibraltar 16.11.1b • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers Cisco IOS XE Gibraltar 16.12.1 • Cisco Catalyst 9200 Series Switches **Note** This feature is not supported on C9200L SKUs. • Cisco Catalyst 9300L SKUs • Cisco Catalyst 9600 Series Switches Cisco IOS XE Amsterdam 17.3.1 • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms

Feature	Release & Platform
Python 3 Support in Guest Shell	Cisco IOS XE Amsterdam 17.1.1 • Cisco 1000 Series Integrated Services Routers • Cisco ASR 1000 Aggregation Services Routers (ASR1000-RP1, ASR1000-RP2, ASR1000-RP3, ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco ISR 4000 Series Integrated Services Routers
NETCONF Access from Guest Shell	Cisco IOS XE Bengaluru 17.6.1 • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches

Feature	Release & Platform
Python APIs	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) • Cisco Cloud Services Router 1000V Series Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500-High Performance Series Switches

Feature	Release & Platform
Python CLI Module	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) • Cisco Cloud Services Router 1000V Series Cisco IOS XE Fuji 16.8.1 • Cisco 4000 Series Integrated Services Router models with a minimum of 4 GB RAM. • Cisco ASR 1000 Series Aggregation Services Routers (ASR1004, ASR1006, ASR1006-X, ASR1009-X, ASR1013) Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500-High Performance Series Switches

Feature	Release & Platform
EEM Python Module	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2. • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) • Cisco Cloud Services Router 1000V Series Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500-High Performance Series Switches
Model-Driven Programmability	

Feature	Release & Platform
NETCONF Protocol	Cisco IOS XE Denali 16.3.1 • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.7.1 • Cisco ASR 900 Series Aggregation Services Routers • Cisco Network Convergence System 4200 Series Cisco IOS XE Fuji 16.9.2 • Cisco Catalyst 9200 Series Switches Cisco IOS XE Gibraltar 16.10.1 • Cisco Catalyst 9800 Series Wireless Controllers • Cisco Network Convergence System 520 Series • Cisco IR1101 Integrated Services Router Rugged Cisco IOS XE Gibraltar 16.11.1 • Cisco Catalyst IE 3200, 3300, 3400 Rugged Series • Cisco Embedded Service 3300 Series Switches

Feature	Release & Platform
NETCONF and RESTCONF IPv6 Support	Cisco IOS XE Fuji 16.8.1a • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series Cisco IOS XE Gibraltar 16.11.1 • Cisco Catalyst 9500-High Performance Series Switches
NETCONF Global Lock and Kill Session	Cisco IOS XE Fuji 16.8.1a • Cisco 1100 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000v Series

Feature	Release & Platform
NETCONF-YANG SSH Server Support	Cisco IOS XE Gibraltar 16.12.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800 Series Wireless Controllers • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature	Release & Platform
Named Method List	Cisco IOS XE Cupertino 17.9.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 Series and 8500L Series Edge Platforms • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800 Series Wireless Controllers • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
Side-Effect Synchronization of the Configuration Database	Cisco IOS XE Bengaluru 17.4.1 • Cisco ASR 1000 Series Aggregated Services Routers • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches

Feature	Release & Platform
YANG Model Version 1.1	Cisco IOS XE Cupertino 17.7.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Aggregation Services Routers • Cisco ASR 920 Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature	Release & Platform
Upgrade YANG Models to YANG 1.1	Cisco IOS XE Dublin 17.10.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Aggregation Services Routers • Cisco ASR 920 Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
Converting IOS Commands to XML	Cisco IOS XE Cupertino 17.7.1 • This feature is supported on all platforms that support NETCONF-YANG.

Feature	Release & Platform
RESTCONF Protocol	Cisco IOS XE Everest 16.6.1 • Cisco 4000 Series Integrated Services Router • Cisco ASR 1000 Aggregation Services Routers • Cisco Cloud Services Router 1000V Series Cisco IOS XE Fuji 16.8.1a • Cisco 1000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Router • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Network Convergence System 4200 Series Cisco IOS XE Fuji 16.9.2 • Cisco Catalyst 9200 Series Switches Cisco IOS XE Gibraltar 16.11.1 • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CL Wireless Controller for Cloud • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco Network Convergence System 520 Series Cisco IOS XE Gibraltar 16.12.1 • Cisco Catalyst 9800-L Wireless Controllers Cisco IOS XE Amsterdam 17.3.1 • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms

Feature	Release & Platform
RESTCONF YANG-Patch Support	Cisco IOS XE Amsterdam 17.1.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers (ASR1000-RP2, ASR1000-RP3, ASR1001-HX, ASR1001-X, ASR1002-HX, ASR1002-X) • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Catalyst IE3200 Rugged Series • Cisco Catalyst IE3300 Rugged Series • Cisco Catalyst IE3400 Rugged Series • Cisco IR1101 Integrated Services Router Rugged • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature	Release & Platform
NETCONF and RESTCONF Service-Level ACLs	Cisco IOS XE Gibraltar 16.11.1 • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst IE3200 Rugged Series • Cisco Catalyst IE3300 Rugged Series • Cisco Catalyst IE3400 Rugged Series • Cisco Embedded Services 3300 Series Switches • Cisco IR1101 Integrated Services Router Rugged • Cisco Network Convergence System 4200 Series • Cisco Network Convergence System 520 Series

Feature	Release & Platform
gNMI Protocol	Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Gibraltar 16.10.1 • Cisco Catalyst 9500-High Performance Series Switches Cisco IOS XE Gibraltar 16.11.1 • Cisco Catalyst 9600 Series Switches Cisco IOS XE Gibraltar 16.12.1 • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs • Cisco cBR-8 Converged Broadband Router Cisco IOS XE Amsterdam 17.1.1 • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Router • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series Cisco IOS XE Amsterdam 17.2.1r • Cisco ASR 1000 Series Aggregation Services Routers Cisco IOS XE Cupertino 17.8.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 9800-CL Wireless Controller for Cloud • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers

Feature	Release & Platform
gNMI IPv6 support	Cisco IOS XE Dublin 17.10.1 • Cisco Catalyst 9200, 9200L Series Switches • Cisco Catalyst 9300, 9300L, 9300X Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches
gNMI Username and Password Authentication	Cisco IOS XE Gibraltar 16.12.1 • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst IE3200 Rugged Series • Cisco Catalyst IE3200 Rugged Series • Cisco Catalyst IE3300 Rugged Series • Cisco Catalyst IE3400 Rugged Series • Cisco Embedded Service 3300 Series Switches
PROTO Encoding	Cisco IOS XE Dublin 17.11.1 • Cisco Catalyst 9200, 9200L, and 9200X Series Switches • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 and 9400X Series Switches • Cisco Catalyst 9500, 9500-High Performance, and 9500X Series Switches • Cisco Catalyst 9600 Series Switches

Feature	Release & Platform
gRPC Network Operations Interface: gNOI Certificate Management & gNOI Bootstrapping with Certificate Service	Cisco IOS XE Amsterdam 17.3.1 • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches
gNOI OS Installation Service	Cisco IOS XE Bengaluru 17.5.1 • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches
gNOI Factory Reset Service	Cisco IOS XE Cupertino 17.7.1 • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers
gNMI Dial-Out Using the gRPC Tunnel Service	Cisco IOS XE Dublin 17.11.1 • Cisco Catalyst 9200, 9200L, and 9200CX Series Switches • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 and 9400X Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 and 9600X Series Switches • Cisco Network Convergence System 4200 Series

Feature	Release & Platform
Model-Based AAA	Cisco IOS XE Fuji 16.8.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco Cloud Services Router 1000v Series • Cisco Network Convergence System 4200 Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches

Feature	Release & Platform
Model-Driven Telemetry NETCONF Dial-In	

Feature	Release & Platform
	Cisco IOS XE Everest 16.6.1 • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.7.1 • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Aggregation Services Routers (ASR1001-HX, ASR1001-X, ASR1002-HX, ASR1002-X Cisco IOS XE Fuji 16.8.1 • Cisco 1000 Series Integrated Services Routers • Cisco ASR 1000 RP2 and RP3 Series Aggregation Services Routers Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500-High Performance Series Switches Cisco IOS XE Fuji 16.9.1 • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Router • Cisco cBR-8 Converged Broadband Router • Cisco Network Convergence System 4200 Series Cisco IOS XE Fuji 16.9.2 • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs Cisco IOS XE Gibraltar 16.10.1 • Cisco Cloud Services Router 1000v • Cisco Network Convergence System 520 Series Cisco IOS XE Gibraltar 16.11.1 • Cisco Catalyst 9600 Series Switches

Feature	Release & Platform
	Cisco IOS XE Amsterdam 17.3.1 • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms
Model-Driven Telemetry gRPC Dial-out	Cisco IOS XE Gibraltar 16.10.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series Cisco IOS XE Gibraltar 16.11.1 • Cisco Catalyst 9600 Series Switches Cisco IOS XE Amsterdam 17.3.1 • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms

Feature	Release & Platform
Model-Driven Telemetry gNMI Dial-In	Cisco IOS XE Gibraltar 16.12.1 • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco cBR-8 Converged Broadband Router Cisco IOS XE Amsterdam 17.1.1 • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series Cisco IOS XE Amsterdam 17.2.1 • Cisco ASR 1000 Series Aggregation Services Routers Cisco IOS XE Amsterdam 17.3.1 • Cisco ASR 1000 Series Aggregation Services Routers Cisco IOS XE Cupertino 17.8.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms

Feature	Release & Platform
TLS for gRPC Dial-Out	Cisco IOS XE Amsterdam 17.1.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 1000 Series Aggregation Services Routers ASR1000-RP1, ASR1000-RP2, ASR1000-RP3, ASR1001-HX, ASR1001-X, ASR1002-HX, ASR1002-X) • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Catalyst IE3200 Rugged Series • Cisco Catalyst IE3300 Rugged Series • Cisco Catalyst IE3400 Rugged Series • Cisco IR1101 Integrated Services Router Rugged • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
Address Resolution Protocol On-Change Notification	Cisco IOS XE 17.14.1 • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300X Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches

Feature	Release & Platform
gNMI: Stream Subscriptions with On-Change Mode	Cisco IOS XE Amsterdam 17.14.1 • Cisco 1000 Series Integrated Services Routers • Cisco Catalyst IE3200 Rugged Series • Cisco Catalyst IE3300 Rugged Series • Cisco Catalyst IE3400 Rugged Series • Cisco Catalyst IR1101 Rugged Series Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CLWireless Controller for Cloud • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco Cloud Services Router 1000V Series • Cisco Embedded Services 3300 Series • Cisco ESR6300 Embedded Series Router

Feature	Release & Platform
gNMI sync_response Message	Cisco IOS XE Amsterdam 17.14.1 • Cisco 1000 Series Integrated Services Routers • Cisco Catalyst IE3200 Rugged Series • Cisco Catalyst IE3300 Rugged Series • Cisco Catalyst IE3400 Rugged Series • Cisco Catalyst IR1101 Rugged Series Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CLWireless Controller for Cloud • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco Cloud Services Router 1000V Series • Cisco Embedded Services 3300 Series • Cisco ESR6300 Embedded Series Router
TLDP On-Change Notifications	Cisco IOS XE Amsterdam 17.2.1 • Cisco 4000 Series Integrated Services Routers • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches
Subscription Dampening Period for On-Change Telemetry	Cisco IOS XE Dublin 17.11.1 • Cisco Catalyst 9800-CL Series Wireless Controller for Cloud • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller

Feature	Release & Platform
Kill Telemetry Subscription	Cisco IOS XE Gibraltar 16.11.1 • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco IR1101 Integrated Services Router Rugged • Cisco Network Convergence System 4200 Series • Cisco Network Convergence System 520 Series Cisco IOS XE Gibraltar 16.11.1a • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco Cloud Services Router 1000V Series
FQDN Support for gRPC Subscriptions	Cisco IOS XE Bengaluru 17.6.1 • Cisco ASR 900 Series Aggregation Services Routers (RSP2) • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers
Leaf-Level Filtering	Cisco IOS XE Cupertino 17.7.1 • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches

Feature	Release & Platform
Mutual Authentication for gRPC Telemetry	Cisco IOS XE Cupertino 17.9.1 • Cisco Catalyst 9800-CL Series Wireless Controller for Cloud • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller
Pubd Restartability	Cisco IOS XE Cupertino 17.9.1 • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CL Series Wireless Controller for Cloud • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature	Release & Platform
In-Service Model Updates	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.1 • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.7.1 • Cisco ASR 1000 Aggregation Services Routers Cisco IOS XE Fuji 16.8.1a • Cisco Catalyst 9500-High Performance Series Switches.
Application Hosting	
Application Hosting	Cisco IOS XE Gibraltar 16.12.1 • Cisco Catalyst 9300 Series Switches Cisco IOS XE Amsterdam 17.1.1 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Amsterdam 17.2.1 • Cisco Catalyst 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches Cisco IOS XE Bengaluru 17.5.1 • Cisco Catalyst 9410 Series Switches Cisco IOS XE Cupertino 17.7.1 • Cisco Catalyst 9500-X Series Switches

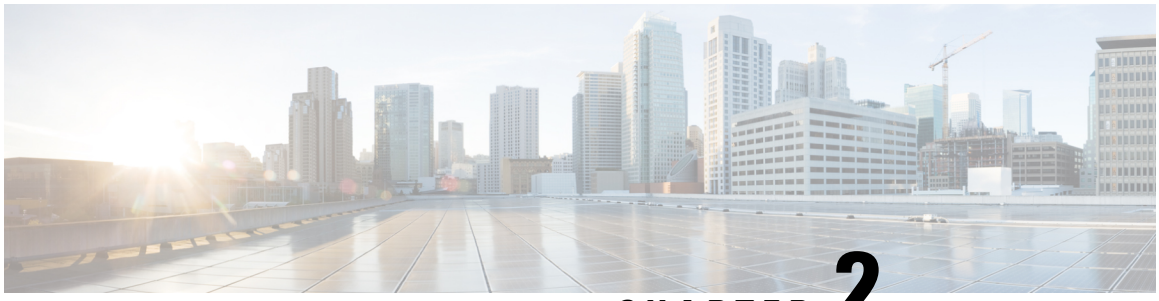
Feature	Release & Platform
Application Hosting: Front-Panel Network Port Access	Cisco IOS XE Gibraltar 16.12.1 • Cisco Catalyst 9300 Series Switches Cisco IOS XE Amsterdam 17.1.1 • Cisco Catalyst 9400 Series Switches **Note** Cisco Catalyst 9410R Switch does not support front-panel application-hosting.
Application Hosting: Front-Panel USB Port Access	Cisco IOS XE Gibraltar 16.12.1 • Cisco Catalyst 9300 Series Switches Cisco IOS XE Amsterdam 17.1.1 • Cisco Catalyst 9400 Series Switches **Note** Cisco Catalyst 9410R Switch does not support front-panel application-hosting.
Native Docker Container: Application Auto-Restart	Cisco IOS XE Amsterdam 17.2.1 • Cisco Catalyst 9300 Series Switches Cisco IOS XE Bengaluru 17.5.1 • Cisco Catalyst 9410 Series Switches
Application Hosting: ERSPAN Support on the AppGigabitEthernet Port	Cisco IOS XE Dublin 17.10.1 • Cisco Catalyst 9300, 9300L, 9300X Series Switches • Cisco Catalyst 9400 Series Switches
Multicast Routing on the AppGigabitEthernet Port	Cisco IOS XE Dublin 17.11.1 • Cisco Catalyst 9300, 9300L, 9300X Series Switches • Cisco Catalyst 9400 and 9400X Series Switches • Cisco Catalyst 9500-High Performance Series Switches • Cisco Catalyst 9600 and 9600X Series Switches

Feature	Release & Platform
ThousandEyes Integration	Cisco IOS XE Amsterdam 17.3.3 • Cisco Catalyst 9300 and 9300L Series Switches Cisco IOS XE Bengaluru 17.5.1 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Bengaluru 17.6.1 • Cisco Catalyst 9300X Series Switches
ThousandEyes BrowserBot	Cisco IOS XE Bengaluru 17.6.1 • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 Series Switches
ThousandEyes 5.0 Support	Cisco IOS XE 17.16.1 • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 Series Switches

PART I

Provisioning

- Zero-Touch Provisioning, on page 39
- iPXE, on page 85



CHAPTER 2

Zero-Touch Provisioning

To address network provisioning challenges, Cisco introduces a zero-touch provisioning model. This module describes the Zero-Touch Provisioning feature.

Note The Zero-Touch Provisioning feature is enabled automatically; no configuration is required.

- Restrictions for Zero-Touch Provisioning, on page 39
- Information About Zero-Touch Provisioning, on page 39
- Sample Zero-Touch Provisioning Configurations, on page 48
- Additional References for Zero-Touch Provisioning, on page 76
- Feature Information for Zero-Touch Provisioning, on page 77

Restrictions for Zero-Touch Provisioning

- Zero-touch provisioning is not supported on Cisco Catalyst 9200L SKUs.
- Zero-touch provisioning is not supported on Cisco Catalyst 9800-L Wireless Controller.

Information About Zero-Touch Provisioning

This section provides information about the DHCP server configuration, DHCPv6 support, Secure ZTP, bootstrapping information, and so on.

Zero-Touch Provisioning Overview

Zero-touch provisioning (ZTP) provides open bootstrap interfaces to automate network device provisioning in heterogeneous network environments.

When a device that supports ZTP starts up, and does not find the startup configuration (during initial installation), the device enters the zero-touch provisioning mode. The device searches for a DHCP server, bootstraps itself with its interface IP address, gateway, and Domain Name System (DNS) server IP address, and enables Guest Shell. The device then obtains the IP address or URL of an HTTP or a TFTP server, and downloads the Python script from an HTTP or a TFTP server to configure the device.

ZTP is supported through data ports on all supported platforms; however, on Cisco Catalyst 9500X Series Switches it is not supported through data ports.

Guest Shell provides the environment for the Python script to run. Guest Shell runs the downloaded Python script and applies an initial configuration to the device.

After initial provisioning is complete, Guest Shell remains enabled. For more information, see the Guest Shell chapter.

Note If ZTP fails, the device falls back on AutoInstall to load the configuration file. For more information, see Using AutoInstall and Setup.

DHCP Server Configuration for Zero-Touch Provisioning

In ZTP, a DHCP server must be running on the same network as the new device that is being provisioned. ZTP is supported on both management ports and in-band ports.

When the new device is switched on, it retrieves the IP address information of the HTTP or TFTP server where the Python script resides, and the folder path of the Python script from the DHCP server. For more information on Python Scripts, see the *Python API* chapter.

The DHCP server responds to DHCP discovery events with the following options:

- Option 150: (Optional) Contains a list of IP addresses that point to the HTTP or TFTP server in the management network that hosts the Python scripts to be run.
- Option 67: Contains the Python script file path in the HTTP/TFTP server.

After receiving these DHCP options, the device connects to the HTTP or TFTP server, and downloads the Python script. At this point, because the device does not have a route to reach the HTTP or TFTP server, it uses the default route provided by the DHCP server.

DHCP Client Behavior

From Cisco IOS XE Amsterdam 17.2.1 release onwards, the device product identification (PID), Cisco Network Plug and Play (PnP), the serial number, and the MAC address is set in the DHCP client options.

These client options can be used as variables at the DHCP server. The DHCP client options can be used to identify the device and send a unique Python file through option 67 or boot filename.

Table 2: DHCP Client Options

Device Information	DHCPv4	DHCPv6
Product ID	124	16/toggles
Serial number	61/toggles	16/toggles
MAC address	61/toggles	1
Cisco Network PnP	60	15

DHCPv6 Support

In Cisco IOS XE Fuji 16.9.1, DHCP Version 6 support is added to the Zero-Touch Provisioning feature. DHCPv6 is enabled by default, and will work on any device that boots without a startup configuration.

Note DHCPv6 is only supported on Catalyst 9300 and 9500 Series Switches.

DHCPv6 is supported by both TFTP and HTTP download of Python scripts. If this download fails, the device reverts to the initial or factory settings (without any configuration). For both DHCPv4 and DHCPv6 to work, the correct HTTP or TFTP file path must be available in the DHCP configuration.

There can be scenarios where the same interface can have both IPv4 and IPv6 addresses, or two different interfaces in a network—one can receive IPv4 traffic and the other, IPv6 traffic. We recommend that you use either the DHCPv4 or DHCPv6 option in your deployment.

The following is a sample DHCPv4: {/etc/dhcp/dhcpd.conf:}

```
host <hostname> {
  hardware ethernet xx:xx:xx:xx:xx:xx;
  option dhcp-client-identifier "xxxxxxxxxxxxxx";
  option host-name "<hostname>".
  option log-servers x.x.x.x;
  fixed-address x.x.x.x;
  if option vendor-class-identifier = "..." {
    option vendor-class-identifier "...";
    if exists user-class and option user-class = "iPXE" {
      filename "http://x.x.x.x/…/<image>";
    } else {
      filename "http://x.x.x.x/…/<script-name>";
    }
  }
}
```

The following is a sample ISC DHCPv6 server configuration:

```
 option dhcp6.bootfile-url  "http://[2001:DB8::21]/sample_day0_script.py";
```

Secure ZTP

As per RFC 8572, Secure ZTP is a technique to securely provision a device, while it is booting in a factory default state. The provisioning updates the boot image, commits an initial configuration, and runs customer-specific scripts.

The existing ZTP can download a configuration script from an HTTP or a TFTP server, and run it on the device, but it does not securely provision a device.

To securely provision a device, the following requirements must be met:

- The management system must validate that it is provisioning a valid device.
- The device must validate that it is being deployed in the correct network.
- The device must validate that the provisioning data has not been tampered with.
- Provisioning must use a secure transport protocol for data communication.

The Secure ZTP feature is enabled by default, and it cannot be disabled. The feature coexists with the classic ZTP. Based on the response that comes from the DHCP server, either ZTP or Secure ZTP is enabled. Option 67 triggers ZTP, and Option 143 or 136 triggers Secure ZTP.

To use Secure ZTP, users must get an ownership voucher from a Cisco Manufacturer Authorized Signing Authority (MASA) server.

Note Secure ZTP supports only Python scripts.

Secure ZTP Workflow

This section describes the Secure ZTP workflow:

1. When a device that supports the Secure ZTP boots up and does not find the startup configuration, it enters the ZTP mode, and triggers the Plug-n-Play (PnP) agent.
2. The PnP agent locates a DHCP server, obtains an IP address from the server, and checks the response for Options 143 (DHCPv4) or 136 (DHCPv6). If either of these options is available, Secure ZTP is started.

Note The DHCP server must already be configured.

3. The device starts a recursive algorithm of iterating over the bootstrapping servers, trying to get valid bootstrapping data.

   The algorithm is recursive because a bootstrapping server can return redirect data, in which case, new bootstrapping servers are added to the top of the list. The algorithm stops when either the bootstrapping data is received and successfully applied, or when the list of servers is exhausted after not receiving any bootstrapping data.

Note A bootstrapping server is a RESTCONF server, and the data transport protocol is HTTPS.

4. The server validates and provisions the device. The device presents its Cisco Secure Unique Device Identifier (SUDI) certificate to the bootstrapping server during TLS handshake.

   The device can accept the server certificate either by verifying it against a server-trust anchor that the device learned previously, or without verification.

5. The device requests bootstrapping data from the server through a RESTCONF POST request by using the ietf-sztp-bootstrap-server:get-bootstrapping-data RPC.

   The bootstrapping server can provide either redirect or onboarding data in the response message. The onboarding information that is received from the bootstrapping server will be signed data.

6. The device checks the headers of the CMS structures that contain the bootstrapping artifacts to decide whether decrypting is required, and the format (JSON or XML) of the data used. If required, decryption is performed by using the device's private SUDI key.

7. The device uses its trust anchor (the Cisco root certificate) to validate the ownership voucher. The ownership voucher must be signed by the IOSXE_SZTP private key. The IOSXE_SZTP certificate is signed by a Cisco root, but the trust chain for the voucher must use the IOSXE_SZTP key: any other certificate signed by the Cisco root will fail.

   Note that only the ownership voucher is validated against the device trust anchor. All other bootstrapping artifacts are validated with the customer's pinned-domain-certificate that is found in the ownership voucher.

8. If the onboarding information includes a requirement for a specific OS version, the device checks the running OS version. In case of a mismatch, the device downloads and installs the specific version.

   After the OS installation is complete, the device reboots and the Secure ZTP process restarts.

9. The device starts the Guest Shell after the onboarding information is downloaded from the bootstrap server.

   Guest Shell is used to run customer-specific scripts, and to apply the downloaded configuration. The configuration can be in Cisco IOS CLI format or in the form of NETCONF edit-config requests. The Guest Shell is shut down after bootstrapping is completed.

10. The device sends progress messages to the server if the server is trusted.

Secure ZTP Transport Protocol

Secure ZTP uses HTTPS as the transport protocol when communicating with the RESTCONF bootstrapping servers.

When establishing a connection, the device provides its Cisco Secure Unique Device Identifier (SUDI) certificate during the TLS handshake. The certificate serves as an authentication token. No other authentication is supported. The SUDI certificate validates the device on the server side.

The bootstrapping server provides its own certificate during the TLS handshake to the device. If the device has the server-trust anchor corresponding to this server, the device performs an X.509 certificate validation of the server certificate against the server-trust anchor. (The device can get the server-trust anchor from the redirect server.)

If the device does have a server-trust anchor, but the X.509 certificate is not validated successfully, the device must discard the server-trust anchor, accept the connection, and mark the server as not trusted.

If the device does not have a server-trust anchor, it accepts the server certificate, but marks it as not trusted.

According to the RESTCONF protocol, a device must first initiate the root-resource discovery, before it can proceed with RPC requests. The following is a sample root-discovery request:

```
GET /.well-known/host-meta HTTP/1.1
Host: example.com
Accept: application/xrd+xml
```

The following is a sample root-discovery response:

```
HTTP/1.1 200 OK
Content-Type: application/xrd+xml
Content-Length: nnn
<XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0'>
    <Link rel='restconf' href='/restconf_root'/>
```

```
</XRD>
```

The value in the HREF attribute is used in the follow-up requests to the RESTCONF server, for example, when

```
href='/restconf_root'
```

is received, the header for the get-bootstrapping-data RPC is

```
POST /restconf_root/operations/ietf-sztp-bootstrap-server:
get-bootstrapping-data HTTP/1.1
```

The device receives a plain XML file, and the fields in the file are Base64 encoded. The device runs Base64 decoding before extracting the information and attempting to recognize its structure. The decoded structure format can either be JSON or XML.

DHCP Options for Secure ZTP

DHCP redirect Options 136 (DHCPv6) and 143 (DHCPv4) are supported for Secure ZTP. These options are used to provision a client with one or more Uniform Resource Identifiers (URIs) for bootstrap servers.

A device in the Zero-Touch Provisioning mode carries out a DHCP discovery to find a DHCP server, and to receive the IP address and other information. When the DHCP client receives Options 136 and 143, the Secure ZTP procedures are started, and the Secure ZTP handling functions are invoked. If the Secure ZTP handling functions fail, the handling procedure is repeated indefinitely, until it is either successful, or is interrupted by the user.

Bootstrapping Servers

A bootstrap server is a RESTCONF server that uses the HTTPS data transport protocol. A bootstrap server can be a redirect server or an onboarding server.

A redirect server sends a list of bootstrapping servers to a device. The device can attempt bootstrapping from any server in the list. Each server entry has a server address, a server port, and a trust anchor (X.509 certificate) that verifies the server's certificate presented during the TLS handshake. The trust anchor is accepted only if the redirect information is signed, or if the current redirect server is trusted. Otherwise, all the trust anchors received from the redirect server are discarded.

An onboarding server is a server that sends the actual bootstrapping data. The bootstrapping data contains the following types of information:

- Required updated image that the device must run: The required version, URLs for downloading the image, and information about how to validate the downloaded image.
- Preconfiguration script: A Python script that must be run before the initial device configuration is applied.
- Configuration: The initial device configuration. This configuration can be in the Cisco IOS CLI format of a NETCONF edit-config format.
- Postconfiguration script: A Python script that must be run after the initial device configuration is applied.

A bootstrapping server can be trusted or untrusted. A trusted server can send bootstrapping data that is not necessarily signed. A trusted server can be demoted into an untrusted server if the server certificate it sends during the TLS handshake fails to validate the server-trust anchor on the device.

Bootstrapping Data

Bootstrapping data refers to a collection of data that a device obtains during the bootstrapping process.

Bootstrapping data (both redirect and onboarding information) can be signed or unsigned. An ownership voucher, owner certificate, and conveyed information must be present in signed bootstrapping data.

Signed data is trusted. If the signed data is redirect information, trust anchors for the bootstrap servers that are being redirected to are saved for future TLS handshakes. Signed onboarding information is trusted and is applied to the device.

When a bootstrap server provides signed data, the data is considered trusted. However; this does not make the server trusted. If the server is considered untrusted before receiving the data, it remains untrusted, and the server does not receive progress reports.

Unsigned data do not need to present the ownership voucher and the owner certificate artifacts; infact, these are discarded, if present. Unsigned data is accepted only if it is redirect data or is received from a trusted server. Receiving unsigned data from an untrusted onboarding server is considered an error, and the device continues to the next server on its list.

Ownership Voucher

Assigning ownership is important to bootstrapping mechanisms. The ownership voucher is defined in Section 5.3 of RFC 8366. The primary purpose of a voucher is to securely assign a pledge to an owner. The voucher provides the pledge with details of the entity, which the pledge should consider as its owner.

The ownership voucher has a Cryptographic Message Syntax (CMS) structure, is Distinguished Encoding Rules-encoded (DER-encoded), and encloses YANG-modeled data.

This is a sample ownership voucher:

```
yang-data voucher-artifact:
+---- voucher
+---- created-on                     yang:date-and-time
+---- expires-on?                    yang:date-and-time
+---- assertion                      enumeration
+---- serial-number                  string
+---- idevid-issuer?                 binary
+---- pinned-domain-cert             binary
+---- domain-cert-revocation-checks? boolean
+---- nonce?                         binary
+---- last-renewal-date?             yang:date-and-time
```

The CMS structure is signed using a private key that corresponds to the device's trust anchor. The device carries the public key (the trust anchor), which is the only thing it needs to verify the signature on the voucher. The private key is used by the Cisco MASA server when creating the voucher. The private key is securely stored in the Cisco Software Image Management (SWIM) server.

The device uses its trust anchor to extract the actual ownership voucher from the CMS structure. The signature on the ownership voucher is verified by using the public key enclosed in the device's trust anchor. This verification may require additional intermediate X.509 certificates, in which case, these must be attached to the ownership voucher.

After the signature on the ownership voucher is verified, the YANG-modeled data fields are extracted. If the *created-on* field is present, the device verifies whether the voucher was created in the past. If the *expires-on* field is present, the device verifies whether the voucher is expired or not. If these fields are not present, the voucher is rejected.

The device then verifies the required *serial-number* data field. The serial number of the device is learned by the server from the device's SUDI certificate sent to the server during the TLS handshake. The serial number helps the server to send the right configuration data to the right device.

When all the checks are successful, the device extracts the data field *pinned-domain-cert*, which is the main payload of the ownership voucher. The pinned domain certificate must be Base64 encoded. The device performs a Base64 decoding before handling the certificate. The decoded certificate is an X.509 certificate in DER format.

If an error occurs during validation, the ownership voucher is not verified, and the bootstrapping data is considered invalid.

Owner Certificate

The owner certificate is an X.509 certificate that binds an owner's identity to a public key, which a device can use to validate a signature in the conveyed information.

After verifying the ownership voucher, the device uses the pinned domain certificate extracted from the ownership voucher to validate the owner certificate.

The data field in the bootstrapping data that represents the owner certificate is Base64 encoded. The device then performs a Base64 decoding. The output of this decoding is a degenerate CMS structure, is DER-encoded, and is of the content type signedData. (Degenerate structure is a format that is commonly used to distribute X.509 certificates. These structures do not contain any signatures, but can have attached intermediate certificates.) The owner certificate is an x509 certificate, and the owner certificate artifact is a degenerate CMS structure that carries the x509 certificate.

Conveyed Information

The conveyed information artifact encodes the essential bootstrapping data for a device. This artifact is used to encode the redirect and onboarding information types.

The final step in validating the signed information is the verification of the signed-CMS structure that contains the conveyed information. The data field for the conveyed information is Base64 encoded. The device first performs Base64 decoding.

The device then verifies the signature using the certificates from the owner certificate's CMS structure and the pinned domain certificate as the trust anchor.

Image Update Support

When the bootstrapping data requires a specific OS version, the device checks the version of the software that it is currently running. In case of a mismatch, even if the running version is newer, the device uses the URL provided in the bootstrapping data, and downloads the specified image. No modifications are done to the URL, and no additional authentication information is added by the device. If several URLs are specified in the bootstrapping data, the device will loop over the list of URLs until the first successful attempt or until the list is exhausted.

After an image is downloaded, if the *image-verification* data is included in the bootstrapping information, the device uses the specified algorithm to calculate the hash of the image and compares the result with the hash string included in the bootstrapping data. If multiple algorithms or hashes are provided in the bootstrapping data, the device picks the first-supported algorithm in the list to calculate and compare the hash.

After the image is verified, the image is installed, and the device reboots. State information is not stored before the reboot. When the device boots up, it enters the Secure ZTP process again and repeats all the steps till the

image updating. Because the device is running the correct image, it does not need an image update, and the device will proceed with applying the on-boarding information.

Progress Reporting

The device sends progress reports to the bootstrapping server. RFC 8572 provides more information on progress reporting.

When a bootstrapping server is trusted, it receives appropriate POST-request messages with the *ietf-sztp-bootstrap-server:report-progress* RPC.

There are two types of progress report messages—mandatory and optional. The mandatory reports indicate the start or termination of the bootstrapping process and include the following types of reports:

- bootstrap-initiated
- bootstrap-error
- bootstrap-complete
- boot-image-installed-rebooting
- config-error
- parsing-error
- pre-script-error
- post-script-error

The rest of progress report messages are optional. Optional progress report messages are only sent to the server if the bootstrapping data has set the *reporting-level* parameter to *verbose*.

This is a sample report-progress request from the device to the server:

```
POST /restconf/operations/ietf-sztp-bootstrap-server:report-progress/HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml

<input xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
<progress-type>bootstrap-error</progress-type>
<message>Failed to decode data</message>
</input>
```

This is a sample *No content* response from the server:

```
HTTP/1.1 204 No Content
Date: Sat, 31 May 2021 17:02:40 GMT
Server: example-server
```

Sample Zero-Touch Provisioning Configurations

Sample DHCP Server Configuration on a Management Port Using TFTP Copy

The following is a sample DHCP server configuration using TFTP copy when the DHCP server is connected through the management port on a device:

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp excluded-address 10.1.1.1
Device(config)# ip dhcp excluded-address vrf Mgmt-vrf 10.1.1.1 10.1.1.10
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# vrf Mgmt-vrf
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 150 ip 203.0.113.254
Device(config-dhcp)# option 67 ascii /sample_python_dir/python_script.py
Device(config-dhcp)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# no ip dhcp client request tftp-server-address
Device(config-if)# end
```

Sample DHCP Server Configuration on a Management Port Using HTTP Copy

The following is a sample DHCP server configuration using HTTP copy when the DHCP server is connected through the management port on a device:

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# vrf Mgmt-vrf
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 67 ascii http://198.51.100.1:8000/sample_python_2.py
Device(config-dhcp)# end
```

Sample DHCP Server Configuration on an In-Band Port Using TFTP Copy

The following is a sample DHCP server configuration using TFTP copy when the DHCP server is connected through the in-band port on a device:

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp excluded-address 10.1.1.1
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 150 ip 203.0.113.254
Device(config-dhcp)# option 67 ascii /sample_python_dir/python_script.py
```

```
Device(config-dhcp)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# no ip dhcp client request tftp-server-address
Device(config-if)# end
```

Sample DHCP Server Configuration on an In-Band Port Using HTTP Copy

The following is a sample DHCP server configuration using HTTP copy when the DHCP server is connected through the in-band port on a device:

```
Device> enable
Device# configure terminal
Device(config)# ip dhcp excluded-address 10.1.1.1
Device(config)# ip dhcp pool pnp_device_pool
Device(config-dhcp)# network 10.1.1.0 255.255.255.0
Device(config-dhcp)# default-router 10.1.1.1
Device(config-dhcp)# option 67 ascii http://192.0.2.1:8000/sample_python_2.py
Device(config-dhcp)# end
```

Sample DHCP Server Configuration on a Linux Ubuntu Device

The following sample DHCP server configuration shows that the server is either connected to the management port or the in-band port in a device, and that a Python script is copied from a TFTP server.

```
root@ubuntu-server:/etc/dhcp# more dhcpd.conf
subnet 10.1.1.0 netmask 255.255.255.0 {
range 10.1.1.2  10.1.1.255;
      host 3850 {
          fixed-address                     10.1.1.246 ;
          hardware ethernet                 CC:D8:C1:85:6F:00;
          option bootfile-name !<opt 67>     " /python_dir/python_script.py";
         option tftp-server-name !<opt 150>  "203.0.113.254";
      }
  }
```

The following sample DHCP configuration shows that a Python script is copied from an HTTP server to the device:

```
Day0_with_mgmt_port_http
------------------------
subnet 192.168.1.0 netmask 255.255.255.0 {
 range 192.168.1.2 192.168.1.255;
      host C2-3850 {
          fixed-address                  192.168.1.246 ;
          hardware ethernet              CC:D8:C1:85:6F:00;
          option bootfile-name           "http://192.168.1.46/sample_python_2.py";
      }
  }
```

After the DHCP server starts running, you must start the management network-connected device. The rest of the configuration is automatic.

Sample DHCPv6 Server Configuration on a Management Port Using TFTP Copy

The following is a sample configuration using TFTP copy when the DHCPv6 server is connected through the management port on a device:

```
Device> enable
Device# configure terminal
Device(config)# ipv6 dhcp pool ztp
Device(config-dhcpv6)# address prefix 2001:DB8::1/64
Device(config-dhcpv6)# domain-name cisco.com
Device(config-dhcpv6)# bootfile-url tftp://[2001:db8::46]/sample_day0_script.py
Device(config-dhcpv6)# exit
Device(config)# interface vlan 20
Device(config-if)# ipv6 dhcp server ztp
Device(config-if)# end
```

Sample Python Provisioning Script

The following is a sample Python script that can be used from either an HTTP server or a TFTP server:

```
print "\n\n *** Sample ZTP Day0 Python Script *** \n\n"

# Importing cli module
import cli


print "\n\n *** Executing show platform  *** \n\n"
cli_command = "show platform"
cli.executep(cli_command)

print "\n\n *** Executing show version *** \n\n"
cli_command = "show version"
cli.executep(cli_command)

print "\n\n *** Configuring a Loopback Interface *** \n\n"
cli.configurep(["interface loop 100", "ip address 10.10.10.10 255.255.255.255", "end"])


print "\n\n *** Executing show ip interface brief  *** \n\n"
cli_command = "sh ip int brief"
cli.executep(cli_command)

print "\n\n *** ZTP Day0 Python Script Execution Complete *** \n\n"
```

Boot Log for Cisco 4000 Series Integrated Services Routers

The following sample Zero-Touch Provisioning boot log displays that Guest Shell is successfully enabled, the Python script is downloaded to the Guest Shell, and the Guest Shell is running the downloaded Python script and configuring the device for day zero.

```
% failed to initialize nvram
! <This message indicates that the startup configuration
is absent on the device. This is the first indication that the Day Zero work flow is
going to start.>

This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html

If you require further assistance please contact us by sending email to
export@cisco.com.

cisco ISR4451-X/K9 (2RU) processor with 7941237K/6147K bytes of memory.
Processor board ID FJC1950D091
4 Gigabit Ethernet interfaces
32768K bytes of non-volatile configuration memory.
16777216K bytes of physical memory.
7341807K bytes of flash memory at bootflash:.
0K bytes of WebUI ODM Files at webui:.

%INIT: waited 0 seconds for NVRAM to be available



         --- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: %
!!<DO NOT TOUCH. This is Zero-Touch Provisioning>>
Generating 2048 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 1 seconds)
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
Guestshell enabled successfully


 *** Sample ZTP Day0 Python Script ***


 *** Configuring a Loopback Interface ***
```

```
Line 1 SUCCESS: interface loop 100
Line 2 SUCCESS: ip address 10.10.10.10 255.255.255.255
Line 3 SUCCESS: end


 *** Executing show ip interface brief  ***


Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  down                  down
GigabitEthernet0/0/1   unassigned      YES unset  down                  down
GigabitEthernet0/0/2   unassigned      YES unset  down                  down
GigabitEthernet0/0/3   192.168.1.246   YES DHCP   up                    up
GigabitEthernet0       192.168.1.246   YES DHCP   up                    up
Loopback100            10.10.10.10     YES TFTP   up                    up


 *** ZTP Day0 Python Script Execution Complete ***



Press RETURN to get started!
```

The day zero provisioning is complete, and the Cisco IOS prompt is accessible.

Boot Log for Cisco Catalyst 9000 Series Switches

The following sections display sample Zero-Touch Provisioning boot logs. These logs shows that Guest Shell is successfully enabled, the Python script is downloaded to the Guest Shell, and the Guest Shell executes the downloaded Python script and configures the device for Day Zero.

=

```
% Checking backup nvram
% No config present. Using default config

FIPS: Flash Key Check : Begin
FIPS: Flash Key Check : End, Not Found, FIPS Mode Not Enabled

! <This message indicates that the startup configuration
is absent on the device. This is the first indication that the Day Zero work flow is
going to start.>
```

Cisco IOS XE Everest 16.6.x to Cisco IOS XE Fuji 16.8.x

The following example shows the sample boot logs before the .py script is run:

```
Press RETURN to get started!

The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
```

```
The process for the command is not responding or is otherwise unavailable


 *** Sample ZTP Day0 Python Script ***

...

 *** ZTP Day0 Python Script Execution Complete ***
```

The following example shows how to configure the device for Day Zero provisioning:

```
Initializing Hardware...

System Bootstrap, Version 17.2.1r[FC1], RELEASE SOFTWARE (P)
Compiled Thu 02/20/2020 23:47:51.50 by rel

Current ROMMON image : Primary
Last reset cause     : SoftwareReload
C9300-48UXM platform with 8388608 Kbytes of main memory

Preparing to autoboot. [Press Ctrl-C to interrupt]  0
boot: attempting to boot from [flash:cat9k_iosxe.16.06.05.SPA.bin]
boot: reading file cat9k_iosxe.16.06.05.SPA.bin
###############################################################################################


Both links down, not waiting for other switches
Switch number is 1

              Restricted Rights Legend

Use, duplication, or disclosure by the Government is
subject to restrictions as set forth in subparagraph
(c) of the Commercial Computer Software - Restricted
Rights clause at FAR sec. 52.227-19 and subparagraph
(c) (1) (ii) of the Rights in Technical Data and Computer
Software clause at DFARS sec. 252.227-7013.

           cisco Systems, Inc.
           170 West Tasman Drive
           San Jose, California 95134-1706



Cisco IOS Software [Everest], Catalyst L3 Switch Software (CAT9K_IOSXE),
Version 16.6.5, RELEASE SOFTWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
Compiled Mon 10-Dec-18 12:52 by mcpre



Cisco IOS-XE software, Copyright (c) 2005-2018 by cisco Systems, Inc.
All rights reserved.  Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0.  The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY.  You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0.  For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.
```

```
% Checking backup nvram
% No config present. Using default config


FIPS: Flash Key Check : Begin
FIPS: Flash Key Check : End, Not Found, FIPS Mode Not Enabled

This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html

If you require further assistance please contact us by sending email to
export@cisco.com.

cisco C9300-48UXM (X86) processor with 1392780K/6147K bytes of memory.
Processor board ID FCW2144L045
2048K bytes of non-volatile configuration memory.
8388608K bytes of physical memory.
1638400K bytes of Crash Files at crashinfo:.
11264000K bytes of Flash at flash:.
0K bytes of WebUI ODM Files at webui:.

Base Ethernet MAC Address          : ec:1d:8b:0a:68:00
Motherboard Assembly Number        : 73-17959-06
Motherboard Serial Number          : FOC21418FPQ
Model Revision Number              : B0
Motherboard Revision Number        : A0
Model Number                       : C9300-48UXM
System Serial Number               : FCW2144L045


%INIT: waited 0 seconds for NVRAM to be available

SETUP: new interface Vlan1 placed in "shutdown" state


Press RETURN to get started!


*Sep  4 20:35:07.330: %SMART_LIC-6-AGENT_READY: Smart Agent for Licensing is initialized
*Sep  4 20:35:07.493: %IOSXE_RP_NV-3-NV_ACCESS_FAIL: Initial read of NVRAM contents failed
*Sep  4 20:35:07.551: %IOSXE_RP_NV-3-BACKUP_NV_ACCESS_FAIL: Initial read of backup NVRAM
contents failed
*Sep  4 20:35:10.932: dev_pluggable_optics_selftest attribute table internally inconsistent
 @ 0x1D4

*Sep  4 20:35:13.406: %CRYPTO-4-AUDITWARN: Encryption audit check could not be performed
*Sep  4 20:35:13.480: %SPANTREE-5-EXTENDED_SYSID: Extended SysId enabled for type vlan
*Sep  4 20:35:13.715: %LINK-3-UPDOWN: Interface Lsmpi18/3, changed state to up
*Sep  4 20:35:13.724: %LINK-3-UPDOWN: Interface EOBC18/1, changed state to up
*Sep  4 20:35:13.724: %LINEPROTO-5-UPDOWN: Line protocol on Interface LI-Null0, changed
state to up
*Sep  4 20:35:13.724: %LINK-3-UPDOWN: Interface GigabitEthernet0/0, changed state to down
*Sep  4 20:35:13.725: %LINK-3-UPDOWN: Interface LIIN18/2, changed state to up
*Sep  4 20:35:13.749: WCM-PKI-SHIM: buffer allocation failed for SUDI support check
*Sep  4 20:35:13.749: PKI/SSL unable to send Sudi support to WCM
```

```
*Sep  4 20:35:14.622: %IOSXE_MGMTVRF-6-CREATE_SUCCESS_INFO: Management vrf Mgmt-vrf created
 with ID 1,
    ipv4 table-id 0x1, ipv6 table-id 0x1E000001
*Sep  4 20:34:42.022: %STACKMGR-6-STACK_LINK_CHANGE: Switch 1 R0/0: stack_mgr:  Stack port
 1 on Switch 1 is nocable
*Sep  4 20:34:42.022: %STACKMGR-6-STACK_LINK_CHANGE: Switch 1 R0/0: stack_mgr:  Stack port
 2 on Switch 1 is down
*Sep  4 20:34:42.022: %STACKMGR-6-STACK_LINK_CHANGE: Switch 1 R0/0: stack_mgr:  Stack port
 2 on Switch 1 is nocable
*Sep  4 20:34:42.022: %STACKMGR-6-SWITCH_ADDED: Switch 1 R0/0: stack_mgr:  Switch 1 has
been added to the stack.
*Sep  4 20:34:42.022: %STACKMGR-6-SWITCH_ADDED: Switch 1 R0/0: stack_mgr:  Switch 1 has
been added to the stack.
*Sep  4 20:34:42.022: %STACKMGR-6-SWITCH_ADDED: Switch 1 R0/0: stack_mgr:  Switch 1 has
been added to the stack.
*Sep  4 20:34:42.022: %STACKMGR-6-ACTIVE_ELECTED: Switch 1 R0/0: stack_mgr:  Switch 1 has
been elected ACTIVE.
*Sep  4 20:35:14.728: %LINEPROTO-5-UPDOWN: Line protocol on Interface Lsmpi18/3, changed
state to up
*Sep  4 20:35:14.728: %LINEPROTO-5-UPDOWN: Line protocol on Interface EOBC18/1, changed
state to up
*Sep  4 20:35:15.506: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for switch
1: EMP_RELAY: Channel UP!
*Sep  4 20:35:15.510: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan1, changed state
 to down
*Sep  4 20:35:34.501: %LINK-5-CHANGED: Interface Vlan1, changed state to administratively
down
*Sep  4 20:35:34.717: %SYS-5-RESTART: System restarted --
Cisco IOS Software [Everest], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.6.5,
RELEASE SOFTWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
Compiled Mon 10-Dec-18 12:52 by mcpre
*Sep  4 20:35:34.796: %LINK-3-UPDOWN: Interface GigabitEthernet0/0, changed state to up
*Sep  4 20:35:35.266: %SYS-6-BOOTTIME: Time taken to reboot after reload =  283 seconds
*Sep  4 20:35:35.796: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0,
changed state to up
*Sep  4 20:35:36.607: %LINK-3-UPDOWN: Interface GigabitEthernet1/1/1, changed state to down
*Sep  4 20:35:36.607: %LINK-3-UPDOWN: Interface GigabitEthernet1/1/2, changed state to down
*Sep  4 20:35:36.607: %LINK-3-UPDOWN: Interface GigabitEthernet1/1/3, changed state to down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface GigabitEthernet1/1/4, changed state to down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/1, changed state to
down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/2, changed state to
down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/3, changed state to
down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/4, changed state to
down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/5, changed state to
down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/6, changed state to
down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/7, changed state to
down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/8, changed state to
down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface FortyGigabitEthernet1/1/1, changed state
to down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface FortyGigabitEthernet1/1/2, changed state
to down
*Sep  4 20:35:37.607: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/1/1,
 changed state to down
*Sep  4 20:35:37.608: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/1/2,
```

```
 changed state to down
*Sep  4 20:35:37.608: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/1/3,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/1/4,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/1/1,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/1/2,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/1/3,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/1/4,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/1/5,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/1/6,
 changed state to down
*Sep  4 20:35:43.511: AUTOINSTALL: Obtain tftp server address (opt 150) 159.14.27.2
*Sep  4 20:35:43.511: PNPA: Setting autoinstall complete to true for 159.14.27.2
*Sep  4 20:35:57.673: %PLATFORM_PM-6-FRULINK_INSERTED: 8x10G uplink module inserted in the
 switch 1 slot 1
*Sep  4 20:36:19.562: [IOX DEBUG] Guestshell start API is being invoked

*Sep  4 20:36:19.562: [IOX DEBUG] provided idb is mgmt interface

*Sep  4 20:36:19.562: [IOX DEBUG] Setting up guestshell to use mgmt-intf

*Sep  4 20:36:19.562: [IOX DEBUG] Setting up chasfs for iox related activity

*Sep  4 20:36:19.562: [IOX DEBUG] Setting up for iox pre-clean activity if needed

*Sep  4 20:36:19.562: [IOX DEBUG] Waiting for iox pre-clean setup to take affect

*Sep  4 20:36:19.562: [IOX DEBUG] Waited for 1 sec(s) for iox pre-clean setup to take affect

*Sep  4 20:36:19.562: [IOX DEBUG] Auto-configuring iox feature

*Sep  4 20:36:19.563: [IOX DEBUG] Waiting for CAF and ioxman to be up, in that order

*Sep  4 20:36:20.076: %UICFGEXP-6-SERVER_NOTIFIED_START: Switch 1 R0/0: psd:  Server iox
has been notified to start
*Sep  4 20:36:23.564: [IOX DEBUG] Waiting for another 5 secs

*Sep  4 20:36:28.564: [IOX DEBUG] Waiting for another 5 secs
The process for the command is not responding or is otherwise unavailable

*Sep  4 20:36:33.564: [IOX DEBUG] Waiting for another 5 secs
The process for the command is not responding or is otherwise unavailable

*Sep  4 20:36:34.564: [IOX DEBUG] Waited for 16 sec(s) for CAF and ioxman to come up

*Sep  4 20:36:34.564: [IOX DEBUG] Validating if CAF and ioxman are running

*Sep  4 20:36:34.564: [IOX DEBUG] CAF and ioxman are up and running

*Sep  4 20:36:34.564: [IOX DEBUG] Building the simple mgmt-intf enable command string

*Sep  4 20:36:34.564: [IOX DEBUG] Enable command is: request platform software iox-manager

    app-hosting guestshell enable

*Sep  4 20:36:34.564: [IOX DEBUG] Issuing guestshell enable command and waiting for it to
be up
The process for the command is not responding or is otherwise unavailable
```

```
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable

*Sep  4 20:36:38.578: [IOX DEBUG] Waiting for another 5 secs
The process for the command is not responding or is otherwise unavailable

*Sep  4 20:36:39.416: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/0/48, changed state to
 up
*Sep  4 20:36:40.416: %LINEPROTO-5-UPDOWN: Line protocol on Interface
TenGigabitEthernet1/0/48,
    changed state to upThe process for the command is not responding or is otherwise
unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable

*Sep  4 20:36:43.586: [IOX DEBUG] Waiting for another 5 secs
Guestshell enabled successfully

*Sep  4 20:37:45.321: [IOX DEBUG] Checking for guestshell mount path

*Sep  4 20:37:45.321: [IOX DEBUG] Validating if guestshell is ready for use

*Sep  4 20:37:45.321: [IOX DEBUG] Guestshell enabled successfully


 *** Sample ZTP Day0 Python Script ***



 *** Executing show platform  ***


Switch  Ports    Model               Serial No.   MAC address     Hw Ver.       Sw Ver.
------  -----   ---------            -----------  --------------  -------       --------
 1       62     C9300-48UXM          FCW2144L045  ec1d.8b0a.6800  V01           16.6.5

Switch/Stack Mac Address : ec1d.8b0a.6800 - Local Mac Address
Mac persistency wait time: Indefinite
                                     Current
Switch#   Role        Priority      State
-----------------------------------------
*1       Active          1          Ready


 *** Executing show version ***


Cisco IOS XE Software, Version 16.06.05
Cisco IOS Software [Everest], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.6.5,
RELEASE SOFTWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
Compiled Mon 10-Dec-18 12:52 by mcpre
Cisco IOS-XE software, Copyright (c) 2005-2018 by cisco Systems, Inc.
All rights reserved.  Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0.  The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY.  You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0.  For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.
```

```
ROM: IOS-XE ROMMON
BOOTLDR: System Bootstrap, Version 17.2.1r[FC1], RELEASE SOFTWARE (P)
Switch uptime is 2 minutes
Uptime for this control processor is 4 minutes
System returned to ROM by Reload Command
System image file is "flash:cat9k_iosxe.16.06.05.SPA.bin"
Last reload reason: Reload Command
This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.
A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html
If you require further assistance please contact us by sending email to
export@cisco.com.
Technology Package License Information:
------------------------------------------------------------------
Technology-package                  Technology-package
Current             Type            Next reboot
------------------------------------------------------------------
network-advantage   Permanent       network-advantage
cisco C9300-48UXM (X86) processor with 1392780K/6147K bytes of memory.
Processor board ID FCW2144L045
36 Ethernet interfaces
1 Virtual Ethernet interface
4 Gigabit Ethernet interfaces
20 Ten Gigabit Ethernet interfaces
2 Forty Gigabit Ethernet interfaces
2048K bytes of non-volatile configuration memory.
8388608K bytes of physical memory.
1638400K bytes of Crash Files at crashinfo:.
11264000K bytes of Flash at flash:.
0K bytes of WebUI ODM Files at webui:.
Base Ethernet MAC Address          : ec:1d:8b:0a:68:00
Motherboard Assembly Number        : 73-17959-06
Motherboard Serial Number          : FOC21418FPQ
Model Revision Number              : B0
Motherboard Revision Number        : A0
Model Number                       : C9300-48UXM
System Serial Number               : FCW2144L045
Switch Ports Model              SW Version        SW Image              Mode
------ ----- -----              ----------        ----------            ----
*    1 62    C9300-48UXM        16.6.5            CAT9K_IOSXE           BUNDLE
Configuration register is 0x102


 *** Configuring a Loopback Interface ***


Line 1 SUCCESS: interface loop 100
Line 2 SUCCESS: ip address 10.10.10.10 255.255.255.255
Line 3 SUCCESS: end


 *** Executing show ip interface brief  ***


Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES unset  administratively down down
GigabitEthernet0/0     10.127.128.3    YES DHCP   up                    up
```

```
Tw1/0/1                unassigned      YES unset  down                  down
Tw1/0/2                unassigned      YES unset  down                  down
Tw1/0/3                unassigned      YES unset  down                  down
Tw1/0/4                unassigned      YES unset  down                  down
Tw1/0/5                unassigned      YES unset  down                  down
Tw1/0/6                unassigned      YES unset  down                  down
Tw1/0/7                unassigned      YES unset  down                  down
Tw1/0/8                unassigned      YES unset  down                  down
Tw1/0/9                unassigned      YES unset  down                  down
Tw1/0/10               unassigned      YES unset  down                  down
Tw1/0/11               unassigned      YES unset  down                  down
Tw1/0/12               unassigned      YES unset  down                  down
Tw1/0/13               unassigned      YES unset  down                  down
Tw1/0/14               unassigned      YES unset  down                  down
Tw1/0/15               unassigned      YES unset  down                  down
Tw1/0/16               unassigned      YES unset  down                  down
Tw1/0/17               unassigned      YES unset  down                  down
Tw1/0/18               unassigned      YES unset  down                  down
Tw1/0/19               unassigned      YES unset  down                  down
Tw1/0/20               unassigned      YES unset  down                  down
Tw1/0/21               unassigned      YES unset  down                  down
Tw1/0/22               unassigned      YES unset  down                  down
Tw1/0/23               unassigned      YES unset  down                  down
Tw1/0/24               unassigned      YES unset  down                  down
Tw1/0/25               unassigned      YES unset  down                  down
Tw1/0/26               unassigned      YES unset  down                  down
Tw1/0/27               unassigned      YES unset  down                  down
Tw1/0/28               unassigned      YES unset  down                  down
Tw1/0/29               unassigned      YES unset  down                  down
Tw1/0/30               unassigned      YES unset  down                  down
Tw1/0/31               unassigned      YES unset  down                  down
Tw1/0/32               unassigned      YES unset  down                  down
Tw1/0/33               unassigned      YES unset  down                  down
Tw1/0/34               unassigned      YES unset  down                  down
Tw1/0/35               unassigned      YES unset  down                  down
Tw1/0/36               unassigned      YES unset  down                  down
Te1/0/37               unassigned      YES unset  down                  down
Te1/0/38               unassigned      YES unset  down                  down
Te1/0/39               unassigned      YES unset  down                  down
Te1/0/40               unassigned      YES unset  down                  down
Te1/0/41               unassigned      YES unset  down                  down
Te1/0/42               unassigned      YES unset  down                  down
Te1/0/43               unassigned      YES unset  down                  down
Te1/0/44               unassigned      YES unset  down                  down
Te1/0/45               unassigned      YES unset  down                  down
Te1/0/46               unassigned      YES unset  down                  down
Te1/0/47               unassigned      YES unset  down                  down
Te1/0/48               unassigned      YES unset  up                    up
GigabitEthernet1/1/1   unassigned      YES unset  down                  down
GigabitEthernet1/1/2   unassigned      YES unset  down                  down
GigabitEthernet1/1/3   unassigned      YES unset  down                  down
GigabitEthernet1/1/4   unassigned      YES unset  down                  down
Te1/1/1                unassigned      YES unset  down                  down
Te1/1/2                unassigned      YES unset  down                  down
Te1/1/3                unassigned      YES unset  down                  down
Te1/1/4                unassigned      YES unset  down                  down
Te1/1/5                unassigned      YES unset  down                  down
Te1/1/6                unassigned      YES unset  down                  down
Te1/1/7                unassigned      YES unset  down                  down
Te1/1/8                unassigned      YES unset  down                  down
Fo1/1/1                unassigned      YES unset  down                  down
Fo1/1/2                unassigned      YES unset  down                  down
Loopback100            10.10.10.10     YES TFTP   up                    up
```

```
 *** Configuring username, password, SSH  ***


Line 1 SUCCESS: username cisco privilege 15 password cisco
Line 2 SUCCESS: ip domain name domain
Line 3 SUCCESS: line vty 0 15
Line 4 SUCCESS: login local
Line 5 SUCCESS: transport input all
Line 6 SUCCESS: end


 *** ZTP Day0 Python Script Execution Complete ***
```

Cisco IOS XE Fuji 16.9.x to Cisco IOS XE Gibraltar 16.11.x

The following example shows the sample boot logs before the .py script is run:

```
--- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: The process for the
command is not
responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
guestshell installed successfully
Current state is: DEPLOYED
guestshell activated successfully
Current state is: ACTIVATED
guestshell started successfully
Current state is: RUNNING
Guestshell enabled successfully
```

The following example shows how to configure the device for Day Zero provisioning:

```
Both links down, not waiting for other switches
Switch number is 1

              Restricted Rights Legend

Use, duplication, or disclosure by the Government is
subject to restrictions as set forth in subparagraph
(c) of the Commercial Computer Software - Restricted
Rights clause at FAR sec. 52.227-19 and subparagraph
(c) (1) (ii) of the Rights in Technical Data and Computer
Software clause at DFARS sec. 252.227-7013.

           Cisco Systems, Inc.
           170 West Tasman Drive
           San Jose, California 95134-1706



Cisco IOS Software [Fuji], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.9.4, RELEASE
 SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
```

```
Copyright (c) 1986-2019 by Cisco Systems, Inc.
Compiled Thu 22-Aug-19 18:14 by mcpre


PLEASE READ THE FOLLOWING TERMS CAREFULLY. INSTALLING THE LICENSE OR
LICENSE KEY PROVIDED FOR ANY CISCO SOFTWARE PRODUCT, PRODUCT FEATURE,
AND/OR SUBSEQUENTLY PROVIDED SOFTWARE FEATURES (COLLECTIVELY, THE
"SOFTWARE"), AND/OR USING SUCH SOFTWARE CONSTITUTES YOUR FULL
ACCEPTANCE OF THE FOLLOWING TERMS. YOU MUST NOT PROCEED FURTHER IF YOU
ARE NOT WILLING TO BE BOUND BY ALL THE TERMS SET FORTH HEREIN.

Your use of the Software is subject to the Cisco End User License Agreement
(EULA) and any relevant supplemental terms (SEULA) found at
http://www.cisco.com/c/en/us/about/legal/cloud-and-software/software-terms.html.

You hereby acknowledge and agree that certain Software and/or features are
licensed for a particular term, that the license to such Software and/or
features is valid only for the applicable term and that such Software and/or
features may be shut down or otherwise terminated by Cisco after expiration
of the applicable license term (e.g., 90-day trial period). Cisco reserves
the right to terminate any such Software feature electronically or by any
other means available. While Cisco may provide alerts, it is your sole
responsibility to monitor your usage of any such term Software feature to
ensure that your systems and networks are prepared for a shutdown of the
Software feature.


% Checking backup nvram
% No config present. Using default config

FIPS: Flash Key Check : Key Not Found, FIPS Mode Not Enabled
cisco C9300-48UXM (X86) processor with 1419044K/6147K bytes of memory.
Processor board ID FCW2144L045
2048K bytes of non-volatile configuration memory.
8388608K bytes of physical memory.
1638400K bytes of Crash Files at crashinfo:.
11264000K bytes of Flash at flash:.
0K bytes of WebUI ODM Files at webui:.

Base Ethernet MAC Address          : ec:1d:8b:0a:68:00
Motherboard Assembly Number        : 73-17959-06
Motherboard Serial Number          : FOC21418FPQ
Model Revision Number              : B0
Motherboard Revision Number        : A0
Model Number                       : C9300-48UXM
System Serial Number               : FCW2144L045


%INIT: waited 0 seconds for NVRAM to be available


         --- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: The process for the
command is not
    responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
```

```
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
The process for the command is not responding or is otherwise unavailable
guestshell installed successfully
Current state is: DEPLOYED
guestshell activated successfully
Current state is: ACTIVATED
guestshell started successfully
Current state is: RUNNING
Guestshell enabled successfully


HTTP server statistics:
Accepted connections total: 0

 *** Sample ZTP Day0 Python Script ***
```

```
 *** Executing show platform  ***


Switch  Ports    Model               Serial No.   MAC address     Hw Ver.       Sw Ver.
------  -----   ---------            -----------  --------------  -------       --------
 1       64     C9300-48UXM          FCW2144L045  ec1d.8b0a.6800  V01           16.9.4

Switch/Stack Mac Address : ec1d.8b0a.6800 - Local Mac Address
Mac persistency wait time: Indefinite
                                     Current
Switch#   Role        Priority      State
-----------------------------------------
*1       Active          1          Ready


 *** Executing show version ***


Cisco IOS XE Software, Version 16.09.04
Cisco IOS Software [Fuji], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.9.4, RELEASE
 SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2019 by Cisco Systems, Inc.
Compiled Thu 22-Aug-19 18:14 by mcpre
Cisco IOS-XE software, Copyright (c) 2005-2019 by cisco Systems, Inc.
All rights reserved.  Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0.  The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY.  You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0.  For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.
ROM: IOS-XE ROMMON
BOOTLDR: System Bootstrap, Version 17.2.1r[FC1], RELEASE SOFTWARE (P)
Switch uptime is 4 minutes
Uptime for this control processor is 5 minutes
System returned to ROM by Reload Command
System image file is "flash:cat9k_iosxe.16.09.04.SPA.bin"
Last reload reason: Reload Command
This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.
A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html
If you require further assistance please contact us by sending email to
export@cisco.com.
Technology Package License Information:
------------------------------------------------------------------------------
Technology-package                                  Technology-package
Current                       Type                     Next reboot
------------------------------------------------------------------------------
network-advantage       Smart License                  network-advantage
None                    Subscription Smart License     None
Smart Licensing Status: UNREGISTERED/EVAL EXPIRED
cisco C9300-48UXM (X86) processor with 1419044K/6147K bytes of memory.
Processor board ID FCW2144L045
36 Ethernet interfaces
```

```
1 Virtual Ethernet interface
4 Gigabit Ethernet interfaces
20 Ten Gigabit Ethernet interfaces
2 TwentyFive Gigabit Ethernet interfaces
2 Forty Gigabit Ethernet interfaces
2048K bytes of non-volatile configuration memory.
8388608K bytes of physical memory.
1638400K bytes of Crash Files at crashinfo:.
11264000K bytes of Flash at flash:.
0K bytes of WebUI ODM Files at webui:.
Base Ethernet MAC Address          : ec:1d:8b:0a:68:00
Motherboard Assembly Number        : 73-17959-06
Motherboard Serial Number          : FOC21418FPQ
Model Revision Number              : B0
Motherboard Revision Number        : A0
Model Number                       : C9300-48UXM
System Serial Number               : FCW2144L045
Switch Ports Model              SW Version        SW Image              Mode
------ ----- -----              ----------        ----------            ----
*    1 64    C9300-48UXM        16.9.4            CAT9K_IOSXE           BUNDLE
Configuration register is 0x102


 *** Configuring a Loopback Interface ***


Line 1 SUCCESS: interface loop 100
Line 2 SUCCESS: ip address 10.10.10.10 255.255.255.255
Line 3 SUCCESS: end


 *** Executing show ip interface brief  ***


Any interface listed with OK? value "NO" does not have a valid configuration
Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      NO  unset  up                    up
GigabitEthernet0/0     10.127.128.5    YES DHCP   up                    up
Tw1/0/1                unassigned      YES unset  down                  down
Tw1/0/2                unassigned      YES unset  down                  down
Tw1/0/3                unassigned      YES unset  down                  down
Tw1/0/4                unassigned      YES unset  down                  down
Tw1/0/5                unassigned      YES unset  down                  down
Tw1/0/6                unassigned      YES unset  down                  down
Tw1/0/7                unassigned      YES unset  down                  down
Tw1/0/8                unassigned      YES unset  down                  down
Tw1/0/9                unassigned      YES unset  down                  down
Tw1/0/10               unassigned      YES unset  down                  down
Tw1/0/11               unassigned      YES unset  down                  down
Tw1/0/12               unassigned      YES unset  down                  down
Tw1/0/13               unassigned      YES unset  down                  down
Tw1/0/14               unassigned      YES unset  down                  down
Tw1/0/15               unassigned      YES unset  down                  down
Tw1/0/16               unassigned      YES unset  down                  down
Tw1/0/17               unassigned      YES unset  down                  down
Tw1/0/18               unassigned      YES unset  down                  down
Tw1/0/19               unassigned      YES unset  down                  down
Tw1/0/20               unassigned      YES unset  down                  down
Tw1/0/21               unassigned      YES unset  down                  down
Tw1/0/22               unassigned      YES unset  down                  down
Tw1/0/23               unassigned      YES unset  down                  down
Tw1/0/24               unassigned      YES unset  down                  down
Tw1/0/25               unassigned      YES unset  down                  down
Tw1/0/26               unassigned      YES unset  down                  down
```

```
Tw1/0/27               unassigned      YES unset  down                  down
Tw1/0/28               unassigned      YES unset  down                  down
Tw1/0/29               unassigned      YES unset  down                  down
Tw1/0/30               unassigned      YES unset  down                  down
Tw1/0/31               unassigned      YES unset  down                  down
Tw1/0/32               unassigned      YES unset  down                  down
Tw1/0/33               unassigned      YES unset  down                  down
Tw1/0/34               unassigned      YES unset  down                  down
Tw1/0/35               unassigned      YES unset  down                  down
Tw1/0/36               unassigned      YES unset  down                  down
Te1/0/37               unassigned      YES unset  down                  down
Te1/0/38               unassigned      YES unset  down                  down
Te1/0/39               unassigned      YES unset  down                  down
Te1/0/40               unassigned      YES unset  down                  down
Te1/0/41               unassigned      YES unset  down                  down
Te1/0/42               unassigned      YES unset  down                  down
Te1/0/43               unassigned      YES unset  down                  down
Te1/0/44               unassigned      YES unset  down                  down
Te1/0/45               unassigned      YES unset  down                  down
Te1/0/46               unassigned      YES unset  down                  down
Te1/0/47               unassigned      YES unset  down                  down
Te1/0/48               unassigned      YES unset  up                    up
GigabitEthernet1/1/1   unassigned      YES unset  down                  down
GigabitEthernet1/1/2   unassigned      YES unset  down                  down
GigabitEthernet1/1/3   unassigned      YES unset  down                  down
GigabitEthernet1/1/4   unassigned      YES unset  down                  down
Te1/1/1                unassigned      YES unset  down                  down
Te1/1/2                unassigned      YES unset  down                  down
Te1/1/3                unassigned      YES unset  down                  down
Te1/1/4                unassigned      YES unset  down                  down
Te1/1/5                unassigned      YES unset  down                  down
Te1/1/6                unassigned      YES unset  down                  down
Te1/1/7                unassigned      YES unset  down                  down
Te1/1/8                unassigned      YES unset  down                  down
Fo1/1/1                unassigned      YES unset  down                  down
Fo1/1/2                unassigned      YES unset  down                  down
TwentyFiveGigE1/1/1    unassigned      YES unset  down                  down
TwentyFiveGigE1/1/2    unassigned      YES unset  down                  down
Loopback100            10.10.10.10     YES TFTP   up                    up


 *** Configuring username, password, SSH  ***


Line 1 SUCCESS: username cisco privilege 15 password cisco
**CLI Line # 1:  WARNING: Command has been added to the configuration using a type 0 password.

    However, type 0 passwords will soon be deprecated. Migrate to a supported password type
Line 2 SUCCESS: ip domain name domain
Line 3 SUCCESS: line vty 0 15
Line 4 SUCCESS: login local
Line 5 SUCCESS: transport input all
Line 6 SUCCESS: end


 *** ZTP Day0 Python Script Execution Complete ***




Press RETURN to get started!
```

Cisco IOS XE Gibraltar 16.12.x to Cisco IOS XE Amsterdam 17.1.x

The following example shows the sample boot logs before the .py script is run:

```
--- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: day0guestshell installed
 successfully
Current state is: DEPLOYED
day0guestshell activated successfully
Current state is: ACTIVATED
day0guestshell started successfully
Current state is: RUNNING
Guestshell enabled successfully


 *** Sample ZTP Day0 Python Script ***

...

 *** ZTP Day0 Python Script Execution Complete ***


Guestshell destroyed successfully
```

The following example shows how to configure the device for Day Zero provisioning:

```
Both links down, not waiting for other switches
Switch number is 1

              Restricted Rights Legend

Use, duplication, or disclosure by the Government is
subject to restrictions as set forth in subparagraph
(c) of the Commercial Computer Software - Restricted
Rights clause at FAR sec. 52.227-19 and subparagraph
(c) (1) (ii) of the Rights in Technical Data and Computer
Software clause at DFARS sec. 252.227-7013.

           Cisco Systems, Inc.
           170 West Tasman Drive
           San Jose, California 95134-1706



Cisco IOS Software [Gibraltar], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.12.3a,

RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2020 by Cisco Systems, Inc.
Compiled Tue 28-Apr-20 09:37 by mcpre


This software version supports only Smart Licensing as the software licensing mechanism.


PLEASE READ THE FOLLOWING TERMS CAREFULLY. INSTALLING THE LICENSE OR
LICENSE KEY PROVIDED FOR ANY CISCO SOFTWARE PRODUCT, PRODUCT FEATURE,
```

```
AND/OR SUBSEQUENTLY PROVIDED SOFTWARE FEATURES (COLLECTIVELY, THE
"SOFTWARE"), AND/OR USING SUCH SOFTWARE CONSTITUTES YOUR FULL
ACCEPTANCE OF THE FOLLOWING TERMS. YOU MUST NOT PROCEED FURTHER IF YOU
ARE NOT WILLING TO BE BOUND BY ALL THE TERMS SET FORTH HEREIN.

Your use of the Software is subject to the Cisco End User License Agreement
(EULA) and any relevant supplemental terms (SEULA) found at
http://www.cisco.com/c/en/us/about/legal/cloud-and-software/software-terms.html.

You hereby acknowledge and agree that certain Software and/or features are
licensed for a particular term, that the license to such Software and/or
features is valid only for the applicable term and that such Software and/or
features may be shut down or otherwise terminated by Cisco after expiration
of the applicable license term (e.g., 90-day trial period). Cisco reserves
the right to terminate any such Software feature electronically or by any
other means available. While Cisco may provide alerts, it is your sole
responsibility to monitor your usage of any such term Software feature to
ensure that your systems and networks are prepared for a shutdown of the
Software feature.


% Checking backup nvram
% No config present. Using default config

FIPS: Flash Key Check : Key Not Found, FIPS Mode Not Enabled

All TCP AO KDF Tests Pass
cisco C9300-48UXM (X86) processor with 1343703K/6147K bytes of memory.
Processor board ID FCW2144L045
2048K bytes of non-volatile configuration memory.
8388608K bytes of physical memory.
1638400K bytes of Crash Files at crashinfo:.
11264000K bytes of Flash at flash:.
0K bytes of WebUI ODM Files at webui:.

Base Ethernet MAC Address          : ec:1d:8b:0a:68:00
Motherboard Assembly Number        : 73-17959-06
Motherboard Serial Number          : FOC21418FPQ
Model Revision Number              : B0
Motherboard Revision Number        : A0
Model Number                       : C9300-48UXM
System Serial Number               : FCW2144L045


         --- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: day0guestshell installed
 successfully
Current state is: DEPLOYED
day0guestshell activated successfully
Current state is: ACTIVATED
day0guestshell started successfully
Current state is: RUNNING
Guestshell enabled successfully


HTTP server statistics:
Accepted connections total: 0

 *** Sample ZTP Day0 Python Script ***
```

```
 *** Executing show platform  ***


Switch  Ports    Model               Serial No.   MAC address     Hw Ver.       Sw Ver.
------  -----   ---------            -----------  --------------  -------       --------
 1       65     C9300-48UXM          FCW2144L045  ec1d.8b0a.6800  V01           16.12.3a

Switch/Stack Mac Address : ec1d.8b0a.6800 - Local Mac Address
Mac persistency wait time: Indefinite
                                     Current
Switch#   Role        Priority      State
-----------------------------------------
*1       Active          1          Ready


 *** Executing show version ***


Cisco IOS XE Software, Version 16.12.03a
Cisco IOS Software [Gibraltar], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.12.3a,

RELEASE SOFTWARE (fc1)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2020 by Cisco Systems, Inc.
Compiled Tue 28-Apr-20 09:37 by mcpre
Cisco IOS-XE software, Copyright (c) 2005-2020 by cisco Systems, Inc.
All rights reserved.  Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0.  The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY.  You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0.  For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.
ROM: IOS-XE ROMMON
BOOTLDR: System Bootstrap, Version 17.2.1r[FC1], RELEASE SOFTWARE (P)
Switch uptime is 4 minutes
Uptime for this control processor is 9 minutes
System returned to ROM by Reload Command
System image file is "flash:cat9k_iosxe.16.12.03a.SPA.bin"
Last reload reason: Reload Command
This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.
A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html
If you require further assistance please contact us by sending email to
export@cisco.com.
Technology Package License Information:
------------------------------------------------------------------------------
Technology-package                                  Technology-package
Current                       Type                     Next reboot
------------------------------------------------------------------------------
network-advantage       Smart License                  network-advantage
None                    Subscription Smart License     None
AIR License Level: AIR DNA Advantage
Next reload AIR license Level: AIR DNA Advantage
Smart Licensing Status: UNREGISTERED/EVAL EXPIRED
```

```
cisco C9300-48UXM (X86) processor with 1343703K/6147K bytes of memory.
Processor board ID FCW2144L045
1 Virtual Ethernet interface
4 Gigabit Ethernet interfaces
36 2.5 Gigabit Ethernet interfaces
20 Ten Gigabit Ethernet interfaces
2 TwentyFive Gigabit Ethernet interfaces
2 Forty Gigabit Ethernet interfaces
2048K bytes of non-volatile configuration memory.
8388608K bytes of physical memory.
1638400K bytes of Crash Files at crashinfo:.
11264000K bytes of Flash at flash:.
0K bytes of WebUI ODM Files at webui:.
Base Ethernet MAC Address          : ec:1d:8b:0a:68:00
Motherboard Assembly Number        : 73-17959-06
Motherboard Serial Number          : FOC21418FPQ
Model Revision Number              : B0
Motherboard Revision Number        : A0
Model Number                       : C9300-48UXM
System Serial Number               : FCW2144L045
Switch Ports Model              SW Version        SW Image              Mode
------ ----- -----              ----------        ----------            ----
*    1 65    C9300-48UXM        16.12.3a          CAT9K_IOSXE           BUNDLE
Configuration register is 0x102


 *** Configuring a Loopback Interface ***


Line 1 SUCCESS: interface loop 100
Line 2 SUCCESS: ip address 10.10.10.10 255.255.255.255
Line 3 SUCCESS: end


 *** Executing show ip interface brief  ***


Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES unset  up                    up
GigabitEthernet0/0     10.127.128.10   YES DHCP   up                    up
Tw1/0/1                unassigned      YES unset  down                  down
Tw1/0/2                unassigned      YES unset  down                  down
Tw1/0/3                unassigned      YES unset  down                  down
Tw1/0/4                unassigned      YES unset  down                  down
Tw1/0/5                unassigned      YES unset  down                  down
Tw1/0/6                unassigned      YES unset  down                  down
Tw1/0/7                unassigned      YES unset  down                  down
Tw1/0/8                unassigned      YES unset  down                  down
Tw1/0/9                unassigned      YES unset  down                  down
Tw1/0/10               unassigned      YES unset  down                  down
Tw1/0/11               unassigned      YES unset  down                  down
Tw1/0/12               unassigned      YES unset  down                  down
Tw1/0/13               unassigned      YES unset  down                  down
Tw1/0/14               unassigned      YES unset  down                  down
Tw1/0/15               unassigned      YES unset  down                  down
Tw1/0/16               unassigned      YES unset  down                  down
Tw1/0/17               unassigned      YES unset  down                  down
Tw1/0/18               unassigned      YES unset  down                  down
Tw1/0/19               unassigned      YES unset  down                  down
Tw1/0/20               unassigned      YES unset  down                  down
Tw1/0/21               unassigned      YES unset  down                  down
Tw1/0/22               unassigned      YES unset  down                  down
Tw1/0/23               unassigned      YES unset  down                  down
Tw1/0/24               unassigned      YES unset  down                  down
```

```
Tw1/0/25               unassigned      YES unset  down                  down
Tw1/0/26               unassigned      YES unset  down                  down
Tw1/0/27               unassigned      YES unset  down                  down
Tw1/0/28               unassigned      YES unset  down                  down
Tw1/0/29               unassigned      YES unset  down                  down
Tw1/0/30               unassigned      YES unset  down                  down
Tw1/0/31               unassigned      YES unset  down                  down
Tw1/0/32               unassigned      YES unset  down                  down
Tw1/0/33               unassigned      YES unset  down                  down
Tw1/0/34               unassigned      YES unset  down                  down
Tw1/0/35               unassigned      YES unset  down                  down
Tw1/0/36               unassigned      YES unset  down                  down
Te1/0/37               unassigned      YES unset  down                  down
Te1/0/38               unassigned      YES unset  down                  down
Te1/0/39               unassigned      YES unset  down                  down
Te1/0/40               unassigned      YES unset  down                  down
Te1/0/41               unassigned      YES unset  down                  down
Te1/0/42               unassigned      YES unset  down                  down
Te1/0/43               unassigned      YES unset  down                  down
Te1/0/44               unassigned      YES unset  down                  down
Te1/0/45               unassigned      YES unset  down                  down
Te1/0/46               unassigned      YES unset  down                  down
Te1/0/47               unassigned      YES unset  down                  down
Te1/0/48               unassigned      YES unset  up                    up
GigabitEthernet1/1/1   unassigned      YES unset  down                  down
GigabitEthernet1/1/2   unassigned      YES unset  down                  down
GigabitEthernet1/1/3   unassigned      YES unset  down                  down
GigabitEthernet1/1/4   unassigned      YES unset  down                  down
Te1/1/1                unassigned      YES unset  down                  down
Te1/1/2                unassigned      YES unset  down                  down
Te1/1/3                unassigned      YES unset  down                  down
Te1/1/4                unassigned      YES unset  down                  down
Te1/1/5                unassigned      YES unset  down                  down
Te1/1/6                unassigned      YES unset  down                  down
Te1/1/7                unassigned      YES unset  down                  down
Te1/1/8                unassigned      YES unset  down                  down
Fo1/1/1                unassigned      YES unset  down                  down
Fo1/1/2                unassigned      YES unset  down                  down
TwentyFiveGigE1/1/1    unassigned      YES unset  down                  down
TwentyFiveGigE1/1/2    unassigned      YES unset  down                  down
Ap1/0/1                unassigned      YES unset  up                    up
Loopback100            10.10.10.10     YES TFTP   up                    up


 *** Configuring username, password, SSH  ***


Line 1 SUCCESS: username cisco privilege 15 password cisco
**CLI Line # 1:  WARNING: Command has been added to the configuration using a type 0 password.

However, type 0 passwords will soon be deprecated. Migrate to a supported password type
Line 2 SUCCESS: ip domain name domain
Line 3 SUCCESS: line vty 0 15
Line 4 SUCCESS: login local
Line 5 SUCCESS: transport input all
Line 6 SUCCESS: end


 *** ZTP Day0 Python Script Execution Complete ***


Guestshell destroyed successfully
```

```
Press RETURN to get started!
```

Cisco IOS XE Amsterdam 17.2.x and Later Releases

This following example shows the sample boot logs before the .py script is run:

```
--- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]:
Acquired IPv4 address 10.127.128.8 on Interface GigabitEthernet0/0
Received following DHCPv4 options:
        bootfile        : test.py
        tftp-server-ip  : 159.14.27.2

OK to enter CLI now...

pnp-discovery can be monitored without entering enable mode

Entering enable mode will stop pnp-discovery

Attempting bootfile tftp://159.14.27.2/test.py
day0guestshell activated successfully
Current state is: ACTIVATED
day0guestshell started successfully
Current state is: RUNNING
Guestshell enabled successfully


 *** Sample ZTP Day0 Python Script ***

...

 *** ZTP Day0 Python Script Execution Complete ***

Guestshell destroyed successfully
```

The following example shows how to configure the device for Day Zero provisioning:

```
Both links down, not waiting for other switches
Switch number is 1

              Restricted Rights Legend

Use, duplication, or disclosure by the Government is
subject to restrictions as set forth in subparagraph
(c) of the Commercial Computer Software - Restricted
Rights clause at FAR sec. 52.227-19 and subparagraph
(c) (1) (ii) of the Rights in Technical Data and Computer
Software clause at DFARS sec. 252.227-7013.

           Cisco Systems, Inc.
           170 West Tasman Drive
           San Jose, California 95134-1706


Cisco IOS Software [Amsterdam], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 17.2.1,
```

```
 RELEASE SOFTWARE (fc4)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2020 by Cisco Systems, Inc.
Compiled Thu 26-Mar-20 03:29 by mcpre


This software version supports only Smart Licensing as the software licensing mechanism.


PLEASE READ THE FOLLOWING TERMS CAREFULLY. INSTALLING THE LICENSE OR
LICENSE KEY PROVIDED FOR ANY CISCO SOFTWARE PRODUCT, PRODUCT FEATURE,
AND/OR SUBSEQUENTLY PROVIDED SOFTWARE FEATURES (COLLECTIVELY, THE
"SOFTWARE"), AND/OR USING SUCH SOFTWARE CONSTITUTES YOUR FULL
ACCEPTANCE OF THE FOLLOWING TERMS. YOU MUST NOT PROCEED FURTHER IF YOU
ARE NOT WILLING TO BE BOUND BY ALL THE TERMS SET FORTH HEREIN.

Your use of the Software is subject to the Cisco End User License Agreement
(EULA) and any relevant supplemental terms (SEULA) found at
http://www.cisco.com/c/en/us/about/legal/cloud-and-software/software-terms.html.

You hereby acknowledge and agree that certain Software and/or features are
licensed for a particular term, that the license to such Software and/or
features is valid only for the applicable term and that such Software and/or
features may be shut down or otherwise terminated by Cisco after expiration
of the applicable license term (e.g., 90-day trial period). Cisco reserves
the right to terminate any such Software feature electronically or by any
other means available. While Cisco may provide alerts, it is your sole
responsibility to monitor your usage of any such term Software feature to
ensure that your systems and networks are prepared for a shutdown of the
Software feature.


% Checking backup nvram
% No config present. Using default config

FIPS: Flash Key Check : Key Not Found, FIPS Mode Not Enabled

All TCP AO KDF Tests Pass
cisco C9300-48UXM (X86) processor with 1338934K/6147K bytes of memory.
Processor board ID FCW2144L045
2048K bytes of non-volatile configuration memory.
8388608K bytes of physical memory.
1638400K bytes of Crash Files at crashinfo:.
11264000K bytes of Flash at flash:.

Base Ethernet MAC Address          : ec:1d:8b:0a:68:00
Motherboard Assembly Number        : 73-17959-06
Motherboard Serial Number          : FOC21418FPQ
Model Revision Number              : B0
Motherboard Revision Number        : A0
Model Number                       : C9300-48UXM
System Serial Number               : FCW2144L045
CLEI Code Number                   :


No startup-config, starting autoinstall/pnp/ztp...

Autoinstall will terminate if any input is detected on console

Autoinstall trying DHCPv4 on GigabitEthernet0/0

Autoinstall trying DHCPv6 on GigabitEthernet0/0
```

```
         --- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]:
Acquired IPv4 address 10.127.128.8 on Interface GigabitEthernet0/0
Received following DHCPv4 options:
        bootfile        : test.py
        tftp-server-ip  : 159.14.27.2

OK to enter CLI now...

pnp-discovery can be monitored without entering enable mode

Entering enable mode will stop pnp-discovery

Attempting bootfile tftp://159.14.27.2/test.py
day0guestshell activated successfully
Current state is: ACTIVATED
day0guestshell started successfully
Current state is: RUNNING
Guestshell enabled successfully


 *** Sample ZTP Day0 Python Script ***



 *** Executing show platform  ***


Switch  Ports    Model                Serial No.   MAC address     Hw Ver.       Sw Ver.
------  -----   ---------             -----------  --------------  -------       --------
 1       65     C9300-48UXM           FCW2144L045  ec1d.8b0a.6800  V01           17.02.01

Switch/Stack Mac Address : ec1d.8b0a.6800 - Local Mac Address
Mac persistency wait time: Indefinite
                                     Current
Switch#   Role        Priority      State
-----------------------------------------
*1       Active          1          Ready


 *** Executing show version ***


Cisco IOS XE Software, Version 17.02.01
Cisco IOS Software [Amsterdam], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 17.2.1,
 RELEASE SOFTWARE (fc4)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2020 by Cisco Systems, Inc.
Compiled Thu 26-Mar-20 03:29 by mcpre
Cisco IOS-XE software, Copyright (c) 2005-2020 by cisco Systems, Inc.
All rights reserved.  Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0.  The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY.  You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0.  For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.
ROM: IOS-XE ROMMON
BOOTLDR: System Bootstrap, Version 17.2.1r[FC1], RELEASE SOFTWARE (P)
Switch uptime is 2 minutes
Uptime for this control processor is 8 minutes
```

```
System returned to ROM by Reload Command
System image file is "flash:cat9k_iosxe.17.02.01.SPA.bin"
Last reload reason: Reload Command
This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.
A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html
If you require further assistance please contact us by sending email to
export@cisco.com.
Technology Package License Information:
------------------------------------------------------------------------------
Technology-package                                          Technology-package
Current                       Type                             Next reboot
------------------------------------------------------------------------------
network-advantage       Smart License                    network-advantage
None                    Subscription Smart License       None
AIR License Level: AIR DNA Advantage
Next reload AIR license Level: AIR DNA Advantage
Smart Licensing Status: UNREGISTERED/EVAL EXPIRED
cisco C9300-48UXM (X86) processor with 1338934K/6147K bytes of memory.
Processor board ID FCW2144L045
1 Virtual Ethernet interface
4 Gigabit Ethernet interfaces
36 2.5 Gigabit Ethernet interfaces
20 Ten Gigabit Ethernet interfaces
2 TwentyFive Gigabit Ethernet interfaces
2 Forty Gigabit Ethernet interfaces
2048K bytes of non-volatile configuration memory.
8388608K bytes of physical memory.
1638400K bytes of Crash Files at crashinfo:.
11264000K bytes of Flash at flash:.
Base Ethernet MAC Address          : ec:1d:8b:0a:68:00
Motherboard Assembly Number        : 73-17959-06
Motherboard Serial Number          : FOC21418FPQ
Model Revision Number              : B0
Motherboard Revision Number        : A0
Model Number                       : C9300-48UXM
System Serial Number               : FCW2144L045
CLEI Code Number                   :
Switch Ports Model              SW Version        SW Image              Mode
------ ----- -----              ----------        ----------            ----
*    1 65    C9300-48UXM        17.02.01          CAT9K_IOSXE           BUNDLE
Configuration register is 0x102


 *** Configuring a Loopback Interface ***


Line 1 SUCCESS: interface loop 100
Line 2 SUCCESS: ip address 10.10.10.10 255.255.255.255
Line 3 SUCCESS: end


 *** Executing show ip interface brief  ***


Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES unset  up                    up
```

```
GigabitEthernet0/0     10.127.128.8    YES DHCP   up                    up
Tw1/0/1                unassigned      YES unset  down                  down
Tw1/0/2                unassigned      YES unset  down                  down
Tw1/0/3                unassigned      YES unset  down                  down
Tw1/0/4                unassigned      YES unset  down                  down
Tw1/0/5                unassigned      YES unset  down                  down
Tw1/0/6                unassigned      YES unset  down                  down
Tw1/0/7                unassigned      YES unset  down                  down
Tw1/0/8                unassigned      YES unset  down                  down
Tw1/0/9                unassigned      YES unset  down                  down
Tw1/0/10               unassigned      YES unset  down                  down
Tw1/0/11               unassigned      YES unset  down                  down
Tw1/0/12               unassigned      YES unset  down                  down
Tw1/0/13               unassigned      YES unset  down                  down
Tw1/0/14               unassigned      YES unset  down                  down
Tw1/0/15               unassigned      YES unset  down                  down
Tw1/0/16               unassigned      YES unset  down                  down
Tw1/0/17               unassigned      YES unset  down                  down
Tw1/0/18               unassigned      YES unset  down                  down
Tw1/0/19               unassigned      YES unset  down                  down
Tw1/0/20               unassigned      YES unset  down                  down
Tw1/0/21               unassigned      YES unset  down                  down
Tw1/0/22               unassigned      YES unset  down                  down
Tw1/0/23               unassigned      YES unset  down                  down
Tw1/0/24               unassigned      YES unset  down                  down
Tw1/0/25               unassigned      YES unset  down                  down
Tw1/0/26               unassigned      YES unset  down                  down
Tw1/0/27               unassigned      YES unset  down                  down
Tw1/0/28               unassigned      YES unset  down                  down
Tw1/0/29               unassigned      YES unset  down                  down
Tw1/0/30               unassigned      YES unset  down                  down
Tw1/0/31               unassigned      YES unset  down                  down
Tw1/0/32               unassigned      YES unset  down                  down
Tw1/0/33               unassigned      YES unset  down                  down
Tw1/0/34               unassigned      YES unset  down                  down
Tw1/0/35               unassigned      YES unset  down                  down
Tw1/0/36               unassigned      YES unset  down                  down
Te1/0/37               unassigned      YES unset  down                  down
Te1/0/38               unassigned      YES unset  down                  down
Te1/0/39               unassigned      YES unset  down                  down
Te1/0/40               unassigned      YES unset  down                  down
Te1/0/41               unassigned      YES unset  down                  down
Te1/0/42               unassigned      YES unset  down                  down
Te1/0/43               unassigned      YES unset  down                  down
Te1/0/44               unassigned      YES unset  down                  down
Te1/0/45               unassigned      YES unset  down                  down
Te1/0/46               unassigned      YES unset  down                  down
Te1/0/47               unassigned      YES unset  down                  down
Te1/0/48               unassigned      YES unset  up                    up
GigabitEthernet1/1/1   unassigned      YES unset  down                  down
GigabitEthernet1/1/2   unassigned      YES unset  down                  down
GigabitEthernet1/1/3   unassigned      YES unset  down                  down
GigabitEthernet1/1/4   unassigned      YES unset  down                  down
Te1/1/1                unassigned      YES unset  down                  down
Te1/1/2                unassigned      YES unset  down                  down
Te1/1/3                unassigned      YES unset  down                  down
Te1/1/4                unassigned      YES unset  down                  down
Te1/1/5                unassigned      YES unset  down                  down
Te1/1/6                unassigned      YES unset  down                  down
Te1/1/7                unassigned      YES unset  down                  down
Te1/1/8                unassigned      YES unset  down                  down
Fo1/1/1                unassigned      YES unset  down                  down
Fo1/1/2                unassigned      YES unset  down                  down
TwentyFiveGigE1/1/1    unassigned      YES unset  down                  down
```

```
TwentyFiveGigE1/1/2    unassigned      YES unset  down                  down
Ap1/0/1                unassigned      YES unset  up                    up
Loopback100            10.10.10.10     YES TFTP   up                    up


 *** Configuring username, password, SSH  ***


Line 1 SUCCESS: username cisco privilege 15 password cisco
**CLI Line # 1:  WARNING: Command has been added to the configuration using a type 0 password.

However, type 0 passwords will soon be deprecated. Migrate to a supported password type
Line 2 SUCCESS: ip domain name domain
Line 3 SUCCESS: line vty 0 15
Line 4 SUCCESS: login local
Line 5 SUCCESS: transport input all
Line 6 SUCCESS: end


 *** ZTP Day0 Python Script Execution Complete ***



Guestshell destroyed successfully
Script execution success!



Press RETURN to get started!
```

Additional References for Zero-Touch Provisioning

Standards and RFCs

Standard/RFC	Title
RFC 5652	*Cryptographic Message Syntax (CMS)*
RFC 8040	*RESTCONF Protocol*
RFC 8366	*A Voucher Artifact for Bootstrapping Protocols*
RFC 8572	*Secure Zero Touch Provisioning (SZTP)*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	https://www.cisco.com/c/en/us/support/index.html

Feature Information for Zero-Touch Provisioning

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 3: Feature Information for Zero-Touch Provisioning

Feature Name	Release	Feature Information
Zero-Touch Provisioning	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b Cisco IOS XE Fuji 16.7.1 Cisco IOS XE Fuji 16.8.2 Cisco IOS XE Gibraltar 16.12.1 Cisco IOS XE Amsterdam 17.2.1 Cisco IOS XE Amsterdam 17.3.1 Cisco IOS XE Cupertino 17.8.1 Cisco IOS XE Dublin 17.10.1b	

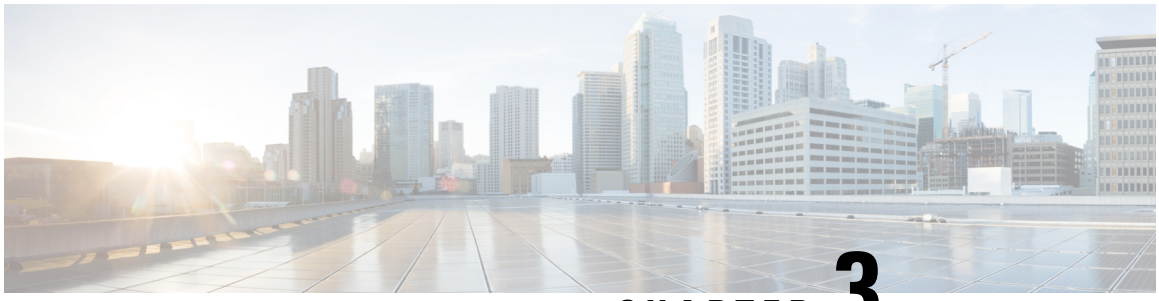
Feature Name	Release	Feature Information
		To address network provisioning challenges, Cisco introduces a zero-touch provisioning model. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platform: • Cisco 4000 Series Integrated Services Router models with a minimum of 8 GB RAM to support Guest Shell. In Cisco IOS XE Fuji 16.7.1, this feature was implemented on the following platform: • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) In Cisco IOS XE Fuji 16.8.2, this feature was implemented on the following platform: • Cisco ASR 1000 Series Aggregation Services Routers (ASR1004, ASR1006, ASR1006-X, ASR1009-X, ASR1013) In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on the following platforms: • Cisco Catalyst 9200 Series Switches **Note** This feature is not supported on C9200L SKUs. • Cisco Catalyst 9300L SKUs • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers

Feature Name	Release	Feature Information
		In Cisco IOS XE Amsterdam 17.2.1, this feature was implemented on the following platforms: • Cisco Cloud Services Router 1000V Series • Cisco C1100 Terminal Services Gateway (Supported only on C1100TGX-1N24P32A) In Cisco IOS XE Amsterdam 17.3.1, this feature was implemented on the following platforms: • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 and 8500L Series Edge Platforms In Cisco IOS XE Bengaluru 17.4.1, this feature was implemented on the following platform: • Cisco Catalyst 8000V Edge Software In Cisco IOS XE Cupertino 17.8.1, this feature was implemented on the following platform: • Cisco Catalyst 9800-L Wireless Controller In Cisco IOS XE Dublin 17.10.1b, this feature was implemented on the following platform: • Cisco Catalyst 9500X Series Switches

Feature Name	Release	Feature Information
Zero-Touch Provisioning: HTTP Download	Cisco IOS XE Fuji 16.8.1 Cisco IOS XE Fuji 16.8.1a	Zero-Touch Provisioning supports HTTP and TFTP file download. In Cisco IOS XE Everest 16.8.1, this feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches.
DHCPv6 Support for Zero-Touch Provisioning	Cisco IOS XE Fuji 16.9.1 Cisco IOS XE Amsterdam 17.3.2a	In Cisco IOS XE Fuji 16.9.1, this feature was implemented on the following platforms: • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches In Cisco IOS XE Amsterdam 17.3.2a, this feature was implemented on the following platforms: • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers

Feature Name	Release	Feature Information
Side-Effect Synchronization of the Configuration Database	Cisco IOS XE Bengaluru 17.4.1	During configuration changes in the DMI, a partial synchronization of the changes that are triggered when a command or RPC is configured occurs. This is called the side-effect synchronization, and it reduces the synchronization time and NETCONF downtime. This feature was implemented on the following platforms: • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 8500 and 8500L Series Edge Platforms • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches
Zero-Touch Provisioning Through YANG Models	Cisco IOS XE Cupertino 17.7.1	ZTP is enabled through YANG models when NETCONF is enabled. This feature is supported on all platforms that support NETCONF-YANG.
Zero-Touch Provisioning Support on Data Port	Cisco IOS XE Cupertino 17.7.1	ZTP is supported on data port for both IPv4 and IPv6. This feature is implemented on the following platform: • Cisco Catalyst 9800-L Wireless Controller

Feature Name	Release	Feature Information
Secure Zero-Touch Provisioning	Cisco IOS XE Dublin 17.11.1	Secure ZTP provisions a device securely while booting in factory default state. This feature is implemented on the following platforms: • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches



CHAPTER 3

iPXE

iPXE is an enhanced version of the Pre-boot eXecution Environment (PXE), which is an open standard for network booting. This module describes the iPXE feature and how to configure it.

- Information About iPXE, on page 85
- How to Configure iPXE, on page 93
- Configuration Examples for iPXE, on page 95
- Troubleshooting Tips for iPXE, on page 98
- Additional References for iPXE, on page 99
- Feature Information for iPXE, on page 99

Information About iPXE

About iPXE

iPXE is an enhanced version of the Pre-boot eXecution Environment (PXE), which is an open standard for network booting.

iPXE netboot provides:

- IPv4 and IPv6 protocols
- FTP/HTTP/TFTP boot image download
- Embedded scripts into the image
- Stateless and stateful address auto-configuration (SLAAC) using Dynamic Host Configuration Protocol Version 4 (DHCPv4) and/or DHCPv6, boot URI, and parameters for DHCPv6 options depending on the IPv6 router advertisement.

Netboot Requirements

The following are the primary requirements for netbooting:

- DHCP server with proper configuration.
- Boot image available on the FTP/HTTP/TFTP server.
- Device configured to boot from a network-based source.

iPXE Overview

Network bootloaders support booting from a network-based source. The bootloaders boot an image located on an HTTP, FTP, or TFTP server. A network boot source is detected automatically by using an iPXE-like solution.

iPXE enables network boot for a device that is offline. The following are the three types of boot modes:

- iPXE Timeout—Boots through iPXE network boot. Configures a timeout in seconds for iPXE network boot by using the IPXE_TIMEOUT rommon variable. Use the **boot ipxe timeout** command to configure iPXE timeout. When the timeout expires, device boot is activated.
- iPXE Forever—Boots through iPXE network boot. The device sends DHCP requests forever, when the **boot ipxe forever** command is configured. This is an iPXE-only boot (which means that the bootloader will not fall back to a device boot or a command prompt, because it will send DHCP requests forever until it receives a valid DHCP response.)
- Device—Boots using the local device BOOT line configured on it. When device boot is configured, the configured IPXE_TIMEOUT rommon variable is ignored. You can activate device boot as specified below:
  - If BOOTMODE=ipxe-forever, device boot is not activated without user intervention (this is possible only if ENABLE_BREAK=yes).
  - If BOOTMODE=ipxe-timeout, device boot is activated when the specified IPXE_TIMEOUT variable (in seconds) has elapsed.
  - If BOOTMODE=device, device boot is activated. This is the default active mode.
  - Device boot can also be activated through the CLI.

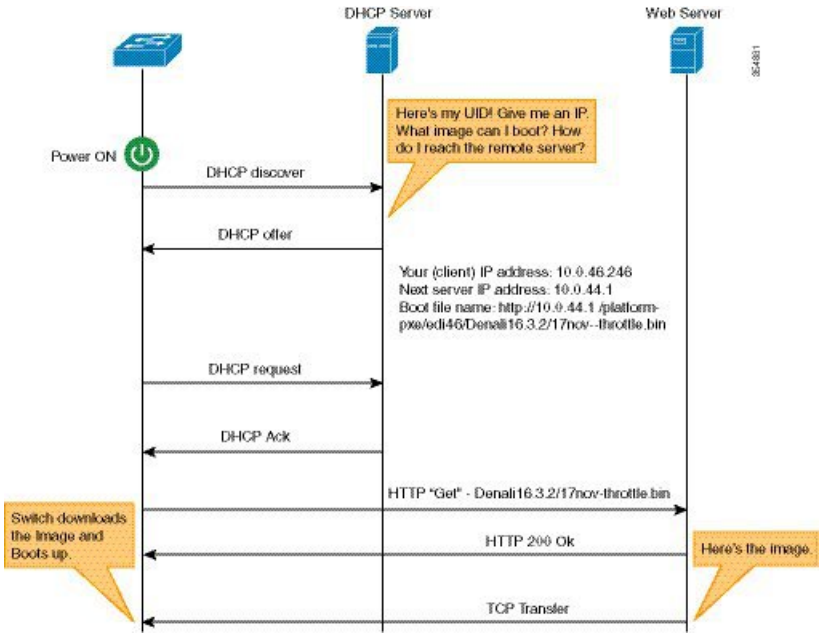
Note Device boot is the default boot mode.

Note Manual boot is another term used in this document. Manual boot is a flag that determines whether to do a rommon reload or not. When the device is in rommon mode, you have to manually issue the **boot** command.

If manual boot is set to YES, the rommon or device prompt is activated. If manual boot is set to NO, the autoboot variable is executed; this means that the value set in the BOOT variable is followed.

The following section describes how an iPXE bootloader works:

Figure 1: iPXE Bootloader Workflow



1. Bootloader sends a DHCP discover message, and when the server replies, the Bootloader sends a DHCP request.
2. The DHCP response includes the IP address and boot file name. The boot file name indicates that the boot image is to be retrieved from a TFTP server (tftp://server/filename), FTP server (ftp://userid:password@server/filename), or an HTTP server (http://server/filename).
3. Bootloader downloads and boots the image from the network source.
4. If no DHCP response is received, the bootloader keeps sending DHCP requests forever or for a specified period of time, based on the boot mode configured. When a timeout occurs, the bootloader reverts to a device-based boot. The device sends DHCP requests forever only if the configured boot mode is **ipxe-forever**. If the **ipxe-timeout** boot mode command is configured, DHCP requests are sent for the specified amount of time, and when the timeout expires, device boot mode is activated.

Note Because the current iPXE implementation works only via the management port (GigabitEthernet0/0), DHCP requests sent through the front panel ports are not supported.

When using a static network configuration to network boot, ROMMON uses the following environment variables (and all of them are required):

- BOOT—URLs separated by semicolon (;) to boot from.
- IP_ADDRESS—Statically assigned IP address of a device.
- DEFAULT_GATEWAY—Default gateway of the device.
- IP_SUBNET_MASK—IPv4 or IPv6 prefix information.

  IPv4—Subnet mask of the device in the format WWW.XXX.YYY.ZZZ eg. 255.255.255.0.

IPv6—Subnet prefix length of the device in the format NNN eg. 64 or 112.

When manual boot is disabled, the bootloader determines whether to execute a device boot or a network boot based on the configured value of the rommon iPXE variable. Irrespective of whether manual boot is enabled or disabled, the bootloader uses the BOOTMODE variable to determine whether to do a device boot or a network boot. Manual boot means that the user has configured the **boot manual switch** command. When manual boot is disabled, and when the device reloads, the boot process starts automatically.

When iPXE is disabled, the contents of the existing BOOT variable are used to determine how to boot the device. The BOOT variable may contain a network-based uniform resource identifier (URI) (for example, http://, ftp://, tftp://), and a network boot is initiated; however DHCP is not used to get the network image path. The static network configuration is taken from the IP_ADDRESS, DEFAULT_GATEWAY, and IP_SUBNET_MASK variables. The BOOT variable may also contain a device filesystem-based path, in which case, a device filesystem-based boot is initiated.

The DHCP server used for booting can identify a device through the Product ID (PID) (available in DHCP Option 60), chassis serial number (available in DHCP option 61), or the MAC address of the device. The **show inventory** and **show switch** commands also display these values on the device.

The following is sample output from the **show inventory** command:

```
Device# show inventory

NAME:“c38xx Stack”, DESCR:“c38xx Stack”
PID:WS-3850-12X-48U-L, VID:V01 ,  SN: F0C1911V01A

NAME:“Switch 1”, DESCR:“WS-C3850-12X48U-L”
PID:WS-C3850-12X48U-L, VID:V01 , SN:F0C1911V01A

NAME:”Switch1 -Power Supply B”, DESCR:“Switch1 -Power Supply B”
PID:PWR-C1-1100WAC, VID:V01, SN:LIT1847146Q
```

The following is sample output from the **show switch** command:

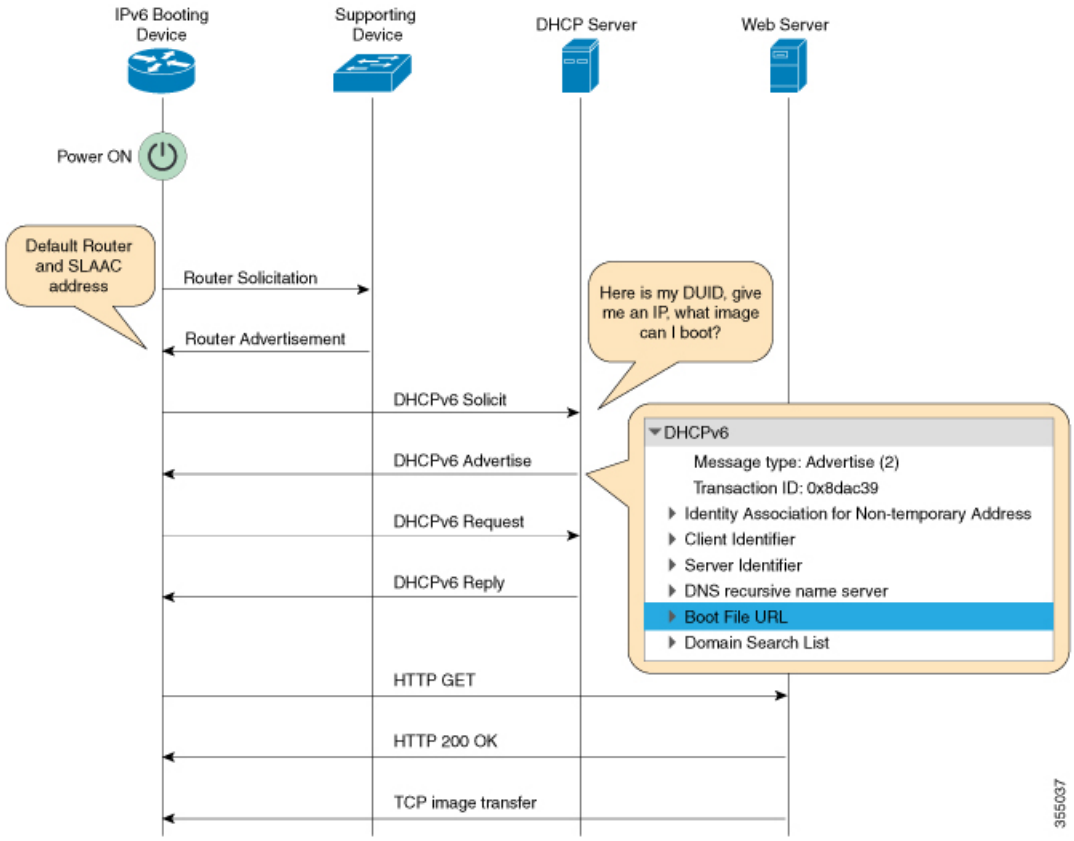
```
Device# show switch

Switch/Stack Mac Address : 046c.9d01.7d80 - Local Mac Address
Mac persistency wait time: Indefinite
                                             H/W   Current
Switch#   Role    Mac Address     Priority Version  State
------------------------------------------------------------------------------------
 1       Member   046c.9d1e.1a00     1              Ready
 2       Standby  046c.9d01.7d80     1              Ready
*3       Active   f8b7.e24e.9a00     1      P2B     Ready
```

The following rommon variables should be configured for iPXE:

- BOOTMODE = ipxe-forever | ipxe-timeout | device
- IPXE_TIMEOUT = seconds

IPv6 iPXE Network Boot

This illustration displays how IPv6 iPXE network boot works on a Cisco device:



The four elements in the above illustration are described below:

- IPv6 Booting Device—The device that is booting through iPXE boot.
- Supporting Device—A Cisco device that is configured with an IPv6 address to generate Router Advertisement (RA) messages.

Note In this illustration, the IPv6 booting device, the supporting device, and the DHCP server are on the same subnet. However; if the supporting device and the DHCP server are on different subnets, then there must be a relay agent in the network.

- DHCP server—Any DHCP server.
- Web server—Any web server.

This section describes the IPv6 iPXE boot process:

1. The device sends a router solicitation Internet Control Message Protocol IPv6 (ICMPv6) type 133 packet to the IPv6 device on the local subnet.
2. The IPv6 device on the local subnet replies with a router advertisement (RA) message, ICMPv6 type 134 packet. The device that sent the router solicitation message, gets the default router and prefix information for Stateless Address AutoConfiguration (SLAAC) address completion from the RA packet.

3. The device sends a DHCPv6 solicit message to the multicast group address of ff02::1:2 for all DHCP agents.

   The following sample displays the fields in a DHCPv6 solicit packet during iPXE boot:

   ```
   DHCPv6
    Message type: Solicit (1)
    Transaction ID: 0x36f5f1
    Client Identifier
    Vendor Class
    Identity Association for Non-Temporary Address
    Option Request
    User Class
    Vendor-specific Information
   ```

   The DHCPv6 solicit message contains the following information:

   - DHCP Unique Identifier (DUID)—Identifies the client. iPXE supports DUID-EN. EN stands for Enterprise Number, and this DUID is based on the vendor-assigned unique identifier.
   - DHCP and DHCPv6 Options

4. If the DHCPv6 server is configured, it responds with a DHCPv6 advertise packet that contains the 128 Bit IPv6 address, the boot file Uniform Resource Identifier (URI), the Domain Name System (DNS) server and domain search list, and the client and server IDs. The client ID contains the DUID of the client (In this illustration, the IPv6 Booting Device), and the Server ID contains the DUID of the DHCPv6 server.
5. The client then sends a DHCPv6 request packet to the multicast group address ff02::1:2, requesting for advertised parameters.
6. The server responds with a unicast DHCPv6 reply to the Link Local (FE80::) IPv6 address of the client. The following sample displays the fields in a DHCPv6 reply packet:

   ```
   DHCPv6
    Message type: Reply (7)
    Transaction ID: 0x790950
    Identity Association for Non-Temporary Address
    Client Identifier
    Server Identifier
    DNS recursive name server
    Boot File URL
    Domain Search List
   ```

7. The device then sends an HTTP GET request to the web server.
8. If the requested image is available at the specified path, the web server responds with an OK for the HTTP GET request.
9. The TCP image transfer copies the image, and the device boots up.

IPv6 Address Assignment in Rommon Mode

The DHCP client uses the following order-of-precedence to decide which IPv6 address to use in rommon mode:

1. DHCP Server-assigned address

2. Stateless Address Auto-Configuration (SLAAC) address
3. Link-local address
4. Site-local address

The device uses the DHCP server-assigned address to boot an image. If the DHCPv6 server fails to assign an address, the device tries to use the SLAAC address. If both the DHCP server-assigned address and the SLAAC address are not available, the device uses the link-local address. However, the remote FTP/HTTP/TFTP servers must be on the same local subnet as that of the device for the image copy to succeed.

If the first three addresses are not available, the device uses the automatically generated site-local address.

Supported ROMMON Variables

The following ROMMON variables are supported in Cisco IOS XE Fuji 16.8.1:

- BAUD: Changes the device console BAUD rate to one of the Cisco standard baud rate; such as 1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200). Any invalid value will be rejected. If the BAUD variable is not set, the default will be 9600. The corresponding CLI command is
- ENABLE_BREAK: Enables a rommon break. The default value is NO.
- MANUAL_BOOT: If manual boot is set to 1, the rommon or device prompt is activated. If manual boot is set to 0, the device is reloaded; but rommon mode is not activated.
- SWITCH_IGNORE_STARTUP_CFG: If the value is 1, it causes the device to ignore the startup configuration. If the value is not set, the value is treated as zero. This is a read-only variable, and can only be modified by IOS.

iPXE-Supported DHCP Options

iPXE boot supports the following DHCPv4 and DHCPv6 options in rommon mode.

Note Catalyst 9000 Series Switches support DHCP Option 60, Option 77, DHCPv6 Options 1, Option 15, and Option 16. DHCP Option 61 is only supported on Catalyst 9300 and 9500 Series Switches.

- DHCP Option 60—Vendor Class Identifier. This option is populated with the value of the ROMMON environment variable MODEL_NUM.
- DHCP Option 61—Client Identifier. This option is populated with the value of theROMMON environment variable SYSTEM_SERIAL_NUM.

Note This option is not supported on Catalyst 9400 Series Switches.

- DHCP Option 77—User Class Option. This option is added to a DHCP Discover packet, and contains the value equal to the string *iPXE*. This option helps to isolate iPXE DHCP clients looking for an image to boot from a DHCP server.

The following is sample DHCPv4 configuration from the ISC DHCP Server that displays the use of Option 77. The *if* condition in this sample implies that if Option 77 exists, and is equal to the string *iPXE*, then advertise the Boot File URI for the image.

```
host Switch2 {
     fixed-address 192.168.1.20 ;
     hardware ethernet CC:D8:C1:85:6F:11 ;
           #user-class = length of string + ASCII code for iPXE
     if exists user-class and option user-class = 04:68:50:58:45 {
         filename "http://192.168.1.146/test-image.bin"
     }
}
```

- DHCPv6 Option 1—Client Identifier Option. This option is populated with the value of the ROMMON environment variable SYSTEM_SERIAL_NUM as specified in RFC 3315. The recommended format for the ROMMON environment variable is MAC_ADDR.
- DHCPv6 Option 15—User Class Option. This option is the IPv6 User Class option in a DHCPv6 solicit message, and is populated with the string, iPXE. The following sample shows Option 15 defined in the ISC DHCP server:

```
option dhcp6.user-class code 15 = string ;
```

The following is a sample DHCP Server configuration that uses the DHCPv6 Option 15:

```
#Client-specific parameters
host switch1 {
     #assigning a fixed IPv6 address
     fixed-address6 2001:DB8::CAFE ;
     #Client DUID in hexadecimal format contains: DUID-type"2" + "EN=9" + "Chassis
serial number"
     host-identifier option dhcp6.client-id       00:02:00:00:00:09:46:4F:43:31:38:33:
31:58:31:41:53;
    #User class 00:04:69:50:58:45 is len 4 + "iPXE"
    if option dhcp6.user-class = 00:04:69:50:58:45 {
       option dhcp6.bootfile-url
"http://[2001:DB8::461/platform-pxe/edi46/test-image.bin";
    }
}
```

- DHCPv6 Option 16—Vendor Class Option. Contains the device product ID (PID). The PID can be determined from the output of the **show inventory** command or from the MODEL_NUM rommon variable. Option 16 is not a default option in the ISC DHCP Server and can be defined as follows:

```
option dhcp6.vendor-class-data code 16 = string;
```

The following sample configuration illustrates the use of DHCPv6 Option 16:

```
# Source: dhcpd6ConfigPD

host host1-ipxe6-auto-host1 {
     fixed-address6 2001:DB8::1234;
     host-identifier option dhcp6.client-id 00:02:00:00:00:09:46:4F:
     43:31:38:33:31:58:31:41:53;
     if option dhcp6.vendor-class-data = 00:00:00:09:00:0E:57:53:2D:
     43:33:38:35:30:2D:32:34:50:2D:4D {
     option dhcp6.bootfile-url
```

```
"http://[2001:DB8::46]/platform-pxe/host1/17jan-polaris.bin";
```

The table below describes the significant fields shown in the display.

Table 4: Sample Output Field Descriptions

Field	Description
dhcp6.client-id	DHCP Unique Identifier (DUID) to identify the client.
dhcp6.user-class	DHCPv6 Option 15, the User Class option
dhcp6.vendor-class-data	DHCPv6 Option 16, the Vendor Class option that contains the switch Product ID (PID).
dhcp6.bootfile-url	DHCPv6 Option 6 to request for the Boot File URI

DHCPv6 Unique Identifiers

There are three types of DHCPv6 Identifiers (DUIDs) defined by RFC 3315; these are:

- DUID-LLT—DUID Link Layer address plus time, this is the link layer address of the network interface connected to the DHCP device plus the time stamp at which it is generated.
- DUID-EN—EN stands for Enterprise Number, this DUID is based on vendor-assigned unique ID.
- DUID-LL—DUID formed using the Link Layer address of any network interface that is permanently connected to the DHCP (client/server) device.

Cisco devices that support this feature use the DUID-EN (DUID Type 2) to identify the DHCP client (that is the device in the DHCPv6 Solicit packet). Catalyst 9000 Series Switches support not only DUID-EN, but also DUID-LL (DUID Type 3). DUID-EN is the preferred type; however, if switches are unable to create it, then DUID-LL is constructed and used.

How to Configure iPXE

Configuring iPXE

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. • **boot ipxe forever** [*switch number*]
   • **boot ipxe timeout** *seconds* [*switch number*]
4. **boot system** {**switch** *switch-number* | **all**} {**flash:** | **ftp:** | **http:** | **usbflash0** | **tftp:**}
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 3	• **boot ipxe forever** [*switch number*]
	• **boot ipxe timeout** *seconds* [*switch number*]
	Example:
	`Device(config)# boot ipxe forever switch 2`
	Example:
	`Device(config)# boot ipxe timeout 30 switch 2`
		Configures the BOOTMODE rommon variable.
		• The **forever** keyword configures the BOOTMODE rommon variable as IPXE-FOREVER.
		• The **timeout** keyword configures the BOOTMODE rommon variable as IPXE-TIMEOUT.

Step 4	**boot system** {**switch** *switch-number* | **all**} {**flash:** | **ftp:** | **http:** | **usbflash0** | **tftp:**}
	Example:
	`Device(config)# boot system switch 1 http://192.0.2.42/image-filename`
	or
	`Device(config)# boot system switch 1 http://[2001:db8::1]/image-filename`
		Boots an image from the specified location.
		• You can either use an IPv4 or an IPv6 address for the remote FTP/HTTP/TFTP servers.
		• You must enter the IPv6 address inside the square brackets (as per RFC 2732); if not the device will not boot.

Step 5	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring Device Boot

You can either use the **no boot ipxe** or the **default boot ipxe** command to configure device boot.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. • **no boot ipxe**
   • **default boot ipxe**
4. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 3	• **no boot ipxe**
	• **default boot ipxe**
	Example:
	`Device(config)# no boot ipxe`
	Example:
	`Device(config)# default boot ipxe`
		Configures device boot. The default bot mode is device boot.
		Enables default configuration on the device.

Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuration Examples for iPXE

Example: iPXE Configuration

The following example shows that iPXE is configured to send DHCP requests forever until the device boots with an image:

```
Device# configure terminal
Device(config)# boot ipxe forever switch 2
Device(config)# end
```

The following example shows how to configure the boot mode to ipxe-timeout. The configured timeout is 200 seconds. If an iPXE boot failure occurs after the configured timeout expires, the configured device boot is activated. In this example, the configured device boot is http://[2001:db8::1]/image-filename.

```
Device# configure terminal
Device(config)# boot ipxe timeout 200 switch 2
Device(config)# boot system http://[2001:db8::1]/image-filename
Device(config)# end
```

Sample iPXE Boot Logs

The following are sample boot logs from a device in rommon mode. Here, manual boot using the **ipxe-timeout** command is configured:

```
switch: boot

pxemode:(ipxe-timeout) 60s timeout
00267.887 ipxe_get_booturl: Get URL from DHCP; timeout 60s
00267.953 ipxe_get_booturl: trying DHCPv6 (#1) for 10s
IPv4:
           ip addr 192.168.1.246
           netmask 255.255.255.0
           gateway 192.168.1.46
IPv6:
link-local addr fe80::ced8:c1ff:fe85:6f00
site-local addr fec0::ced8:c1ff:fe85:6f00
      DHCP addr 2001:db8::cafe
    router addr fe80::f29e:63ff:fe42:4756
     SLAAC addr 2001:db8::ced8:c1ff:fe85:6f00 /64
Common:
        macaddr cc:d8:c1:85:6f:00
            dns 2001:db8::46
       bootfile
http://[2001:DB8::461/platform-pxe/edi46/17jan-dev.bin--13103--2017-Feb28--13-54-50
         domain cisco.com
00269.321 ipxe_get_booturl: got URL
(http://[2001:DB8::461/platform-pxe/edi46/17jan-dev.bin--13103--2017-Feb-28--13-54-50)
Reading full image into memory …….…….…….…….…….…….…….…….…….…….…….…….…..
Bundle Image
-----------------------------------------------
Kernel Address    : 0x5377a7e4
Kernel Size       : 0x365e3c/3563068
Initramfs Address : 0x53ae0620
Initramfs Size    : 0x13a76f0/20608752
Compression Format: mzip
```

Sample DHCPv6 Server Configuration for iPXE

The following is a sample DHCPv6 server configuration taken from an Internet Systems Consortium (ISC) DHCP Server for reference. The lines preceded by the character #, are comments that explain the configuration that follows.

```
Default-least-time 600;
max-lease-time-7200;
log-facility local7;

#Global configuration
#domain search list
option dhcp6.domain-search "cisco.com" ;
#User-defined options:new-name code new-code = definition ;
option dhcp6.user-class code 15 = string ;
option dhcp6.vendor-class-data code 16 = string;
```

```
subnet6 2001:db8::/64 {
 #subnet range for clients requiring an address
 range6 2001:db8:0000:0000::/64;

#DNS server options
option dhcp6.name-servers 2001:db8::46;

}
#Client-specific parameters
host switch1 {
    #assigning a fixed IPv6 address
    fixed-address6 2001:DB8::CAFE ;
    #Client DUID in hexadecimal that contains: DUID-type "2" + "EN=9" + "Chassis serial
number"
    host-identifier option dhcp6.client-id 00:02:00:00:00:09:46:4F:43:31:38:33:
31:58:31:41:53;
   option dhcp6.bootfile-url "http://[2001:DB8::461/platform-pxe/edi46/test-image.bin";
}
```

For more information on DHCP server commands, see the *ISC DHCP Server* website.

In this sample configuration, the dhcp6.client-id option identifies the switch, and it is followed by the Enterprise Client DUID. The client DUID can be broken down for understanding as 00:02 + 00:00:00:09 + chassis serial number in hexadecimal format, where 2 refers to the Enterprise Client DUID Type, 9 refers to the reserved code for Cisco's Enterprise DUID, followed by the ASCII code for the Chassis serial number in hexadecimal format. The chassis serial number for the switch in this sample is FOC1831X1AS.

The Boot File URI is advertised to the switch only using the specified DUID.

The DHCPv6 Vendor Class Option 16 can also be used to identify the switch on the DHCP Server. To define Option 16 as a user-defined option, configure the following:

```
option dhcp6.vendor-class-data code 16 = string;
```

The following is a sample DHCP server configuration that identifies the switch based on the DHCPv6 Vendor Class Option 16 that is formed by using the switch Product ID:

```
# Source: dhcp6ConfigPID

host edi-46-ipxe6-auto-edi46 {
    fixed-address6 2001:DB8::1234;
    host-identifier option dhcp6.client-id 00:02:00:00:00:09:
    46:4F:43:31:38:33:31:58:31:58:31:41:53;
    if option dhcp6.vendor-class-data = 00:00:00:09:00:0E:57:
    53:2D:43:33:38:35:30:2D:32:34:50:2D:4C {
       option dhcp6.bootfile-url "http://[2001:DB8::461/platform-pxe/edi46/17jan-dev.bin";
    }
}
```

In this sample configuration, the dhcp6.vendor-class-data option refers to the DHCPv6 Option 16. In the dhcp6.vendor-class-data, 00:00:00:09 is Cisco's Enterprise DUID, 0E is the length of the PID, and the rest is the PID in hexadecimal format. The PID can also be found from the output of the **show inventory** command or from the CFG_MODEL_NUM rommon variable. The PID used in this sample configuration is WS-C3850-24P-L.

DHCPv6 options and DUIDs in the server configuration must be specified in the hexadecimal format, as per the ISC DHCP server guidelines.

Troubleshooting Tips for iPXE

This section provides troubleshooting tips.

- When iPXE boot is enabled on power up, the device first attempts to send a DHCPv6 Solicit message, followed by a DHCPv4 Discover message. If boot mode is **ipxe-forever** the device keeps iterating between the two forever.
- If the boot-mode is iPXE timeout, the device first sends a DHCPv6 Solicit message, and then a DHCPv4 Discover message, and the device falls back to device boot after the timeout expires.
- To interrupt iPXE boot, send a serial break to the console.

  When using a UNIX telnet client, type CTRL-] and then send break. When you are using a different TELNET client, or you are directly attached to a serial port, sending a break may be triggered by a different keystroke or command.
- If the DHCP server responds with an image, but the DNS server cannot resolve the hostname, enable DNS debugs.

Note We recommend the use of ISC DHCP server. This feature has not been verified on IOS DHCP.

- To test the HTTP server connectivity, use HTTP copy to copy a small sample file from your HTTP server to your device. For example, at the rommon prompt, enter **copy http://192.168.1.1/test null:** (the flash is normally locked and you need to use the null device for testing) or **http://[2001:db8::99]/test**.
- When manual boot is enabled, and boot mode is ipxe-timeout, the device will not automatically boot on power up. Issue the **boot** command in rommon mode. To automate the boot process on power up, disable manual boot.
- Use the **net6-show** command to display the current IPv6 parameters, including IPv6 addresses and the default router in rommon mode

Note On Catalyst 9000 Series Switches, use the **net-show** show command.

- Use the **net-dhcp** or the **net6-dhcp** commands based on your configuration, The **net-dhcp** command is a test command for DHCPv4 and the **net6-dhcp** command is for DHCPv6.

Note On Catalyst 9000 Series Switches, use the **net-dhcp -6** command for DHCPv6.

- Use the **dig** command to resolve names.

Note On Catalyst 9000 Series Switches, use the **dns-lookup** commmand to resolve names.

- Enable HTTP debug logs to view the HTTP response code from the web server.
- If Stateless Address Auto-Configuration (SLAAC) addresses are not generated, there is no router that is providing IPv6 RA messages. iPXE boot for IPv6 can still work but only with link or site-local addresses.

Additional References for iPXE

Related Documents

Related Topic	Document Title
Programmability commands	Programmability Command Reference, Cisco IOS XE Everest 16.6.1

Standards and RFCs

Standard/RFC	Title
RFC 3315	*Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*
RFC 3986	*Uniform Resource Identifier (URI): Generic Syntax*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for iPXE

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

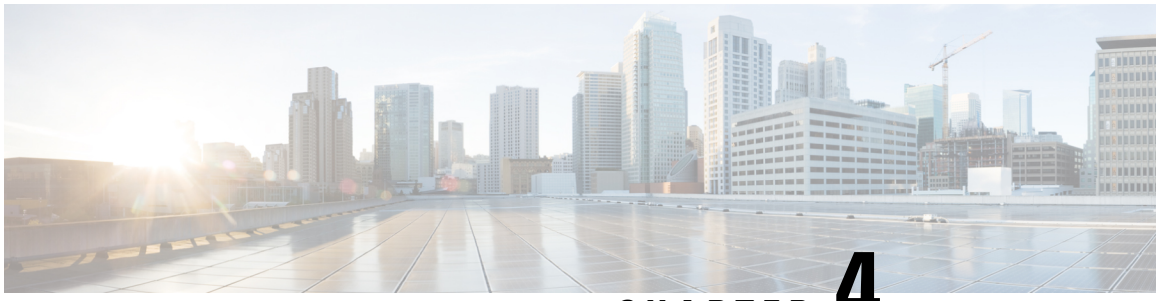
Table 5: Feature Information for iPXE

Feature Name	Release	Feature Information
iPXE	Cisco IOS XE Denali 16.5.1a	Network Bootloaders support booting from an IPv4/IPv6 device-based or network-based source. A network boot source must be detected automatically by using an iPXE-like solution. This feature was implemented on the following platforms: • Catalyst 3650 Series Switches • Catalyst 3850 Series Switches
	Cisco IOS XE Denali 16.6.1	In Cisco IOS XE Denali 16.6.1, this feature was implemented on the following platforms: • Catalyst 9300 Series Switches • Catalyst 9500 Series Switches
	Cisco IOS XE Everest 16.6.2	In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.
	Cisco IOS XE Fuji 16.9.2	In Cisco IOS XE Fuji 16.9.2, this feature was implemented on the following platforms. • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300L SKUs
	Cisco IOS XE Gibraltar 16.11.1	In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on Cisco Catalyst 9600 Series Switches.
IPXE IPv6 Support	Cisco IOS XE 16.8.1a	IPXE supports the IPv6 protocol. This feature was implemented on the following platforms: • Catalyst 9300 Series Switches • Catalyst 9400 Series Switches • Catalyst 9500 Series Switches

PART II

Shells and Scripting

- Guest Shell, on page 103
- Python API, on page 133
- EEM Python Module, on page 143



CHAPTER 4

Guest Shell

Guestshell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python for automated control and management of Cisco devices. It also includes the automated provisioning (Day zero) of systems. This container shell provides a secure environment, decoupled from the host device, in which users can install scripts or software packages and run them.

This module describes Guest Shell and how to enable it.

- Restrictions for Guest Shell, on page 103
- Information About the Guest Shell, on page 104
- How to Enable the Guest Shell, on page 114
- Configuration Examples for the Guest Shell, on page 123
- Additional References for Guest Shell, on page 129
- Feature Information for Guest Shell, on page 129

Restrictions for Guest Shell

- Guest Shell is not supported on Cisco Catalyst 9200L SKUs.
- NETCONF sessions cannot be established on the standby Route Processor (RP).
- Python scripts fail when running commands like the **show tech-support wireless** command, when the scale is set to 2000Aps, and clients are set to 10000.

  The output of commands like **show tech-support wireless** is huge and can cause memory exhaustion inside the Guest Shell. When using commands with huge output, redirect the output to a file. The IOS CLI can write the output to a file in the */bootflash/guest-share* directory, and it can be accessed from the Guest Shell.
- Cisco Catalyst 9200CX Series Switches do not support the Management interface, AppGigabitEthernet interface, or VirtualPortGroup interface. Applications or scripts running in the Guest Shell will not be able to communicate with the external network.
- In Cisco IOS XE 17.16.1 and later releases, Catalyst 9200CX Series Switches will use VirtualPortGroup interface with Network Address Translation (NAT) to communicate with the external network. This communication is only supported for day zero ZTP running in Guest Shell.

Information About the Guest Shell

Guest Shell Overview

The Guest Shell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python, for automated control and management of Cisco devices. Using the Guest Shell, you can also install, update, and operate third-party Linux applications. The Guest Shell is bundled with the system image and can be installed using the **guestshell enable** Cisco IOS command.

The Guest Shell environment is intended for tools, Linux utilities, and manageability rather than networking.

Guest Shell shares the kernel with the host (Cisco switches and routers) system. Users can access the Linux shell of Guest Shell and update scripts and software packages in the container root filesystem. However, users within the Guest Shell cannot modify the host file system and processes.

Guest Shell container is managed using IOx. IOx is Cisco's Application Hosting Infrastructure for Cisco IOS XE devices. IOx enables hosting of applications and services developed by Cisco, partners, and third-party developers in network edge devices, seamlessly across diverse and disparate hardware platforms.

Guest Shell Software Requirements

The Guest Shell container allows users to run their scripts and apps on the system. The Guest Shell container on Intel x86 platforms will be a Linux container (LXC) with a CentOS 8.0 minimal rootfs. You can install other Python libraries such as, Python Version 3.0 during runtime using the Yum utility in CentOS 8.0. You can also install or update python packages using PIP.

Table 6: Guest Shell Software Requirements

	Guest Shell (LXC Container)
Operating System	Cisco IOS XE
Platform	All supported Cisco IOS XE platforms
Guest Shell Environment	• CentOS 7 supported in Cisco IOS XE Amsterdam 17.2.1 and previous releases. • CentOS 8 supported in Cisco IOS XE Amsterdam 17.3.1 and later releases. **Note** CentOS supports only Python 3.6.
Python 2.7	Supported till Cisco IOS XE Amsterdam 17.3.1

	Guest Shell (LXC Container)
Python 3.6	Supported in Cisco IOS XE Amsterdam 17.1.1 and later releases. In Cisco IOS XE Amsterdam 17.1.1 and Cisco IOS XE Amsterdam 17.2.1, Python V2 is the default. However, in Cisco IOS XE Amsterdam 17.3.1 and later releases, Python V3 is the default. **Note** Cisco Catalyst 9200 Series Switches support Python version 3 in Cisco IOS XE Amsterdam 17.3.1 and later releases.
Pre-installed Custom Python Libraries	• Cisco Embedded Event Manager • Cisco IOS XE CLIs • NCCLIENT library for the NETCONF API
Supported Rootfs	SSH, Yum install, and Python PIP install
GNU C Compiler	Not supported
RPM Install	Supported
Architecture	x86 and ARM

Guest Shell Security

Cisco provides security to ensure that users or apps in the Guest Shell do not compromise the host system. Guest Shell is isolated from the host kernel, and it runs as an unprivileged container.

Hardware Requirements for the Guest Shell

This section provides information about the hardware requirements for supported platforms which have variable memory configurations.

Table 7: Guest Shell Resource Requirements

Platforms	Minimum Memory
Cisco 1000 Series Integrated Services Routers	4 GB
Cisco Cloud Services Router 1000V Series	4 GB
Cisco ISR 4000 Series Integrated Services Routers	8 GB DRAM (In Cisco IOS XE Fuji 16.8.1 and previous releases.) 4GB DRAM (In Cisco IOS XE Fuji 16.8.1 and later releases.)

All other platforms are shipped with sufficient resources to support Guest Shell.

Note Virtual-service installed applications and the Guest Shell container cannot co-exist.

Guest Shell Storage Requirements

Cisco Catalyst 9300 Series Switches and Cisco Catalyst 9500 Series Switches require 1100 MB free hard disk space for Guest Shell to install successfully.

On Cisco 4000 Series Integrated Services Routers, the Guest Shell is installed on the Network Interface Module (NIM)-Solid State Drive (SSD) (hard disk), if available. If the hard disk drive is available, there is no option to select bootflash to install Guest Shell. Cisco 4000 Series Integrated Services Routers require 1100 MB free hard disk (NIM-SSD) space for Guest Shell to install successfully.

For Cisco 4000 Series Integrated Services Routers and Cisco ASR 1000 Series Aggregation Services Routers (when an optional hard disk has been added to that router) you can only do resource resizing if you have installed the Guest Shell on the hard disk and inserted the hard disk into the router.

Note A Guest Shell installed via bootflash does not allow you to do resource resizing using application hosting configuration commands.

During Guest Shell installation, if enough hard disk space is not available, an error message is displayed.

The following is a sample error message on an Cisco ISR 4000 Series Integrated Services Router

```
% Error:guestshell_setup.sh returned error:255, message:
Not enough storage for installing guestshell. Need 1100 MB free space.
```

Bootflash or hard disk space can be used to store additional data by Guest Shell. On Cisco 4000 Series Integrated Services Routers, Guest Shell has 800 MB of storage space available. Because Guest Shell accesses the bootflash, it can use the entire space available.

Table 8: Resources Available to Guest Shell and Guest Shell Lite

Resource	Default	Minimum/Maximum
CPU	1% **Note** 1% is not standard; 800 CPU units/ total system CPU units.	1/100%
Memory	256 MB 512 MB (Cisco Cloud Services Router 1000V Series)	256/256 MB 512/512 MB (Cisco Cloud Services Router 1000V Series)

Enabling and Running the Guest Shell

The **guestshell enable** command installs Guest Shell. This command is also used to reactivate Guest Shell, if it is disabled.

When Guest Shell is enabled and the system is reloaded, Guest Shell remains enabled.

Note IOx must be configured before the **guestshell enable** command is used.

The **guestshell run bash** command opens the Guest Shell bash prompt. Guest Shell must already be enabled for this command to work.

Note If the following message is displayed on the console, it means that IOx is not enabled; check the output of the **show iox-service** command to view the status of IOx.

```
The process for the command is not responding or is otherwise unavailable
```

For more information on how to enable Guest Shell, see the "Configuring the AppGigabitEthernet Interface for Guest Shell" and "Enabling Guest Shell on the Management Interface" sections.

Disabling and Destroying the Guest Shell

The **guestshell disable** command shuts down and disables Guest Shell. When Guest Shell is disabled and the system is reloaded, Guest Shell remains disabled.

The **guestshell destroy** command removes the rootfs from the flash filesystem. All files, data, installed Linux applications and custom Python tools and utilities are deleted, and are not recoverable.

Accessing Guest Shell on a Device

Network administrators can use Cisco IOS commands to manage files and utilities in the Guest Shell.

During the Guest Shell installation, SSH access is setup with a key-based authentication. The access to the Guest Shell is restricted to the user with the highest privilege (15) in Cisco IOS. This user is granted access into the Linux container as the *guestshell* Linux user, who is a sudoer, and can perform all root operations. Commands executed through the Guest Shell are executed with the same privilege that a user has when logged into the Cisco IOS terminal.

At the Guest Shell prompt, you can execute standard Linux commands.

Accessing Guest Shell Through the Management Port

By default, Guest Shell allows applications to access the management network. Users cannot change the management VRF networking configurations from inside the Guest Shell.

Note For platforms without a management port, a VirtualPortGroup can be associated with Guest Shell in the Cisco IOS configuration. For more information, see the *Sample VirtualPortGroup Configuration* section.

Cisco Catalyst 9200 Series Switches, Cisco Catalyst 9300 Series Switches, and Cisco Catalyst 9400 Series Switches support the AppGigabitEthernet interface and management interface (mgmt-if) to access Guest Shell.

Cisco Catalyst 9500 and 9500 High-Performance Series Switches and Cisco Catalyst 9600 Series Switches do not support AppGigabitEthernet interfaces.

Note Cisco Catalyst 9200L SKUs do not support Guest Shell.

Day Zero Guest Shell Provisioning Using Front-Panel Port or Fiber Uplink

On Day Zero, when the device has no management connectivity, and the only connectivity is either through the front-panel port or fibre uplink port, Guest Shell is internally configured to use the available port. The AppGigabitEthernet interface connects Guest Shell to the server.

When Guest Shell is connected to the server, the device downloads the configuration script, and configures the device. This configuration also includes downloading, setting, and starting of the virtual machine (VM). After the day zero configuration is complete, based on your configuration the system may reboot. Ensure that the system boots with only the user-specific configuration.

Guest Shell Connectivity Using the USB Port

The device uses a serial adapter to connect to multiple other devices. This serial adapter is connected through the USB port that is present on the front panel of the device.

The VM controls the serial adapter, and if there are any changes to the connected devices that are attached to the USB interface while VM is running, the VM is notified.

Stacking with Guest Shell

Guest Shell supports 1+1 high availability. 1+1 high availability is when one device is designated as the active, and the other is designated as the standby. N+1 high availability is not supported.

When Guest Shell is installed, a *guest-share* directory is automatically created in the flash file system. This directory is synchronized across stack members. Any files stored in the *guest-share* folder will be maintained when the active device goes down and the standby takes over. To preserve up to 50 MB of data during high availability switchover, ensure that data is placed in this directory. If the size of the *guest-share* folder is more than 50 MB, it will not be synched to stack members.

During a high availability switchover, the new active device creates its own Guest Shell installation and restores Guest Shell to the synchronized state; the old file system is not maintained. Guest Shell state is internally synchronized across all stack members.

Cisco IOx Overview

Cisco IOx (IOs + linuX) is an end-to-end application framework that provides application-hosting capabilities for different application types on Cisco network platforms. The Cisco Guest Shell, a special container deployment, is one such application, that is useful in system deployment.

Cisco IOx facilitates the life cycle management of applications and data exchange by providing a set of services that helps developers to package prebuilt applications, and host them on a target device. IOx life cycle management includes distribution, deployment, hosting, starting, stopping (management), and monitoring of applications and data. IOx services also include application distribution and management tools that help users discover and deploy applications to the IOx framework.

Cisco IOx application hosting provides the following features:

- Hides network heterogeneity.
- Cisco IOx application programming interfaces (APIs) remotely manage the life cycle of applications hosted on a device.
- Centralized application life cycle management.
- Cloud-based developer experience.

IOx Tracing and Logging Overview

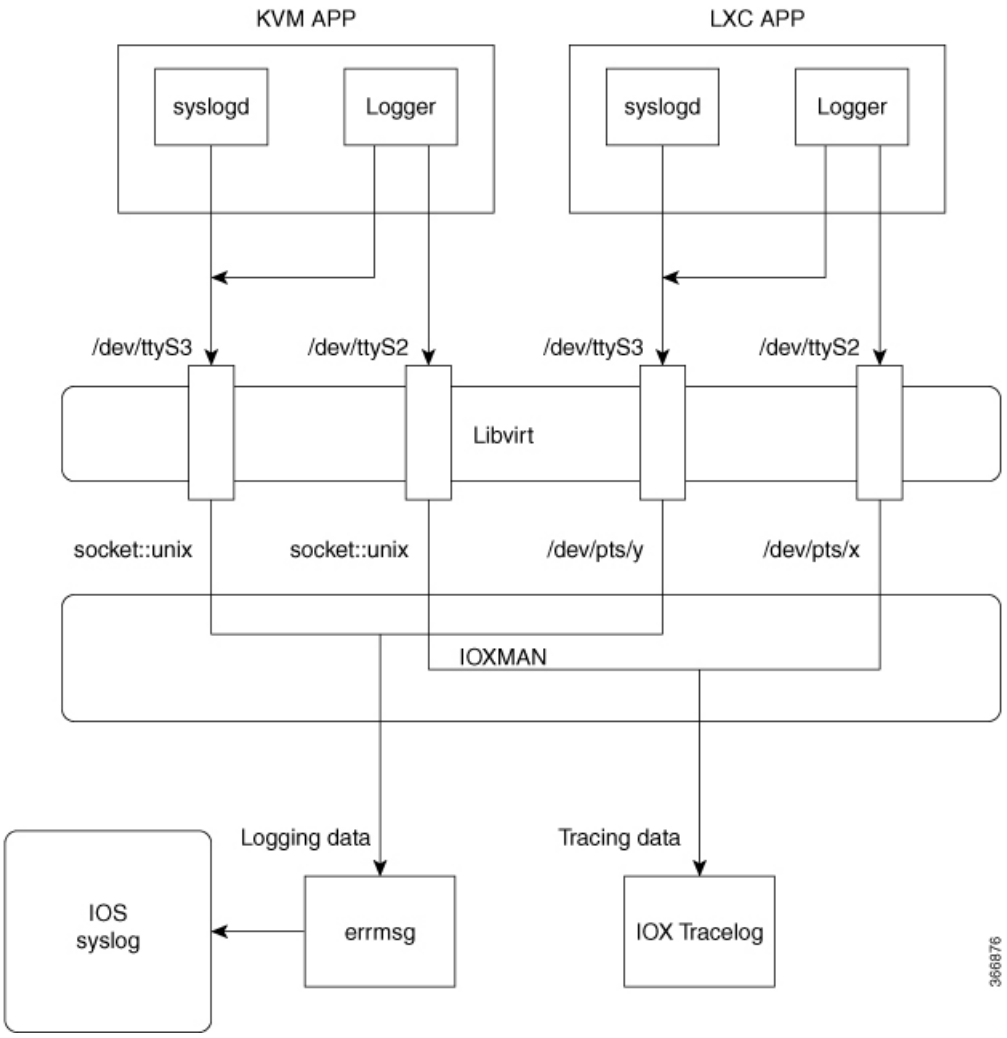
IOx tracing and logging feature allows guest application to run separately on the host device that can help reporting the logging and tracing of the data to the host. The tracing data is saved into IOx tracelog, and the logging data is saved into the Cisco IOS syslog on the host device.

You can redirect the tracing data to the appropriate storage device on the host device which can help in debugging of guest application.

IOXMAN Structure

Each guest application, a system LXC or a KVM instance is configured with its own syslogd and logfiles stored within a visible file system and are not accessible to the host device. To support logging data to the Cisco IOS syslog and tracing data to IOx tracelog on the host, two serial devices, **/dev/ttyS2** and **/dev/ttyS3**, are designated on the guest application for delivering data to the host as shown in the following figure.

Figure 2: IOXMAN Structure



IOXMAN is a process to establish the tracing infrastructure to provide logging or tracing services for the guest application, except Libvirt that emulates serial devices. IOXMAN is based on the lifecycle of the guest application to enable and disable tracing service, to send logging data to the Cisco IOS syslog, to save tracing data to IOx tracelog, and to maintain IOx tracelog for each guest application.

NETCONF Access from Guest Shell

NETCONF-YANG can be accessed from within the Guest Shell, so that users can run Python scripts and invoke Cisco-custom package CLIs using the NETCONF protocol.

The Guest Shell application will establish an SSH connection without a passwordless SSH connection to the localhost and NETCONF port, by using *guestshell* as the username. This username does not correspond to any actual users configured on the device. Even if the device does have a *guestshell* user configured, there is no connection to this passwordless access. Only users with PRIV15 privilege level can access NETCONF from within the Guest Shell.

Authentication and authorization is not bypassed; instead, authentication and authorization happens while granting access to Guest Shell. Only users with the maximum privilege are granted this access.

Users can access the NETCONF service from Guest Shell without opening any external ports. Before connecting to the NETCONF-YANG server on the device, you must run the initializing commands in Guest Shell. These commands are:

```
iosp_client -f netconf_enable guestshell <port-number> and
iosp_client -f netconf_enable_passwordless guestshell <username>
```

The **iosp_client -f netconf_enable guestshell** *port-number* command configures the **netconf-yang ssh local-vrf guestshell** command, and blocks connections until NETCONF-YANG is up and running.

The **iosp_client -f netconf_enable_passwordless guestshell <username>** command creates the SSH keys required for Guest Shell access.

To remove the NETCONF-YANG access from Guest Shell, use the following commands:

```
iosp_client -f netconf_disable guestshell and
iosp_client -f netconf_disable_passwordless guestshell  <username>
```

The **iosp_client -f netconf_disable guestshell** command disables access to NETCONF from within the Guest Shell; however, the NETCONF-YANG configuration will still exist. To shut down NETCONF-YANG, use the **no netconf-yang** command.

The **iosp_client -f netconf_disable_passwordless guestshell** *username* command removes the SSH keys for the specified user. The user will not be able to access NETCONF without a password; however, the user would still be able to connect by using a password.

The *netconf_enable_guestshell* python API runs a combination of the iosp_client functions, *iosp_client -f netconf_enable guestshell 830* and *iosp_client -f netconf_enable_passwordless guestshell guestshell*. This API hides the *unfamiliar-to-user iosp_client* function. When this function is called, it does not return a response until all commands are completed. Unless the function returns an error, you can be sure that NETCONF is running, and the passwordless setup is complete; and you can start creating connections.

Logging and Tracing System Flow

The following sections describes how the IOx logging and tracing works:

LXC Logging

1. Guest OS enables **/dev/ttyS2** on the guest application.
2. Guest application writes data to **/dev/ttyS2**.
3. Libvirt emulates **/dev/ttyS2** to **/dev/pts/x** on the host.
4. IOXMAN gets the emulated serial device, **/dev/pts/x** from the XML file.
5. IOXMAN listens and reads available data from **/dev/pts/x**, sets the severity for the message, filters, parses and queues the message.
6. Start timer to send the message to **/dev/log** device on the host using **errmsg**.
7. Data is saved to the Cisco IOS syslog.

KVM Logging

1. Guest OS enables **/dev/ttyS2** on the guest application.
2. Guest application writes data to **/dev/ttyS2**.
3. Libvirt emulates **/dev/ttyS2** to **/dev/pts/x** on the host.
4. IOXMAN gets the emulated TCP path from the XML file.
5. IOXMAN opens an UNIX socket, and connects to the remote socket.
6. IOXMAN reads available data from the socket, sets the severity for the message, filters, parses, and queues the message.
7. Starts the timer to send the message to **/dev/log** device on the host using **errmsg**.
8. Data is saved to the Cisco IOS syslog.

LXC Tracing

1. Guest OS enables **/dev/ttyS3** on the guest application.
2. Configures **syslogd** to copy message to **/dev/ttyS3**.
3. Guest application writes data to **/dev/ttyS3**.
4. Libvirt emulates **/dev/ttyS3** to **/dev/pts/y** on the host.
5. IOXMAN gets the emulated serial device, **/dev/pts/y** from the XML file.
6. IOXMAN listens and reads available data from **/dev/pts/y**, filters, parses, and saves the message to IOx tracelog.
7. If IOx tracelog is full, IOXMAN rotates the tracelog file to **/bootflash/tracelogs**.

KVM Tracing

1. Guest OS enables **/dev/ttyS3** on the guest application.
2. Configures syslog to copy the message to **/dev/ttyS3**.
3. Guest application writes data to **/dev/ttyS3**.
4. Libvirt emulates **/dev/ttyS3** to TCP path on the host.
5. IOXMAN gets the emulated TCP path from the XML file.
6. IOXMAN opens an UNIX socket, and connects to the remote socket.
7. IOXMAN reads the available data from the socket, sets the severity level for the message, filters, parses, and saves the message to IOx tracelog.
8. If IOx tracelog is full, IOXMAN rotates the tracelog file to **/bootflash/tracelogs**.

Logging and Tracing of Messages

The following sections explains the logging and tracing of messages in to the Cisco IOS syslog.

Logging Messages in Cisco IOS Syslog

For any logging messages received from a guest application, IOXMAN sets the severity of the message to NOTICE by default, before sending it to the Cisco IOS syslog. When a message is received by IOSd, it is displayed on the console and saved on the syslog in the following message format:

***Apr 7 00:48:21.911: %IM-5-IOX_INST_NOTICE:ioxman: IOX SERVICE guestshell LOG: Guestshell test**

To comply with the Cisco IOS syslog, the IOXMAN does support severity levels for logging messages. To report logging messages with severity, a guest application must append a header to the front of the message.

```
[a123b234,version,severity]

a123b234 is magic number.
Version:        severity support version.  Current version is 1.
Severity:       CRIT is 2
                ERR is 3
                WARN is 4
                NOTICE is 5
                INFO is 6
                DEBUG is 7
```

The following is an example of a message log:

```
echo "[a123b234,1,2]Guestshell failed" > /dev/ttyS2
```

Perform the following steps to report logging data from a guest application to the Cisco IOS syslog:

1. If you are using C programming, use **write()** to send logging data to the host.

   ```
   #define SYSLOG_TEST       "syslog test"
        int fd;
        fd = open("/dev/ttyS2", O_WRONLY);
        write(fd, SYSLOG_TEST, strlen(SYSLOG_TEST));
        close(fd);
   ```

2. If you are using a Shell console, use **echo** to send logging data to the host.

   ```
   echo "syslog test" > /dev/ttyS2
   ```

Tracing Message to IOx Tracelog

Perform the following steps to report tracing messages from a guest application to IOx tracelog:

1. If you are using C programming, use **write()** to send tracing message to the host.

   ```
   #define SYSLOG_TEST       "tracelog test"
        int fd;
        fd = open("/dev/ttyS3", O_WRONLY);
        write(fd, SYSLOG_TEST, strlen(SYSLOG_TEST));
        close(fd);
   ```

2. If you are using C programming, use **syslog()** to send tracing message to the host.

   ```
   #define SYSLOG_TEST       "tracelog test"

   syslog(LOG_INFO, "%s\n", SYSLOG_TEST);
   ```

3. If you are using a Shell console, use **echo** to send tracing data to the host.

```
echo "tracelog test" > /dev/ttyS3
   or
logger "tracelog test"
```

How to Enable the Guest Shell

Managing IOx

Before you begin

IOx takes upto two minutes to start. CAF, IOXman, and Libvirtd services must be running to enable Guest Shell successfully.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **iox**
4. **exit**
5. **show iox-service**
6. **show app-hosting list**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**iox** **Example:** `Device(config)# iox`	Configures IOx services.
Step 4	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 5	**show iox-service** **Example:** `Device# show iox-service`	Displays the status of the IOx service
Step 6	**show app-hosting list** **Example:** `Device# show app-hosting list`	Displays the list of app-hosting services enabled on the device.

Example

The following is sample output from the **show iox-service** command:

```
Device# show iox-service

IOx Infrastructure Summary:
---------------------------
IOx service (CAF) 1.10.0.0 : Running
IOx service (HA)           : Running
IOx service (IOxman)       : Running
IOx service (Sec storage) : Not Running
Libvirtd   1.3.4           : Running
Dockerd    18.03.0         : Running
Application DB Sync Info  : Available
Sync Status               : Disabled
```

The following is sample output from the **show app-hosting list** command:

```
Device# show app-hosting list

App id                           State
------------------------------------------------------
guestshell                       RUNNING
```

Managing the Guest Shell

Note VirtualPortGroups are supported only on routing platforms.

Before you begin

IOx must be configured, and running for Guest Shell access to work. If IOx is not configured, the following message is displayed on the device console.

```
iox feature is not enabled.
```

Removing IOx removes access to the Guest Shell, but the rootfs remains unaffected.

An application or management interface must also be configured to enable and operate Guest Shell. See "Configuring the AppGigabitEthernet Interface for Guest Shell" and "Enabling Guest Shell on the Management Interface" sections for more information on enabling an interface for Guest Shell.

SUMMARY STEPS

1. **enable**
2. **guestshell enable**
3. **guestshell run** *linux-executable*
4. **guestshell run bash**
5. **guestshell disable**
6. **guestshell destroy**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**guestshell enable** **Example:** `Device# guestshell enable`	Enables the Guest Shell service. **Note** • The **guestshell enable** command uses the management virtual routing and forwarding (VRF) instance for networking. • When using VirtualPortGroups (VPGs) for front panel networking, the VPG must be configured first. • The guest IP address and the gateway IP address must be in the same subnet.

Step 3	**guestshell run** *linux-executable*
	Example:
	`Device# guestshell run python`
	or
	`Device# guestshell run python3`
		Executes or runs a Linux program in the Guest Shell.
		Note
		In Cisco IOS XE Amsterdam 17.3.1 and later releases, only Python version 3 is supported.

Step 4	**guestshell run bash** **Example:** `Device# guestshell run bash`	Starts a Bash shell to access the Guest Shell.
Step 5	**guestshell disable** **Example:**	Disables the Guest Shell service.

	Command or Action	Purpose
	`Device# guestshell disable`	
Step 6	**guestshell destroy** **Example:** `Device# guestshell destroy`	Deactivates and uninstalls the Guest Shell service.

Managing the Guest Shell Using Application Hosting

Note This section is applicable to Cisco routing platforms. VirtualPortGroups are not supported on Cisco Catalyst Switching platforms.

IOx must be configured, and running for Guest Shell access to work. If IOx is not configured, the following message is displayed on the device console.

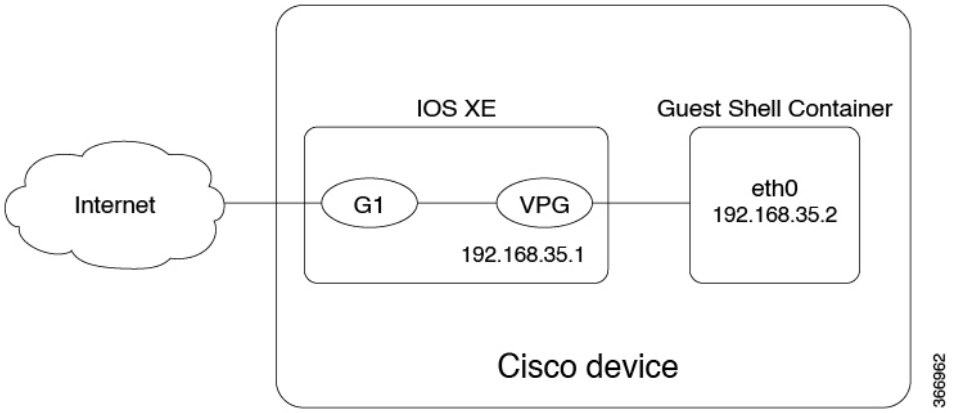
```
iox feature is not enabled.
```

Removing IOx removes access to the Guest Shell, but the rootfs remains unaffected.

Note Use this procedure (Managing the Guest Shell Using Application Hosting) to enable the Guest Shell in Cisco IOS XE Fuji 16.7.1 and later releases. For Cisco IOS XE Everest 16.6.x and previous releases, use the procedure in Managing the Guest Shell, on page 115.

```
Device(config)# interface GigabitEthernet1
Device(config-if)# ip address dhcp
Device(config-if)# ip nat outside
Device(config-if)# exit

Device(config-if)# interface VirtualPortGroup0
Device(config-if)# ip address 192.168.35.1 255.255.255.0
Device(config-if)# ip nat inside
Device(config-if)# exit

Device(config)# ip nat inside source list GS_NAT_ACL interface GigabitEthernet1 overload
Device(config)# ip access-list standard GS_NAT_ACL
Device(config)# permit 192.168.0.0 0.0.255.255

Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 0
Device(config-app-hosting-gateway)# guest-ipaddress 192.168.35.2 netmask 255.255.255.0
Device(config-app-hosting-gateway)# exit
Device(config-app-hosting)# app-default-gateway 192.168.35.1 guest-interface 0
Device(config-app-hosting)# end

Device# guestshell enable
Device# guestshell run python
```

Figure 3: Managing the Guest Shell using Application Hosting



For front panel networking, you must configure the GigabitEthernet and VirtualPortGroup interfaces as shown above. The Guest Shell uses a Virtualportgroup as the source interface to connect to the outside network through NAT.

The following commands are used to configure inside NAT. They allow the Guest Shell to reach the internet; for example, to obtain Linux software updates:

```
ip nat inside source list
ip access-list standard
permit
```

The **guestshell run** command in the example above, runs a python executable. You can also use the **guestshell run** command to run other Linux executables; for example, see the example **guestshell run bash** command, which starts a Bash shell or the **guestshell disable** command which shuts down and disables the Guest Shell. If the system is later reloaded, the Guest Shell remains disabled.

Configuring the AppGigabitEthernet Interface for Guest Shell

Note The following task is applicable only to Catalyst switches that have the AppGigabitEthernet interface. All other Catalyst switches use the management port.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface AppGigabitEthernet** *interface-number*
4. **switchport mode trunk**
5. **exit**
6. **app-hosting appid** *name*
7. **app-vnic AppGigabitEthernet trunk**
8. **vlan** *vlan-ID* **guest-interface** *guest-interface-number*
9. **guest-ipaddress** *ip-address* **netmask** *netmask*
10. **exit**

11. **exit**
12. **app-default-gateway** *ip-address* **guest-interface** *network-interface*
13. **nameserver#** *ip-address*
14. **end**
15. **guestshell enable**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface AppGigabitEthernet** *interface-number* **Example:** `Device(config)# interface AppGigabitEthernet 1/0/1`	Configures the AppGigabitEthernet interface and enters interface configuration mode.
Step 4	**switchport mode trunk** **Example:** `Device(config-if)# switchport mode trunk`	Sets the interface into permanent trunking mode and negotiates to convert the neighboring link into a trunk link.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid guestshell`	Configures an application and enters application-hosting configuration mode.
Step 7	**app-vnic AppGigabitEthernet trunk** **Example:** `Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk`	Configures a trunk port as the front-panel port for application hosting, and enters application-hosting trunk configuration mode.
Step 8	**vlan** *vlan-ID* **guest-interface** *guest-interface-number* **Example:** `Device(config-config-app-hosting-trunk)# vlan 4094 guest-interface 0`	Configures a VLAN guest interface and enters application-hosting VLAN-access IP configuration mode.

	Command or Action	Purpose
Step 9	**guest-ipaddress** *ip-address* **netmask** *netmask* **Example:** `Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.2.2 netmask 255.255.255.0`	(Optional) Configures a static IP address.
Step 10	**exit** **Example:** `Device(config-config-app-hosting-vlan-access-ip)# exit`	Exits application-hosting VLAN-access IP configuration mode and returns to application-hosting trunk configuration mode
Step 11	**exit** **Example:** `Device(config-config-app-hosting-trunk)# exit`	Exits application-hosting trunk configuration mode and returns to application-hosting configuration mode.
Step 12	**app-default-gateway** *ip-address* **guest-interface** *network-interface* **Example:** `Device(config-app-hosting)# app-default-gateway 192.168.2.1 guest-interface 0`	Configures the default management gateway.
Step 13	**nameserver#** *ip-address* **Example:** `Device(config-app-hosting)# name-server0 172.16.0.1`	Configures the Domain Name System (DNS) server.
Step 14	**end** **Example:** `Device(config-app-hosting)# end`	Exits application-hosting configuration mode and returns to privileged EXEC mode.
Step 15	**guestshell enable** **Example:** `Device# guestshell enable`	Enables the Guest Shell service.

Enabling Guest Shell on the Management Interface

Note This task is applicable to Cisco Catalyst 9200 Series Switches, Cisco Catalyst 9300 Series Switches, Cisco Catalyst 9400 Series Switches, Cisco Catalyst 9500 Series Switches, and Cisco Catalyst 9600 Series Switches.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *name*

4. **app-vnic management guest-interface** *interface-number*
5. **end**
6. **show app-hosting list**
7. **guestshell enable**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid guestshell`	Configures an application and enters application-hosting configuration mode.
Step 4	**app-vnic management guest-interface** *interface-number* **Example:** `Device(config-app-hosting)# app-vnic management guest-interface 0`	Configures the management gateway of the virtual network interface and guest interface, and enters application-hosting management-gateway configuration mode.
Step 5	**end** **Example:** `Device(config-app-hosting-mgmt-gateway)# end`	Exits application-hosting management-gateway configuration mode and returns to privileged EXEC mode.
Step 6	**show app-hosting list** **Example:** `Device# show app-hosting list`	Displays the current status of the installed applications. **Note** Guest Shell is displayed in the list of applications, only if it is installed.
Step 7	**guestshell enable** **Example:** `Device# guestshell enable`	Enables the Guest Shell service.

Enabling and Disabling NETCONF Access from Guest Shell

Before you begin

Initialize the following commands from within the Guest Shell to initialize the NETCONF-YANG access:

SUMMARY STEPS

1. **iosp_client -f netconf_enable guestshell** *port-number*
2. **iosp_client -f netconf_enable_passwordless guestshell** *username*
3. **iosp_client -f netconf_disable guestshell**
4. **iosp_client -f netconf_disable_passwordless guestshell** *username*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**iosp_client -f netconf_enable guestshell** *port-number* **Example:** `Guest Shell: iosp_client -f netconf_enable guestshell 3`	Configures the **netconf-yang ssh local-vrf guestshell** command, and blocks connections until NETCONF-YANG is up and running.
Step 2	**iosp_client -f netconf_enable_passwordless guestshell** *username* **Example:** `Guest Shell: iosp_client -f netconf_enable guestshell guestshell`	Creates the SSH keys required for Guest Shell access.
Step 3	**iosp_client -f netconf_disable guestshell** **Example:** `GuestShell: iosp_client -f netconf_disable guestshell`	Removes access to NETCONF from within the Guest Shell. • NETCONF-YANG configuration will still exist. To shut down NETCONF-YANG use the **no netconf-yang** command.
Step 4	**iosp_client -f netconf_disable_passwordless guestshell** *username* **Example:** `Guest Shell: iosp_client -f netconf_disable_passwordless guestshell guestshell`	Removes the access keys for the specified user. • NETCONF access is still enabled for the user; however the user will have to use a password to connect to NETCONF.

Example

Accessing the Python Interpreter

Python can be used interactively or Python scripts can be run in the Guest Shell. Use the **guestshell run python** command to launch the Python interpreter in Guest Shell and open the Python terminal.

Note In releases prior to Cisco IOS XE Amsterdam 17.3.1, Python V2 is the default. Python V3 is supported in Cisco IOS XE Amsterdam 17.1.1, and Cisco IOS XE Amsterdam 17.2.1. In Cisco IOS XE Amsterdam 17.3.1 and later releases, Python V3 is the default.

In Releases Prior to Cisco IOS XE Amsterdam 17.3.1

The **guestshell run** command is the Cisco IOS equivalent of running Linux executables, and when running a Python script from Cisco IOS, specify the absolute path. The following example shows how to specify the absolute path for the command:

```
Guestshell run python /flash/guest-share/sample_script.py parameter1 parameter2
```

The following example shows how to enable Python on a Cisco Catalyst 3650 Series Switch or a Cisco Catalyst 3850 Series Switch:

```
Device# guestshell run python

Python 2.7.11 (default, March 16 2017, 16:50:55)
[GCC 4.7.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>>>
```

The following example shows how to enable Python on a Cisco ISR 4000 Series Integrated Services Router:

```
Device# guestshell run python

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>>>
```

In Cisco IOS XE Amsterdam 17.3.1 and Later Releases

The following example shows how to enable Python on Cisco Catalyst 9000 Series Switches:

```
Device# guestshell run python3

Python 3.6.8 (default, Nov 21 2019, 22:10:21)
[GCC 8.3.1 20190507 (Red Hat 8.3.1-4)] on linux
Type "help", "copyright", "credits" or "license" for more information.>>>>>
```

Configuration Examples for the Guest Shell

Example: Managing the Guest Shell

In Cisco IOS XE Amsterdam 17.1.x to Cisco IOS XE Amsterdam 17.2.x

The following example shows how to enable Guest Shell. In Cisco IOS XE Amsterdam 17.1.x and Cisco IOS XE Amsterdam 17.2.x, both Python V2.7 and Python V3.6 are supported. However, Python V2.7 is the default in these releases.

```
Device> enable
Device# guestshell enable
```

```
Management Interface will be selected if configured
Please wait for completion
Guestshell enabled successfully

Device# guestshell run python
or
Device# guestshell run python3

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>>>


Device# guestshell run bash

[guestshell@guestshell ~]$

Device# guestshell disable

Guestshell disabled successfully

Device# guestshell destroy

Guestshell destroyed successfully
```

In Cisco IOS XE Amsterdam 17.3.1 and Later Releases

The following example shows how to enable Guest Shell. In Cisco IOS XE Amsterdam 17.3.1 and later releases, only Python V3.6 is supported.

```
Device> enable
Device# guestshell enable

Management Interface will be selected if configured
Please wait for completion
Guestshell enabled successfully

Device# guestshell run python3

Python 3.6.8 (default, Nov 21 2019, 22:10:21)
[GCC 8.3.1 20190507 (Red Hat 8.3.1-4)] on linux
Type "help", "copyright", "credits" or "license" for more information.>>>>>

>>>>>

Device# guestshell run bash

[guestshell@guestshell ~]$

Device# guestshell disable

Guestshell disabled successfully

Device# guestshell destroy

Guestshell destroyed successfully
```

Sample VirtualPortGroup Configuration

Note VirtualPortGroups are supported only on Cisco routing platforms.

When using the VirtualPortGroup interface for Guest Shell networking, the VirtualPortGroup interface must have a static IP address configured. The front port interface must be connected to the Internet and Network Address Translation (NAT) must be configured between the VirtualPortGroup and the front panel port.

The following is a sample VirtualPortGroup configuration:

```
Device> enable
Device# configure terminal
Device(config)# interface VirtualPortGroup 0
Device(config-if)# ip address 192.168.35.1 255.255.255.0
Device(config-if)# ip nat inside
Device(config-if)# no mop enabled
Device(config-if)# no mop sysid
Device(config-if)# exit
Device(config)# interface GigabitEthernet 0/0/3
Device(config-if)# ip address 10.0.12.19 255.255.0.0
Device(config-if)# ip nat outside
Device(config-if)# negotiation auto
Device(config-if)# exit
Device(config)# ip route 0.0.0.0 0.0.0.0 10.0.0.1
Device(config)# ip route 10.0.0.0 255.0.0.0 10.0.0.1
!Port forwarding to use ports for SSH and so on.
Device(config)# ip nat inside source static tcp 192.168.35.2 7023 10.0.12.19 7023 extendable
Device(config)# ip nat outside source list NAT_ACL interface GigabitEthernet 0/0/3 overload
Device(config)# ip access-list standard NAT_ACL
Device(config-std-nacl)# permit 192.168.0.0 0.0.255.255
Device(config-std-nacl)# exit

! App-hosting configuration
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 0
Device(config-app-hosting-gateway)# guest-ipaddress 192.168.35.2 netmask 255.255.255.0
Device(config-app-hosting-gateway)# exit
Device(config-app-hosting)# app-resource profile custom
Device(config-app-resource-profile-custom)# cpu 1500
Device(config-app-resource-profile-custom)# memory 512
Device(config-app-resource-profile-custom)# end

Device# guestshell enable
Device# guestshell run python
```

Example: Configuring the AppGigabitEthernet Interface for Guest Shell

Note The following task is applicable only to Catalyst switches that have the AppGigabitEthernet interface. All other Catalyst switches use the management port.

The following example shows how to configure an AppGigabitEthernet interface for Guest Shell. Here, VLAN 4094 creates a Network Address Translation (NAT) this is used for Guest Shell. VLAN 1 is an external interface.

```
Device> enable
Device# configure terminal
Device(config)# ip nat inside source list NAT_ACL interface vlan 1 overload
Device(config)# ip access-list standard NAT_ACL
Device(config-std-nacl)# permit 192.168.0.0 0.0.255.255
Device(config-std-nacl)# exit
Device(config)# vlan 4094
Device(config-vlan)# exit
Device(config)# interface vlan 4094
Device(config-if)# ip address 192.168.2.1 255.255.255.0
Device(config-if)# ip nat inside
Device(config-if)# exit
Device(config)# interface vlan 1
Device(config-if)# ip nat outside
Device(config-if)# exit
Device(config)# ip routing
Device(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1
Device(config)# interface AppGigabitEthernet 1/0/1
Device(config-if)# switchport mode trunk
Device(config-if)# exit
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic AppGigEthernet trunk
Device(config-config-app-hosting-trunk)# vlan 4094 guest-interface 0
Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.2.2 netmask
255.255.255.0
Device(config-config-app-hosting-vlan-access-ip)# exit
Device(config-config-app-hosting-trunk)# exit
Device(config-app-hosting)# app-default-gateway 192.168.2.1 guest-interface 0
Device(config-app-hosting)# name-server0 172.16.0.1
Device(config-app-hosting)# name-server1 198.51.100.1
Device(config-app-hosting)# end
Device# guestshell enable
```

Example: Enabling Guest Shell on the Management Interface

This example is applicable to Cisco Catalyst 9200 Series Switches, Cisco Catalyst 9300 Series Switches, Cisco Catalyst 9400 Series Switches, Cisco Catalyst 9500 Series Switches, and Cisco Catalyst 9600 Series Switches.

```
Device> enable
Device# configure terminal
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic management guest-interface 0
Device(config-app-hosting-mgmt-gateway)# end
```

```
Device# guestshell enable
```

Example: Guest Shell Usage

From the Guest Shell prompt, you can run Linux commands. The following example shows the usage of some Linux commands.

```
[guestshell@guestshell~]$ pwd
/home/guestshell

[guestshell@guestshell~]$ whoami
guestshell

[guestshell@guestshell~]$ uname -a
Linux guestshell 5.4.85 #1 SMP Tue Dec 22 10:50:44 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

Cisco 4000 Series Integrated Services Routers use the **dohost** provided by CentOS Linux release 7.1.1503.

Note The **dohost** command requires the **ip http server** command to be configured on the device.

Example: Guest Shell Networking Configuration

The following is a sample Guest Shell networking configuration.

- Configure Domain Name System (DNS)
- Configure proxy settings
- Configure YUM or PIP to use proxy settings

Sample DNS Configuration for Guest Shell

The following is a sample DNS configuration for Guest Shell:

```
[guestshell@guestshell ~]$ cat/etc/resolv.conf
nameserver 192.0.2.1

Other Options:
[guestshell@guestshell ~]$ cat/etc/resolv.conf
domain cisco.com
search cisco.com
nameserver 192.0.2.1
search cisco.com
nameserver 198.51.100.1
nameserver 172.16.0.6
domain cisco.com
```

```
nameserver 192.0.2.1
nameserver 172.16.0.6
nameserver 192.168.255.254
```

Example: Configuring Proxy Environment Variables

If your network is behind a proxy, configure proxy variables in Linux. If required, add these variables to your environment.

The following example shows how to configure your proxy variables:

```
[guestshell@guestshell ~]$cat /bootflash/proxy_vars.sh
export http_proxy=http://proxy.example.com:80/
export https_proxy=http://proxy.example.com:80/
export ftp_proxy=http://proxy.example.com:80/
export no_proxy=example.com
export HTTP_PROXY=http://proxy.example.com:80/
export HTTPS_PROXY=http://proxy.example.com:80/
export FTP_PROXY=http://proxy.example.com:80/
guestshell ~] source /bootflash/proxy_vars.sh
```

Example: Configuring Yum and PIP for Proxy Settings

The following example shows how to use Yum for setting proxy environment variables:

```
cat /etc/yum.conf | grep proxy
[guestshell@guestshell~]$ cat/bootflash/yum.conf | grep proxy
proxy=http://proxy.example.com:80/
```

PIP install picks up environment variable used for proxy settings. Use sudo with -E option for PIP installation. If the environment variables are not set, define them explicitly in PIP commands as shown in following example:

```
sudo pip --proxy http://proxy.example.com:80/install requests
sudo pip install --trusted-bost pypi.example.com --index-url
http://pypi.example.com/simple requests
```

The following example shows how to use PIP install for Python:

```
Sudo -E pip install requests
[guestshell@guestshell ~]$ python
Python 2.17.11 (default, Feb 3 2017, 19:43:44)
[GCC 4.7.0] on linux2
Type "help", "copyright", "credits" or "license" for more information
>>>import requests
```

Additional References for Guest Shell

Related Documents

Related Topic	Document Title
Python module	CLI Python Module
Zero-Touch Provisioning	Zero-Touch Provisioning

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for Guest Shell

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

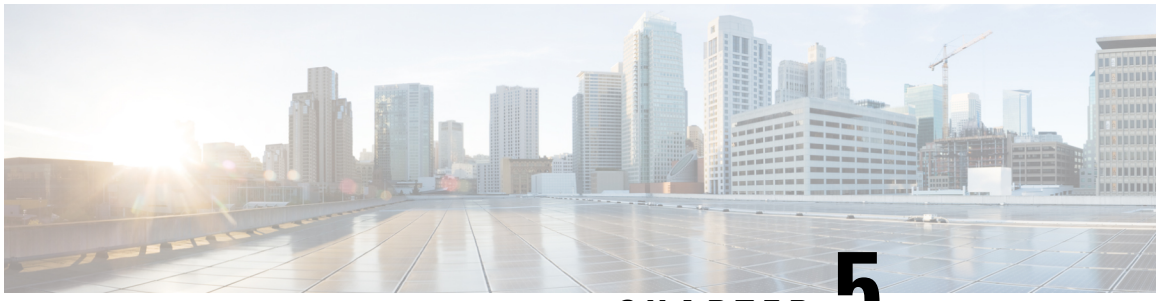
Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 9: Feature Information for Guest Shell

Feature Name	Release	Feature Information
Guest Shell	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	Guest Shell is a secure container that is an embedded Linux environment that allows customers to develop and run Linux and custom Python applications for automated control and management of Cisco switches. It also includes the automated provisioning of systems. This container shell provides a secure environment, decoupled from the host device, in which users can install scripts or software packages and run them. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches In Cisco IOS Everest 16.5.1b, this feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers
	Cisco IOS XE Everest 16.6.2	In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.
	Cisco IOS XE Fuji 16.7.1	

Feature Name	Release	Feature Information
		In Cisco IOS XE Fuji 16.7.1, this feature was implemented on the following platforms: • Cisco ASR 1000 Series Aggregation Services Routers • Cisco Cloud Services Router 1000v Series In Cisco IOS XE Fuji 16.7.1, for Guest Shell feature, the Logging and Tracing support was implemented on Cisco ASR 1000 Aggregation Services Routers.
	Cisco IOS XE Fuji 16.8.1	In Cisco IOS XE Fuji 16.8.1, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches.
	Cisco IOS XE Fuji 16.9.1	In Cisco IOS XE Fuji 16.9.1, this feature was implemented on Cisco 1000 Series Integrated Services Routers.
	Cisco IOS XE Gibraltar 16.11.1b	In Cisco IOS XE Gibraltar 16.11.1b, this feature was implemented on the following platforms: • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers
	Cisco IOS XE Gibraltar 16.12.1	In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on the following platforms: • Cisco Catalyst 9200 Series Switches **Note** This feature is not supported on C9200L SKUs. • Cisco Catalyst 9300L SKUs • Cisco Catalyst 9600 Series Switches

Feature Name	Release	Feature Information
	Cisco IOS XE Amsterdam 17.3.1	In Cisco IOS XE Amsterdam 17.3.1, this feature was implemented on the following platforms: • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 and 8500L Series Edge Platforms
NETCONF Access from Guest Shell	Cisco IOS XE Bengaluru 17.6.1	NETCONF can be accessed from within the Guest Shell, so that users can run Python scripts and invoke Cisco-custom package CLIs using the NETCONF protocol. In 17.6.1, this feature was implemented on the following platforms: • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches
Python 3 Support in Guest Shell	Cisco IOS XE Amsterdam 17.1.1	Python Version 3.6 is supported in Guest Shell. Python Version 3.6 is available on all supported platforms.



CHAPTER 5

Python API

Python programmabililty supports Python APIs.

- About Python , on page 133
- Additional References for Python API, on page 141
- Feature Information for Python API, on page 141

About Python

The Cisco IOS XE devices support Python Version 2.7 in both interactive and non-interactive (script) modes within the Guest Shell. The Python scripting capability gives programmatic access to a device's CLI to perform various tasks and Zero Touch Provisioning or Embedded Event Manager (EEM) actions.

Cisco Python Module

Cisco provides a Python module that provides access to run EXEC and configuration commands. You can display the details of the Cisco Python module by entering the **help()** command. The **help()** command displays the properties of the Cisco CLI module.

The following example displays information about the Cisco Python module:

```
Device# guestshell run python

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> >>> from cli import cli,clip,configure,configurep, execute, executep
>>> help(configure)
Help on function configure in module cli:

configure(configuration)
Apply a configuration (set of Cisco IOS CLI config-mode commands) to the device
and return a list of results.

configuration = '''interface gigabitEthernet 0/0
no shutdown'''

# push it through the Cisco IOS CLI.
try:
results = cli.configure(configuration)
print "Success!"
```

```
except CLIConfigurationError as e:
print "Failed configurations:"
for failure in e.failed:
print failure

Args:
configuration (str or iterable): Configuration commands, separated by newlines.

Returns:
list(ConfigResult): A list of results, one for each line.

Raises:
CLISyntaxError: If there is a syntax error in the configuration.

>>> help(configurep)
Help on function configurep in module cli:

configurep(configuration)
Apply a configuration (set of Cisco IOS CLI config-mode commands) to the device
and prints the result.

configuration = '''interface gigabitEthernet 0/0
no shutdown'''

# push it through the Cisco IOS CLI.
configurep(configuration)

Args:
configuration (str or iterable): Configuration commands, separated by newlines.
>>> help(execute)
Help on function execute in module cli:

execute(command)
Execute Cisco IOS CLI exec-mode command and return the result.

command_output = execute("show version")

Args:
command (str): The exec-mode command to run.

Returns:
str: The output of the command.

Raises:
CLISyntaxError: If there is a syntax error in the command.

>>> help(executep)
Help on function executep in module cli:

executep(command)
Execute Cisco IOS CLI exec-mode command and print the result.

executep("show version")

Args:
command (str): The exec-mode command to run.


>>> help(cli)
Help on function cli in module cli:

cli(command)
    Execute Cisco IOS CLI command(s) and return the result.
```

```
    A single command or a delimited batch of commands may be run. The
    delimiter is a space and a semicolon, " ;". Configuration commands must be
    in fully qualified form.

    output = cli("show version")
    output = cli("show version ; show ip interface brief")
    output = cli("configure terminal ; interface gigabitEthernet 0/0 ; no shutdown")

    Args:
        command (str): The exec or config CLI command(s) to be run.

    Returns:
        string: CLI output for show commands and an empty string for
            configuration commands.

    Raises:
        errors.cli_syntax_error: if the command is not valid.
        errors.cli_exec_error: if the execution of command is not successful.
```

```
>>> help(clip)
Help on function clip in module cli:

clip(command)
    Execute Cisco IOS CLI command(s) and print the result.

    A single command or a delimited batch of commands may be run. The
    delimiter is a space and a semicolon, " ;". Configuration commands must be
    in fully qualified form.

    clip("show version")
    clip("show version ; show ip interface brief")
    clip("configure terminal ; interface gigabitEthernet 0/0 ; no shutdown")

    Args:
        command (str): The exec or config CLI command(s) to be run.
```

Cisco Python Module to Execute IOS CLI Commands

Note Guest Shell must be enabled for Python to run. For more information, see the *Guest Shell* chapter.

The Python programming language uses six functions that can execute CLI commands. These functions are available from the Python CLI module. To use these functions, execute the **import cli** command.

Arguments for these functions are strings of CLI commands. To execute a CLI command through the Python interpreter, enter the CLI command as an argument string of one of the following six functions:

- **cli.cli(command)**—This function takes an IOS command as an argument, runs the command through the IOS parser, and returns the resulting text. If this command is malformed, a Python exception is raised. The following is sample output from the **cli.cli(command)** function:

```
>>> import cli
>>> cli.clip('configure terminal; interface loopback 10; ip address
10.10.10.10 255.255.255.255')
*Mar 13 18:39:48.518: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback10, changed
```

```
 state to up
>>> cli.clip('show clock')
'\n*18:11:53.989 UTC Mon Mar 13 2017\n'
>>> output=cli.cli('show clock')
>>> print(output)
*18:12:04.705 UTC Mon Mar 13 2017
```

- **cli.clip(command)**—This function works exactly the same as the **cli.cli(command)** function, except that it prints the resulting text to *stdout* rather than returning it. The following is sample output from the **cli.clip(command)** function:

```
>>> cli
>>> cli.clip('configure terminal; interface loopback 11; ip address
10.11.11.11 255.255.255.255')
*Mar 13 18:42:35.954: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback11, changed
 state to up
*Mar 13 18:42:35.954: %LINK-3-UPDOWN: Interface Loopback11, changed state to up
>>> cli.clip('show clock')
*18:13:35.313 UTC Mon Mar 13 2017
>>> output=cli.clip('show clock')
*18:19:26.824 UTC Mon Mar 13 2017
>>> print (output)
None
```

- **cli.execute(command)**—This function executes a single EXEC command and returns the output; however, does not print the resulting text No semicolons or newlines are allowed as part of this command. Use a Python list with a for-loop to execute this function more than once. The following is sample output from the **cli.execute(command)**

  function:

```
>>> cli.execute("show clock")
'15:11:20.816 UTC Thu Jun 8 2017'
>>>
>>> cli.execute('show clock'; 'show ip interface brief')
  File "<stdin>", line 1
    cli.execute('show clock'; 'show ip interface brief')
                            ^
SyntaxError: invalid syntax
>>>
```

- **cli.executep(command)**—This function executes a single command and prints the resulting text to *stdout* rather than returning it. The following is sample output from the **cli.executep(command)** function:

```
>>> cli.executep('show clock')
*18:46:28.796 UTC Mon Mar 13 2017
>>> output=cli.executep('show clock')
*18:46:36.399 UTC Mon Mar 13 2017
>>> print(output)
None
```

- **cli.configure(command)**—This function configures the device with the configuration available in commands. It returns a list of named tuples that contains the command and its result as shown below:

```
[Think: result = (bool(success), original_command, error_information)]
```

The command parameters can be in multiple lines and in the same format that is displayed in the output of the **show running-config** command. The following is sample output from the **cli.configure(command)** function:

```
>>>cli.configure(["interface GigabitEthernet1/0/7", "no shutdown",
"end"])
[ConfigResult(success=True, command='interface GigabitEthernet1/0/7',
line=1, output='', notes=None), ConfigResult(success=True, command='no shutdown',
line=2, output='', notes=None), ConfigResult(success=True, command='end',
line=3, output='', notes=None)]
```

- **cli.configurep(command)**—This function works exactly the same as the **cli.configure(command)** function, except that it prints the resulting text to *stdout* rather than returning it. The following is sample output from the **cli.configurep(command)** function:

```
>>> cli.configurep(["interface GigabitEthernet1/0/7", "no shutdown",
"end"])
Line 1 SUCCESS: interface GigabitEthernet1/0/7
Line 2 SUCCESS: no shut
Line 3 SUCCESS: end
```

Python Scripts Overview

Python run in a virtualized Linux-based environment, Guest Shell. For more information, see the *Guest Shell* chapter. Cisco provides a Python module that allows user's Python scripts to run IOS CLI commands on the host device.

Interactive Python Prompt

When you execute the **guestshell run python** command on a device, the interactive Python prompt is opened inside the Guest Shell. The Python interactive mode allows users to execute Python functions from the Cisco Python CLI module to configure the device.

The following example shows how to enable the interactive Python prompt:

```
Device# guestshell run python

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>

Device#
```

Python Script

Python scripts can run in non-interactive mode by providing the Python script name as an argument in the Python command. Python scripts must be accessible from within the Guest Shell. To access Python scripts from the Guest Shell, save the scripts in bootflash/flash that is mounted within the Guest Shell.

The following sample Python script uses different CLI functions to configure and print **show** commands:

```
Device# more flash:sample_script.py


import sys
import cli

intf= sys.argv[1:]
intf = ''.join(intf[0])

print "\n\n *** Configuring interface %s with 'configurep' function  *** \n\n" %intf
cli.configurep(["interface loopback55","ip address 10.55.55.55 255.255.255.0","no
shut","end"])

print "\n\n *** Configuring interface %s with 'configure' function  *** \n\n"
cmd='interface %s,logging event link-status ,end' % intf
cli.configure(cmd.split(','))

print "\n\n *** Printing show cmd with 'executep' function  *** \n\n"
cli.executep('show ip interface brief')

print "\n\n *** Printing show cmd with 'execute' function  *** \n\n"
output= cli.execute('show run interface %s' %intf)
print (output)

print "\n\n *** Configuring interface %s with 'cli' function  *** \n\n"
cli.cli('config terminal; interface %s; spanning-tree portfast edge default' %intf)

print "\n\n *** Printing show cmd with 'clip' function  *** \n\n"
cli.clip('show run interface %s' %intf)


To run a Python script from the Guest Shell, execute the guestshell run python
/bootflash/guest-share/script.py command
at the device prompt.
```

The following example shows how to run a Python script from the Guest Shell:

```
Device# guestshell run python /bootflash/guest-share/sample_script.py loop55

 *** Configuring interface loop55 with 'configurep' function  ***

Line 1 SUCCESS: interface loopback55
Line 2 SUCCESS: ip address 10.55.55.55 255.255.255.0
Line 3 SUCCESS: no shut
Line 4 SUCCESS: end

 *** Configuring interface %s with 'configure' function  ***


 *** Printing show cmd with 'executep' function  ***

Interface              IP-Address      OK? Method Status                Protocol
Vlan1                  unassigned      YES NVRAM  administratively down down
GigabitEthernet0/0     192.0.2.1       YES NVRAM  up                    up
GigabitEthernet1/0/1   unassigned      YES unset  down                  down
GigabitEthernet1/0/2   unassigned      YES unset  down                  down
GigabitEthernet1/0/3   unassigned      YES unset  down                  down
    :
         :
         :
Te1/1/4                unassigned      YES unset  down                  down
```

```
Loopback55             10.55.55.55     YES TFTP   up                    up
Loopback66             unassigned      YES manual up                    up


 *** Printing show cmd with 'execute' function  ***

Building configuration...
Current configuration : 93 bytes
!
interface Loopback55
 ip address 10.55.55.55 255.255.255.0
 logging event link-status
end

 *** Configuring interface %s with 'cli' function  ***

 *** Printing show cmd with 'clip' function  ***

Building configuration...
Current configuration : 93 bytes
!
interface Loopback55
 ip address 10.55.55.55 255.255.255.0
 logging event link-status
end
```

Supported Python Versions

Guest Shell is pre-installed with Python Version 2.7. Guest Shell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python applications for automated control and management of Cisco devices. Platforms with Montavista CGE7 support Python Version 2.7.11, and platforms with CentOS 7 support Python Version 2.7.5.

The following table provides information about Python versions and the supported platforms:

Table 10: Python Version Support

Python Version	Platform
Python Version 2.7.5	All supported platforms except for Cisco Catalyst 3650 Series Switches and Cisco Catalyst 3850 Series Switches.
Python Version 2.7.11	• Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches

Python Version	Platform
Python Version 3.6	Supported in Cisco IOS XE Amsterdam 17.1.1 and later releases. In Cisco IOS XE Amsterdam 17.1.1 and Cisco IOS XE Amsterdam 17.2.1, Python V2 is the default. However, in Cisco IOS XE Amsterdam 17.3.1 and later releases, Python V3 is the default. **Note** Cisco Catalyst 9200 Series Switches do not support Python Version 3.6 in Cisco IOS XE Amsterdam 17.1.1 and Cisco IOS XE Amsterdam 17.2.1. Cisco Catalyst 9200 Series Switches support Python V3 in Cisco IOS XE Amsterdam 17.3.1 and later releases. **Note** Not supported by Cisco Catalyst 3650 Series Switches and Cisco Catalyst 3850 Series Switches.

Platforms with CentOS 7 support the installation of Redhat Package Manager (RPM) from the open source repository.

Updating the Cisco CLI Python Module

The Cisco CLI Python module and EEM module are pre-installed on devices. However, when you update the Python version by using either Yum or prepackaged binaries, the Cisco-provided CLI module must also be updated.

Note When you update to Python Version 3 on a device that already has Python Version 2, both versions of Python exist on the device. Use one of the following IOS commands to run Python:

- The **guestshell run python2** command enables Python Version 2.
- The **guestshell run python3** command enables Python Version 3.
- The **guestshell run python** command enables Python Version 2.

Use one of the following methods to update the Python version:

- Standalone tarball installation
- PIP install for the CLI module

Additional References for Python API

Related Documents

Related Topic	Document Title
Guest Shell	Guest Shell
EEM Python Module	Python Scripting in EEM

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

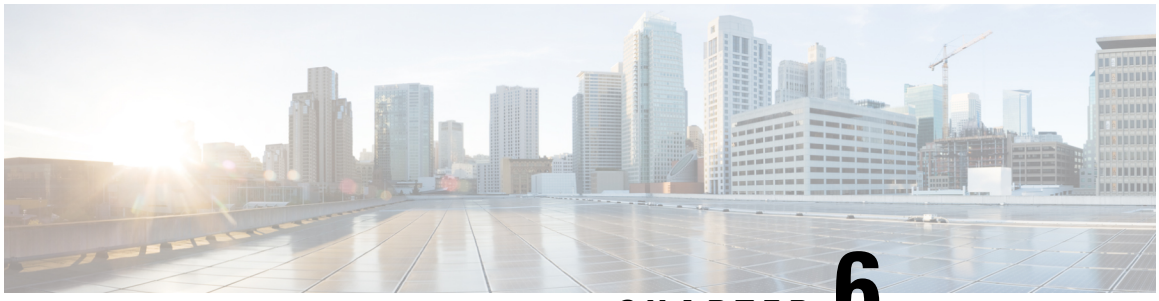
Feature Information for Python API

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 11: Feature Information for the CLI Python Module

Feature Name	Release	Feature Information
CLI Python Module	Cisco IOS XE Everest 16.5.1a	Python programmabilty provides a Python module that allows users to interact with IOS using CLIs. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers
	Cisco IOS XE Everest 16.6.2	This feature was implemented on Cisco Catalyst 9400 Series Switches.
	Cisco IOS XE Fuji 16.7.1	This feature was implemented on the following platforms: • Cisco ASR 1000 Aggregation Services Routers • Cisco CSR 1000v Series Cloud Services Routers
	Cisco IOS XE Fuji 16.8.1 Cisco IOS XE Fuji 16.8.1a	In Cisco IOS XE Fuji 16.8.1, this feature was implemented on following platforms: • Cisco ASR 1004 Router • Cisco ASR 1006 Router • Cisco ASR 1006-X Router • Cisco ASR 1009-X Router • Cisco ASR 1013 Router • Cisco 4000 Series Integrated Services Router models with a minimum of 4 GB RAM. In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches



CHAPTER 6

EEM Python Module

Embedded Event Manager (EEM) policies support Python scripts. Python scripts can be executed as part of EEM actions in EEM applets.

- Prerequisites for the EEM Python Module, on page 143
- Information About EEM Python Module, on page 143
- How to Configure the EEM Python Policy, on page 146
- Additional References EEM Python Module, on page 151
- Feature Information for EEM Python Module, on page 152

Prerequisites for the EEM Python Module

Guest Shell must be working within the container. Guest Shell is not enabled by default. For more information see the *Guest Shell* feature.

Information About EEM Python Module

Python Scripting in EEM

Embedded Event Manager (EEM) policies support Python scripts. You can register Python scripts as EEM policies, and execute the registered Python scripts when a corresponding event occurs. The EEM Python script has the same event specification syntax as the EEM TCL policy.

Configured EEM policies run within the Guest Shell. Guest Shell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python for automated control and management of Cisco devices. The Guest Shell container provides a Python interpreter.

EEM Python Package

The EEM Python package can be imported to Python scripts for running EEM-specific extensions.

Note The EEM Python package is available only within the EEM Python script (The package can be registered with EEM, and has the EEM event specification in the first line of the script.) and not in the standard Python script (which is run using the Python script name).

The Python package includes the following application programming interfaces (APIs):

- Action APIs—Perform EEM actions and have default parameters.
- CLI-execution APIs—Run IOS commands, and return the output. The following are the list of CLI-execution APIs:
  - eem_cli_open()
  - eem_cli_exec()
  - eem_cli_read()
  - eem_cli_read_line()
  - eem_cli_run()
  - eem_cli_run_interactive()
  - eem_cli_read_pattern()
  - eem_cli_write()
  - eem_cli_close()
- Environment variables-accessing APIs—Get the list of built-in or user-defined variables. The following are the environment variables-accessing APIs:
  - eem_event_reqinfo ()-Returns the built-in variables list.
  - eem_user_variables()-Returns the current value of an argument.

Python-Supported EEM Actions

The Python package (is available only within the EEM script, and not available for the standard Python script) supports the following EEM actions:

- Syslog message printing
- Send SNMP traps
- Reload the box
- Switchover to the standby device
- Run a policy
- Track Object read
- Track Object Set
- Cisco Networking Services event generation

The EEM Python package exposes the interfaces for executing EEM actions. You can use the Python script to call these actions, and they are forwarded from the Python package via Cisco Plug N Play (PnP) to the action handler.

EEM Variables

An EEM policy can have the following types of variables:

- Event-specific built-in variables—A set of predefinied variables that are populated with details about the event that triggered the policy. The eem_event_reqinfo () API returns the builtin variables list. These variables can be stored in the local machine and used as local variables. Changes to local variables do not reflect in builtin variables.
- User-defined variables—Variables that can be defined and used in policies. The value of these variables can be referred in the Python script. While executing the script, ensure that the latest value of the variable is available. The eem_user_variables() API returns the current value of the argument that is provided in the API.

EEM CLI Library Command Extensions

The following CLI library commands are available within EEM for the Python script to work:

- eem_cli_close()—Closes the EXEC process and releases the VTY and the specified channel handler connected to the command.
- eem_cli_exec—Writes the command to the specified channel handler to execute the command. Then reads the output of the command from the channel and returns the output.
- eem_cli_open—Allocates a VTY, creates an EXEC CLI session, and connects the VTY to a channel handler. Returns an array including the channel handler.
- eem_cli_read()—Reads the command output from the specified CLI channel handler until the pattern of the device prompt occurs in the contents read. Returns all the contents read up to the match.
- eem_cli_read_line()—Reads one line of the command output from the specified CLI channel handler. Returns the line read.
- eem_cli_read_pattern()—Reads the command output from the specified CLI channel handler until the pattern that is to be matched occurs in the contents read. Returns all the contents read up to the match.
- eem_cli_run()—Iterates over the items in the *clist* and assumes that each one is a command to be executed in the enable mode. On success, returns the output of all executed commands and on failure, returns error.
- eem_cli_run_interactive()—Provides a sublist to the *clist* which has three items. On success, returns the output of all executed commands and on failure, returns the error. Also uses arrays when possible as a way of making things easier to read later by keeping expect and reply separated.
- eem_cli_write()—Writes the command that is to be executed to the specified CLI channel handler. The CLI channel handler executes the command.

How to Configure the EEM Python Policy

For the Python script to work, you must enable the Guest Shell. For more information, see the *Guest Shell* chapter.

Registering a Python Policy

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager directory user policy** *path*
4. **event manager policy** *policy-filename*
5. **exit**
6. **show event manager policy registered**
7. **show event manager history events**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**event manager directory user policy** *path* **Example:** `Device(config)# event manager directory user policy flash:/user_library`	Specifies a directory to use for storing user library files or user-defined EEM policies. **Note** You must have a policy in the specified path. For example, in this step, the eem_script.py policy is available in the flash:/user_library folder or path.
Step 4	**event manager policy** *policy-filename* **Example:** `Device(config)# event manager policy eem_script.py`	Registers a policy with EEM. • The policy is parsed based on the file extension. If the file extension is .py, the policy is registered as Python policy. • EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. When the **event manager policy** command is

	Command or Action	Purpose
		invoked, EEM examines the policy and registers it to be run when the specified event occurs.
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 6	**show event manager policy registered** **Example:** `Device# show event manager policy registered`	Displays the registered EEM policies.
Step 7	**show event manager history events** **Example:** `Device# show event manager history events`	Displays EEM events that have been triggered.

Example

The following is sample output from the **show event manager policy registered** command:

```
Device# show event manager policy registered

No.  Class     Type    Event Type          Trap  Time Registered           Name
1    script    user    multiple            Off   Tue Aug 2 22:12:15 2016   multi_1.py
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi_1.py} sync {yes}
 trigger delay 10.000
  correlate event 1 or event 2
  attribute tag 1 occurs 1
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none


2    script    user    multiple            Off   Tue Aug 2 22:12:20 2016   multi_2.py
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi_2.py} sync {yes}
 trigger
  correlate event 1 or event 2
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none


3    script    user    multiple            Off   Tue Aug 2 22:13:31 2016   multi.tcl
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi.tcl} sync {yes}
 trigger
  correlate event 1 or event 2
  attribute tag 1 occurs 1
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none
```

Running Python Scripts as Part of EEM Applet Actions

Python Script: eem_script.py

An EEM applet can include a Python script with an action command. In this example, an user is trying to run a standard Python script as part of the EEM action, however; EEM Python package is not available in the standard Python script. The standard Python script in IOS has a package named *from cli import cli,clip* and this package can be used to execute IOS commands.

```
import sys
from cli import cli,clip,execute,executep,configure,configurep

intf= sys.argv[1:]
intf = ''.join(intf[0])


print ('This script is going to unshut interface %s and then print show ip interface
brief'%intf)

if intf == 'loopback55':
configurep(["interface loopback55","no shutdown","end"])
else :
cmd='int %s,no shut ,end' % intf
configurep(cmd.split(','))

executep('show ip interface brief')
```

This following is sample output from the **guestshell run python** command.

```
Device#  guestshell run python /flash/eem_script.py loop55

This script is going to unshut interface loop55 and then print show ip interface brief
Line 1 SUCCESS: int loop55
Line 2 SUCCESS: no shut
Line 3 SUCCESS: end
Interface IP-Address OK? Method Status Protocol
Vlan1 unassigned YES NVRAM administratively down down
GigabitEthernet0/0 5.30.15.37 YES NVRAM up up
GigabitEthernet1/0/1 unassigned YES unset down down
GigabitEthernet1/0/2 unassigned YES unset down down
GigabitEthernet1/0/3 unassigned YES unset down down
GigabitEthernet1/0/4 unassigned YES unset up up
GigabitEthernet1/0/5 unassigned YES unset down down
GigabitEthernet1/0/6 unassigned YES unset down down
GigabitEthernet1/0/7 unassigned YES unset down down
GigabitEthernet1/0/8 unassigned YES unset down down
GigabitEthernet1/0/9 unassigned YES unset down down
GigabitEthernet1/0/10 unassigned YES unset down down
GigabitEthernet1/0/11 unassigned YES unset down down
GigabitEthernet1/0/12 unassigned YES unset down down
GigabitEthernet1/0/13 unassigned YES unset down down
GigabitEthernet1/0/14 unassigned YES unset down down
GigabitEthernet1/0/15 unassigned YES unset down down
GigabitEthernet1/0/16 unassigned YES unset down down
GigabitEthernet1/0/17 unassigned YES unset down down
GigabitEthernet1/0/18 unassigned YES unset down down
GigabitEthernet1/0/19 unassigned YES unset down down
GigabitEthernet1/0/20 unassigned YES unset down down
GigabitEthernet1/0/21 unassigned YES unset down down
GigabitEthernet1/0/22 unassigned YES unset down down
```

```
GigabitEthernet1/0/23 unassigned YES unset up up
GigabitEthernet1/0/24 unassigned YES unset down down
GigabitEthernet1/1/1 unassigned YES unset down down
GigabitEthernet1/1/2 unassigned YES unset down down
GigabitEthernet1/1/3 unassigned YES unset down down
GigabitEthernet1/1/4 unassigned YES unset down down
Te1/1/1 unassigned YES unset down down
Te1/1/2 unassigned YES unset down down
Te1/1/3 unassigned YES unset down down
Te1/1/4 unassigned YES unset down down
Loopback55 10.55.55.55 YES manual up up

Device#
Jun 7 12:51:20.549: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback55,
changed state to up
Jun 7 12:51:20.549: %LINK-3-UPDOWN: Interface Loopback55, changed state to up
```

The following is a sample script for printing messages to the syslog. This script must be stored in a file, copied to the file system on the device, and registered using the event manager policy file.

```
::cisco::eem::event_register_syslog tag "1"    pattern     COUNTER    maxrun 200

import eem
import time

eem.action_syslog("SAMPLE SYSLOG MESSAGE","6","TEST")
```

The following is sample script to print EEM environment variables. This script must be stored in a file, copied to the file system on the device, and registered using the event manager policy file.

```
::cisco::eem::event_register_syslog tag "1"    pattern     COUNTER    maxrun 200

import eem
import time

c = eem.env_reqinfo()


print "EEM Environment Variables"
for k,v in c.iteritems():
      print "KEY : " + k + str(" ---> ") + v

print "Built in Variables"
for i,j in a.iteritems() :
      print "KEY : " + i + str(" ---> ") + j
```

Adding a Python Script in an EEM Applet

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event** [**tag** *event-tag*] **syslog pattern** *regular-expression*
5. **action** *label* **cli command** *cli-string*

6. **action** *label* **cli command** *cli-string* [**pattern** *pattern-string*]
7. **end**
8. **show event manager policy active**
9. **show event manager history events**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Device(config)# event manager applet interface_Shutdown`	Registers an applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 4	**event** [**tag** *event-tag*] **syslog pattern** *regular-expression* **Example:** `Device(config-applet)# event syslog pattern "Interface Loopback55, changed state to administratively down"`	Specifies a regular expression to perform the syslog message pattern match.
Step 5	**action** *label* **cli command** *cli-string* **Example:** `Device(config-applet)# action 0.0 cli command "en"`	Specifies the IOS command to be executed when an EEM applet is triggered.
Step 6	**action** *label* **cli command** *cli-string* [**pattern** *pattern-string*] **Example:** `Device(config-applet)# action 1.0 cli command "guestshell run python3 /bootflash/eem_script.py loop55"`	Specifies the action to be specified with the **pattern** keyword. • Specify a regular expression pattern string that will match the next solicited prompt.
Step 7	**end** **Example:** `Device(config-applet)# end`	Exits applet configuration mode and returns to privileged EXEC mode.
Step 8	**show event manager policy active** **Example:**	Displays EEM policies that are executing.

	Command or Action	Purpose
	`Device# show event manager policy active`	
Step 9	**show event manager history events** **Example:** `Device# show event manager history events`	Displays the EEM events that have been triggered.

What to do next

The following example shows how to trigger the Python script configured in the task:

```
Device(config)# interface loopback 55
Device(config-if)# shutdown
Device(config-if)# end
Device#

Mar 13 10:53:22.358 EDT: %SYS-5-CONFIG_I: Configured from console by console
Mar 13 10:53:24.156 EDT: %LINK-5-CHANGED:  Line protocol on Interface Loopback55, changed
state to down
Mar 13 10:53:27.319 EDT: %LINK-3-UPDOWN:  Interface Loopback55, changed state to
administratively down
Enter configuration commands, one per line.  End with CNTL/Z.
Mar 13 10:53:35.38 EDT:  %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback55, changed
 state to up
*Mar 13 10:53:35.39 EDT %LINK-3-UPDOWN: Interface Loopback55, changed state to up
+++ 10:54:33 edi37(default) exec +++
show ip interface br
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  down                  down
GigabitEthernet0/0/1   unassigned      YES unset  down                  down
GigabitEthernet0/0/2   10.1.1.31       YES DHCP   up                    up
GigabitEthernet0/0/3   unassigned      YES unset  down                  down
GigabitEthernet0       192.0.2.1       YES manual up                    up
Loopback55             198.51.100.1    YES manual up                    up
Loopback66             172.16.0.1      YES manual up                    up
Loopback77             192.168.0.1     YES manual up                    up
Loopback88             203.0.113.1     YES manual up                    up
```

Additional References EEM Python Module

Related Documents

Related Topic	Document Title
EEM configuration	*Embedded Event Manager Configuration Guide*
EEM commands	*Embedded Event Manager Command Reference*
Guest Shell configuration	Guest Shell

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for EEM Python Module

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

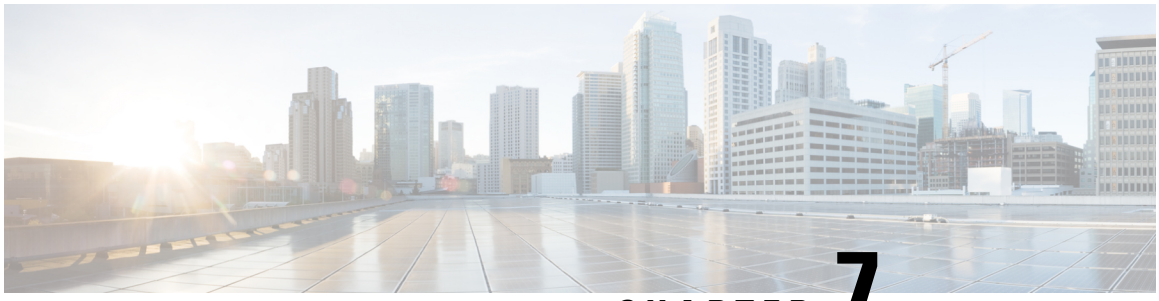
Table 12: Feature Information for EEM Python Module

Feature Name	Release	Feature Information
EEM Python Module	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	This feature supports Python scripts as EEM policies. No new commands were introduced. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms: • Cisco ISR 4000 Series Integrated Service Routers
	Cisco IOS XE Everest 16.6.2	In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.
	Cisco IOS XE Fuji 16.8.1a	In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches

PART III

Model-Driven Programmability

- NETCONF Protocol, on page 157
- RESTCONF Protocol, on page 205
- NETCONF and RESTCONF Service-Level ACLs, on page 241
- gNMI Protocol, on page 247
- gRPC Network Operations Interface, on page 275
- gNMI Dial-Out Using the gRPC Tunnel Service, on page 295
- Model Based AAA, on page 303
- Model-Driven Telemetry, on page 311
- In-Service Model Update, on page 383



CHAPTER 7

NETCONF Protocol

- Information About the NETCONF Protocol, on page 157
- How to Configure the NETCONF Protocol, on page 180
- Verifying the NETCONF Protocol Configuration Through the CLI, on page 186
- Example: Named Method List, on page 189
- Displaying NETCONF-YANG Diagnostics Through RPCs, on page 189
- Additional References for NETCONF Protocol, on page 192
- Feature Information for the NETCONF Protocol, on page 194

Information About the NETCONF Protocol

Introduction to Data Models: Programmatic and Standards-Based Configuration

The traditional way of managing network devices is by using Command Line Interfaces (CLIs) for configurational (configuration commands) and operational data (show commands). For network management, Simple Network Management Protocol (SNMP) is widely used, especially for exchanging management information between various network devices. Although CLIs and SNMP are heavily used, they have several restrictions. CLIs are highly proprietary, and human intervention is required to understand and interpret their text-based specification. SNMP does not distinguish between configurational and operational data.

The solution lies in adopting a programmatic and standards-based way of writing configurations to any network device, replacing the process of manual configuration. Network devices running on Cisco IOS XE support the automation of configuration for multiple devices across the network using data models. Data models are developed in a standard, industry-defined language, that can define configuration and state information of a network.

Cisco IOS XE supports the Yet Another Next Generation (YANG) data modeling language. YANG can be used with the Network Configuration Protocol (NETCONF) to provide the desired solution of automated and programmable network operations. NETCONF (RFC 6241) is an XML-based protocol that client applications use to request information from and make configuration changes to the device. YANG is primarily used to model the configuration and state data used by NETCONF operations.

In Cisco IOS XE, model-based interfaces interoperate with existing device CLI, Syslog, and SNMP interfaces. These interfaces are optionally exposed northbound from network devices. YANG is used to model each protocol based on RFC 6020.

From Cisco IOS XE 17.15.1, use the **netconf-yang feature enable-deprecated** command or the Cisco-IOS-XE-Yang-interfaces-cfg.yang model to disable the deprecated NETCONF-YANG model elements. By default deprecated model elements remain enabled.

Note To access Cisco YANG models in a developer-friendly way, clone the *GitHub repository*, and navigate to the *vendor/cisco* subdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.

NETCONF

NETCONF provides a mechanism to install, manipulate, and delete the configuration of network devices.

It uses an Extensible Markup Language (XML)-based data encoding for the configuration data as well as the protocol messages.

NETCONF uses a simple Remote Procedure Call (RPC) based mechanism to facilitate communication between a client and a server. The client can be a script or application running as part of a network manager. The server is typically a network device (switch or router). It uses Secure Shell (SSH) as the transport layer across network devices. It uses SSH port number 830 as the default port. The port number is a configurable option.

NETCONF also supports capability discovery and model downloads. Supported models are discovered using the *ietf-netconf-monitoring* model. Revision dates for each model are shown in the capabilities response. Data models are available for optional download from a device using the *get-schema* RPC. You can use these YANG models to understand or export the data model. For more details on NETCONF, see *RFC 6241*.

In releases prior to Cisco IOS XE Fuji 16.8.1, an operational data manager (based on polling) was enabled separately. In Cisco IOS XE Fuji 16.8.1 and later releases, operational data works on platforms running NETCONF (similar to how configuration data works), and is enabled by default. For more information on the components that are enabled for operational data queries or streaming, see the GitHub respository, to view **-oper* in the naming convention.

Restrictions for the NETCONF Protocol

- The NETCONF feature is not supported on a device running dual IOSd configuration or software redundancy.
- If RP addresses from the NETCONF datastore are removed using the **no ip pim rp-address** command, there could be inconsistencies in the datastore, due to parser limitations. To remove RP address entries from the NETCONF datastore, use the RPC.

YANG Model Version 1.1

YANG Version 1.1 is described by the *RFC 7950, The YANG 1.1 Data Modeling Language*. YANG Version 1.1 is a maintenance release of the YANG language that addresses ambiguities and defects in the YANG Version 1.0 specification.

The YANG module in YANG Version 1.1 is advertised through the *ietf-yang-library* instead of the NETCONF hello messages.

The following example shows the NETCONF <*get*> RPC that retrieves a list of all the YANG modules supported by a device:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get>
    <filter>
      <modules-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"/>
    </filter>
  </get>
</rpc>
```

The output of the RPC reply contains a list of all the YANG modules regardless of the YANG version each module uses.

Cisco IOS XE Cupertino 17.7.1 uses the YANG Version 1.0; however, you can still download the YANG Version 1.1 from GitHub at https://github.com/YangModels/yang/tree/master/vendor/cisco/xe.

Alternatively, you can also download the YANG models from the device using the NETCONF *get-schema* operation, and migrate the downloaded models to this version using the *migrate_yang_version.py* script.

The following example shows how to migrate from YANG Version 1.0 to YANG Version 1.1 using the script:

```
migrate_yang_version.py [-h] [--out OUT] path
```

Use the **help** command to view the options available with the script:

```
python migrate_yang_version.py --help
usage: migrate_yang_version.py [-h] [--out OUT] path

positional arguments:
  path        Path to the YANG files

optional arguments:
  -h, --help  show this help message and exit
  --out OUT   Path to the output YANG file
```

The following example shows how to use the *out* argument to move a file from its original location to another folder:

```
python migrate_yang_version.py --out testdir/outdir testdir/indir
```

In the above example, *testdir/outdir* is the directory in which the YANG model Version 1.1 resides or where the output of the script is placed. This directory will be created, if it is not available.

The *testdir/indir* directory is where the YANG model Version 1.0 resides; the input for the script.

After the YANG model Version 1.1 is created, either by downloading it from GitHub or by using the *migrate_yang_version.py* script and compiled on the client application, end-to-end YANG model tests can be executed and validated against Cisco IOS XE devices.

Note The YANG models on the device is still YANG Version 1.0. However; there is no need to change the RPC payload of the client test cases.

For inquiries related to the *migrate_yang_version.py* script or the Cisco IOS XE YANG migration process, send an email to *xe-yang-migration@cisco.com*.

Cisco IOS XE Cupertino 17.8.1 uses YANG Version 1.1. The difference between YANG Version 1.1 and Version 1.0 is documented at https://tools.ietf.org/html/rfc7950#page-10.

YANG Version in Cisco IOS XE Dublin 17.10.1

Cisco-defined YANG models are in YANG Version 1.1 in Cisco IOS XE Dublin 17.10.1 and later releases. You can download this version from GitHub at https://github.com/YangModels/yang/tree/master/vendor/cisco/xe.

In YANG Version 1.1, the critical change that impacts client applications that use NETCONF is in the <*hello*> message content. As per RFC 7950, a server advertises support for YANG 1.1 modules by using the *ietf-yang-library*, instead of listing them as capabilities in the <*hello*> message. We recommend that you use the *ietf-yang-library* to gather the list of supported YANG modules, instead of deriving this list from the <*hello*> message content.

NETCONF RESTCONF IPv6 Support

Data model interfaces (DMIs) support the use of IPv6 protocol. DMI IPv6 support helps client applications to communicate with services that use IPv6 addresses. External facing interfaces will provide dual-stack support; both IPv4 and IPv6.

DMIs are a set of services that facilitate the management of network elements. Application layer protocols such as, NETCONF and RESTCONF access these DMIs over a network.

If IPv6 addresses are not configured, external-facing applications will continue to listen on IPv6 sockets; but these sockets will be unreachable.

Converting IOS Commands to XML

In Cisco IOS XE Cupertino 17.7.1 and later releases, you can automatically translate IOS commands into relevant NETCONF-YANG XML or RESTCONF-JSON request messages. You can analyze the generated configuration messages and familiarize with the Xpaths used in these messages. The generated configuration in the structured format can be used to provision other devices in the network; however, this configuration cannot be modified.

Use the **show running-config** | **format netconf-xml** command or the **show running-config** | **format restconf-json** command to translate IOS commands.

If the **netconf-xml** keyword is selected, the IOS commands are translated into the NETCONF-YANG XML format, and if the **restconf-json** keyword is selected, the IOS commands are translated into the RESTCONF-JSON format.

The translation of IOS commands into a structured format is disabled by default. You must initially configure NETCONF-YANG, and once the data model interfaces (DMIs) are initialized, use the appropriate format option to translate the commands.

The following is sample output from the **show running-config** | **format netconf-xml** command:

```
Device# show running-config | format netconf-xml

<config xmlns="http://tail-f.com/ns/config/1.0">
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <version>17.8</version>
```

```
    <boot-start-marker/>
    <boot>
      <system>
        <flash>
          <flash-list-ordered-by-user>

<flash-leaf>bootflash:c8000v-universalk9.BLD_POLARIS_DEV_LATEST_20211020_005209.SSA.bin</
              flash-leaf>
          </flash-list-ordered-by-user>
        </flash>
      </system>
    </boot>
    <boot-end-marker/>
    <memory>
      <free>
        <low-watermark>
          <processor>64219</processor>
        </low-watermark>
      </free>
    </memory>
    <call-home>
      <contact-email-addr xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">
        sch-smart-licensing@cisco.com</contact-email-addr>
      <tac-profile xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">
        <profile>
          <CiscoTAC-1>
            <active>true</active>
            <destination>
              <transport-method>http</transport-method>
            </destination>
          </CiscoTAC-1>
        </profile>
      </tac-profile>
    </call-home>
    <service>
      <timestamps>
        <debug-config>
          <datetime>
            <msec/>
            <localtime/>
            <show-timezone/>
          </datetime>
        </debug-config>
        <log-config>
          <datetime>
            <msec/>
            <localtime/>
            <show-timezone/>
          </datetime>
        </log-config>
      </timestamps>
      <call-home/>
    </service>
    <platform>
      <console xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
        <output>serial</output>
      </console>
      <qfp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
        <utilization>
          <monitor>
            <load>80</load>
          </monitor>
        </utilization>
      </qfp>
```

```
  <punt-keepalive xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
    <disable-kernel-core>true</disable-kernel-core>
  </punt-keepalive>
</platform>
<hostname>pi-prog-csr1</hostname>
<enable>
  <password>
    <secret>lab</secret>
  </password>
</enable>
<username>
  <name>admin</name>
  <privilege>15</privilege>
  <password>
    <encryption>0</encryption>
    <password>lab</password>
  </password>
</username>
<vrf>
  <definition>
    <name>Mgmt-intf</name>
    <address-family>
      <ipv4>
      </ipv4>
      <ipv6>
      </ipv6>
    </address-family>
  </definition>
</vrf>
<ip>
  <domain>
    <name>cisco</name>
  </domain>
  <forward-protocol>
    <protocol>nd</protocol>
  </forward-protocol>
  <route>
    <ip-route-interface-forwarding-list>
      <prefix>10.0.0.0</prefix>
      <mask>255.255.0.0</mask>
      <fwd-list>
        <fwd>10.45.0.1</fwd>
      </fwd-list>
    </ip-route-interface-forwarding-list>
    <vrf>
      <name>Mgmt-intf</name>
      <ip-route-interface-forwarding-list>
        <prefix>0.0.0.0</prefix>
        <mask>0.0.0.0</mask>
        <fwd-list>
          <fwd>10.104.54.129</fwd>
        </fwd-list>
      </ip-route-interface-forwarding-list>
    </vrf>
  </route>
  <ssh>
    <ssh-version>2</ssh-version>
  </ssh>
  <tftp>
    <source-interface>
      <GigabitEthernet>1</GigabitEthernet>
    </source-interface>
    <blocksize>8192</blocksize>
  </tftp>
```

```
  <http xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-http">
    <authentication>
      <local/>
    </authentication>
    <server>true</server>
    <secure-server>true</secure-server>
  </http>
</ip>
<ipv6>
  <unicast-routing/>
</ipv6>
<interface>
  <GigabitEthernet>
    <name>1</name>
    <vrf>
      <forwarding>Mgmt-intf</forwarding>
    </vrf>
    <ip>
      <address>
        <primary>
          <address>10.104.54.222</address>
          <mask>255.255.255.128</mask>
        </primary>
      </address>
    </ip>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>2</name>
    <ip>
      <address>
        <primary>
          <address>9.45.21.231</address>
          <mask>255.255.0.0</mask>
        </primary>
      </address>
    </ip>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>3</name>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>4</name>
    <mop>
```

```
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>5</name>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
</interface>
<control-plane>
</control-plane>
<clock>
  <timezone>
    <zone>IST</zone>
    <hours>5</hours>
    <minutes>30</minutes>
  </timezone>
</clock>
<logging>
  <console-config>
    <console>false</console>
  </console-config>
</logging>
<aaa>
  <new-model xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa"/>
  <authentication xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">
    <login>
      <name>default</name>
      <a1>
        <local/>
      </a1>
    </login>
  </authentication>
  <authorization xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">
    <exec>
      <name>default</name>
      <a1>
        <local/>
      </a1>
    </exec>
  </authorization>
  <common-criteria xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">
    <policy>enable_secret_policy</policy>
    <char-changes>4</char-changes>
    <lower-case>1</lower-case>
    <max-length>127</max-length>
    <min-length>10</min-length>
    <numeric-count>1</numeric-count>
    <upper-case>1</upper-case>
  </common-criteria>
  <session-id xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">common</session-id>
</aaa>
<login>
  <on-success>
    <log>
```

```
        </log>
      </on-success>
    </login>
    <multilink>
      <bundle-name
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ppp">authenticated</bundle-name>
    </multilink>
    <redundancy>
    </redundancy>
    <spanning-tree>
      <extend xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-spanning-tree">
        <system-id/>
      </extend>
    </spanning-tree>
    <subscriber>
      <templating/>
    </subscriber>
    <crypto>
      <pki xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-crypto">
        <certificate>
          <chain>
            <name>SLA-TrustPoint</name>
            <certificate>
              <serial>01</serial>
              <certtype>ca</certtype>
            </certificate>
          </chain>
          <chain>
            <name>TP-self-signed-2685563505</name>
            <certificate>
              <serial>01</serial>
              <certtype>self-signed</certtype>
            </certificate>
          </chain>
        </certificate>
        <trustpoint>
          <id>SLA-TrustPoint</id>
          <enrollment>
            <pkcs12/>
          </enrollment>
          <revocation-check>crl</revocation-check>
        </trustpoint>
        <trustpoint>
          <id>TP-self-signed-2685563505</id>
          <enrollment>
            <selfsigned/>
          </enrollment>
          <revocation-check>none</revocation-check>
          <rsakeypair>
            <key-label>TP-self-signed-2685563505</key-label>
          </rsakeypair>
          <subject-name>cn=IOS-Self-Signed-Certificate-2685563505</subject-name>
        </trustpoint>
      </pki>
    </crypto>
    <license>
      <udi>
        <pid>C8000V</pid>
        <sn>93SHKMJKOC6</sn>
      </udi>
      <boot>
        <level>
          <network-advantage>
            <addon>dna-advantage</addon>
```

```
          </network-advantage>
        </level>
      </boot>
    </license>
    <line>
      <aux>
        <first>0</first>
      </aux>
      <console>
        <first>0</first>
        <exec-timeout>
          <minutes>0</minutes>
          <seconds>0</seconds>
        </exec-timeout>
        <stopbits>1</stopbits>
      </console>
      <vty>
        <first>0</first>
        <last>4</last>
        <exec-timeout>
          <minutes>0</minutes>
          <seconds>0</seconds>
        </exec-timeout>
        <password>
          <secret>lab</secret>
        </password>
        <transport>
          <input>
            <all/>
          </input>
          <output>
            <all/>
          </output>
        </transport>
      </vty>
      <vty>
        <first>5</first>
        <last>31</last>
        <transport>
          <input>
            <all/>
          </input>
          <output>
            <all/>
          </output>
        </transport>
      </vty>
    </line>
    <diagnostic xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-diagnostics">
      <bootup>
        <level>minimal</level>
      </bootup>
    </diagnostic>
  </native>
</config>
pi-prog-csr1#
pi-prog-csr1#
pi-prog-csr1#show running-config | format restconf-json
{
  "data": {
    "Cisco-IOS-XE-native:native": {
      "version": "17.8",
      "boot-start-marker": [null],
      "boot": {
```

```
        "system": {
          "flash": {
            "flash-list-ordered-by-user": [
              {
                "flash-leaf":
"bootflash:c8000v-universalk9.BLD_POLARIS_DEV_LATEST_20211020_005209.SSA.bin"
              }
            ]
          }
        }
      },
      "boot-end-marker": [null],
      "memory": {
        "free": {
          "low-watermark": {
            "processor": 64219
          }
        }
      },
      "call-home": {
        "Cisco-IOS-XE-call-home:contact-email-addr": "sch-smart-licensing@cisco.com",
        "Cisco-IOS-XE-call-home:tac-profile": {
          "profile": {
            "CiscoTAC-1": {
              "active": true,
              "destination": {
                "transport-method": "http"
              }
            }
          }
        }
      },
      "service": {
        "timestamps": {
          "debug-config": {
            "datetime": {
              "msec": [null],
              "localtime": [null],
              "show-timezone": [null]
            }
          },
          "log-config": {
            "datetime": {
              "msec": [null],
              "localtime": [null],
              "show-timezone": [null]
            }
          }
        },
        "call-home": [null]
      },
      "platform": {
        "Cisco-IOS-XE-platform:console": {
          "output": "serial"
        },
        "Cisco-IOS-XE-platform:qfp": {
          "utilization": {
            "monitor": {
              "load": 80
            }
          }
        },
        "Cisco-IOS-XE-platform:punt-keepalive": {
          "disable-kernel-core": true
```

```
  }
},
"hostname": "pi-prog-csr1",
"enable": {
  "password": {
    "secret": "lab"
  }
},
"username": [
  {
    "name": "admin",
    "privilege": 15,
    "password": {
      "encryption": "0",
      "password": "lab"
    }
  }
],
"vrf": {
  "definition": [
    {
      "name": "Mgmt-intf",
      "address-family": {
        "ipv4": {
        },
        "ipv6": {
        }
      }
    }
  ]
},
"ip": {
  "domain": {
    "name": "cisco"
  },
  "forward-protocol": {
    "protocol": "nd"
  },
  "route": {
    "ip-route-interface-forwarding-list": [
      {
        "prefix": "10].0.0.0",
        "mask": "255.255.0.0",
        "fwd-list": [
          {
            "fwd": "9.45.0.1"
          }
        ]
      }
    ],
    "vrf": [
      {
        "name": "Mgmt-intf",
        "ip-route-interface-forwarding-list": [
          {
            "prefix": "0.0.0.0",
            "mask": "0.0.0.0",
            "fwd-list": [
              {
                "fwd": "10.104.54.129"
              }
            ]
          }
        ]
```

```
        }
      ]
    },
    "ssh": {
      "ssh-version": "2"
    },
    "tftp": {
      "source-interface": {
        "GigabitEthernet": "1"
      },
      "blocksize": 8192
    },
    "Cisco-IOS-XE-http:http": {
      "authentication": {
        "local": [null]
      },
      "server": true,
      "secure-server": true
    }
  },
  "ipv6": {
    "unicast-routing": [null]
  },
  "interface": {
    "GigabitEthernet": [
      {
        "name": "1",
        "vrf": {
          "forwarding": "Mgmt-intf"
        },
        "ip": {
          "address": {
            "primary": {
              "address": "10.104.54.222",
              "mask": "255.255.255.128"
            }
          }
        },
        "mop": {
          "enabled": false,
          "sysid": false
        },
        "Cisco-IOS-XE-ethernet:negotiation": {
          "auto": true
        }
      },
      {
        "name": "2",
        "ip": {
          "address": {
            "primary": {
              "address": "10.45.21.231",
              "mask": "255.255.0.0"
            }
          }
        },
        "mop": {
          "enabled": false,
          "sysid": false
        },
        "Cisco-IOS-XE-ethernet:negotiation": {
          "auto": true
        }
      },
```

```
      {
        "name": "3",
        "mop": {
          "enabled": false,
          "sysid": false
        },
        "Cisco-IOS-XE-ethernet:negotiation": {
          "auto": true
        }
      },
      {
        "name": "4",
        "mop": {
          "enabled": false,
          "sysid": false
        },
        "Cisco-IOS-XE-ethernet:negotiation": {
          "auto": true
        }
      },
      {
        "name": "5",
        "mop": {
          "enabled": false,
          "sysid": false
        },
        "Cisco-IOS-XE-ethernet:negotiation": {
          "auto": true
        }
      }
    ]
  },
  "control-plane": {
  },
  "clock": {
    "timezone": {
      "zone": "IST",
      "hours": 5,
      "minutes": 30
    }
  },
  "logging": {
    "console-config": {
      "console": false
    }
  },
  "aaa": {
    "Cisco-IOS-XE-aaa:new-model": [null],
    "Cisco-IOS-XE-aaa:authentication": {
      "login": [
        {
          "name": "default",
          "a1": {
            "local": [null]
          }
        }
      ]
    },
    "Cisco-IOS-XE-aaa:authorization": {
      "exec": [
        {
          "name": "default",
          "a1": {
            "local": [null]
```

```
        }
      }
    ]
  },
  "Cisco-IOS-XE-aaa:common-criteria": [
    {
      "policy": "enable_secret_policy",
      "char-changes": 4,
      "lower-case": 1,
      "max-length": 127,
      "min-length": 10,
      "numeric-count": 1,
      "upper-case": 1
    }
  ],
  "Cisco-IOS-XE-aaa:session-id": "common"
},
"login": {
  "on-success": {
    "log": {
    }
  }
},
"multilink": {
  "Cisco-IOS-XE-ppp:bundle-name": "authenticated"
},
"redundancy": {
},
"spanning-tree": {
  "Cisco-IOS-XE-spanning-tree:extend": {
    "system-id": [null]
  }
},
"subscriber": {
  "templating": [null]
},
"crypto": {
  "Cisco-IOS-XE-crypto:pki": {
    "certificate": {
      "chain": [
        {
          "name": "SLA-TrustPoint",
          "certificate": [
            {
              "serial": "01",
              "certtype": "ca"
            }
          ]
        },
        {
          "name": "TP-self-signed-2685563505",
          "certificate": [
            {
              "serial": "01",
              "certtype": "self-signed"
            }
          ]
        }
      ]
    },
    "trustpoint": [
      {
        "id": "SLA-TrustPoint",
        "enrollment": {
```

```
          "pkcs12": [null]
        },
        "revocation-check": ["crl"]
      },
      {
        "id": "TP-self-signed-2685563505",
        "enrollment": {
          "selfsigned": [null]
        },
        "revocation-check": ["none"],
        "rsakeypair": {
          "key-label": "TP-self-signed-2685563505"
        },
        "subject-name": "cn=IOS-Self-Signed-Certificate-2685563505"
      }
    ]
  }
},
"license": {
  "udi": {
    "pid": "C8000V",
    "sn": "93SHKMJKOC6"
  },
  "boot": {
    "level": {
      "network-advantage": {
        "addon": "dna-advantage"
      }
    }
  }
},
"line": {
  "aux": [
    {
      "first": "0"
    }
  ],
  "console": [
    {
      "first": "0",
      "exec-timeout": {
        "minutes": 0,
        "seconds": 0
      },
      "stopbits": "1"
    }
  ],
  "vty": [
    {
      "first": 0,
      "last": 4,
      "exec-timeout": {
        "minutes": 0,
        "seconds": 0
      },
      "password": {
        "secret": "lab"
      },
      "transport": {
        "input": {
          "all": [null]
        },
        "output": {
          "all": [null]
```

```
            }
          }
        },
        {
          "first": 5,
          "last": 31,
          "transport": {
            "input": {
              "all": [null]
            },
            "output": {
              "all": [null]
            }
          }
        }
      ]
    },
    "Cisco-IOS-XE-diagnostics:diagnostic": {
      "bootup": {
        "level": "minimal"
      }
    }
  }
 }
}
```

NETCONF Global Session Lock

The NETCONF protocol provides a set of operations to manage device configurations and retrieve device state information. NETCONF supports a global lock, and the ability to kill non-responsive sessions are introduced in NETCONF.

To ensure consistency and prevent conflicting configurations through multiple simultaneous sessions, the owner of the session can lock the NETCONF session. The NETCONF lock RPC locks the configuration parser and the running configuration database. All other NETCONF sessions (that do not own the lock) cannot perform edit operations; but can perform read operations. These locks are intended to be short-lived and allow the owner to make changes without interaction with other NETCONF clients, non-NETCONF clients (such as, SNMP and CLI scripts), and human users.

A global lock held by an active session is revoked when the associated session is killed. The lock gives the session holding the lock exclusive write access to the configuration. When a configuration change is denied due to a global lock, the error message will specify that a NETCONF global lock is the reason the configuration change has been denied.

The <lock> operation takes a mandatory parameter, <target> that is the name of the configuration datastore that is to be locked. When a lock is active, the <edit-config> and <copy-config> operations are not allowed.

If the **clear configuration lock** command is specified while a NETCONF global lock is being held, a full synchronization of the configuration is scheduled and a warning syslog message is produced. This command clears only the parser configuration lock.

The following is a sample RPC that shows the <lock> operation:

```
<rpc message-id="101"
          xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
       <lock>
         <target>
           <running/>
         </target>
```

```
        </lock>
</rpc>
```

NETCONF Kill Session

During a session conflict or client misuse of the global lock, NETCONF sessions can be monitored via the **show netconf-yang sessions** command, and non-responsive sessions can be cleared using the **clear netconf-yang session** command. The **clear netconf-yang session** command clears both the NETCONF lock and the configuration lock.

A <kill-session> request will force a NETCONF session to terminate. When a NETCONF entity receives a <kill-session> request for an open session, it stops all operations in process, releases all locks and resources associated with the session, and closes any associated connections.

A <kill-session> request requires the session-ID of the NETCONF session that is to be terminated. If the value of the session-ID is equal to the current session ID, an invalid-value error is returned. If a NETCONF session is terminated while its transaction is still in progress, the data model infrastructure will request a rollback, apply it to the network element, and trigger a synchronization of all YANG models.

If a session kill fails, and a global lock is held, enter the **clear configuration lock** command via the console or vty. At this point, the data models can be stopped and restarted.

NETCONF-YANG SSH Server Support

NETCONF-YANG uses the IOS Secure Shell (SSH) Rivest, Shamir, and Adleman (RSA) public keys to authenticate users as an alternative to password-based authentication.

For public-key authentication to work on NETCONF-YANG, the IOS SSH server must be configured. To authenticate users to the SSH server, use one of the RSA keys configured by using the **ip ssh pubkey-chain** and **user** commands.

NACM is a group-based access control mechanism. When users are authenticated, they are automatically placed in an NACM privilege group based on their configured privilege level. Users can also be manually placed in other user-defined groups. The default privilege level is 1. There are 16 privilege levels, PRIV00 to PRIV15.

If a user authenticates via the public-key; but does not have a corresponding Authentication, Authorization, and Accounting (AAA) configuration, this user is rejected. If a user authenticates via a public-key; but the AAA configuration for NETCONF is using a AAA source other than the local, this user is also rejected. Local and TACACS+ AAA authorization are supported.

Token-based RESTCONF authentication is not supported. SSH user certificates are not supported.

NETCONF SSH Algorithms

The NETCONF-SSH server configuration file contains the list of all supported algorithms. From Cisco IOS XE Dublin 17.12.1, you can enable or disable these algorithms at runtime by using commands or YANG models.

Use the **netconf-yang ssh server algorithm** {**encryption** | **kex** | **mac** | **hostkey**} command to enable the algorithms. Use the **no** form of this command to disable the algorithms. Also, the output of the **show netconf-yang status** command will display the list of configured algorithms.

Users can also enable or disable the NETCONF-SSH algorithms through YANG models. The following is a sample of the NETCONF request for the corresponding model:

```
<yang-interfaces-cfg-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-yang-interfaces-cfg">

      <ssh-server>
            <kex-algorithms>
                  <dh-group14-sha1>false</dh-group14-sha1>
            </kex-algorithms>
            <macs>
                  <hmac-sha1>false</hmac-sha1>
            </macs>
            <ciphers>
                  <aes128-cbc>true</aes128-cbc>
            </ciphers>
            <hostkey-algorithms>
                  <rsa-sha2-256>true </rsa-sha2-256>
            </hostkey-algorithms>
      </ssh-server>
</yang-interfaces-cfg-data>
```

Named Method List

gNMI, NETCONF, and RESTCONF uses the Cisco IOS authentication, authorization, and accounting (AAA) server to authenticate and authorize an user. Prior to the introduction of the named method-list, only the *default* method list was supported for authentication and authorization. This meant that the administrator could not use a custom method-list name for authentication and authorization.

With the introduction of the Named Method List feature, it is possible to use a custom method-list name for gNMI, NETCONF, and RESTCONF authentication and authorization, without changing the existing AAA configuration of a device. You can use the **yang-interfaces aaa {authentication | authorization} method-list** *named-method-list* command to create a custom method-list. Named method lists can provide multiple authentication and authorization options.

A method list is a named list that describes the authentication and authorization methods to be queried, such as, AAA, Lightweight Directory Access Protocol (LDAP), RADIUS, or TACACS+. Method lists define the method and the sequence in which authorization is performed. Method lists enable one or more security protocols for authentication and authorization, ensuring that a backup system is available in case of a failure.

The following is a sample NETCONF RPC that displays the named method list:

```
<edit-config>
<target><running/></target>
<config>
  <yang-interfaces-cfg-data xmlns=http://cisco.com/ns/yang/Cisco-IOS-XE-yang-interfaces-cfg>

    <aaa>
      <authn>
        <login-method-list>test</login-method-list>
      </authn>
      <authz>
        <exec-method-list>test</exec-method-list>
      </authz>
    </aaa>
  </yang-interfaces-cfg-data>
</config>
</edit-config>
```

Candidate Configuration Support

The Candidate Configuration feature enables support for candidate capability by implementing RFC 6241 with a simple commit option.

The candidate datastore provides a temporary work space in which a copy of the device's running configuration is stored. You can create and modify the running configuration before committing the running configuration to the device. Candidate capability is indicated by the following NETCONF capability: urn:ietf:params:netconf:capability:candidate:1.0. This NETCONF capability indicates that the device supports the candidate datastore.

This is a shared data store which enables the user to create, add, delete and make changes to the device configuration without affecting the running configuration on the device. A commit operation pushes the configuration from the candidate to the running configuration on the device. When the candidate data store is enabled, the running data store is *not* writable through NETCONF sessions, and all configurations get committed only through the candidate. In other words, the writable-running NETCONF capability is not enabled with the candidate configuration.

Note It must be kept in mind that candidate datastore is a shared data store. Multiple NETCONF sessions can modify it contents simultaneously. Therefore, it is important to lock the datastore before modifying its contents, to prevent conflicting commits that can eventually lead to the loss of any configuration changes.

NETCONF Operations on Candidate

The following operations can be performed on the candidate data store.

Note The information in this section has been referenced from section 8.3.4 of RFC 6241. Please refer to the RFC for more details and the exact RPCs.

Lock

A <lock> RPC is used to lock the target data store. This prevents others users from modifying the configuration in the locked data store. Both candidate and running data can be locked through the lock operation.

Note Locking the candidate datastore does not affect the Cisco IOS config lock or the running configuration lock and vice versa.

Commit

A <commit> RPC, copies the candidate configuration to the device's running configuration. A *commit* operation must be performed after you have updated the candidate configuration to push the configuration to the device.

If either the running or the candidate datastore is locked by another NETCONF session, the <commit> RPC will fail with an RPC error reply. The <error-tag> should be <in-use> and <error-info> should have the session ID of the NETCONF session holding the lock. You can also lock the running configuration by using the global

lock by entering the conf t lock mode, but, the commit operation will fail with an RPC error reply, with error-tag value <in-use> and the session-id will be “0”.

Edit-config

The candidate configuration can be used as a target for the edit-config operation to modify a configuration. You can change the candidate configuration without affecting the running configuration on the device.

Discard

To remove the changes made to the candidate configuration, perform a discard operation to revert the candidate configuration to running configuration.

If contents of the candidate datastore are modified by NETCONF session A, and session B tries to lock the candidate datastore, the lock fails. NETCONF session B must perform a <discard> operation to remove any outstanding configuration changes on the candidate datastore from other NETCONF sessions before locking a candidate.

Unlock

After working on candidate configuration, such as, lock, edit-config, or commit operations, you can unlock the datastore, by specifying candidate as target in the unlock RPC. The candidate datastore is now available for all operations in other sessions.

If a failure occurs with outstanding changes to the candidate datastore, it can be challenging to recover the configuration, and may create problems for other sessions. To avoid any issues, outstanding changes must be discarded when the lock is released—either implicitly on “NETCONF session failure” or explicitly by using the unlock operation.

Get-config, Copy-config, Validate

The candidate datastore can be used as a source or target for any of the get-config, copy-config or validate config operations. If you do not want to commit the changes in the candidate datastore to the device; but only to validate the configuration, you ca nuse the <validate> RPC followed by a discard operation.

Modifying the Candidate Datastore

The following diagram explains the recommended best practice when modifying the device configuration through candidate datastore:

Figure 4: Modifying Candidate Datastore Steps

1. Lock the running datastore.
2. Lock the candidate datastore.
3. Make modifications to the candidate configuration through edit-config RPCs with the target candidate.

4. Commit the candidate configuration to the running configuration.
5. Unlock the candidate and running datastores.

Confirmed Candidate Configuration Commit

The candidate configuration supports the confirmed commit capability. This implementation is as specified in RFC 6241 for the confirmed commit capability which, when issued, sets the running configuration to the current contents of the candidate configuration and starts a confirmed commit timer. The confirmed commit operation will be rolled back if the commit is not issued within the timeout period. The default timeout period is 600 seconds or 10 minutes.

When you commit the candidate configuration, you can require an explicit confirmation for the commit to become permanent. The confirmed commit operation is useful for verifying that a configuration change works correctly and does not prevent management access to the device. If the change prevents access or causes other errors, the automatic rollback to the previous configuration restores access after the rollback deadline passes. If the commit is not confirmed within the specified amount of time,by default, the device automatically retrieves and commits (rolls back to) the previously committed configuration.

Note RESTCONF does not support confirmed commit.

In a NETCONF session, to commit the candidate configuration and to explicitly confirm the commit to become permanent, a client application encloses the empty <confirmed/> tag in the <commit> and <rpc> tag elements:

```
<rpc>
  <commit>
    <confirmed/>
  </commit>
</rpc>
```

The following sample RPC shows how to change the default rollback timer:

```
<rpc>
    <commit>
        <confirmed/>
        <confirm-timeout>nnn</confirm-timeout> !nnn is the rollback-delay in seconds.
    </commit>
</rpc>
```

The following sample RPC shows that the NETCONF server confirms that the candidate configuration is committed temporarily:

```
<rpc-reply xmlns="URN" xmlns:nc="URL">
    <ok/>
</rpc-reply>
```

If the NETCONF server cannot commit the candidate configuration, the <rpc-reply> element will enclose an <rpc-error> element explaining the reason for the failure. The most common causes are semantic or syntactic errors in the candidate configuration.

To delay the rollback to a time later than the current rollback timer, the client application sends a <confirmed/> tag inside a <commit> tag element again before the deadline passes. Optionally, it includes the <confirm-timeout> element to specify how long to delay the next rollback. The client application can delay the rollback indefinitely by sending the <confirmed/> tag repeatedly.

To commit the configuration permanently, the client application sends the <commit/> tag enclosed in an <rpc> tag element before the rollback deadline passes. The rollback is canceled and the candidate configuration is committed immediately. If the candidate configuration is the same as the temporarily committed configuration, the temporarily committed configuration is recommitted.

If another application uses the <kill-session/> tag element to terminate this application's session while a confirmed commit is pending (this application has committed changes but not yet confirmed them), the NETCONF server that is using this session restores the configuration to its state before the confirmed commit instruction was issued.

The candidate datastore is disabled by using the **no netconf-yang feature candidate-datastore** command. Because the candidate datastore confirmed commit is enabled when the candidate datastore is enabled, the confirmed commit is disabled when the candidate datastore is disabled. All sessions in progress are terminated, and the confd program is restarted.

Candidate Support Configuration

The candidate datastore functionality can be enabled by using the **netconf-yang feature candidate-datastore** command. When the datastore state changes from running to candidate or back, a warning message is displayed, notifying the user that a restart of NETCONF or RESTCONF will occur in order for the change to take effect.

If the selection of the candidate or running datastore is specified in the configuration when a NETCONF-YANG or RESTCONF confd process starts, a warning message appears as shown below:

```
Device(config)# netconf-yang feature candidate-datastore

netconf-yang initialization in progress - datastore transition not allowed, please try again
 after 30 seconds
```

If the selection of the candidate or running datastore is made after the NETCONF-YANG or RESTCONF confd process starts, the following apply:

- If the **netconf-yang feature candidate-datastore** command is configured, the command enables the candidate datastore and prints the following warning:

  ```
  "netconf-yang and/or restconf is transitioning from running to candidate netconf-yang
  and/or restconf will now be restarted,
  and any sessions in progress will be terminated".
  ```

- If the **netconf-yang feature candidate-datastore** command is removed, the command disables the candidate datastore, enables the running datastore and prints the following warning:

  ```
  netconf-yang and/or restconf is transitioning from candidate to running netconf-yang
  and/or restconf will now be restarted,
  and any sessions in progress will be terminated".
  ```

- When NETCONF-YANG or RESTCONF are restarted, sessions in progress will be lost.

Side-Effect Synchronization of the Configuration Database

During configuration changes in the data model interface (DMI), a partial synchronization of the changes that are triggered when a command or RPC is configured happens. This is called the side-effect synchronization,

and it reduces the synchronization time and NETCONF downtime. Prior to the side-effect synchronization, any configuration change used to trigger a time-consuming full synchronization of the configuration database.

The side-effect synchronization is enabled by the **netconf-yang feature side-effect-sync** command.

Some commands, when they are configured, triggers changes in some already configured commands. For example, the following is the configuration on a device before the NETCONF edit-config RPC is configured:

```
hostname device123
```

The NETCONF edit-config RPC:

```
<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
 <hostname xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="delete"/>
</native>
```

The following is the configuration on the device after the NETCONF edit-config RPC is configured:

```
hostname Switch
```

Here, the side-effect of the NETCONF edit-config RPC is a change to the running configuration that is not directly intended by the RPC. The edit-config request is supposed to delete the host name, but instead the hostname is changed back to Switch. The side-effect synchronization does a synchronization of this configuration change to the NETCONF database without synchronizing the entire configuration, thereby improving performance.

The side-effect synchronization is based on the CLI-mode tree concept, where the commands are maintained with modes and submodes structure. This CLI-mode tree data structure consists of three main nodes:

- Same-Level Node: This node points to the list of CLI nodes that belongs to the same parent and on the same level.
- Parent Node: This node points to the CLI nodes parent, its mode, and submode node.
- Child Node: This node points to the child CLI; the CLI under the current mode or submode. If the node has multiple child nodes then those child nodes are linked as part of the same-level node pointers.

How to Configure the NETCONF Protocol

NETCONF-YANG uses the primary trustpoint of a device. If a trustpoint does not exist, when NETCONF-YANG is configured, it creates a self-signed trustpoint. For more information, see the Public Key Infrastructure Configuration Guide, Cisco IOS XE Gibraltar 16.10.x.

Providing Privilege Access to Use NETCONF

To start working with NETCONF APIs, you must be a user with privilege level 15.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **username** *name* **privilege** *level* **password** *password*
4. **aaa new-model**
5. **aaa authentication login default local**

6. **aaa authorization exec default local**
7. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**username** *name* **privilege** *level* **password** *password* **Example:** `Device(config)# username example-name privilege 15 password example_password`	Establishes a user name-based authentication system. Configure the following keywords: • **privilege** *level*: Sets the privilege level for the user. For the NETCONF protocol, it must be 15. • **password** *password*: Sets a password to access the CLI view.
Step 4	**aaa new-model** **Example:** `Device(config)# aaa new-model`	(Optional) Enables authorisation, authentication, and accounting (AAA). If the **aaa new-model** command is configured, AAA authentication and authorization is required.
Step 5	**aaa authentication login default local** **Example:** `Device(config)# aaa authentication login default local`	Sets the login authentication to use the local username database. **Note** Prior to Cisco IOS XE Cupertino 17.9.1, only the default AAA authentication login method is supported for the NETCONF protocol. From Cisco IOS XE Cupertino 17.9.1, named method-list is supported. • For a remote AAA server, replace *local* with your AAA server. The **default** keyword applies the local user database authentication to all ports.
Step 6	**aaa authorization exec default local** **Example:** `Device(config)# aaa authorization exec default local`	Configures user AAA authorization, check the local database, and allows the user to run an EXEC shell. **Note**

	Command or Action	Purpose
		Prior to Cisco IOS XE Cupertino 17.9.1, only the default AAA authentication login method is supported for the NETCONF protocol. From Cisco IOS XE Cupertino 17.9.1, named method-list is supported. • For a remote AAA server, replace *local* with your AAA server. • The **default** keyword applies the local user database authentication to all ports.
Step 7	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring NETCONF-YANG

If the legacy NETCONF protocol is enabled on your device, the RFC-compliant NETCONF protocol does not work. Disable the legacy NETCONF protocol by using the **no netconf legacy** command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **netconf-yang**
4. **netconf-yang feature candidate-datastore**
5. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**netconf-yang** **Example:** `Device (config)# netconf-yang`	Enables the NETCONF interface on your network device. **Note** After the initial enablement through the CLI, network devices can be managed subsequently through a model

	Command or Action	Purpose
		based interface. The complete activation of model-based interface processes may require up to 90 seconds.
Step 4	**netconf-yang feature candidate-datastore** **Example:** `Device(config)# netconf-yang feature candidate-datastore`	Enables candidate datastore.
Step 5	**exit** **Example:** `Device (config)# exit`	Exits global configuration mode.

Configuring NETCONF Options

Configuring SNMP

Enable the SNMP Server in IOS to enable NETCONF to access SNMP MIB data using YANG models generated from supported MIBs, and to enable supported SNMP traps in IOS to receive NETCONF notifications from the supported traps.

Perform the following steps:

SUMMARY STEPS

1. Enable SNMP features in IOS.
2. After NETCONF-YANG starts, enable SNMP Trap support by sending the following RPC <edit-config> message to the NETCONF-YANG port.
3. Send the following RPC message to the NETCONF-YANG port to save the running configuration to the startup configuration.

DETAILED STEPS

Procedure

Step 1 Enable SNMP features in IOS.

Example:

```
configure terminal
logging history debugging
logging snmp-trap emergencies
logging snmp-trap alerts
logging snmp-trap critical
logging snmp-trap errors
logging snmp-trap warnings
logging snmp-trap notifications
logging snmp-trap informational
logging snmp-trap debugging
!
```

```
snmp-server community public RW
snmp-server trap link ietf
snmp-server enable traps snmp authentication linkdown linkup
snmp-server enable traps syslog
snmp-server manager
exit
```

Step 2 After NETCONF-YANG starts, enable SNMP Trap support by sending the following RPC <edit-config> message to the NETCONF-YANG port.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <netconf-yang xmlns="http://cisco.com/yang/cisco-self-mgmt">
        <cisco-ia xmlns="http://cisco.com/yang/cisco-ia">
          <snmp-trap-control>
            <trap-list>
              <trap-oid>1.3.6.1.4.1.9.9.41.2.0.1</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.3</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.4</trap-oid>
            </trap-list>
          </snmp-trap-control>
        </cisco-ia>
      </netconf-yang>
    </config>
  </edit-config>
</rpc>
```

Step 3 Send the following RPC message to the NETCONF-YANG port to save the running configuration to the startup configuration.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <cisco-ia:save-config xmlns:cisco-ia="http://cisco.com/yang/cisco-ia"/>
</rpc>
```

Configuring the SSH Server to Perform RSA-Based User Authentication

Perform this task to configure the SSH public key for NETCONF-YANG to authenticate users.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip ssh pubkey-chain**
4. **username** *username*

5. **key-string**
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip ssh pubkey-chain** **Example:** `Device(config)# ip ssh pubkey-chain`	Configures SSH-RSA keys for user and server authentication on the SSH server and enters public-key configuration mode. • The user authentication is successful if the RSA public key stored on the server is verified with the public or the private key pair stored on the client.
Step 4	**username** *username* **Example:** `Device(conf-ssh-pubkey)# username user1`	Configures the SSH username and enters public-key user configuration mode.
Step 5	**key-string** **Example:** `Device(conf-ssh-pubkey-user)# key-string`	Specifies the RSA public key of the remote peer and enters public-key data configuration mode. **Note** You can obtain the public key value from an open SSH client; that is, from the .ssh/id_rsa.pub file.
Step 6	**end** **Example:** `Device(conf-ssh-pubkey-data)# end`	Exits public-key data configuration mode and returns to privileged EXEC mode. • Use **no hostname** command to return to the default host.

Configuring a Named Method List

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **yang-interfaces aaa authentication method-list** *named-method-list*
4. **yang-interfaces aaa authorization method-list** *named-method-list*
5. **exit**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**yang-interfaces aaa authentication method-list** *named-method-list* **Example:** `Device(config)# yang-interfaces aaa authentication method-list authn-method`	Configures a named method-list for authentication.
Step 4	**yang-interfaces aaa authorization method-list** *named-method-list* **Example:** `Device(config)# yang-interfaces aaa authorization method-list authr-method`	Configures a named method-list for authorization.
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Verifying the NETCONF Protocol Configuration Through the CLI

Use the following commands to verify your NETCONF configuration.

SUMMARY STEPS

1. **show netconf-yang datastores**
2. **show netconf-yang sessions**
3. **show netconf-yang sessions detail**
4. **show netconf-yang diagnostics summary**
5. **show netconf-yang statistics**
6. **show platform software yang-management process**

DETAILED STEPS

Procedure

Step 1 **show netconf-yang datastores**

Displays information about NETCONF-YANG datastores.

Example:

```
Device# show netconf-yang datastores

Device# show netconf-yang datastores
Datastore Name : running
Globally Locked By Session : 42
Globally Locked Time : 2018-01-15T14:25:14-05:00
```

Step 2 **show netconf-yang sessions**

Displays information about NETCONF-YANG sessions.

Example:

```
Device# show netconf-yang sessions

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore
Number of sessions : 10
session-id transport username source-host global-lock
--------------------------------------------------------------------------------
40 netconf-ssh admin 10.85.70.224 None
42 netconf-ssh admin 10.85.70.224 None
44 netconf-ssh admin 10.85.70.224 None
46 netconf-ssh admin 10.85.70.224 None
48 netconf-ssh admin 10.85.70.224 None
50 netconf-ssh admin 10.85.70.224 None
52 netconf-ssh admin 10.85.70.224 None
54 netconf-ssh admin 10.85.70.224 None
56 netconf-ssh admin 10.85.70.224 None
58 netconf-ssh admin 10.85.70.224 None
```

Step 3 **show netconf-yang sessions detail**

Displays detailed information about NETCONF-YANG sessions.

Example:

```
Device# show netconf-yang sessions detail

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions     : 1

session-id             : 19
transport              : netconf-ssh
username               : admin
source-host            : 2001:db8::1
login-time             : 2018-10-26T12:37:22+00:00
```

```
in-rpcs                : 0
in-bad-rpcs            : 0
out-rpc-errors         : 0
out-notifications      : 0
global-lock            : None
```

Step 4 **show netconf-yang diagnostics summary**

Displays a summary of the NETCONF-YANG diagnostic information.

Example:

```
Device# show netconf-yang diagnostics summary

Diagnostic Debugging is ON
Diagnostic Debugging Level: Maximum
Total Log Size (bytes): 20097
Total Transactions: 1
message username session-id transaction-id start-time        end-time          log size
-------------------------------------------------------------------------------------------
1       admin    35          53             03/12/21 14:31:03 03/12/21 14:31:04 20097
```

Step 5 **show netconf-yang statistics**

Displays information about NETCONF-YANG statistics.

Example:

```
Device# show netconf-yang statistics

netconf-start-time : 2018-01-15T12:51:14-05:00
in-rpcs : 0
in-bad-rpcs : 0
out-rpc-errors : 0
out-notifications : 0
in-sessions : 10
dropped-sessions : 0
in-bad-hellos : 0
```

Step 6 **show platform software yang-management process**

Displays the status of the software processes required to support NETCONF-YANG.

Example:

```
Device# show platform software yang-management process

confd            : Running
nesd             : Running
syncfd           : Running
ncsshd           : Running
dmiauthd         : Running
vtyserverutild   : Running
opdatamgrd       : Running
nginx            : Running
ndbmand          : Running
```

Note
The process *nginx* runs if **ip http secure-server** or **ip http server** is configured on the device. This process is not required to be in the *running* state for NETCONF to function properly. However, the *nginx* process is required for RESTCONF.

Table 13: show platform software yang-management process Field Descriptions

Field	Description
confd	Configuration daemon
nesd	Network element synchronizer daemon
syncfd	Sync from daemon
ncsshd	NETCONF Secure Shell (SSH) daemon
dmiauthd	Device management inteface (DMI) authentication daemon
vtyserverutild	VTY server util daemon
opdatamgrd	Operational Data Manager daemon
nginx	NGINX web server
ndbmand	NETCONF database manager

Example: Named Method List

Along with the default method-list, you can enable multiple authentication or authorization options with the named method-list. Method lists enable one or more security protocols to be used for authorization. The method lists are processed serially by the Cisco IOS software. If the first configured method-list fails, the next one is processed. This process continues until a successful authentication or authorization, or until all configured methods are exhausted.

The following example shows how to configure named method-lists for NETCONF AAA:

```
Device> enable
Device# configure terminal
Device(config)# netconf-yang
Device(config)# yang-interfaces aaa authentication method-list netconf-authn
Device(config)# yang-interfaces aaa authorization method-list netconf-authr
Device(config)# end
```

Displaying NETCONF-YANG Diagnostics Through RPCs

You can either use the **show netconf-yang diagnostics** command or the following RPCs to view the diagnostics information.

The following is a sample RPC that enables NETCONF-YANG diagnostics, and the RPC response received from the host:

```
#308
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:b0f45ac0-3fe2-4e1d-a3a1-f57985965be6">
```

```
  <enable-netconf-diag xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-rpc">
    <diag-level>diag-maximum</diag-level>
  </enable-netconf-diag>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:b0f45ac0-3fe2-4e1d-a3a1-f57985965be6"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

The following is a sample RPC that shows the current status and the RPC response received from the host:

```
#294
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:c6c986ac-fc44-45e2-9390-f8a5968dc8d4">
  <nc:get>
    <nc:filter>
      <netconf-diag-oper-data
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-oper"/>
    </nc:filter>
  </nc:get>
</nc:rpc>

#

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:c6c986ac-fc44-45e2-9390-f8a5968dc8d4"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
   <netconf-diag-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-oper">

      <diag-summary>
        <level>diag-maximum</level>
        <log-size>0</log-size>
        <trans-count>0</trans-count>
      </diag-summary>
    </netconf-diag-oper-data>
  </data>
</rpc-reply>
```

The following is a sample RPC to change the host name and the RPC response received from the host:

```
#
#364
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:f3005ee6-8a11-4146-b616-dd95a92b97d1">
  <nc:edit-config>
    <nc:target>
      <nc:running/>
    </nc:target>
```

```
    <nc:config>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <hostname>new-ott-c9300-35</hostname>
      </native>
    </nc:config>
  </nc:edit-config>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:f3005ee6-8a11-4146-b616-dd95a92b97d1"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

The following is a sample RPC to display the current status and the RPC response received from the host:

```
 #294
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:9bffb8d5-3866-48ef-b59d-0486e508fbc4">
  <nc:get>
    <nc:filter>
      <netconf-diag-oper-data
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-oper"/>
    </nc:filter>
  </nc:get>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:9bffb8d5-3866-48ef-b59d-0486e508fbc4"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <netconf-diag-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-oper">

      <diag-summary>
        <level>diag-maximum</level>
        <log-size>20775</log-size>
        <trans-count>1</trans-count>
      </diag-summary>
      <diag-trans>
        <message>1</message>
        <username>lab</username>
        <session-id>31</session-id>
        <trans-id>50</trans-id>
        <start-time>2021-03-12T14:08:26.830334+00:00</start-time>
        <end-time>2021-03-12T14:08:28.279414+00:00</end-time>
        <log-size>20775</log-size>
      </diag-trans>
    </netconf-diag-oper-data>
  </data>
</rpc-reply>
```

The following is a sample RPC to archive the collected system error messages, and the RPC response from the host:

```
#
#256
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:1dbc795c-f594-4194-a89b-fd4d88446b69">
  <archive-netconf-diag-logs xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-rpc"/>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:1dbc795c-f594-4194-a89b-fd4d88446b69"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <log-file xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-rpc">
    bootflash:netconf-yang-diag.20210312141009.tar.gz</log-file>


</rpc-reply>
```

The following is a sample RPC that disables NETCONF-YANG diagnostics, and the RPC response received from the host:

```
#309
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:d253a313-4aec-42bc-80a2-672e9bb9ad56">
  <enable-netconf-diag xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-rpc">
    <diag-level>diag-disabled</diag-level>
  </enable-netconf-diag>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:d253a313-4aec-42bc-80a2-672e9bb9ad56"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Additional References for NETCONF Protocol

Related Documents

Related Topic	Document Title
YANG data models for various release of IOS-XE, IOS-XR, and NX-OS platforms	To access Cisco YANG models in a developer-friendly way, please clone the *GitHub repository*, and navigate to the *vendor/cisco* subdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.

Standards and RFCs

Standard/RFC	Title
RFC 6020	*YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*
RFC 6241	*Network Configuration Protocol (NETCONF)*
RFC 6536	*Network Configuration Protocol (NETCONF) Access Control Model*
RFC 7950	*The YANG 1.1 Data Modeling Language*
RFC 8040	*RESTCONF Protocol*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for the NETCONF Protocol

Table 14: Feature Information for NETCONF Protocol

Feature Name	Release	Feature Information
NETCONF Protocol	Cisco IOS XE Denali 16.3.1	The NETCONF Protocol feature facilitates a programmatic and standards-based way of writing configurations and reading operational data from network devices. The following command was introduced: **netconf-yang**. This feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco Cloud Services Router 1000V Series
	Cisco IOS XE Everest 16.5.1a	This feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches
	Cisco IOS XE Everest 16.6.2	This feature was implemented on the following platforms: • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches
	Cisco IOS XE Fuji 16.8.1a	

Feature Name	Release	Feature Information
		In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco Catalyst 9500-High Performance Series Switches • Cisco CBR-8 Series Routers • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Fuji 16.9.2	This feature was implemented on the following platforms: • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs
	Cisco IOS XE Gibraltar 16.10.1	In Cisco IOS XE Gibraltar 16.10.1, this feature was implemented on the following platforms: • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco Network Convergence System 520 Series
	Cisco IOS XE Gibraltar 16.11.1	In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on Cisco Catalyst 9600 Series Switches.
	Cisco IOS XE Gibraltar 16.12.1	In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on Cisco Catalyst 9800-L Wireless Controllers.
	Cisco IOS XE Amsterdam 17.3.1	

Feature Name	Release	Feature Information
		In Cisco IOS XE Amsterdam 17.3.1, this feature was implemented on the following platforms: • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 and 8500L Series Edge Platforms
NETCONF and RESTCONF IPv6 Support	Cisco IOS XE Fuji 16.8.1a	IPv6 support for the NETCONF and RESTCONF protocols. This feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco CBR-8 Series Routers • Cisco Cloud Services Router 1000V Series
	Cisco IOS XE Gibraltar 16.11.1	In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches.

Feature Name	Release	Feature Information
NETCONF Global Lock and Kill Session	Cisco IOS XE Fuji 16.8.1a	The NETCONF protocol supports a global lock, and the ability to kill non-responsive sessions. This feature is implemented on the following platforms: • Cisco 1100 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco CBR-8 Series Routers • Cisco Cloud Services Router 1000v Series

Feature Name	Release	Feature Information
NETCONF: Candidate Configuration Support	Cisco IOS XE Fuji 16.9.1	The Candidate Config Support feature enables support for candidate capability by implementing RFC 6241 with a simple commit option. This feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco CBR-8 Series Routers • Cisco Cloud Services Router 1000V Series The following command was introduced: **netconf-yang feature candidate-datastore**.

Feature Name	Release	Feature Information
NETCONF: Candidate Configuration Commit Confirm	Cisco IOS XE Amsterdam 17.1.1	The candidate configuration supports the confirmed commit capability. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

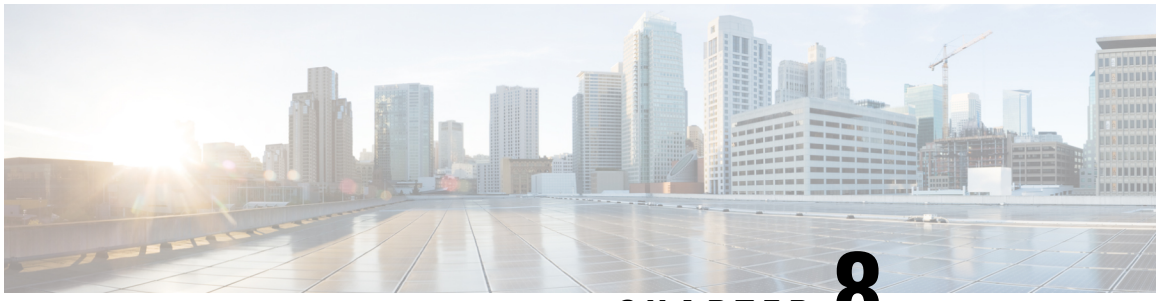
Feature Name	Release	Feature Information
NETCONF-YANG SSH Server Support	Cisco IOS XE Gibraltar 16.12.1	This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800 Series Wireless Controllers • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
NETCONF-SSH Algorithms	Cisco IOS XE Dublin 17.12.1	You can enable or disable the NETCONF-SSH algorithms during runtime by using Cisco IOS commands or U YANG models. This feature was implemented on the following platform: • Cisco Catalyst 9500 Series Switches

Feature Name	Release	Feature Information
Named Method List	Cisco IOS XE Cupertino 17.9.1	With the introduction of the Named Method List feature, it is possible to use a custom method-list name for authentication and authorization, without changing the existing AAA configuration of a device. Prior to this feature, only the default method-list was supported. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 Series and 8500L Series Edge Platforms • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800 Series Wireless Controllers • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature Name	Release	Feature Information
Side-Effect Synchronization of the Configuration Database	Cisco IOS XE Bengaluru 17.4.1	During configuration changes in the DMI, a partial synchronization of the changes that are triggered when a command or RPC is configured happens. This is called the side-effect synchronization, and it reduces the synchronization time and NETCONF downtime. This feature was implemented on the following platforms: • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 8500 and 8500L Series Edge Platforms • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches

Feature Name	Release	Feature Information
YANG Model Version 1.1	Cisco IOS XE Cupertino 17.7.1	Cisco IOS XE Cupertino 17.7.1 uses the YANG Version 1.0; however, you can also use YANG Version 1.1. Download the YANG Version 1.1 from GitHub at https://github.com/YangModels/yang/tree/master/vendor/cisco/xe folder. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Aggregation Services Routers • Cisco ASR 920 Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature Name	Release	Feature Information
	Cisco IOS XE Cupertino 17.8.1	Cisco IOS XE Cupertino 17.8.1 uses YANG Version 1.1. The difference between YANG Version 1.1 and Version 1.0 is documented at https://tools.ietf.org/html/rfc7950#page-10
	Cisco IOS XE Dublin 17.10.1	Cisco-defined YANG models are in YANG Version 1.1 in Cisco IOS XE Dublin 17.10.1 and later releases.
Converting IOS Commands to XML	Cisco IOS XE Cupertino 17.7.1	This feature helps to automatically translate IOS commands into relevant NETCONF-YANG XML or RESTCONF-JSON request messages. This feature is supported on all platforms that support NETCONF-YANG.



CHAPTER 8

RESTCONF Protocol

This chapter describes how to configure the HTTP-based Representational State Transfer Configuration Protocol (RESTCONF). RESTCONF provides a programmatic interface based on standard mechanisms for accessing configuration data, state data, data-model-specific Remote Procedure Call (RPC) operations and events, defined in the YANG model.

- Prerequisites for the RESTCONF Protocol, on page 205
- Restrictions for the RESTCONF Protocol, on page 205
- Information About the RESTCONF Protocol, on page 206
- How to Configure the RESTCONF Protocol, on page 226
- Configuration Examples for the RESTCONF Protocol, on page 231
- Additional References for the RESTCONF Protocol, on page 234
- Feature Information for the RESTCONF Protocol, on page 235

Prerequisites for the RESTCONF Protocol

- Enable the Cisco IOS-HTTP services for RESTCONF. For more information, see Examples for RESTCONF RPCs

Restrictions for the RESTCONF Protocol

The following restrictions apply to the RESTCONF protocol:

- Notifications and event streams
- YANG patch
- Optional query parameters, such as, filter, start-time, stop-time, replay, and action
- The RESTCONF feature is not supported on a device running dual IOSd configuration or software redundancy.

Information About the RESTCONF Protocol

Overview of RESTCONF

This section describes the protocols and modelling languages that enable a programmatic way of writing configurations to a network device.

- RESTCONF—Uses structured data (XML or JSON) and YANG to provide a REST-like APIs, enabling you to programmatically access different network devices. RESTCONF APIs use HTTPs methods.
- YANG—A data modelling language that is used to model configuration and operational features . YANG determines the scope and the kind of functions that can be performed by NETCONF and RESTCONF APIs.

In releases prior to Cisco IOS XE Fuji 16.8.1, an operational data manager (based on polling) was enabled separately. In Cisco IOS XE Fuji 16.8.1 and later releases, operational data works on platforms running NETCONF (similar to how configuration data works), and is enabled by default. For more information on the components that are enabled for operational data queries or streaming, see the GitHub respository, and view **-oper* in the naming convention.

HTTPs Methods

The HTTPS-based RESTCONF protocol (RFC 8040), is a stateless protocol that uses secure HTTP methods to provide CREATE, READ, UPDATE, and DELETE (CRUD) operations on a conceptual datastore containing YANG-defined data, which is compatible with a server that implements NETCONF datastores.

The following table shows how the RESTCONF operations relate to NETCONF protocol operations:

OPTIONS	SUPPORTED METHODS
GET	Read
PATCH	Update
PUT	Create or Replace
POST	Create or Operations (reload, default)
DELETE	Deletes the targeted resource
HEAD	Header metadata (no response body)

RESTCONF Root Resource

- A RESTCONF device determines the root of the RESTCONF API through the link element: /.well-known/host-meta resource that contains the RESTCONF attribute.
- A RESTCONF device uses the RESTCONF API root resource as the initial part of the path in the request URI.

Example:

```
Example returning /restconf:

   The client might send the following:

      GET /.well-known/host-meta HTTP/1.1
      Host: example.com
      Accept: application/xrd+xml

   The server might respond as follows:

      HTTP/1.1 200 OK
      Content-Type: application/xrd+xml
      Content-Length: nnn

      <XRD xmlns='http://docs.oasis-open.org/ns/xri/xrd-1.0'>
          <Link rel='restconf' href='/restconf'/>
      </XRD>
```

Example of URIs:

- GigabitEthernet0/0/2 - https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2
- fields=name – https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2?fields=name
- depth=1 - https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet?depth=1
- Name and IP - https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet/ip/address/primary;name
- MTU (fields) - https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet(mtu)
- MTU - https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=3/mtu
- Port-Channel - https://10.85.116.59/restconf/data/Cisco-IOS-XE-native:native/interface/Port-channel
- “Char” to “Hex” conversion chart: http://www.columbia.edu/kermit/ascii.html

Displaying Version Information

The *Cisco-IOS-XE-install-oper* module that has various nodes to display the version information.

The following sample RPC shows the some of the supported nodes of the *Cisco-IOS-XE-install-oper* module and the response from the host that contains the major and minor release version:

```
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
        message-id="urn:uuid:7d0908d8-0d5f-4521-9d7b-380b81304776">
  <nc:get>
    <nc:filter>
      <install-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-install-oper">
        <install-location-information>
```

```
          <install-version-info>
            <version/>
            <version-extension/>
            <current/>
            <src-filename/>
          </install-version-info>
        </install-location-information>
      </install-oper-data>
    </nc:filter>
  </nc:get>
</nc:rpc>

##
Received message from host

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:7d0908d8-0d5f-4521-9d7b-380b81304776">
  <data>
    <install-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-install-oper">
      <install-location-information>
        <install-version-info>
          <version>17.06.04.0.3870</version>
          <version-extension>1651661105</version-extension>
          <current>install-version-state-present</current>
          <src-filename/>
        </install-version-info>
        <install-version-info>
          <version>17.09.01.0.158212</version>
          <version-extension>1651125381</version-extension>
          <current>install-version-state-present</current>
          <src-filename/>
        </install-version-info>
        <install-version-info>
          <version>17.10.01.0.158658</version>
          <version-extension>1651754624</version-extension>
          <current>install-version-state-present</current>

<src-filename>/bootflash/c8000v-universalk9nic.2022-05-05_18.13.SSA.bin</src-filename>
        </install-version-info>
        <install-version-info>
          <version>17.10.01.0.160585</version>
          <version-extension>1656581638</version-extension>
          <current>install-version-state-provisioned-committed</current>
         <src-filename>/bootflash/c8000v-universalk9.2022-06-30_15.03.SSA.bin</src-filename>

        </install-version-info>
        <install-version-info>
          <version>17.10.01.0.162616</version>
          <version-extension>1657120419</version-extension>
          <current>install-version-state-present</current>
          <src-filename>/bootflash/c8000v-universalk9.BLD_POLARIS_DEV_LATEST_20220706_
            143733.SSA.bin</src-filename>
        </install-version-info>
      </install-location-information>
    </install-oper-data>
  </data>
</rpc-reply>
```

When using the protocol, gNMI, NETCONF, or RESTCONF, the *Cisco-IOS-XE-native:version* module only displays the major release version.

RESTCONF API Resource

The API resource is the top-level resource located at +restconf. It supports the following media types:

Note Media is the type of YANG formated RPC that is sent to the RESCONF server (XML or JSON).

- Application/YANG-Data+XML OR Application/YANG-Data+JSON
- The API resource contains the RESTCONF root resource for the RESTCONF DATASTORE and OPERATION resources. For example:

```
The client may then retrieve the top-level API resource, using the
   root resource "/restconf".

      GET /restconf HTTP/1.1
      Host: example.com
      Accept: application/yang-data+json

   The server might respond as follows:

      HTTP/1.1 200 OK
      Date: Thu, 26 Jan 2017 20:56:30 GMT
      Server: example-server
      Content-Type: application/yang-data+json

      {
        "ietf-restconf:restconf" : {
          "data" : {},
          "operations" : {},
          "yang-library-version" : "2016-06-21"
        }
      }
```

For more information, refer to RFC 3986

Methods

Methods are HTTPS operations (GET/PATCH/POST/DELETE/OPTIONS/PUT) performed on a target resource. A YANG-formated RPC invokes a particular method on a given resource that pertains to a target YANG model residing in the RESTCONF server. The uniform resource identifier (URI) acts as a location identification for a given resource, so that the client RESTCONF method can locate that particular resource to take an action specified by an HTTPS method or property.

For more information, see *RFC 8040 - RESTCONF Protocol*

RESTCONF YANG-Patch Support

RESTCONF supports YANG-Patch media type as specified by RFC 8072. A YANG-Patch is an ordered list of edits that are applied to the target datastore by the RESTCONF server. The YANG Patch operation is invoked by the RESTCONF client by sending a Patch method request with a representation using either the media type *application/yang-patch+xml* or *application/yang-patch+json*.

A YANG-Patch is identified by a unique patch-id. A patch is an ordered collection of edits and each edit is identified by an edit-id. It has an edit operation ("create", "delete", "insert", "merge", "move", "replace", or "remove") that is applied to the target resource.

To verify if the RESTCONF YANG-Patch is supported issue the following RESTCONF Get request:

```
$ curl -k -s -u admin:DMIdmi1! --location-trusted
"https://10.1.1.1/restconf/data/ietf-restconf-monitoring:restconf-state/capabilities" -X
GET

<capabilities xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring"
xmlns:rcmon="urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring">

<capability>urn:ietf:params:restconf:capability:defaults:1.0?basic-mode=explicit</capability>

  <capability>urn:ietf:params:restconf:capability:depth:1.0</capability>
  <capability>urn:ietf:params:restconf:capability:fields:1.0</capability>
  <capability>urn:ietf:params:restconf:capability:with-defaults:1.0</capability>
  <capability>urn:ietf:params:restconf:capability:filter:1.0</capability>
  <capability>urn:ietf:params:restconf:capability:replay:1.0</capability>

<capability>urn:ietf:params:restconf:capability:yang-patch:1.0</capability>

  <capability>http://tail-f.com/ns/restconf/collection/1.0</capability>
  <capability>http://tail-f.com/ns/restconf/query-api/1.0</capability>
</capabilities>
```

This section provides a few RESTCONF YANG-Patch examples.

Add Resource Error

While trying to edit a file, the first edit already exists and an error is reported. The rest of the edits are not attempted because the first edit failed. XML encoding is used in this example

The following example show an add resource request from the RESTCONF client:

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
        <patch-id>add-hostname-patch</patch-id>
        <edit>
          <edit-id>edit1</edit-id>
          <operation>create</operation>
          <target>/hostname</target>
          <value>
              <hostname
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">Cat9K-test</hostname>
          </value>
        </edit>
        <edit>
          <edit-id>edit2</edit-id>
          <operation>create</operation>
          <target>/interface/Loopback=1</target>
          <value>
              <interface xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
              <Loopback>
                <name>1</name>
              </Loopback>
              </interface>
          </value>
        </edit>
</yang-patch>
```

The following examples shows a JSON response from the RESTCONF server:

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native" -X PATCH -H "Accept:
application/yang-data+json" -d
'@yang_patch_create_hostname' -H "Content-type: application/yang-patch+xml"
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "add-hostname-patch",
    "edit-status": {
      "edit": [
        {
          "edit-id": "edit1",
          "errors": {
            "error": [
              {
                "error-type": "application",
                "error-tag": "data-exists",
                "error-path": "/Cisco-IOS-XE-native:native/hostname",
                "error-message": "object already exists: /ios:native/ios:hostname"
              }
            ]
          }
        }
      ]
    }
  }
}
```

The following example shows an XML response from the RESTCONF server:

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native" -X PATCH -H "Accept:
application/yang-data+xml" -d
'@yang_patch_create_hostname' -H "Content-type: application/yang-patch+xml"

 <yang-patch-status xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
   <patch-id>add-hostname-patch</patch-id>
   <edit-status>
     <edit>
       <edit-id>edit1</edit-id>
       <errors>
         <error>
           <error-type>application</error-type>
           <error-tag>data-exists</error-tag>
           <error-path
xmlns:ios="http://cisco.com/ns/yang/Cisco-IOS-XE-native">/ios:native/ios:hostname</error-path>

           <error-message>object already exists: /ios:native/ios:hostname</error-message>
         </error>
       </errors>
     </edit>
   </edit-status>
 </yang-patch-status>device:/nobackup/folder1/confd_6313/bin $
```

Add Resource Success

The following example shows an edit request:

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
        <patch-id>add-Loopback-patch</patch-id>
        <edit>
          <edit-id>edit1</edit-id>
          <operation>create</operation>
          <target>/Loopback=1</target>
```

```
          <value>
              <Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
                <name>1</name>
              </Loopback>
          </value>
        </edit>
</yang-patch>
```

The following example shows that the edit request is successful:

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native/interface" -X PATCH -H "Accept:
 application/yang-data+json"
-d '@yang_patch_create_Loopback_interface' -H "Content-type: application/yang-patch+xml"
Device:/nobackup/folder1/confd_6313/bin
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "add-Loopback-patch",
    "ok" : [null]
  }
}
```

Insert List Entry

The following example shows that the Loopback 1 is inserted after Loopback 0:

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
        <patch-id>insert-Loopback-patch</patch-id>
        <edit>
          <edit-id>edit1</edit-id>
          <operation>insert</operation>
          <target>/Loopback=1</target>
          <point>/Loopback=0</point>
          <where>after</where>
          <value>
              <Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
                <name>1</name>
              </Loopback>
          </value>
        </edit>
</yang-patch>
```

The following example shows that the insert list request is successful:

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native/interface" -X PATCH -H "Accept:
 application/yang-data+json" -d
'@yang_patch_create_Loopback_interface' -H "Content-type: application/yang-patch+xml"
Device:/nobackup/folder1/confd_6313/bin
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "insert-Loopback-patch",
    "ok" : [null]
  }
}
```

Move List Entry

The following example shows Loopback 1 is moved before Loopback 0:

```
<yang-patch xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-patch">
        <patch-id>move-Loopback-patch</patch-id>
```

```
        <edit>
          <edit-id>edit1</edit-id>
          <operation>move</operation>
          <target>/Loopback=1</target>
          <point>/Loopback=0</point>
          <where>before</where>
          <value>
              <Loopback xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
                <name>1</name>
              </Loopback>
          </value>
        </edit>
</yang-patch>
```

The following example shows that the move request is successful:

```
Device:/nobackup/folder1/confd_6313/bin $ curl -k -s -u admin:DMIdmi1! --location-trusted
"https://10.1.1.1/restconf/data/Cisco-IOS-XE-native:native/interface" -X PATCH -H "Accept:
 application/yang-data+json" -d
'@yang_patch_create_Loopback_interface' -H "Content-type: application/yang-patch+xml"
Device:/nobackup/folder1/confd_6313/bin
{
  "ietf-yang-patch:yang-patch-status": {
    "patch-id": "move-Loopback-patch",
    "ok" : [null]
  }
}
```

NETCONF RESTCONF IPv6 Support

Data model interfaces (DMIs) support the use of IPv6 protocol. DMI IPv6 support helps client applications to communicate with services that use IPv6 addresses. External facing interfaces will provide dual-stack support; both IPv4 and IPv6.

DMIs are a set of services that facilitate the management of network elements. Application layer protocols such as, NETCONF and RESTCONF access these DMIs over a network.

If IPv6 addresses are not configured, external-facing applications will continue to listen on IPv6 sockets; but these sockets will be unreachable.

Converting IOS Commands to XML

In Cisco IOS XE Cupertino 17.7.1 and later releases, you can automatically translate IOS commands into relevant NETCONF-YANG XML or RESTCONF-JSON request messages. You can analyze the generated configuration messages and familiarize with the Xpaths used in these messages. The generated configuration in the structured format can be used to provision other devices in the network; however, this configuration cannot be modified.

Use the **show running-config | format netconf-xml** command or the **show running-config | format restconf-json** command to translate IOS commands.

If the **netconf-xml** keyword is selected, the IOS commands are translated into the NETCONF-YANG XML format, and if the **restconf-json** keyword is selected, the IOS commands are translated into the RESTCONF-JSON format.

The translation of IOS commands into a structured format is disabled by default. You must initially configure NETCONF-YANG, and once the data model interfaces (DMIs) are initialized, use the appropriate format option to translate the commands.

The following is sample output from the **show running-config | format netconf-xml** command:

```
Device# show running-config | format netconf-xml

<config xmlns="http://tail-f.com/ns/config/1.0">
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <version>17.8</version>
    <boot-start-marker/>
    <boot>
      <system>
        <flash>
          <flash-list-ordered-by-user>

<flash-leaf>bootflash:c8000v-universalk9.BLD_POLARIS_DEV_LATEST_20211020_005209.SSA.bin</
             flash-leaf>
          </flash-list-ordered-by-user>
        </flash>
      </system>
    </boot>
    <boot-end-marker/>
    <memory>
      <free>
        <low-watermark>
          <processor>64219</processor>
        </low-watermark>
      </free>
    </memory>
    <call-home>
      <contact-email-addr xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">
        sch-smart-licensing@cisco.com</contact-email-addr>
      <tac-profile xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">
        <profile>
          <CiscoTAC-1>
            <active>true</active>
            <destination>
              <transport-method>http</transport-method>
            </destination>
          </CiscoTAC-1>
        </profile>
      </tac-profile>
    </call-home>
    <service>
      <timestamps>
        <debug-config>
          <datetime>
            <msec/>
            <localtime/>
            <show-timezone/>
          </datetime>
        </debug-config>
        <log-config>
          <datetime>
            <msec/>
            <localtime/>
            <show-timezone/>
          </datetime>
        </log-config>
      </timestamps>
      <call-home/>
    </service>
```

```
<platform>
  <console xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
    <output>serial</output>
  </console>
  <qfp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
    <utilization>
      <monitor>
        <load>80</load>
      </monitor>
    </utilization>
  </qfp>
  <punt-keepalive xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
    <disable-kernel-core>true</disable-kernel-core>
  </punt-keepalive>
</platform>
<hostname>pi-prog-csr1</hostname>
<enable>
  <password>
    <secret>lab</secret>
  </password>
</enable>
<username>
  <name>admin</name>
  <privilege>15</privilege>
  <password>
    <encryption>0</encryption>
    <password>lab</password>
  </password>
</username>
<vrf>
  <definition>
    <name>Mgmt-intf</name>
    <address-family>
      <ipv4>
      </ipv4>
      <ipv6>
      </ipv6>
    </address-family>
  </definition>
</vrf>
<ip>
  <domain>
    <name>cisco</name>
  </domain>
  <forward-protocol>
    <protocol>nd</protocol>
  </forward-protocol>
  <route>
    <ip-route-interface-forwarding-list>
      <prefix>10.0.0.0</prefix>
      <mask>255.255.0.0</mask>
      <fwd-list>
        <fwd>10.45.0.1</fwd>
      </fwd-list>
    </ip-route-interface-forwarding-list>
    <vrf>
      <name>Mgmt-intf</name>
      <ip-route-interface-forwarding-list>
        <prefix>0.0.0.0</prefix>
        <mask>0.0.0.0</mask>
        <fwd-list>
          <fwd>10.104.54.129</fwd>
        </fwd-list>
      </ip-route-interface-forwarding-list>
```

```
    </vrf>
  </route>
  <ssh>
    <ssh-version>2</ssh-version>
  </ssh>
  <tftp>
    <source-interface>
      <GigabitEthernet>1</GigabitEthernet>
    </source-interface>
    <blocksize>8192</blocksize>
  </tftp>
  <http xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-http">
    <authentication>
      <local/>
    </authentication>
    <server>true</server>
    <secure-server>true</secure-server>
  </http>
</ip>
<ipv6>
  <unicast-routing/>
</ipv6>
<interface>
  <GigabitEthernet>
    <name>1</name>
    <vrf>
      <forwarding>Mgmt-intf</forwarding>
    </vrf>
    <ip>
      <address>
        <primary>
          <address>10.104.54.222</address>
          <mask>255.255.255.128</mask>
        </primary>
      </address>
    </ip>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>2</name>
    <ip>
      <address>
        <primary>
          <address>9.45.21.231</address>
          <mask>255.255.0.0</mask>
        </primary>
      </address>
    </ip>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>3</name>
```

```
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>4</name>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>5</name>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
</interface>
<control-plane>
</control-plane>
<clock>
  <timezone>
    <zone>IST</zone>
    <hours>5</hours>
    <minutes>30</minutes>
  </timezone>
</clock>
<logging>
  <console-config>
    <console>false</console>
  </console-config>
</logging>
<aaa>
  <new-model xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa"/>
  <authentication xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">
    <login>
      <name>default</name>
      <a1>
        <local/>
      </a1>
    </login>
  </authentication>
  <authorization xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">
    <exec>
      <name>default</name>
      <a1>
        <local/>
      </a1>
    </exec>
  </authorization>
  <common-criteria xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">
    <policy>enable_secret_policy</policy>
    <char-changes>4</char-changes>
```

```
          <lower-case>1</lower-case>
          <max-length>127</max-length>
          <min-length>10</min-length>
          <numeric-count>1</numeric-count>
          <upper-case>1</upper-case>
        </common-criteria>
        <session-id xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">common</session-id>
      </aaa>
      <login>
        <on-success>
          <log>
          </log>
        </on-success>
      </login>
      <multilink>
        <bundle-name
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ppp">authenticated</bundle-name>
      </multilink>
      <redundancy>
      </redundancy>
      <spanning-tree>
        <extend xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-spanning-tree">
          <system-id/>
        </extend>
      </spanning-tree>
      <subscriber>
        <templating/>
      </subscriber>
      <crypto>
        <pki xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-crypto">
          <certificate>
            <chain>
              <name>SLA-TrustPoint</name>
              <certificate>
                <serial>01</serial>
                <certtype>ca</certtype>
              </certificate>
            </chain>
            <chain>
              <name>TP-self-signed-2685563505</name>
              <certificate>
                <serial>01</serial>
                <certtype>self-signed</certtype>
              </certificate>
            </chain>
          </certificate>
          <trustpoint>
            <id>SLA-TrustPoint</id>
            <enrollment>
              <pkcs12/>
            </enrollment>
            <revocation-check>crl</revocation-check>
          </trustpoint>
          <trustpoint>
            <id>TP-self-signed-2685563505</id>
            <enrollment>
              <selfsigned/>
            </enrollment>
            <revocation-check>none</revocation-check>
            <rsakeypair>
              <key-label>TP-self-signed-2685563505</key-label>
            </rsakeypair>
            <subject-name>cn=IOS-Self-Signed-Certificate-2685563505</subject-name>
          </trustpoint>
```

```
  </pki>
</crypto>
<license>
  <udi>
    <pid>C8000V</pid>
    <sn>93SHKMJKOC6</sn>
  </udi>
  <boot>
    <level>
      <network-advantage>
        <addon>dna-advantage</addon>
      </network-advantage>
    </level>
  </boot>
</license>
<line>
  <aux>
    <first>0</first>
  </aux>
  <console>
    <first>0</first>
    <exec-timeout>
      <minutes>0</minutes>
      <seconds>0</seconds>
    </exec-timeout>
    <stopbits>1</stopbits>
  </console>
  <vty>
    <first>0</first>
    <last>4</last>
    <exec-timeout>
      <minutes>0</minutes>
      <seconds>0</seconds>
    </exec-timeout>
    <password>
      <secret>lab</secret>
    </password>
    <transport>
      <input>
        <all/>
      </input>
      <output>
        <all/>
      </output>
    </transport>
  </vty>
  <vty>
    <first>5</first>
    <last>31</last>
    <transport>
      <input>
        <all/>
      </input>
      <output>
        <all/>
      </output>
    </transport>
  </vty>
</line>
<diagnostic xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-diagnostics">
  <bootup>
    <level>minimal</level>
  </bootup>
</diagnostic>
```

```
  </native>
</config>
pi-prog-csr1#
pi-prog-csr1#
pi-prog-csr1#show running-config | format restconf-json
{
  "data": {
    "Cisco-IOS-XE-native:native": {
      "version": "17.8",
      "boot-start-marker": [null],
      "boot": {
        "system": {
          "flash": {
            "flash-list-ordered-by-user": [
              {
                "flash-leaf":
"bootflash:c8000v-universalk9.BLD_POLARIS_DEV_LATEST_20211020_005209.SSA.bin"
              }
            ]
          }
        }
      },
      "boot-end-marker": [null],
      "memory": {
        "free": {
          "low-watermark": {
            "processor": 64219
          }
        }
      },
      "call-home": {
        "Cisco-IOS-XE-call-home:contact-email-addr": "sch-smart-licensing@cisco.com",
        "Cisco-IOS-XE-call-home:tac-profile": {
          "profile": {
            "CiscoTAC-1": {
              "active": true,
              "destination": {
                "transport-method": "http"
              }
            }
          }
        }
      },
      "service": {
        "timestamps": {
          "debug-config": {
            "datetime": {
              "msec": [null],
              "localtime": [null],
              "show-timezone": [null]
            }
          },
          "log-config": {
            "datetime": {
              "msec": [null],
              "localtime": [null],
              "show-timezone": [null]
            }
          }
        },
        "call-home": [null]
      },
      "platform": {
        "Cisco-IOS-XE-platform:console": {
```

```
    "output": "serial"
  },
  "Cisco-IOS-XE-platform:qfp": {
    "utilization": {
      "monitor": {
        "load": 80
      }
    }
  },
  "Cisco-IOS-XE-platform:punt-keepalive": {
    "disable-kernel-core": true
  }
},
"hostname": "pi-prog-csr1",
"enable": {
  "password": {
    "secret": "lab"
  }
},
"username": [
  {
    "name": "admin",
    "privilege": 15,
    "password": {
      "encryption": "0",
      "password": "lab"
    }
  }
],
"vrf": {
  "definition": [
    {
      "name": "Mgmt-intf",
      "address-family": {
        "ipv4": {
        },
        "ipv6": {
        }
      }
    }
  ]
},
"ip": {
  "domain": {
    "name": "cisco"
  },
  "forward-protocol": {
    "protocol": "nd"
  },
  "route": {
    "ip-route-interface-forwarding-list": [
      {
        "prefix": "10].0.0.0",
        "mask": "255.255.0.0",
        "fwd-list": [
          {
            "fwd": "9.45.0.1"
          }
        ]
      }
    ],
    "vrf": [
      {
        "name": "Mgmt-intf",
```

```
        "ip-route-interface-forwarding-list": [
          {
            "prefix": "0.0.0.0",
            "mask": "0.0.0.0",
            "fwd-list": [
              {
                "fwd": "10.104.54.129"
              }
            ]
          }
        ]
      }
    ]
  },
  "ssh": {
    "ssh-version": "2"
  },
  "tftp": {
    "source-interface": {
      "GigabitEthernet": "1"
    },
    "blocksize": 8192
  },
  "Cisco-IOS-XE-http:http": {
    "authentication": {
      "local": [null]
    },
    "server": true,
    "secure-server": true
  }
},
"ipv6": {
  "unicast-routing": [null]
},
"interface": {
  "GigabitEthernet": [
    {
      "name": "1",
      "vrf": {
        "forwarding": "Mgmt-intf"
      },
      "ip": {
        "address": {
          "primary": {
            "address": "10.104.54.222",
            "mask": "255.255.255.128"
          }
        }
      },
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    },
    {
      "name": "2",
      "ip": {
        "address": {
          "primary": {
            "address": "10.45.21.231",
            "mask": "255.255.0.0"
```

```
          }
        }
      },
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    },
    {
      "name": "3",
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    },
    {
      "name": "4",
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    },
    {
      "name": "5",
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    }
  ]
},
"control-plane": {
},
"clock": {
  "timezone": {
    "zone": "IST",
    "hours": 5,
    "minutes": 30
  }
},
"logging": {
  "console-config": {
    "console": false
  }
},
"aaa": {
  "Cisco-IOS-XE-aaa:new-model": [null],
  "Cisco-IOS-XE-aaa:authentication": {
    "login": [
      {
        "name": "default",
        "a1": {
```

```
          "local": [null]
        }
      }
    ]
  },
  "Cisco-IOS-XE-aaa:authorization": {
    "exec": [
      {
        "name": "default",
        "a1": {
          "local": [null]
        }
      }
    ]
  },
  "Cisco-IOS-XE-aaa:common-criteria": [
    {
      "policy": "enable_secret_policy",
      "char-changes": 4,
      "lower-case": 1,
      "max-length": 127,
      "min-length": 10,
      "numeric-count": 1,
      "upper-case": 1
    }
  ],
  "Cisco-IOS-XE-aaa:session-id": "common"
},
"login": {
  "on-success": {
    "log": {
    }
  }
},
"multilink": {
  "Cisco-IOS-XE-ppp:bundle-name": "authenticated"
},
"redundancy": {
},
"spanning-tree": {
  "Cisco-IOS-XE-spanning-tree:extend": {
    "system-id": [null]
  }
},
"subscriber": {
  "templating": [null]
},
"crypto": {
  "Cisco-IOS-XE-crypto:pki": {
    "certificate": {
      "chain": [
        {
          "name": "SLA-TrustPoint",
          "certificate": [
            {
              "serial": "01",
              "certtype": "ca"
            }
          ]
        },
        {
          "name": "TP-self-signed-2685563505",
          "certificate": [
            {
```

```
                "serial": "01",
                "certtype": "self-signed"
              }
            ]
          }
        ]
      },
      "trustpoint": [
        {
          "id": "SLA-TrustPoint",
          "enrollment": {
            "pkcs12": [null]
          },
          "revocation-check": ["crl"]
        },
        {
          "id": "TP-self-signed-2685563505",
          "enrollment": {
            "selfsigned": [null]
          },
          "revocation-check": ["none"],
          "rsakeypair": {
            "key-label": "TP-self-signed-2685563505"
          },
          "subject-name": "cn=IOS-Self-Signed-Certificate-2685563505"
        }
      ]
    }
  },
  "license": {
    "udi": {
      "pid": "C8000V",
      "sn": "93SHKMJKOC6"
    },
    "boot": {
      "level": {
        "network-advantage": {
          "addon": "dna-advantage"
        }
      }
    }
  },
  "line": {
    "aux": [
      {
        "first": "0"
      }
    ],
    "console": [
      {
        "first": "0",
        "exec-timeout": {
          "minutes": 0,
          "seconds": 0
        },
        "stopbits": "1"
      }
    ],
    "vty": [
      {
        "first": 0,
        "last": 4,
        "exec-timeout": {
          "minutes": 0,
```

```
            "seconds": 0
          },
          "password": {
            "secret": "lab"
          },
          "transport": {
            "input": {
              "all": [null]
            },
            "output": {
              "all": [null]
            }
          }
        },
        {
          "first": 5,
          "last": 31,
          "transport": {
            "input": {
              "all": [null]
            },
            "output": {
              "all": [null]
            }
          }
        }
      ]
    },
    "Cisco-IOS-XE-diagnostics:diagnostic": {
      "bootup": {
        "level": "minimal"
      }
    }
  }
}
```

How to Configure the RESTCONF Protocol

Authentication of NETCONF/RESTCONF Using AAA

Before you begin

NETCONF and RESTCONF connections must be authenticated using authentication, authorization, and accounting (AAA). As a result, RADIUS or TACACS+ users defined with privilege level 15 access are allowed access into the system.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **aaa new-model**
4. **aaa group server radius** *server-name*
5. **server-private** *ip-address* **key** *key-name*
6. **ip vrf forwarding** *vrf-name*

7. **exit**
8. **aaa authentication login default group** *group-name* **local**
9. **aaa authentication login** *list-name* **none**
10. **aaa authorization exec default group** *group-name* **local**
11. **aaa session-id common**
12. **line console** *number*
13. **login authentication** *authentication-list*
14. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**aaa new-model** **Example:** `Device(config)# aaa new-model`	Enables AAA.
Step 4	**aaa group server radius** *server-name* **Example:** `Device(config)# aaa group server radius ISE`	Adds the RADIUS server and enters server group RADIUS configuration mode. • The *server-name* argument specifies the RADIUS server group name.
Step 5	**server-private** *ip-address* **key** *key-name* **Example:** `Device(config-sg-radius)# server-private 172.25.73.76 key Cisco123`	Configures a IP address and encryption key for a private RADIUS server.
Step 6	**ip vrf forwarding** *vrf-name* **Example:** `Device(config-sg-radius)# ip vrf forwarding Mgmt-intf`	Configures the virtual routing and forwarding (VRF) reference of a AAA RADIUS or TACACS+ server group.
Step 7	**exit** **Example:** `Device(config-sg-radius)# exit`	Exits server group RADIUS configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 8	**aaa authentication login default group** *group-name* **local** **Example:** `Device(config)# aaa authentication login default group ISE local`	Sets the specified group name as the default local AAA authentication during login.
Step 9	**aaa authentication login** *list-name* **none** **Example:** `Device(config)# aaa authentication login NOAUTH none`	Specifies that no authentication is required while logging into a system.
Step 10	**aaa authorization exec default group** *group-name* **local** **Example:** `Device(config)# aaa authorization exec default group ISE local`	Runs authorization to determine if an user is allowed to run an EXEC shell.
Step 11	**aaa session-id common** **Example:** `Device(config)# aaa session-id common`	Ensures that session identification (ID) information that is sent out for a given call will be made identical.
Step 12	**line console** *number* **Example:** `Device(config)# line console 0`	Identifies a specific line for configuration and enter line configuration mode.
Step 13	**login authentication** *authentication-list* **Example:** `Device(config-line)# login authentication NOAUTH`	Enables AAA authentication for logins.
Step 14	**end** **Example:** `Device(config-line)# end`	Exits line configuration mode and returns to privileged EXEC mode.

Enabling Cisco IOS HTTP Services for RESTCONF

Perform this task to use the RESTCONF interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **restconf**
4. **ip http secure-server**
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**restconf** **Example:** `Device(config)# restconf`	Enables the RESTCONF interface on your network device.
Step 4	**ip http secure-server** **Example:** `Device(config)# ip http secure-server`	Enables a secure HTTP (HTTPS) server.
Step 5	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode

Verifying RESTCONF Configuration

When a device boots up with the startup configuration, the *nginx* process will be running. However; DMI proceses are not enabled.

The following sample output from the **show platform software yang-management process monitor** command shows that the *nginx* process is running:

```
Device# show platform software yang-management process monitor

COMMAND           PID S    VSZ   RSS %CPU %MEM     ELAPSED
nginx           27026 S 332356 18428  0.0  0.4       01:34
nginx           27032 S 337852 13600  0.0  0.3       01:34
```

NGINX is an internal webserver that acts as a proxy webserver. It provides Transport Layer Security (TLS)-based HTTPS. RESTCONF request sent via HTTPS is first received by the NGINX proxy web serve,r and the request is transferred to the confd web server for further syntax/semantics check.

The following sample output from the **show platform software yang-management process** command shows the status of the all processes when a device is booted with the startup-configuration:

```
Device# show platform software yang-management process
```

```
confd            : Not Running
nesd             : Not Running
syncfd           : Not Running
ncsshd           : Not Running
dmiauthd         : Not Running
nginx            : Running
ndbmand          : Not Running
pubd             : Not Running
```

The *nginx* process gets restrated and DMI process are started, when the **restconf** command is configured.

The following sample output from the **show platform software yang-management process** command shows that the *nginx* process and DMI processes are up and running:

```
Device# show platform software yang-management process

confd            : Running
nesd             : Running
syncfd           : Running
ncsshd           : Not Running ! NETCONF-YANG is not configured, hence ncsshd process is
in not running.
dmiauthd         : Running
vtyserverutild   : Running
opdatamgrd       : Running
nginx            : Running ! nginx process is up due to the HTTP configuration, and it is
restarted when RESTCONF is enabled.
ndbmand          : Running
```

The following sample output from the show platform software yang-management process monitor command displays detailed information about all processes:

```
Device#show platform software yang-management process monitor

COMMAND           PID S    VSZ   RSS %CPU %MEM     ELAPSED
confd           28728 S 860396 168496 42.2  4.2       00:12
confd-startup.s 28448 S  19664  4496  0.2  0.1       00:12
dmiauthd        29499 S 275356 23340  0.2  0.5       00:10
ndbmand         29321 S 567232 65564  2.1  1.6       00:11
nesd            29029 S 189952 14224  0.1  0.3       00:11
nginx           29711 S 332288 18420  0.6  0.4       00:09
nginx           29717 S 337636 12216  0.0  0.3       00:09
pubd            28237 S 631848 68624  2.1  1.7       00:13
syncfd          28776 S 189656 16744  0.2  0.4       00:12
```

After AAA and the RESTCONF interface is configured, and nginx process and relevant DMI processes are running; the device is ready to receive RESTCONF requests.

Use the **show netconf-yang sessions** command to view the status of NETCONF/RESTCONF sessions:

```
Device# show netconf-yang sessions

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 1

session-id  transport    username              source-host            global-lock
--------------------------------------------------------------------------------
19          netconf-ssh  admin                 2001:db8::1             None
```

Use the **show netconf-yang sessions detail** command to view detailed information about NETCONF/RESTCONF sessions:

```
Device# show netconf-yang sessions detail

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions     : 1

session-id             : 19
transport              : netconf-ssh
username               : admin
source-host            : 2001:db8::1
login-time             : 2018-10-26T12:37:22+00:00
in-rpcs                : 0
in-bad-rpcs            : 0
out-rpc-errors         : 0
out-notifications      : 0
global-lock            : None
```

Configuration Examples for the RESTCONF Protocol

Example: Configuring the RESTCONF Protocol

RESTCONF Requests (HTTPS Verbs):

The following is a sample RESTCONF request that shows the HTTPS verbs allowed on a targeted resource. In this example, the **logging monitor** command is used..

```
root:~# curl -i -k -X "OPTIONS"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin'
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 23 Apr 2018 15:27:57 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Allow: DELETE, GET, HEAD, PATCH, POST, PUT, OPTIONS    >>>>>>>>>>>    Allowed methods
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Accept-Patch: application/yang-data+xml, application/yang-data+json
Pragma: no-cache

root:~#
```

POST (Create) Request

The POST operation creates a configuration which is not present in the targeted device.

Note Ensure that the **logging monitor** command is not availabel in the running configuration.

The following sample POST request uses the **logging monitor alerts** command.

```
Device:~# curl -i -k -X "POST"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor" \
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin' \
>      -d $'{
>   "severity": "alerts"
> }'
HTTP/1.1 201 Created
Server: nginx
Date: Mon, 23 Apr 2018 14:53:51 GMT
Content-Type: text/html
Content-Length: 0
Location:
https://10.85.116.30/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 14:53:51 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-495231-97239
Pragma: no-cache

Device:~#
```

PUT: (Create or Replace) Request:

If the specified command is not present on the device, the POST request creates it ; however, if it is already present in the running configuration, the command will be replaced by this request.

The following sample PUT request uses the **logging monitor warnings** command.

```
Device:~# curl -i -k -X "PUT"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin' \
>      -d $'{
>   "severity": "warnings"
> }'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 14:58:36 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 14:57:46 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-495466-326956
Pragma: no-cache

Device:~#
```

PATCH: (Update) Request

The following sample PATCH request uses the **logging monitor informational** command.

```
Device:~# curl -i -k -X "PATCH"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native" \
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin' \
>      -d $'{
>   "native": {
>     "logging": {
>       "monitor": {
>         "severity": "informational"
>       }
>     }
>   }
> }'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 15:07:56 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 15:07:56 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-496076-273016
Pragma: no-cache
Device:~#
```

GET Request (To Read)

The following sample GET request uses the **logging monitor informational** command.

```
Device:~# curl -i -k -X "GET"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin'
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 23 Apr 2018 15:10:59 GMT
Content-Type: application/yang-data+json
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Pragma: no-cache

{
  "Cisco-IOS-XE-native:severity": "informational"
}
Device:~#
```

DELETE Request (To Delete the Configuration)

```
Device:~# curl -i -k -X "DELETE"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>       -H 'Content-Type: application/yang-data+json' \
>       -H 'Accept: application/yang-data+json' \
>       -u 'admin:admin'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 15:26:05 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 15:26:05 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-497165-473206
Pragma: no-cache

linux_host:~#
```

Additional References for the RESTCONF Protocol

Related Documents

Related Topic	Document Title
YANG data models for various releases of IOS XE, IOS XR, and NX-OS platforms	To access Cisco YANG models in a developer-friendly way, please clone the GitHub repository, and navigate to the vendor/cisco subdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.

Standards and RFCs

Standard/RFC	Title
RFC 6020	*YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*
RFC 8040	*RESTCONF Protocol*
RFC 8072	*YANG Patch Media Type*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	https://www.cisco.com/c/en/us/support/index.html

Feature Information for the RESTCONF Protocol

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

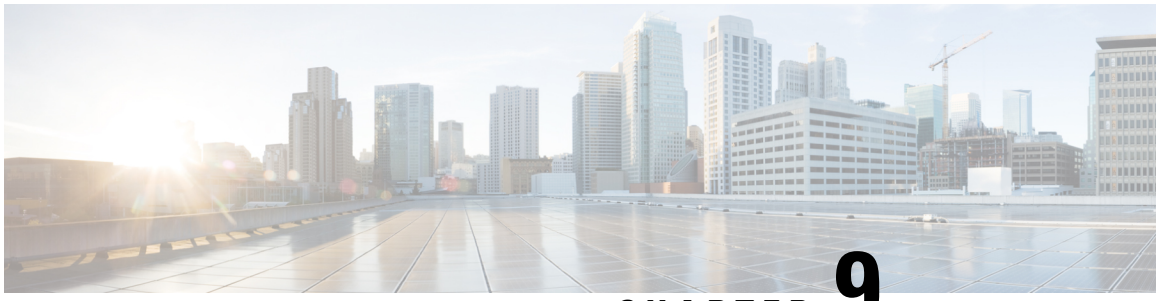
Table 15: Feature Information for the RESTCONF Protocol

Feature Name	Releases	Feature Information
RESTCONF Protocol	Cisco IOS XE Everest 16.6.1	RESTCONF provides a programmatic interface based on standard mechanisms for accessing configuration data, state data, data-model-specific RPC operations and event notifications defined in the YANG model. This feature was introduced on the following platforms: • Cisco 4000 Series Integrated Services Router • Cisco ASR 1000 Aggregation Services Routers • Cisco Cloud Services Router 1000V Series The following commands were introduced or modified: **ip http server** and **restconf**
	Cisco IOS XE Fuji 16.8.1a	In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Router • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Fuji 16.9.2	In Cisco IOS XE Fuji 16.9.2, this feature was implemented on the following platforms: • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs
	Cisco IOS XE Gibraltar 16.11.1	

Feature Name	Releases	Feature Information
		In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on the following platforms: • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CL Wireless Controllers • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco Network Convergence System 520 Series
	Cisco IOS XE Gibraltar 16.12.1	In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on Cisco Catalyst 9800-L Wireless Controllers.
	Cisco IOS XE Amsterdam 17.3.1	In Cisco IOS XE Amsterdam 17.3.1, this feature was implemented on the following platforms: • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 and 8500L Series Edge Platforms
	Cisco IOS XE Bengaluru 17.4.1	In Cisco IOS XE Bengaluru 17.4.1, this feature was implemented on Cisco Catalyst 8000V Edge Software.

Feature Name	Releases	Feature Information
RESTCONF YANG-Patch Support	Cisco IOS XE Amsterdam 17.1.1	RESTCONF supports YANG-Patch media type as specified by RFC 8072. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers (ASR1000-RP2, ASR1000-RP3, ASR1001-HX, ASR1001-X, ASR1002-HX, ASR1002-X) • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
NETCONF and RESTCONF IPv6 Support	Cisco IOS XE Fuji 16.8.1a	• Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco CBR-8 Series Routers • Cisco Cloud Services Router 1000V Series
	Cisco IOS XE Gibraltar 16.11.1	In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches.

Feature Name	Releases	Feature Information
Converting IOS Commands to XML	Cisco IOS XE Cupertino 17.7.1	This feature helps to automatically translate IOS commands into relevant NETCONF-XML or RESTCONF/JSON request messages. This feature is supported on all platforms that support RESTCONF.
Named Method List	Cisco IOS XE Cupertino 17.9.1	With the introduction of the Named Method List feature, it is possible to use a custom method-list name for authentication and authorization, without changing the existing AAA configuration of a device. Prior to this feature, only the default method-list was supported. For more information, see the NETCONF Protocol chapter. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 Series and 8500L Series Edge Platforms • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800 Series Wireless Controllers • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series



CHAPTER 9

NETCONF and RESTCONF Service-Level ACLs

This module describes the service-levels ACLs supported on NETCONF and RESTCONF, and how to configure it.

- Information About NETCONF and RESTCONF Service-Level ACLs, on page 241
- How to Configure NETCONF and RESTCONF Service-Level ACLs, on page 241
- Configuration Examples for NETCONF and RESTCONF Service-Level ACLs, on page 244
- Additional References for NETCONF and RESTCONF Service-Level ACLs, on page 245
- Feature Information for NETCONF and RESTCONF Service-Level ACLs, on page 246

Information About NETCONF and RESTCONF Service-Level ACLs

Overview of NETCONF and RESTCONF Service-Level ACLs

You can configure an IPv4 or IPv6 access control list (ACL) for NETCONF and RESTCONF sessions. Clients that do not conform to the configured ACLs are not allowed to access the NETCONF or RESTCONF subsystems. When service-level ACLs are configured, NETCONF-YANG and RESTCONF connection requests are filtered based on the source IP address.

If no service-level ACLs are configured, all NETCONF-YANG and RESTCONF connection requests are permitted into the subsystems.

Note Only named ACLs are supported; numbered ACLs are not supported.

How to Configure NETCONF and RESTCONF Service-Level ACLs

Configuring an ACL for a NETCONF-YANG Session

You can either configure an IP access-list or an IPv6 access list for your NETCONF-YANG session.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. • **ip access-list** {**standard** | **extended**} *access-list-name*
   • **ipv6 access-list** *access-list-name*
4. **permit** {*host-address* | *host-name* | **any**} [*wildcard*]
5. **deny** {*host-address* | *host-name* | **any**} [*wildcard*]
6. **exit**
7. **netconf-yang ssh** {{**ipv4** | **ipv6** }**access-list name** *access-list-name*} | **port** *port-number*}
8. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 3	• **ip access-list** {**standard**	**extended**} *access-list-name*
		• **ipv6 access-list** *access-list-name*
		Example:
		`Device(config)# ip access-list standard acl1_permit`
		`Device(config)# ipv6 access-list ipv6-acl1_permit`
			• Specifies a standard IP access list and enters standard access-list configuration mode.
			• Specifies an IPv6 access list and enters IPv6 access-list configuration mode.

Step 4	**permit** {*host-address*	*host-name*	**any**} [*wildcard*] **Example:** `Device(config-std-nacl)# permit 192.168.255.0 0.0.0.255`	Sets conditions in an IP/IPv6 access list that will permit packets.
Step 5	**deny** {*host-address*	*host-name*	**any**} [*wildcard*] **Example:** `Device(config-std-nacl)# deny any`	Sets conditions in an IP or IPv6 access list that will deny packets.
Step 6	**exit** **Example:** `Device(config-std-nacl)# exit`	Exits standard access-list configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 7	**netconf-yang ssh** {{**ipv4** | **ipv6** }**access-list name** *access-list-name*} | **port** *port-number*} **Example:** `Device(config)# netconf-yang ssh ipv4 access-list name acl1_permit`	Configures an ACL for the NETCONF-YANG session.
Step 8	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring an ACL for a RESTCONF Session

You can either configure an IP access list or an IPv6 access list for your RESTCONF session.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. • **ip access-list** {**standard** | **extended**} *access-list-name*
   • **ipv6 access-list** *access-list-name*
4. **permit** {*protocol-number* | *ipv6-source-address* | *ipv6-source-prefix* | *protocol*}**any**
5. **deny** {*protocol-number* | *ipv6-source-address* | *ipv6-source-prefix* | *protocol*}**any any**
6. **exit**
7. **restconf** {**ipv4** | **ipv6** }**access-list name** *access-list-name*
8. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	• **ip access-list** {**standard** | **extended**} *access-list-name* • **ipv6 access-list** *access-list-name* **Example:**	• Specifies a standard IP access list and enters standard access-list configuration mode. • Specifes an IPv6 access list and enters IPv6 access list configuration mode.

	Command or Action	Purpose

	`Device(config)# ip access-list standard acl1_permit`
	
	`Device(config)# ipv6 access-list ipv6-acl1_permit`	

Step 4	**permit** {*protocol-number* | *ipv6-source-address* | *ipv6-source-prefix* | *protocol*} **any** **Example:** `Device(config-ipv6-acl)# permit ipv6 2001:db8::1/32 any`	Sets conditions in an IPv6 access list that will permit packets.
Step 5	**deny** {*protocol-number* | *ipv6-source-address* | *ipv6-source-prefix* | *protocol*} **any any** **Example:** `Device(config-ipv6-acl)# deny ipv6 any any`	Sets conditions in an IPv6 access list that will deny packets.
Step 6	**exit** **Example:** `Device(config-ipv6-acl)# exit`	Exits IPv6 access list configuration mode and returns to global configuration mode.
Step 7	**restconf** {**ipv4** | **ipv6** } **access-list name** *access-list-name* **Example:** `Device(config)# restconf ipv6 access-list name ipv6-acl1_permit`	Configures an ACL for the RESTCONF session.
Step 8	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configuration Examples for NETCONF and RESTCONF Service-Level ACLs

Example: Configuring an ACL for a NETCONF Session

```
Device# enable
Device# configure terminal
Device(config)# ip access-list standard acl1_permit
Device(config-std-nacl)# permit 192.168.255.0 0.0.0.255
Device(config-std-nacl)# deny any
Device(config-std-nacl)# exit
Device(config)# netconf-yang ssh ipv4 access-list name acl1_permit
Device(config)# end
```

Example: Configuring an ACL for a RESTCONF Session

```
Device# enable
Device# configure terminal
Device(config)# ipv6 access-list ipv6-acl1_permit
Device(config-ipv6-acl)# permit ipv6 2001:db8::1/32 any
Device(config-ipv6-acl)# deny ipv6 any any
Device(config-ipv6-acl)# exit
Device(config)# restconf ipv6 access-list name ipv6-acl1_permit
Device(config)# end
```

Additional References for NETCONF and RESTCONF Service-Level ACLs

Related Documents

Related Topic	Document Title
NETCONF-YANG	NETCONF Protocol
RESTCONF	RESTCONF Protocol
Programmability commands	*Programmability Command Reference*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

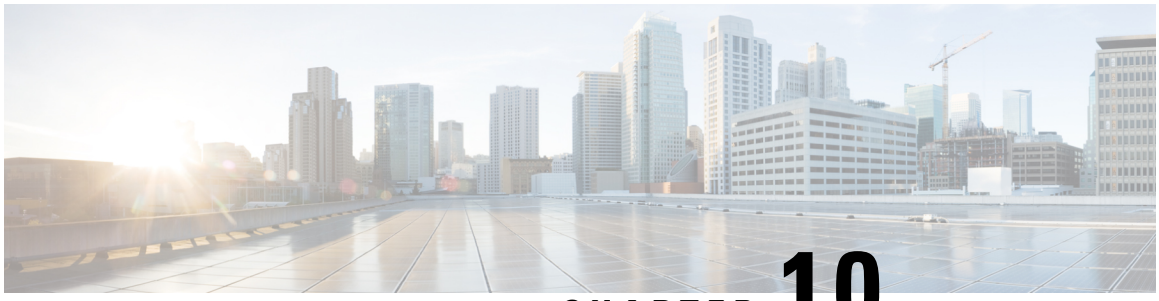
Feature Information for NETCONF and RESTCONF Service-Level ACLs

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 16: Feature Information for NETCONF and RESTCONF Service-Level ACLs

Feature Name	Releases	Feature Information
NETCONF and RESTCONF Service-Level ACLs	Cisco IOS XE Everest 16.11.1	You can configure an access control list (ACL) for NETCONF and RESTCONF sessions. Clients that do not conform to the configured ACL are not allowed to access the NETCONF or RESTCONF subsystems. The following commands were introduced or modified: **netconf-yang ssh access-list** and **restconf access-list** • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Routers (RSP2) • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst IE 3200, 3300, 3400 Rugged Series • Cisco Embedded Services 3300 Series Switches • Cisco IR1101 Integrated Services Router Rugged • Cisco Network Convergence System 4200 Series • Cisco Network Convergence System 520 Series



CHAPTER 10

gNMI Protocol

The gNMI Protocol feature describes the model-driven configuration and retrieval of operational data using the gRPC Network Management Interface (gNMI) capabilities, and the Get, Set, and Subscribe remote procedure calls (RPCs). gNMI Version 0.4.0 is supported.

- Restrictions for gNMI Protocol, on page 247
- Information About the gNMI Protocol, on page 248
- How to Enable the gNMI Protocol, on page 260
- Configuration Examples for the gNMI Protocol, on page 266
- Additional References for the gNMI Protocol, on page 266
- Feature Information for the gNMI Protocol, on page 267

Restrictions for gNMI Protocol

The following restrictions apply to the gNMI Protocol feature:

- BYTES and ASCII encoding options are not supported.

  PROTO encoding is supported from Cisco IOS XE Dublin 17.11.1.

- JSON IETF keys must contain a YANG prefix where the namespace of the child elements differs from that of the parent. This means that the routed VLAN derived from augmentation in openconfig-vlan.yang must be entered as *oc-vlan:routed-vlan* because it is different from the namespace of the parent nodes (parent nodes have the prefix oc-if)

- GetRequest:

  - Operational data filtering is not supported.

  - Use models are not supported. These are a set of model data messages indicating the schema definition modules that define the data elements that must be returned in response to a Get RPC call.

- GetResponse:

  - Alias, which is a string that provides an alternate name for a prefix specified within the GetResponse notification message is not supported.

  - Delete, which is a set of paths that are to be removed from a data tree is not supported.

Information About the gNMI Protocol

About GNMI

gNMI is gRPC Network Management Interface developed by Google. gNMI provides the mechanism to install, manipulate, and delete the configuration of network devices, and also to view operational data. The content provided through gNMI can be modeled using YANG.

gRPC is a remote procedure call developed by Google for low-latency, scalable distributions with mobile clients communicating to a cloud server. gRPC carries gNMI, and provides the means to formulate and transmit data and operation requests.

When a gNMI service failure occurs, the gNMI broker (GNMIB) will indicate an operational change of state from up to down, and all RPCs will return a service unavailable message until the database is up and running. Upon recovery, the GNMIB will indicate a change of operation state from down to up, and resume normal handling of RPCs.

gNMI supports <subscribe> RPC services. For more information, see the Model-Driven Telemetry chapter.

JSON IETF Encoding for YANG Data Trees

RFC 7951 defines JavaScript Object Notation (JSON) encoding for YANG data trees and their subtrees. gNMI uses JSON for encoding data in its content layer.

The JSON type indicates that the value is encoded as a JSON string. JSON_IETF-encoded data must conform to the rules for JSON serialisation described in RFC 7951. Both the client and target must support JSON encoding.

Instances of YANG data nodes (leafs, containers, leaf-lists, lists, anydata nodes, and anyxml nodes) are encoded as members of a JSON object or name/value pairs. Encoding rules are identical for all types of data trees, such as configuration data, state data, parameters of RPC operations, actions, and notifications.

Every data node instance is encoded as a name/value pair where the name is formed from the data node identifier. The value depends on the category of the data node.

The leaf Data Node

A leaf node has a value, but no children, in a data tree. A leaf instance is encoded as a name/value pair. This value can be a string, number, literal true or false, or the special array [null], depending on the type of the leaf. In the case that the data item at the specified path is a leaf node (which means it has no children, and an associated value) the value of that leaf is encoded directly. (A bare JSON value is included; it does not require a JSON object.)

The following example shows a leaf node definition:

```
leaf foo {
 type uint8;
}
```

The following is a valid JSON-encoded instance:

```
"foo": 123
```

Proto Encoding

gNMI protocol supports PROTO encoding along with the already supported JSON and JSON_IETF formats. The *gnmi.proto* file represents the blueprint for generating a complete set of client and server-side procedures that instantiate the framework for the gNMI protocol.

The existing encoding procedure for JSON and JSON_IETF formats push the input data that causes the output values to have premature wraps and other issues, causing loss of data precision.

PROTO encoding allows the querying of applications to retrieve data in scalar (TypedValue) values; each leaf is sent on its own update as per the gNMI specification. This allows customers to have access to float and double values for more accuracy.

PROTO encoding is supported for all the paths that are supported by gNMI. PROTO encoding supports subscribe RPC, but not GET and SET RPCs.

PROTO encoding is part of the gNMI protocol, and is enabled when the gNMI feature is enabled.

The following is a sample subscribeRequest RPC with PROTO:

```
subscribe:
           prefix:
               origin: "legacy"
               elem:
                   name: "mdt-oper-v2:mdt-oper-v2-data"
               >
           >
           subscription:
             path:
              elem:
                  name: "mdt-subscriptions"
                  >
              >
              mode: SAMPLE
              sample_interval: 10000000000
          >
          mode: STREAM
          encoding: PROTO
      >
```

The following is a sample subscribeRequest response with PROTO:

```
update: <
  timestamp: 1652408966709576000
  prefix: <
    origin: "openconfig"
    elem: <
      name: "oc-if:interfaces"
    >
    elem: <
      name: "interface"
      key: <
        key: "name"
        value: "*"
      >
    >
    elem: <
      name: "state"
    >
  update: <
    path: <
      origin: "openconfig"
```

```
      elem: <
        name: "interfaces"
      >
      elem: <
        name: "interface"
        key: <
          key: "name"
          value: "GigabitEthernet3"
        >
      >
      elem: <
        name: "ethernet"
      >
      elem: <
        name: "config"
      >
    >
    val: <
      string_val: "{
\"mac-address\":\"00:0c:29:29:42:e9\"
  }"
    >
  >
update: <
    path: <
      origin: "openconfig"
      elem: <
        name: "interfaces"
      >
      elem: <
        name: "interface"
        key: <
          key: "name"
          value: "GigabitEthernet3"
        >
      >
      elem: <
        name: "ethernet"
      >
      elem: <
        name: "config"
      >
    >
    val: <
      bool_val: "{
\",\"auto-negotiate\":true
  }"
    >
  >
update: <
    path: <
      origin: "openconfig"
      elem: <
        name: "interfaces"
      >
      elem: <
        name: "interface"
        key: <
          key: "name"
          value: "GigabitEthernet3"
        >
      >
      elem: <
        name: "ethernet"
```

```
      >
      elem: <
        name: "config"
      >
    >
    val: <
      bool_val: "{
\"enable-flow-control\":true
  }"
    >
  >
```

Errors are reported using the *status.proto* message in the RPC return message.

gNMI GET Request

The gNMI Get RPC specifies how to retrieve one or more of the configuration attributes, state attributes, derived state attributes, or all attributes associated with a supported mode from a date tree. A GetRequest is sent from a client to the target to retrieve values from the data tree. A GetResponse is sent in response to a GetRequest.

GetRequest JSON Structure

The following is a sample GetRequest JSON structure. Both the GetRequest and GetResponse are displayed.

GetRequest

```
The following is a path for the
openconfig-interfaces model
++++++++ Sending get request: ++++++++
path {
  elem {
    name: "interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
      value: "Loopback111"
    }
  }
}
```

GetResponse

```
encoding: JSON_IETF
++++++++ Received get response: ++++++++
notification {
  timestamp: 1521699434792345469
  update {
    path {
      elem {
        name: "interfaces"
      }
      elem {
        name: "interface"
        key {
          key: "name"
          value: "\"Loopback111\""
        }
```

```
  }
}


val {
  json_ietf_val: "{\n\t\"openconfig-interfaces:name\":\t\
  "Loopback111\",\n\t\
   "openconfig-interfaces:config\":\t{\n\t\t\
   "openconfig-interfaces:type\":\t\"ianaift:
    softwareLoopback\",\n\t\t\
   "openconfig-interfaces:name\":\t\"Loopback111\",\n\t\t\
   "openconfig-interfaces:enabled\":\t\"true\"\n\t},\n\t\
   "openconfig-interfaces:state\":\t{\n\t\t\
   "openconfig-interfaces:type\":\t\"ianaift:
    softwareLoopback\",\n\t\t\
   "openconfig-interfaces:name\":\t\"Loopback111\",\n\t\t\
   "openconfig-interfaces:enabled\":\t\"true\",\n\t\t\
   "openconfig-interfaces:ifindex\":\t52,\n\t\t\


   "openconfig-interfaces:admin-status\":\t\"UP\",\n\t\t\
   "openconfig-interfaces:oper-status\":\t\"UP\",\n\t\t\
   "openconfig-interfaces:last-change\":\t2018,\n\t\t\
   "openconfig-interfaces:counters\":\t{\n\t\t\t\
   "openconfig-interfaces:in-octets\":\t0,\n\t\t\t\
   "openconfig-interfaces:in-unicast-pkts\":\t0,\n\t\t\t\
   "openconfig-interfaces:in-broadcast-pkts\":\t0,\n\t\t\t\
   "openconfig-interfaces:in-multicast-pkts\":\t0,\n\t\t\t\
   "openconfig-interfaces:in-discards\":\t0,\n\t\t\t\
   "openconfig-interfaces:in-errors\":\t0,\n\t\t\t\
   "openconfig-interfaces:in-unknown-protos\":\t0,\n\t\t\t\
   "openconfig-interfaces:out-octets\":\t0,\n\t\t\t\
   "openconfig-interfaces:out-unicast-pkts\":\t0,\n\t\t\t\
   "openconfig-interfaces:out-broadcast-pkts\":\t0,\n\t\t\t\
   "openconfig-interfaces:out-multicast-pkts\":\t0,\n\t\t\t\
   "openconfig-interfaces:out-discards\":\t0,\n\t\t\t\
   "openconfig-interfaces:out-errors\":\t0,\n\t\t\t\
   "openconfig-interfaces:last-clear\":\t2018\n\t\t},\n\t\t\

   "openconfig-platform:hardware-port\":\t\
   "Loopback111\"\n\t},\n\t\
   "openconfig-interfaces:subinterfaces\":\t{\n\t\t\
   "openconfig-interfaces:index\":\t0,\n\t\t\
   "openconfig-interfaces:config\":\t{\n\t\t\t\
   "openconfig-interfaces:index\":\t0,\n\t\t\t\
   "openconfig-interfaces:name\":\t\"Loopback111\",\n\t\t\t\
   "openconfig-interfaces:enabled\":\t\"true\"\n\t\t},\n\t\t\
   "openconfig-interfaces:state\":\t{\n\t\t\t\
   "openconfig-interfaces:index\":\t0,\n\t\t\t\
   "openconfig-interfaces:name\":\t\"Loopback111.0\",\n\t\t\t\
   "openconfig-interfaces:enabled\":\t\"true\",\n\t\t\t\
   "openconfig-interfaces:admin-status\":\t\"UP\",\n\t\t\t\
   "openconfig-interfaces:oper-status\":\t\"UP\",\n\t\t\t\
   "openconfig-interfaces:last-change\":\t2018,\n\t\t\t\
   "openconfig-interfaces:counters\":\t{\n\t\t\t\t\
   "openconfig-interfaces:in-octets\":\t0,\n\t\t\t\t\
   "openconfig-interfaces:in-unicast-pkts\":\t0,\n\t\t\t\t\
   "openconfig-interfaces:in-broadcast-pkts\":\t0,\n\t\t\t\t\
   "openconfig-interfaces:in-multicast-pkts\":\t0,\n\t\t\t\t\
   "openconfig-interfaces:in-discards\":\t0,\n\t\t\t\t\
   "openconfig-interfaces:in-errors\":\t0,\n\t\t\t\t\

   "openconfig-interfaces:out-octets\":\t0,\n\t\t\t\t\
   "openconfig-interfaces:out-unicast-pkts\":\t0,\n\t\t\t\t\
```

```
        "openconfig-interfaces:out-broadcast-pkts\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:out-multicast-pkts\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:out-discards\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:out-errors\":\t0,\n\t\t\t\t\
        "openconfig-interfaces:last-clear\":\
         t2018\n\t\t\t}\n\t\t},\n\t\t\
        "openconfig-if-ip:ipv6\":\t{\n\t\t\t\
        "openconfig-if-ip:config\":\t\"false\",\n\t\t\t\
        "openconfig-if-ip:state\":\t\"false\"\n\t\t}\n\t}\n}"
    }
  }
}
```

GetRequest on a Leaf Value

The following is a sample GetRequest on a Leaf value. Both the GetRequest and the corresponding GetResponse are displayed.

GetRequest

```
++++++++ Sending get request: ++++++++
path {
  elem {
    name: "interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
      value: "Loopback111"
    }
  }
  elem {
    name: "state"
  }
  elem {
    name: "oper-status"
  }
}
```

GetResponse

```
encoding: JSON_IETF
++++++++ Received get response: ++++++++
notification {
  timestamp: 1521699326012374332
  update {
    path {
      elem {
        name: "interfaces"
      }
      elem {
        name: "interface"
        key {
          key: "name"
          value: "\"Loopback111\""
        }
      }
      elem {
        name: "state"
```

```
      }
      elem {
        name: "oper-status"
      }
    }
    val {
      json_ietf_val: "\"UP\""
    }
  }
}
```

gNMI SetRequest

The Set RPC specifies how to set one or more configurable attributes associated with a supported model. A SetRequest is sent from a client to a target to update the values in the data tree.

SetRequests also support JSON keys, and must contain a YANG-prefix, in which the namespace of the element differs from parent.

For example, the *routed-vlan* element derived from augmentation in openconfig-vlan.yang must be entered as *oc-vlan:routed-vlan*, because it is different from the namespace of the parent node (The parent node prefix is oc-if.).

The total set of deletes, replace, and updates contained in any one SetRequest is treated as a single transaction. If any subordinate element of the transaction fails; the entire transaction is disallowed and rolled back. A SetResponse is sent back for a SetRequest.

Table 17: Example of a SetRequest JSON Structure

SetRequest	SetResponse

++++++++ Sending set request: ++++++++
update {
  path {
  elem {
  name: "interfaces"
  }
  elem {
  name: "interface"
  key {
  key: "name"
  value: "Loopback111"
  }
  }
  elem {
  name: "config"
  }
  }
  val {
  json_ietf_val:
"{\"openconfig-interfaces:enabled\":\"false\"}"
 	
  }
}
	++++++++ Received set response: ++++++++
	response {
	  path {
	  elem {
	  name: "interfaces"
	  }
	  elem {
	  name: "interface"
	  key {
	  key: "name"
	  value: "Loopback111"
	  }
	  }
	  elem {
	  name: "config"
	  }
	  }
	  op: UPDATE
	}
	timestamp: 1521699342123890045

Table 18: Example of a SetRequest on Leaf Value

SetRequest	SetResponse

++++++++ Sending set request: ++++++++
update {
  path {
  elem {
  name: "interfaces"
  }
  elem {
  name: "interface"
  key {
  key: "name"
  value: "Loopback111"
  }
  }
  elem {
  name: "config"
  }
  elem {
  name: "description"
  }
  }
  val {
  json_ietf_val: "\"UPDATE DESCRIPTION\""
  }
}
	++++++++ Received set response: ++++++++
	response {
	  path {
	  elem {
	  name: "interfaces"
	  }
	  elem {
	  name: "interface"
	  key {
	  key: "name"
	  value: "Loopback111"
	  }
	  }
	  elem {
	  name: "config"
	  }
	  elem {
	  name: "description"
	  }
	  }
	  op: UPDATE
	}
	timestamp: 1521699342123890045

gNMI Namespace

A namespace specifies the path prefixing to be used in the *origin* field of a message.

This section describes the namespaces used in Cisco IOS XE Gibraltar 16.10.1 and later releases:

- RFC 7951-specified namespaces: Path prefixes use the YANG module name as defined in RFC 7951.

  The RFC 7951-specified value prefixing uses the YANG module name.

  Value prefixing is not affected by the selected path prefix namespace. The following example shows an RFC 7951-specified value prefix:

  ```
  val {
     json_ietf_val:"{
       "openconfig-interfaces:config": {
         “openconfig-interfaces:description":
           “DESCRIPTION”
       }
     }"
   }
  ```

  An RFC 7951-specified namespace prefixing also uses the YANG module name. For example, the openconfig path to a loopback interface will be

  ```
  /openconfig-interfaces:interfaces/interface[name=Loopback111]/
  ```

  The following example shows a gNMI path with RFC7951 namespacing:

  ```
  path {
    origin: “rfc7951"
    elem {
  ```

```
    name: "openconfig-interface:interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
      value: "Loopback111"
    }
  }
}
```

- Openconfig: No path prefixes are used. These can only be used with a path to an openconfig model.

  The behavior of the Openconfig namespace prefixing is the same when no origin or namespace is provided. For example, the openconfig path to a loopback interface will be

  ```
  /interfaces/interface[name=Loopback111]/
  ```

  The following example shows a gNMI path with an Openconfig namespacing:

  ```
  path {
    origin: "openconfig"
    elem {
      name: "interfaces"
    }
    elem {
      name: "interface"
      key {
        key: "name"
        value: "Loopback111"
      }
    }
  }
  ```

- Blank: Same as the openconfig prefix. This is the default.

  The following example shows a gNMI path with a blank Openconfig namespacing:

  ```
  path {
  elem {
      name: "interfaces"
    }
    elem {
      name: "interface"
      key {
        key: "name"
        value: "Loopback111"
      }
    }
  }
  ```

This section describes the path prefixing used in releases prior to Cisco IOS XE Gibraltar 16.10.1.

Here, path prefixing uses the YANG module prefix as defined in the YANG module definition. For example, the openconfig path to a loopback interface will be

```
/oc-if:interfaces/interface[name=Loopback111]/
```

The following example shows a gNMI Path with with legacy namespacing:

```
path {
  origin: "legacy"
  elem {
    name: "oc-if:interfaces"
  }
```

```
  elem {
    name: "interface"
    key {
      key: "name"
      value: "Loopback111"
    }
  }
}
```

gNMI Wildcards

The gNMI protocol supports wildcards for Get paths. This is the ability to use a wildcards in a path to match multiple elements. These wildcards indicate all elements in a given subtree in the schema.

An *elem* is an element, and it is a value between / characters in an xPath. An *elem* is also available in a gNMI path. For example, the position of a wildcard relative to *elem* names implies that the wildcard stands for an interface, and is interpreted as all interfaces.

There are two types of wildcards; implicit and explicit, and both are supported. Get paths support all types and combinations of path wildcards.

- Implicit wildcards: These expand a list of elements in an element tree. Implicit wildcard occurs when a key value is not provided for elements of a list.

  The following is a sample path implicit wildcard. This wildcard will return the descriptions of all interfaces on a device:

  ```
  path {
    elem {
      name: "interfaces"
    }
    elem {
      name: "interface"
    }
    elem {
      name: "config"
    }
    elem {
      name: "description"
    }
  }
  ```

- Explicit wildcards: Provides the same functionality by

  - Specifying an asterisk (*) for either the path element name or key name.

    The following sample shows a path asterisk wildcard as the key name. This wildcard returns the description for all interfaces on a device.

    ```
    path {
      elem {
        name: "interfaces"
      }
      elem {
        name: "interface"
        key {
         key: "name"
          value: "*"
        }
      }
      elem {
        name: "config"
    ```

```
  }
  elem {
    name: "description"
  }
}
```

The following sample shows a path asterisk wildcard as the path name. This wildcard will return the description for all elements that are available in the Loopback111 interface.

```
path {
  elem {
    name: "interfaces"
  }
  elem {
    name: "interface"
    key {
      key: "name"
      value: "Loopback111"
    }
  }
   elem {
    name: "*"
  }
  elem {
    name: "description"
  }
}
```

- Specifying an ellipsis (...) or a blank entry as element names. These wildcards can expand to multiple elements in a path.

  The following sample shows a path ellipsis wildcard. This wildcard returns all description fields available under /interfaces.

```
path {
  elem {
    name: "interfaces"
  }
  elem {
    name: "..."
  }
  elem {
    name: "description"
  }
}
```

The following is a sample GetRequest with an implicit wildcard. This GetRequest will return the oper-status of all interfaces on a device.

```
path {
  elem {
    name: "interfaces"
    }
  elem {
    name: "interface"
  }
  elem {
    name: "state"
  }
  elem {
    name: "oper-status"
  }
```

```
},
type: 0,
encoding: 4
```

The following is a sample GetResponse with an implicit wildcard:

```
notification {
  timestamp: 1520627877608777450
  update {
    path {
      elem {
        name: "interfaces"
      }
      elem {
        name: "interface"
        key {
          key: "name"
          value: "\"FortyGigabitEthernet1/1/1\""
        }
      }
      elem {
        name: "state"
}
      elem {
        name: "oper-status"
      }
    }
    val {
      json_ietf_val: "\"LOWER_LAYER_DOWN\""
    }
  },

<snip>
…
</snip>

update {
    path {
      elem {
        name: "interfaces"
      }
      elem {
        name: "interface"
        key {
          key: "name"
          value: "\"Vlan1\""
        }
      }
      elem {
        name: "state"
      }
      elem {
        name: "oper-status"
      }
    }
    val {
      json_ietf_val: "\"DOWN\""
    }
  }
}
```

gNMI Configuration Persistence

The gNMI Configuration Persistence feature ensures that all successful configuration changes made through the gNMI SetRequest RPC persists across device restarts. Prior to this feature, the gNMI configuration was stored in the running configuration of a device. And the changes were saved by issuing the **write memory** command, or the SaveConfig NETCONF RPC.

All changes in the running configuration, even if the data was modified by processes other than gNMI, the data is saved to the startup configuration, when the SetRequest RPC is issued.

This feature is enabled by default and cannot be disabled.

gNMI Username and Password Authentication

User credentials, the username and password provide authorization as metadata in each gNMI RPC. The following is a sample gNMI Capabilities RPC that use the username and password:

```
metadata = [('username','admin'), ('password','lab')]
cap_request = gnmi_pb2.CapabilityRequest()
# pass metadata to the gnmi_pb2_grpc.gNMIStub object
secure_stub.Capabilities(cap_request, metadata=metadata)
```

gNMI Error Messages

When errors occur, gNMI returns descriptive error messages. The following section displays some gNMI error messages.

The following sample error message is displayed when the path is invalid:

```
gNMI Error Response:
  <_Rendezvous of RPC that terminated with (StatusCode.TERMINATED,
   An error occurred while parsing provided xpath: unknown tag:
   "someinvalidxpath" Additional information: badly formatted or nonexistent path)>
```

The following sample error message is displayed for an unimplemented error:

```
gNMI Error Response:
  <_Rendezvous of RPC that terminated with (StatusCode.UNIMPLEMENTED,
    Requested encoding "ASCII" not supported)>
```

The following sample error message is displayed when the data element is empty:

```
gNMI Error Response:
  <_Rendezvous of RPC that terminated with (StatusCode.NOT_FOUND,
   Empty set returned for path "/oc-if:interfaces/noinfohere")>
```

How to Enable the gNMI Protocol

Perform the following steps to enable the gNMI protocol:

1. Create a set of certs for the gNMI client and device signed by a Certificate Authority (CA).
   a. Create Certs with OpenSSL on Linux.
   b. Install Certs on a device.
   c. Configure gNMI on the device.
   d. Verify whether gNMI is enabled and running.
2. Connect the gNMI client using client and root certificates configured in previous steps.

Creating Certs with OpenSSL on Linux

Certs and trustpoint are only required for secure gNMI servers.

The following example shows how to create Certs with OpenSSL on a Linux machine:

```
# Setting up a CA
openssl genrsa -out rootCA.key 2048
openssl req -subj /C=/ST=/L=/O=/CN=rootCA -x509 -new -nodes -key rootCA.key -sha256 -out
rootCA.pem

# Setting up device cert and key
openssl genrsa -out device.key 2048
openssl req -subj /C=/ST=/L=/O=/CN=<hostnameFQDN> -new -key device.key -out device.csr
openssl x509 -req -in device.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out
device.crt -sha256
openssl pkcs12 -export -out mycert.pfx -inkey device.key -in device.crt -certfile rootCA.pem
 -aes128
Enter Export Password:cisco
Verifying - Enter Export Password:cisco

# Setting up client cert and key
openssl genrsa -out client.key 2048
openssl req -subj /C=/ST=/L=/O=/CN=gnmi_client -new -key client.key -out client.csr
openssl x509 -req -in client.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out
client.crt -sha256
```

Installing Certs on a Device Through the CLI

The following example show how to install certs on a device:

```
# Send:
Device# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Device(config)# crypto pki trustpoint trustpoint1
Device(ca-trustpoint)# revocation-check none
Device(ca-trustpoint)# crypto pki import AESTEST pkcs12 bootflash:mycert.pfx password cisco
% Importing pkcs12...
Source filename [mycert.pfx]?
Reading file from bootflash:mycert.pfx
CRYPTO_PKI: Imported PKCS12 file successfully.
Device(config)#
```

Enabling gNMI in Insecure Mode

Note This task is applicable in Cisco IOS XE Amsterdam 17.3.1 and later releases.

In a Day Zero setup, first enable the device in insecure mode, then disable it, and enable the secure mode. To stop gNxI in insecure mode, use the **no gnxi server** command.

Note gNxI insecure and secure servers can run simultaneously on a device.

Note The **gnxi** commands apply to both gNMI and gRPC Network Operations Interface (gNOI) services. gNxI tools are a collection of tools for Network Management that use the gNMI and gNOI protocols.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **gnxi**
4. **gnxi server**
5. **gnxi port** *port-number*
6. **end**
7. **show gnxi state**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**gnxi** **Example:** `Device(config)# gnxi`	Starts the gNxI process.

	Command or Action	Purpose
Step 4	**gnxi server** **Example:** `Device(config)# gnxi server`	Enables the gNxI server in insecure mode.
Step 5	**gnxi port** *port-number* **Example:** `(Optional) Device(config)# gnxi port 50000`	Sets the gNxI port to listen to. • The default insecure gNxI port is 50052.
Step 6	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.
Step 7	**show gnxi state** **Example:** `Device# show gnxi state`	Displays the status of gNxI interfaces.

Enabling gNMI in Secure Mode

Note This task is applicable in Cisco IOS XE Amsterdam 17.3.1 and later releases.

To stop gNxI in secure mode, use the **no gnxi secure-server** command.

Note gNxI insecure and secure servers can simultaneously run on a device.

Note The **gnxi** commands apply to both gNMI and gRPC Network Operations Interface (gNOI) services. gNxI tools are a collection of tools for Network Management that use the gNMI and gNOI protocols.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **gnxi**
4. **gnxi secure-trustpoint** *trustpoint-name*
5. **gnxi secure-server**
6. **gnxi secure-client-auth**
7. **gnxi secure-port**
8. **end**
9. **show gnxi state**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**gnxi** **Example:** `Device(config)# gnxi`	Starts the gNxI process.
Step 4	**gnxi secure-trustpoint** *trustpoint-name* **Example:** `Device(config)# gnxi secure-trustpoint trustpoint1`	Specifies the trustpoint and cert set that gNxI uses for authentication.
Step 5	**gnxi secure-server** **Example:** `Device(config)# gnxi secure-server`	Enables the gNxI server in secure mode.
Step 6	**gnxi secure-client-auth** **Example:** `Device(config)# gnxi secure-client-auth`	(Optional) The gNxI process authenticates the client certificate against the root certificate.
Step 7	**gnxi secure-port** **Example:** `Device(config)# gnxi secure-port`	(Optional) Sets the gNxI port to listen to. • The default secure gNxI port is 9339.
Step 8	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.
Step 9	**show gnxi state** **Example:** `Device# show gnxi state`	Displays the status of gNxI servers.

Example

The following is sample output from the **show gnxi state** command:

```
Device# show gnxi state

State          Status
--------------------------------
Enabled        Up
```

Connecting the gNMI Client

The gNMI client is connected by using the client and root certificates that are previously configured.

The following example shows how to connect the gNMI client using Python:

```
# gRPC Must be compiled in local dir under path below:
>>> import sys
>>> sys.path.insert(0, "reference/rpc/gnmi/")
>>> import grpc
>>> import gnmi_pb2
>>> import gnmi_pb2_grpc
>>> gnmi_dir = '/path/to/where/openssl/creds/were/generated/'

# Certs must be read in as bytes
>>> with open(gnmi_dir + 'rootCA.pem', 'rb') as f:
>>>     ca_cert = f.read()
>>> with open(gnmi_dir + 'client.crt', 'rb') as f:
>>>     client_cert = f.read()
>>> with open(gnmi_dir + 'client.key', 'rb') as f:
>>>     client_key = f.read()

# Create credentials object
>>> credentials = grpc.ssl_channel_credentials(root_certificates=ca_cert,
private_key=client_key, certificate_chain=client_cert)

# Create a secure channel:
# Default port is 9339, can be changed on ios device with 'gnxi secure-port ####'
>>> port = 9339
>>> host = <HOSTNAME FQDN>
>>> secure_channel = grpc.secure_channel("%s:%d" % (host, port), credentials)

# Create secure stub:
>>> secure_stub = gnmi_pb2_grpc.gNMIStub(secure_channel)

# Done! Let's test to make sure it works:
>>> secure_stub.Capabilities(gnmi_pb2.CapabilityRequest())
supported_models {
<snip>
}
supported_encodings: <snip>
gNMI_version: "0.4.0"
```

Configuration Examples for the gNMI Protocol

Example: Enabling gNMI in Insecure Mode

Note This example is applicable in Cisco IOS XE Amsterdam 17.3.1 and later releases.

The following example shows how to enable the gNxI server in insecure mode:

```
Device> enable
Device# configure terminal
Device(config)# gnxi
Device(config)# gnxi server
Device(config)# gnxi port 50000 <The default port is 50052.>
Device(config)# end
Device#
```

Example: Enabling gNMI in Secure Mode

Note This example is applicable in Cisco IOS XE Amsterdam 17.3.1 and later releases.

The following example shows how to enable the gNxI server in secure mode:

```
Device> enable
Device# configure terminal
Device(config)# gnxi
Device(config)# gnxi secure-trustpoint trustpoint1
Device(config)# gnxi secure-server
Device(config)# gnxi secure-client-auth
Device(config)# gnxi secure-port 50001 <The default port is 9339.>
Device(config)# end
Device#
```

Additional References for the gNMI Protocol

Related Documents

Related Topic	Document Title
DevNet	https://developer.cisco.com/site/ios-xe/
gNMI	https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md

Related Topic	Document Title
gNMI path encoding	https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-path-conventions.md

Standards and RFCs

Standard/RFC	Title
RFC 7951	JSON Encoding of Data Modeled with YANG

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for the gNMI Protocol

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 19: Feature Information for the gNMI Protocol

Feature Name	Release	Feature Information
gNMI Protocol	Cisco IOS XE Fuji 16.8.1a	This feature describes the model-driven configuration and retrieval of operational data using the gNMI capabilities, GET and SET RPCs. This feature was implemented on the following platforms: • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches
	Cisco IOS XE Gibraltar 16.10.1	In Cisco IOS XE Gibraltar 16.10.1, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches.
	Cisco IOS XE Gibraltar 16.11.1	In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on Cisco Catalyst 9600 Series Switches.
	Cisco IOS XE Gibraltar 16.12.1	In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on the following platforms: • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs • Cisco cBR-8 Converged Broadband Router
	Cisco IOS XE Amsterdam 17.1.1	In Cisco IOS XE Amsterdam 17.1.1, this feature was implemented on the following platforms: • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Router • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Amsterdam 17.2.1r	In Cisco IOS XE Amsterdam 17.2.1r, this feature was implemented on Cisco ASR 1000 Series Aggregation Services Routers.

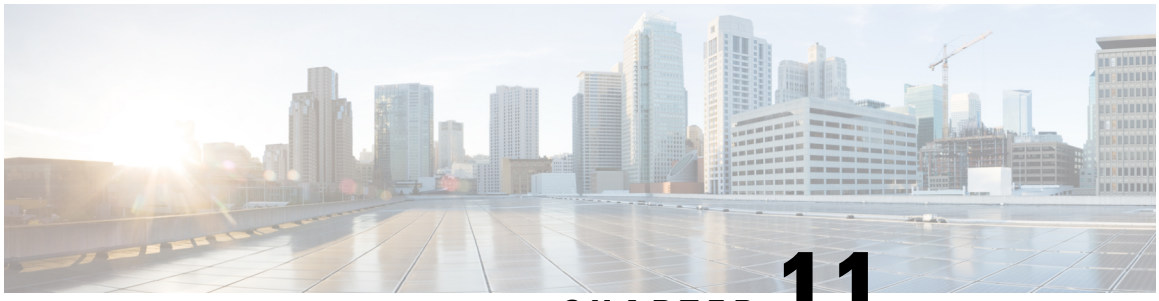
Feature Name	Release	Feature Information
	Cisco IOS XE Cupertino 17.8.1	In Cisco IOS XE Cupertino 17.8.1, this feature was implemented on the following platforms: • Cisco Catalyst 9800-CL Wireless Controllers • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers
gNMI IPv6 Support	Cisco IOS XE Dublin 17.10.1	gNMI IPv6 support was enabled in Cisco IOS XE Dublin 17.10.1. This feature was implemented on the following platforms: • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500 High-Performance Series Switches • Cisco Catalyst 9600 Series Switches
gNMI Username and Password Authentication	Cisco IOS XE Gibraltar 16.12.1	The Username and Password Authentication feature was added to the gNMI protocol. This feature is supported on all IOS XE platforms that support gNMI.
gNMI Configuration Persistence	Cisco IOS XE Amsterdam 17.3.1	All successful configuration changes made through the gNMI SetRequest RPC persists across device restarts. This feature was implemented on the following platforms: • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches

Feature Name	Release	Feature Information
gNOI Certificate Management	Cisco IOS XE Amsterdam 17.3.1	The gNOI Certificate Management Service provides RPCs to install, rotate, get certificate, revoke certificate, and generate certificate signing request. This feature was implemented on the following platforms: • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches

Feature Name	Release	Feature Information
Named Method List	Cisco IOS XE Cupertino 17.9.1	

Feature Name	Release	Feature Information
		With the introduction of the Named Method List feature, it is possible to use a custom method-list name for authentication and authorization, without changing the existing AAA configuration of a device. Prior to this feature, only the default method-list was supported. For more information, see the NETCONF Protocol chapter. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 Series and 8500L Series Edge Platforms • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800 Series Wireless Controllers • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System

Feature Name	Release	Feature Information
		4200 Series
PROTO Encoding	Cisco IOS XE Dublin 17.11.1	gNMI protocol supports PROTO encoding. The *gnmi.proto* file represents the blueprint for generating a complete set of client and server-side procedures that represents the framework for the gNMI protocol. This feature was implemented on the following platforms: • Cisco Catalyst 9200, 9200L, and 9200X Series Switches • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 and 9400X Series Switches • Cisco Catalyst 9500, 9500 High-Performance, and 9500X Series Switches • Cisco Catalyst 9600 Series Switches



CHAPTER 11

gRPC Network Operations Interface

The Google Remote Procedure Call (gRPC) Network Operations Interface (gNOI) is a suite of microservices, each corresponding to a set of operations. This module describes the supported gNOI services.

- Information About the gRPC Network Operations Interface, on page 275
- Additional References for the gRPC Network Operations Interface, on page 290
- Feature Information for the gRPC Network Operations Interface, on page 291

Information About the gRPC Network Operations Interface

gNOI Protocol

gNOI defines a set of gRPC-based microservices for executing operational commands on network devices. The gNMI service defines operations for configuration management, operational state retrieval, and bulk data collection through streaming telemetry. gNOI only allows the adoption of services that a device supports. gNOI supports the OS installation service.

gNOI can be used with or without user authentication. User authentication is disabled by default. Use the **gnxi secure-password-auth** command to enable user authentication. For information about enabling user authentication through the OpenConfig model, see https://github.com/YangModels/yang/blob/master/vendor/cisco/xe/1751/openconfig-system-management.yang.

The gNOI protocol supports the following operations:

- Certificate Management
- Bootstrapping
- OS Installation Service
- Factory Reset Service

Certificate Management Service

The Certificate Management Service primarily exports two main RPCs, Install and Rotate, that are used for the installation of new certificates, and the rotation of existing certificates on a device, respectively.

The following RPCs are supported by the Certificate Management Service:

- Install: Installs a certificate. All certificates are uniquely identified by a certificate ID. The certificate ID is a string.
- Rotate: Rotates an existing certificate.
- RevokeCertificates: Revokes one or more certificates.
- GetCertificates: Queries all certificates.
- CanGenerateCSR: Queries whether the device can generate a Certificate Signing Request (CSR).

Trustpoints and certificates created through the RPCs mentioned above persist across switchovers and device reboots.

The following is a sample Certificate Management Service definition:

```
service CertificateManagement {
 rpc Install(stream InstallCertificateRequest)
  returns (stream InstallCertificateResponse);

 rpc Rotate(stream RotateCertificateRequest)
  returns (stream RotateCertificateResponse);

 rpc RevokeCertificates(RevokeCertificateRequest)
  returns (RevokeCertificateResponse);

 rpc GetCertificates(GetCertificateRequest)
  returns (GetCertificateResponse);

 rpc CanGenerateCSR(CanGenerateCSRRequest)
  returns (CanGenerateCSRResponse);
}
```

Install RPC

The Install RPC adds a new certificate to a device by creating a new CSR request. The new certificate is associated with a new certificate ID on the device. If the device has a pre-existing certificate with the given certificate ID, the operation fails.

The Install RPC is a bidirectional streaming RPC. It has an input (InstallCertificateRequest) and an output (IntsallCertificateResponse) both of which are streaming. If the stream is broken, or any steps in the process fail, the device rolls back the changes.

The following is an example of the Install RPC definition and messages:

```
rpc Install(stream InstallCertificateRequest)
returns (stream InstallCertificateResponse);

// Request messages to install new certificates on the target.
message InstallCertificateRequest {
  // Request Messages.
  oneof install_request {
    GenerateCSRRequest generate_csr = 1;
    LoadCertificateRequest load_certificate = 2;
  }
}
// Request to generate the CSR.
message GenerateCSRRequest {
  // Parameters for creating a CSR.
```

```
  CSRParams csr_params = 1;
  // The certificate id with which this CSR will be associated. The target
  // configuration should bind an entity which wants to use a certificate to
  // the certificate_id it should use.
  string certificate_id = 2;
}
// Parameters to be used when generating a Certificate Signing Request.
message CSRParams {
  // The type of certificate which will be associated for this CSR.
  CertificateType type = 1;

  // Minimum size of the key to be used by the target when generating a
  // public/private key pair.
  uint32 min_key_size = 2;

  // If provided, the target must use the provided key type. If the target
  // cannot use the algorithm specified in the key_type, it should cancel the
  // stream with an Unimplemented error.
  KeyType key_type = 3;

  // --- common set of parameters applicable for any type of certificate --- //
  string common_name = 4;           // e.g "device.corp.google.com"
  string country = 5;               // e.g "US"
  string state = 6;                 // e.g "CA"
  string city = 7;                  // e.g "Mountain View"
  string organization = 8;          // e.g "Google"
  string organizational_unit = 9;   // e.g "Security"
  string ip_address = 10;
  string email_id = 11;
}
// A certificate.
message Certificate {
  // Type of certificate.
  CertificateType type = 1;

  // Actual certificate.
  // The exact encoding depends upon the type of certificate.
  // for X509, this should be a PEM encoded Certificate.
  bytes certificate = 2;
}

message LoadCertificateRequest {
  // The certificate to be Loaded on the target.
  Certificate certificate = 1;

  // The key pair to be used with the certificate. This is provided in the event
  // that the target cannot generate a CSR (and the corresponding public/private
  // keys).
  KeyPair key_pair = 2;

  // Certificate Id of the above certificate. This is to be provided only when
  // there is an externally generated key pair.
  string certificate_id = 3;

  // Optional pool of CA certificates to be used for authenticating the client.
  repeated Certificate ca_certificate = 4;
}

// A message representing a pair of public/private keys.
message KeyPair {
  bytes private_key = 1;
  bytes public_key = 2;
}
```

```
// Response Messages from the target for the InstallCertificateRequest.
message InstallCertificateResponse {
  // Response messages.
  oneof install_response {
    GenerateCSRResponse generated_csr = 1;
    LoadCertificateResponse load_certificate = 2;
  }
}

// GenerateCSRResponse contains the CSR associated with the Certificate ID
// supplied in the GenerateCSRRequest. When a Certificate is subsequently
// installed on the target in the same streaming RPC session, it must be
// associated to that Certificate ID.
//
// An Unimplemented error will be returned if the target cannot generate a CSR
// as per the request. In this case, the caller must generate its own key pair.
message GenerateCSRResponse {
  CSR csr = 1;
}

// A Certificate Signing Request.
message CSR {
  // Type of certificate.
  CertificateType type = 1;

  // Bytes representing the CSR.
  // The exact encoding depends upon the type of certificate requested.
  // for X509: This should be the PEM encoded CSR.
  bytes csr = 2;
}
```

After the target device is up and gNOI is in default state, the controller (a third-party implementation) uses the Install RPC to install a certificate that is signed by a Certificate Authority (CA). The certificate is uniquely identified by a certificate ID. This ID is used as the trustpoint name in the Public Key Infrastructure (PKI) configuration. The installation will fail, if you try to install a certificate that has an existing certificate ID.

The following section describes how a CSR is generated by a device:

1. The device generates a self-signed certificate through the Install RPC. The controller does not require a copy of this certificate because in encrypted mode (or gNMI default state) the controller does not validate the certificate presented by the target device. This is the default state.
2. The controller requests the device to generate a CSR, sends the CSR to the CA, and gets the signed certificate back from the CA.
3. The signed certificate is installed into the device along with the CA certificates used to sign the certificate. The CA certificate is present in the *ca_certificates* bundle, and is required by the PKI to install the device certificate.
4. The gNMI or the gNOI service restarts using the newly installed certificate that is now in the provisioned state.

Rotate RPC

The Rotate RPC renews an existing certificate; a certificate that is already installed. If a certificate is not already installed, the Rotate RPC fails. A certificate that is not in use can be rotated, but the client cannot test it.

The following is a sample Rotate RPC definition:

```
rpc Rotate(stream RotateCertificateRequest)
returns (stream RotateCertificateResponse);

// Request messages to rotate existing certificates on the target.
message RotateCertificateRequest {
  // Request Messages.
  oneof rotate_request {
    GenerateCSRRequest generate_csr = 1;
    LoadCertificateRequest load_certificate = 2;
    FinalizeRequest finalize_rotation = 3;
  }
}

// A Finalize message is sent to the target to confirm the Rotation of
// the certificate and that the certificate should not be rolled back when
// the RPC concludes. The certificate must be rolled back if the target returns
// an error after receiving a Finalize message.
message FinalizeRequest {
}

message RotateCertificateResponse {
  // Response messages.
  oneof rotate_response {
    GenerateCSRResponse generated_csr = 1;
    LoadCertificateResponse load_certificate = 2;
  }
}
```

The Rotate RPC differs from the Install RPC in the following ways:

- PKI has to save or cache the old certificate and the CA certificate when installing a new certificate (for the purpose of rollback).
- The controller creates a new connection to test whether the renewed certificate works, and in case of success, finalizes the certificate rotation.

Revoke RPC

This RPC is used to revoke one or more certificates, each uniquely identified by a certificate ID. Revocation of a certificate results in the corresponding trustpoint to be removed from the Cisco IOS XE configuration. If the corresponding trustpoints are currently in use, or if the trustpoints do not exist, revocation of the certificates may fail.

A RevokeCertificate RPC may have certificates revoked successfully or unsuccessfully. On the target device, revocation is a simple delete operation; the actual revocation with the CA is done by the client. If the client revokes a certificate that is in use, new connections fail, but the existing connections are unaffected.

The following is a sample RevokeCertificate RPC:

```
// An RPC to revoke specific certificates.
// If a certificate is not present on the target, the request should silently
// succeed. Revoking a certificate should render the existing certificate
// unusable by any endpoints.
rpc RevokeCertificates(RevokeCertificatesRequest)
returns (RevokeCertificatesResponse);

message RevokeCertificatesRequest {
  // Certificates to revoke.
```

```
  repeated string certificate_id = 1;
}

message RevokeCertificatesResponse {
  // List of certificates successfully revoked.
  repeated string revoked_certificate_id = 1;

  // List of errors why certain certificates could not be revoked.
  repeated CertificateRevocationError certificate_revocation_error = 2;
}

// An error message indicating why a certificate id could not be revoked.
message CertificateRevocationError {
  string certificate_id = 1;
  string error_message = 2;
}
```

GetCertificate RPC

This RPC queries all certificate IDs.

The response to the query contains the following information:

- Certificate information for all the certificates that are identified by a certificate ID.
- The list of endpoints, for example, tunnels, daemons, and so on, that use this certificate.

Note Endpoints are not supported.

Note Responses do not contain the *ca_certificate* bundle.

The following is a sample GetCertificate RPC:

```
// An RPC to get the certificates on the target.
rpc GetCertificates(GetCertificatesRequest) returns (GetCertificatesResponse);

// The request to query all the certificates on the target.
message GetCertificatesRequest {
}

// Response from the target about the certificates that exist on the target what
// what is using them.
message GetCertificatesResponse {
  repeated CertificateInfo certificate_info = 1;
}

message CertificateInfo {
  string certificate_id = 1;
  Certificate certificate = 2;

  // List of endpoints using this certificate.
  repeated Endpoint endpoints = 3;
```

```
  // System modification time when the certificate was installed/rotated in
  // nanoseconds since epoch.
  int64 modification_time = 4;
}

// An endpoint represents an entity on the target which can use a certificate.
message Endpoint {
  // Type of endpoint that can use a cert. This list is to be extended based on
  // conversation with vendors.
  enum Type {
    EP_UNSPECIFIED = 0;
    EP_IPSEC_TUNNEL = 1;
    EP_DAEMON = 2;
  }
  Type type = 1;

  // Human readable identifier for an endpoint.
  string endpoint = 2;
}
```

CanGenerateCSR RPC

This RPC queries whether a device can generate a CSR for a specific key type, certificate type, and key size. The supported key type is Rivest, Shamir, and Adelman (RSA), and the supported certificate type is X.509.

When this RPC request is made for installing a completely new certificate as part of the Install RPC, the device must ensure that the certificate ID is new and no entities on the device are bound to this certificate ID. If any existing certificate matches the certificate ID, this request fails.

When this RPC request is made for rotating an existing certificate as part of the Rotate RPC, the device must ensure that the certificate ID is already available. If certificate rotation proceeds to load the certificate, it must associate the new certificate with the previously created certificate ID.

The following is a sample CanGenerateCSR RPC:

```
// An RPC to ask a target if it can generate a Certificate.
rpc CanGenerateCSR(CanGenerateCSRRequest) returns (CanGenerateCSRResponse);

// A request to ask the target if it can generate key pairs.
message CanGenerateCSRRequest {
  KeyType key_type = 1;
  CertificateType certificate_type = 2;
  uint32 key_size = 3;
}

// Algorithm to be used for generation the key pair.
enum KeyType {
  // 1 - 500, for known types.
  // 501 and onwards for private use.
  KT_UNKNOWN = 0;
  KT_RSA = 1;
}

// Types of certificates.
enum CertificateType {
  // 1 - 500 for public use.
  // 501 onwards for private use.
  CT_UNKNOWN = 0;
  CT_X509 = 1;
}
```

```
// Response from the target about whether it can generate a CSR with the given
// parameters.
message CanGenerateCSRResponse {
  bool can_generate = 4;
}
```

Mutual Authentication

Mutual authentication is a two-way authentication; two parties authenticate each other at the same time. To enable mutual-authentication, use the **gnmi-yang secure-peer-verify-trustpoint** command. If this command is not enabled, the authentication service validates the gNMI client against all the existing trustpoints and the contents of the trustpool.

Rotation of the CA certificates for mutual authentication requires the client to present a new bundle to the target device, and the old bundle to be removed. However, the CA certificates reside in a trustpool, and cannot be selectively deleted from the trustpool.

Bootstrapping with Certificate Service

After installing gNOI certificates, bootstrapping is used to configure or operate a target device. When a target device does not have any pre-existing certificates, bootstrapping allows the installing of certificates by using the gNOI Certificate Management Service. After the certificate installation, the device is capable of establishing secure gNOI or gNMI connections. This process assumes a pre-existing secure environment.

To enable gNMI bootstrapping, use the **gnxi secure-init** command.

Note The gNOI Certificate Management Service must be installed before bootstapping.

The gNOI Certificate Management Service has two states. These states are supported by both the gNOI service and the gNMI service.

- Default/Encrypted: gNOI and gNMI on the device use a self-signed (default) certificate that the client does not verify; the certificate does not require authentication. In this state, only the gNOI certificate service is enabled on the target device.
- Provisioned: gNOI and gNMI on the device use an installed certificate that is verified by the client, and the client presents its certificate, which the device verifies against its certificate store. The device verifies the client certificate only if mutual authentication is enabled.

OS Installation Service

The OS installation service defines a gNOI API that is used for installation. The OS installation service is supported in the gNOI protocol.

This service provides an interface for the installation of an OS on a device. It supports the following three RPCs:

- Install: This RPC transfers an image to a device. These images are uniquely identified by a version string. This RPC is the similar to the **install add** command; the main difference is that the image is transferred as part of the RPC.
- Activate: This RPC sets the requested OS version, which is part of the input to the RPC, as the version to be used at the next reboot, and reboots the device. This RPC is the same as the **install activate** and the **install commit** commands.
- Verify: This RPC verifies the current OS version.

Cisco IOS XE devices support both install mode and bundle mode to boot software images.

In install mode, you can bring up your device by booting the software package provisioning file that resides in the flash: file system. The ISO file system in each installed package is mounted to the root file system (rootfs) directly from the flash.

In bundle mode, you can boot your device by using the bundle (.bin) file. Packages are extracted from the bundle, and copied to the RAM. The ISO file system in each package is mounted to the rootfs. Unlike install boot mode, additional memory that is equivalent to the size of the bundle is used when booting in bundle mode.

In the following scenarios, an error message is generated when a device starts in bundle mode:

- The device starts with the current image running in bundle mode.
- The install RPC is initiated on the device to install a new image.

The following is a sample error message:

```
May 11 09:24:15.385 PST: %INSTALL-3-OPERATION_ERROR_MESSAGE:
Switch 1 R0/0: install_engine: Failed to install_add package
flash:gNOI_iosxe_17.05.01.0.144.1617180620.bin, Error: [2|install_add(ERR, )]:
Booted in bundle mode. For Bundle-to-Install mode conversion,
please use one-shot CLI - install add file <> activate commit
```

Even though an error message is generated, the install RPC returns a success to the client. The error message can be safely ignored; the subsequent activate RPC is not affected. After rebooting with the new image, the device is in install mode.

Note This error message is not displayed if the device was initially running in install mode. It is applicable only when the device starts in bundle mode.

To view all the error messages, see https://github.com/openconfig/gnoi/blob/master/os/os.proto#L218.

For more information about installation modes, see the "Performing Device Setup Configuration" chapter of the *System Management Configuration Guide* for all the Cisco Catalyst 9000 Series Switches.

Dual Route Processor Support

Cisco devices support both In-Service Software Update (ISSU) (only install mode is supported) and non-ISSU modes. When ISSU is not supported or is not possible through the Install RPC, the gNOI OS installation service will request a non-ISSU install.

If a device supports ISSU upgrade in case of dual Route Processors (RPs), the gNOI OS installation service interface invokes the install activate ISSU workflow. In all other scenarios, where ISSU not is supported, or

the device supports a single RP, the gNOI OS installation service uses a regular non-ISSU image install workflow to process the gRPC activate request.

In bundle mode, the upgrade is done through the **install add file** *filename* **activate commit** command. This upgrade is the same for devices with a single RP. No ISSU support means that both the RPs are reloaded at the same time, and the device is down until one RP comes up.

In install mode without ISSU, both the RPs are reloaded at the same time and the device is down until one RP comes up. In install mode with ISSU, the reload of the RPs is simultaneous, and the device downtime is shorter.

OS Install RPC

The Install RPC transfers an image to a device. The RPC consists of the input InstallRequest RPC, and the output InstallResponse RPC, both of which are bidirectional streaming RPCs.

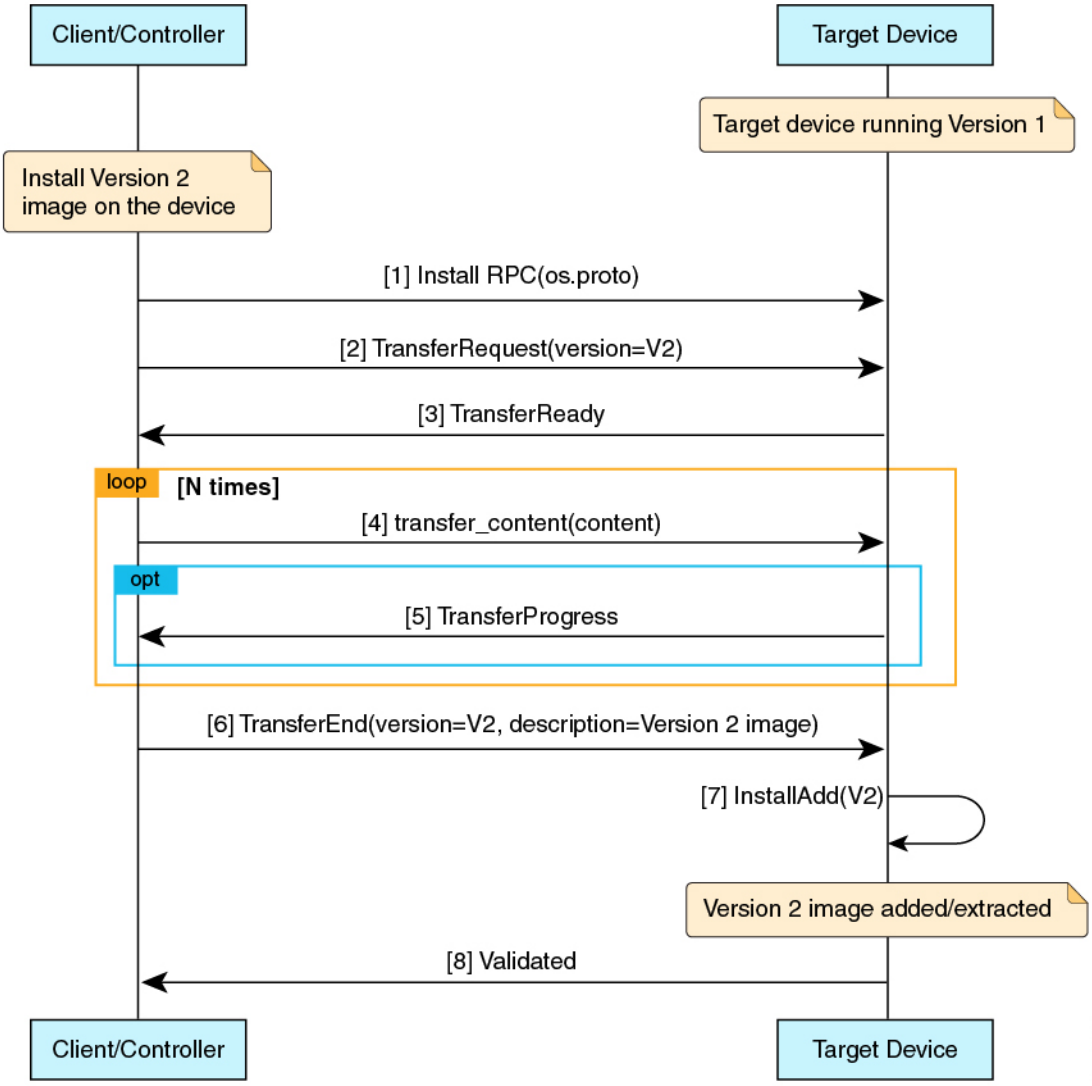
This RPC does not support Software Maintenance Update (SMU).

The following is a high-level message sequence for an Install RPC on a device with a single RP that is running the operating system Version 1:

1. A client initiates an Install RPC to the device.
2. The client sends a *TransferRequest* message to the device, with version set to Version 2.
3. The device responds with a *TransferReady* message to the client. This is required for the client to start transferring the image.
4. The client transfers the image by sending multiple *transfer_content* messages to the device.
5. Optionally, the device sends *TransferProgress* messages to the client.
6. The client sends a *TransferEnd* message to the device, indicating that the image transfer is complete.
7. In *install* mode, the device does an operation equivalent of the **install add** command programmatically. The contents of the package are extracted.
8. The device sends a *Validated* message, which contains the version extracted from the image, to the client, indicating that the image transfer is valid.

Note If the Install RPC is stopped prematurely by the client, or if any part of the operation fails, the local image file is removed, and the **install remove inactive** command is invoked automatically. An appropriate status code is returned to the client.

Figure 5: Single-RP Image Install Workflow



OS Activate RPC

The Activate RPC sets the requested operating system version as the version to be used at the next reboot, and reboots the target device. The RPC activates an installed operating system version. If the version is not already installed, the Activate RPC fails.

The client must provide a version that has been received in the *Validated* message of the Install RPC.

The following is the message sequence for an Activate RPC on a device with a single RP running operating system Version 1:

1. The client initiates an Activate RPC to a device.

2. The client sends an *ActivateRequest* message to the device with Version 2.

   For the purpose of this message sequence, assume that Version 2 is already installed through the Install RPC.

3. The device does a programmatic operation equivalent to the **install activate commit** command, if it is in install mode, or the **install add file activate commit** command if it is in bundle mode.
4. Because no errors are detected in the activate process, the device responds with an *ActivateResponse(OK)* message to the client.
5. The device reloads with Version 2.
6. When the device comes up after the reload, it does a programmatic operation equivalent to the **install commit** command.

Note Only one inactive image version is supported. Because of this, if a client installs Version 2 and then Version 3, the Version 2 files get deleted.

The following images display the image activation workflow.

Figure 6: Single-RP Image Activation Workflow

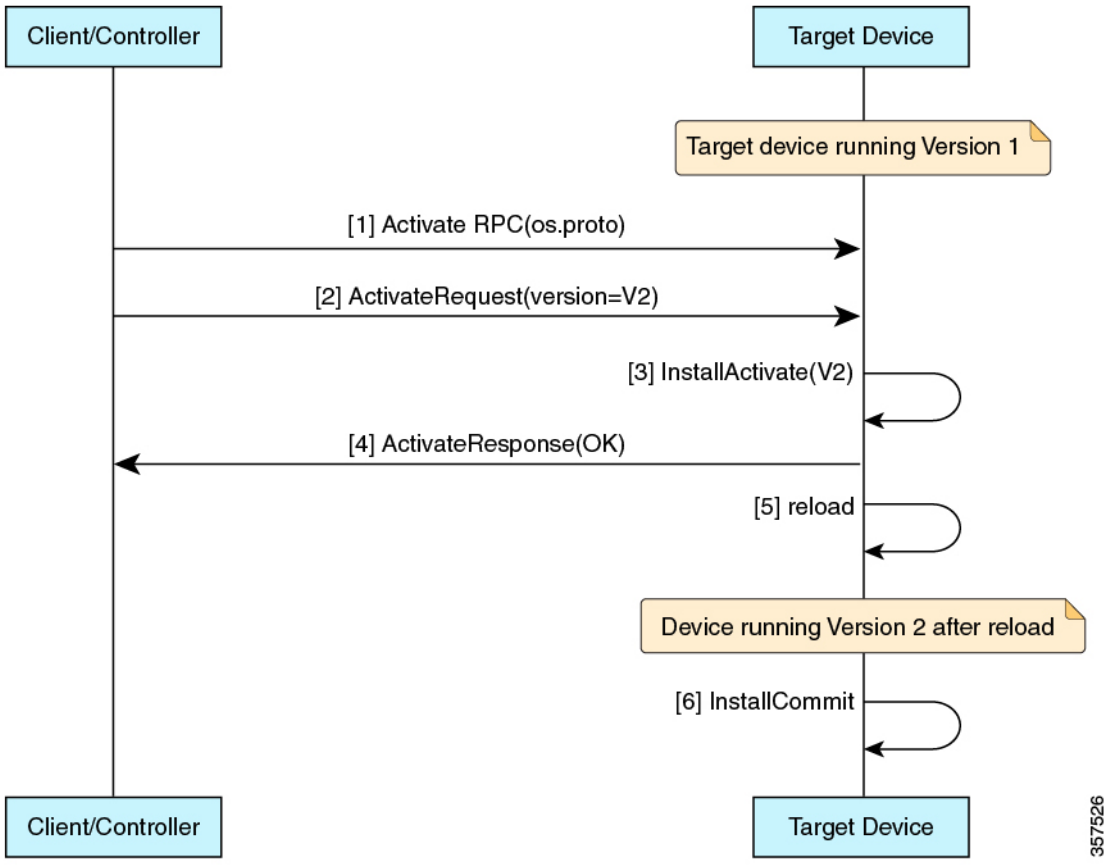


Figure 7: Dual-RP Image Install + non-ISSU Activation Workflow in Bundle Mode

Figure 8: Dual-RP Image Install + non-ISSU Activation Workflow

OS Verify RPC

The Verify RPC verifies the running OS version. The response to the RPC contains information about the support and presence of a standby RP.

If there was an error in the last activate RPC, that error is returned in the response as a string. The gNOI OS installation service uses the install operational model and platform model to populate this information. Currently, the install operational model does not support different versions running on two RPs.

GNOI Factory-Reset Services

Cisco IOS XE Cupertino 17.7.1 supports gNOI factory-reset services as specified in the reset.proto.

The gNOI factory-reset service supports a single RPC, *Start*. This RPC instructs a device to clean the existing state, and boot the device in the same condition as it was shipped from the factory. The state includes, storage, configuration, logs, certificates, licenses, crashinfo, and Rommon variables. Not all Rommon variables are removed, enough are preserved on a per-platform basis to allow the image to automatically reboot with the preserved image. The device then reboots with the current Operating System image, and comes back into the default state, based on the simplified bootstrapping workflow. This RPC is accepted only if the target device is in a provisioned state.

The *Start* RPC is similar to the **factory-reset all** command, however; the RPC preserves the current Operating System image, unlike the command, which deletes the image. As part of the factory-reset scripts, both the flash: or harddisk:, where the current image resides is cleaned up. However, when the factory-reset scripts are run, the boot image or packages are backed up to the /*tmp* folder, and restored.

The regular factory-reset erases all the customer-specific data stored in a device and restores the device to its original configuration at the time of shipping. Data that is erased includes configurations, log files, boot variables, core files, and credentials such as Federal Information Processing Standard-related (FIPS-related) keys. The erasure is consistent with the clear method, as described in NIST SP 800-88 Rev. 1. For more information, see the "Performing Factory Reset Services" module of the *System Management Configuration Guide* for your platform.

gNOI Factory-Reset Error Messages

gNOI factory-reset services return an empty ResetSuccess message upon successfully triggering the factory reset on a device.

Some of the error messages that are returned in the context of gRPC and factory reset services are described in this section:

Table 20: gNOI Factory-Reset Error Messages

Error Message	Error Description
When the *factory_os* field is requested, the GNMIB returns a gRPC error code of INVALID_ARGUMENT along with the message, “Factory OS rollback is not supported.”	In the ResetError message, the client will also receive the factory_os_unsupported field set to TRUE. The other fields in the message will have default values.

Error Message	Error Description
This device does not support the requested zero-fill option.	The *StartRequest* message has an optional field, *zero_fill* that instructs the target device to zero fill the persistent storage state data. When a client requests a zero-fill, and if the device cannot perform a zero fill, then the gRPC error code of INVALID_ARGUMENT is sent back to the client along with the message, "This device does not support the requested zero-fill option." In the *ResetError* message, the zero_fill_unsupported field is set to TRUE.
This device does not support the requested zero-fill option.	When the device can perform a zero-fill, but the client has not made a request for a zero-fill, then the gRPC error code of INVALID_ARGUMENT is sent back to the client with the message, "This device does not support the requested zero-fill option." In the *ResetError* message, the zero_fill_unsupported field is set to FALSE.
Factory reset capability is not present.	When the gNOI factory-reset script is run on an unsupported platform, the factory-reset services returns the gRPC error code of UNIMPLEMENTED with the message, "Factory reset capability is not present."
Factory reset interface is not ready.	When the gNOI factory-reset management interface is down or busy, the factory-reset services returns the gRPC error code of UNAVAILABLE with the message, "Factory reset interface is not ready."

Without using a cert.proto provisioning operation, or configuring gNOI with a signed certificate (not self-signed), the gNOI factory-reset service will always return the FAILED_PRECONDITION error code.

Additional References for the gRPC Network Operations Interface

Related Documents

Related Topic	Document Title
DevNet	https://developer.cisco.com/site/ios-xe/
gNOI	https://github.com/openconfig/gnoi
OS Service	https://github.com/openconfig/gnoi/blob/master/os/os.proto

Related Topic	Document Title
gNOI Factory Reset Service	https://github.com/openconfig/gnoi/blob/master/factory_reset/factory_reset.proto
Performing Device Setup Configuration	• *System Management Configuration Guide, Catalyst 9200 Switches* • *System Management Configuration Guide, Catalyst 9300 Switches* • *System Management Configuration Guide, Catalyst 9400 Switches* • *System Management Configuration Guide, Catalyst 9500 Switches* • *System Management Configuration Guide, Catalyst 9600 Switches*
Performing Factory Reset	• *System Management Configuration Guide, Catalyst 9300 Switches* • *System Management Configuration Guide, Catalyst 9300 Switches* • *System Management Configuration Guide, Catalyst 9300 Switches* • *System Management Configuration Guide, Catalyst 9300 Switches* • *System Management Configuration Guide, Catalyst 9300 Switches*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for the gRPC Network Operations Interface

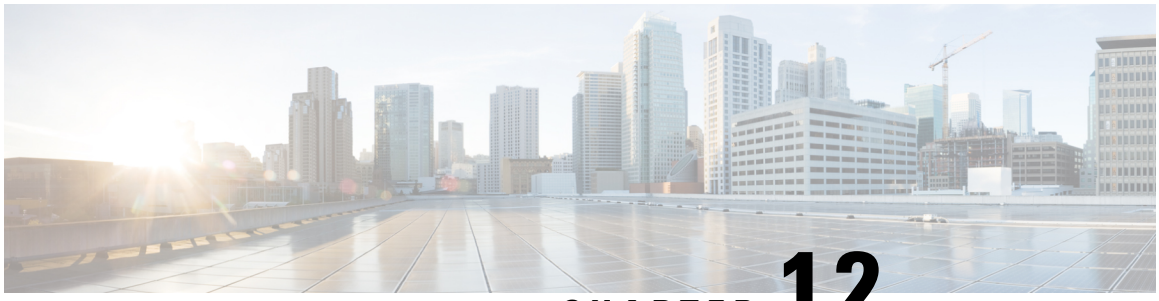
The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 21: Feature Information for the gRPC Network Operations Interface

Feature Name	Release	Feature Information
gNOI Certificate Management	Cisco IOS XE Amsterdam 17.3.1	The gNOI Certificate Management Service provides RPCs to install, rotate, get certificate, revoke certificate, and generate certificate signing request. In Cisco IOS XE Amsterdam 17.3.1, this feature was implemented on the following platforms: • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches
gNOI Bootstrapping with Certificate Service	Cisco IOS XE Amsterdam 17.3.1	After installing gNOI certificates, bootstrapping is used to configure or operate a target device. gNMI bootstrapping is enabled by using the **gnxi-secure-init** command and disabled by using the **secure-allow-self-signed-trustpoint** command. In Cisco IOS XE Amsterdam 17.3.1, this feature was implemented on the following platforms: • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches

Feature Name	Release	Feature Information
gNOI OS Installation Service	Cisco IOS XE Bengaluru 17.5.1	The gNOI OS installation service defines a gNOI API that is used for installation. In Cisco IOS XE Bengaluru 17.5.1, this feature was implemented on the following platforms: • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches
gNOI Factory Reset Services	Cisco IOS XE Cupertino 17.7.1	The gNOI factory reset service provides an interface that instructs target devices to clean the existing state, and boot the devices in same condition as it was shipped from the factory. In Cisco IOS XE Cupertino 17.7.1, this feature was implemented on the following platforms: • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers



CHAPTER 12

gNMI Dial-Out Using the gRPC Tunnel Service

This module describes how to configure a tunnel service for gNMI dial-out connections. You can use the gRPC tunnel server to forward connections from external clients, such as a gRPC Network Management Interface (gNMI) or gRPC Network Operations interface (gNOI), to connect to a network device without establishing a direct connection.

- gNMI Dial-Out Using gRPC Tunnel Service, on page 295
- Information About gNMI Dial-Out Using gRPC Tunnel Service, on page 296
- How to Configure gNMI Dial-Out Using gRPC Tunnel Service, on page 297
- Verifying the gNMI Dial-Out Using gRPC Tunnel Service Configuration, on page 300
- Feature Information for gNMI Dial-Out Using gRPC Tunnel Service, on page 301

gNMI Dial-Out Using gRPC Tunnel Service

In releases prior to Cisco IOS XE Dublin 17.11.1, gNMI supports a dial-in session, where the data collector sends RPCs directly to a network device. From Cisco IOS XE Dublin 17.11.1, gNMI uses a tunnel service for gNMI dial-out connections based on the recommendation from the OpenConfig forum.

With gNMI dial-out through gRPC tunnel service, you can use a router (tunnel client) to dial out to a collector (tunnel server). After establishing a session, the tunnel server acts as a client and requests gNMI services. The tunnel server then forwards requests from one or more gNMI or gNOI clients. Note that the gRPC tunnel server and the gNMI or gNOI client may be distinct entities.

Note The gRPC tunnel design is based on the feature specifications provided in the **tunnel.proto** file.

For more information about gNMI dial-out using gRPC tunnel, see the Github repository.

Note The tunnel service supports only Transport Layer Security (TLS) sessions.

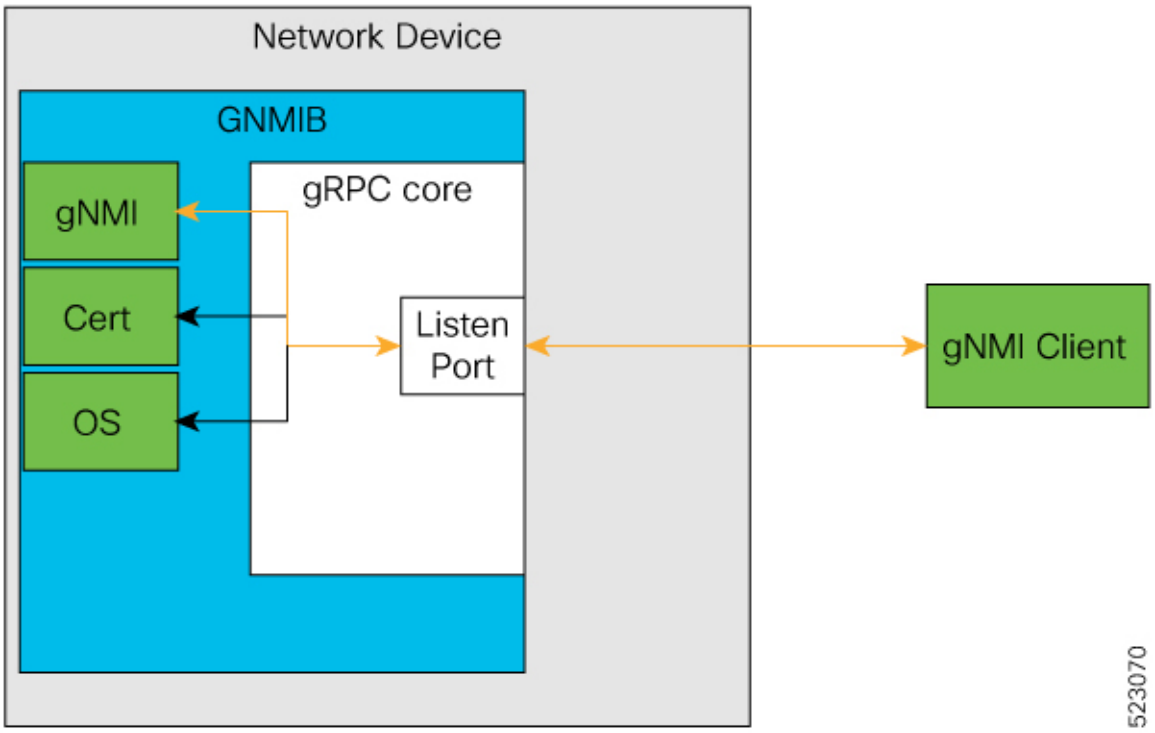
Information About gNMI Dial-Out Using gRPC Tunnel Service

The following sections provide detailed information about a traditional gRPC connection, a gRC tunnel, and connecting to GNMIB using a gRPC tunnel.

Traditional gRPC Connection

The yellow arrow in the following image shows the traditional method of connecting to a network device to access gRPC or gNOI services. The gNMI client connects to the network device only when the gNMI client is allowed to establish a direct connection.

Figure 9: Traditional gRPC Connection



gRPC Tunnel

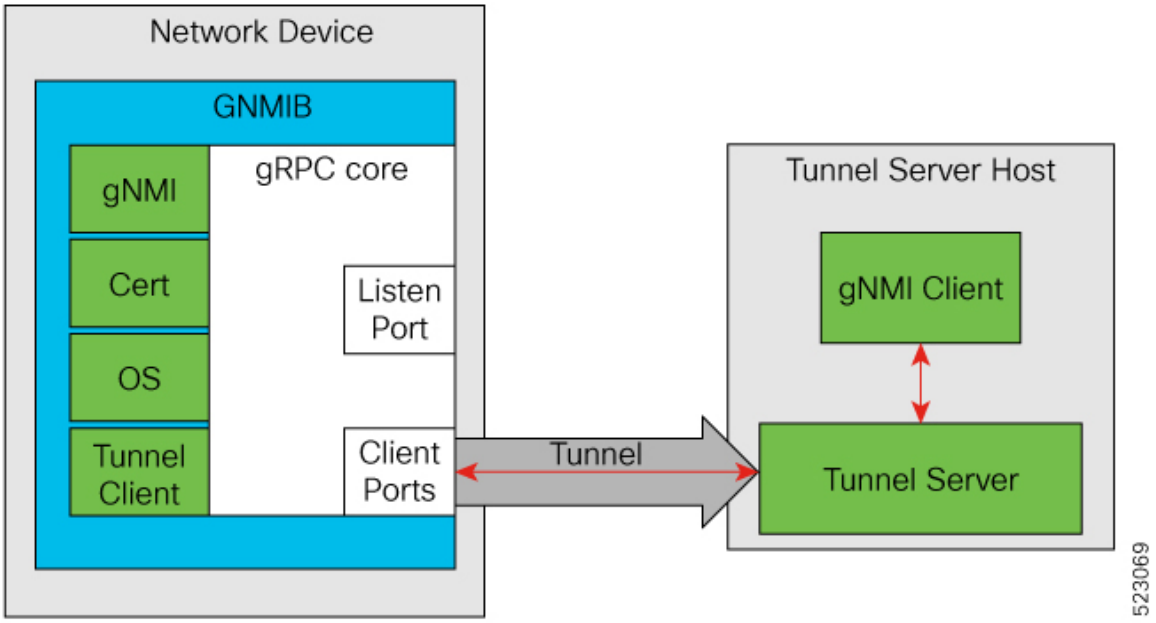
The following are the main components of a gRPC tunnel:

- Target: Represents a single service on a network device. For example, gNMI or gNOI is one target type. A tunnel client can register one or more target types with a tunnel server.
- Tunnel: A bidirectional stream in which data can be forwarded between the tunnel client and the server.
- Tunnel Server: The gRPC server that manages the target subscriptions and registrations.
- Tunnel Client: GNMIB is the gRPC tunnel client.

Connecting to GNMIB Using the gRPC Tunnel

In the gRPC tunnel design, the traditional flow is reversed. The network device dials out to the gRPC tunnel server. This leads to the gRPC tunnel server and any gNMI or gNOI clients to be unaware of the network device addresses and locations. Also, the network devices can access the gRPC tunnel server even if its outgoing connections are blocked.

Figure 10: A New Method of Connecting to GNMIB



How to Configure gNMI Dial-Out Using gRPC Tunnel Service

The following sections provide detailed information about the configurations that comprise the larger gNMI dial-out using gRPC tunnel service configuration.

Configuring and Enabling a Target

Run the following commands on a network device to configure and enable the target:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **gnxi grpctunnel target** {**GNMI_GNOI** | **GNMI_GNOI_INSECURE**}
4. **enable**
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**gnxi grpctunnel target** {**GNMI_GNOI** | **GNMI_GNOI_INSECURE**} **Example:** `Device(config)# gnxi grpctunnel target GNMI_GNOI`	Configures a gRPC tunnel target, and enters target configuration mode. **Note** Only **GNMI_GNOI** targets are supported. The **GNMI_GNOI_INSECURE** target is for testing purposes only and always connects to the GNMIB's insecure port.
Step 4	**enable** **Example:** `Device(config-target)# enable`	Enables the tunnel target.
Step 5	**end** **Example:** `Device(config-target)# end`	Exits target configuration mode and returns to privileged EXEC mode.

Configuring a gRPC Tunnel

Run the following commands to configure and enable a target in a network device that is part of a gRPC tunnel. Configure the IP address of the tunnel server, the port the tunnel server listens on, and the source or outgoing VRF. The following configuration task shows how the target sends data to the server:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **gnxi grpctunnel destination** *destination-name*
4. **enable**
5. **address** *IP-address*
6. **port** *port-number*
7. **identity-trustpoint** *trustpoint-name*
8. **source-address** *IP-address*
9. **source-vrf** *VRF-name*

10. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**gnxi grpctunnel destination** *destination-name* **Example:** `Device(config)# gnxi grpctunnel destination foobar`	Configures a gRPC tunnel destination and enters destination configuration mode.

Step 4	**enable**
	Example:
	`Device(config-destination)# enable`
	
	`% node-1:dbm:gnmi:Destination address must be set when destination is enabled`
		Enables the tunnel destination.

Step 5	**address** *IP-address* **Example:** `Device(config-destination)# address 209.165.200.225`	Configures the destination IP address.
Step 6	**port** *port-number* **Example:** `Device(config-destination)# port 1234`	Configures the destination port. • Valid values for the *port-number* argument are from 0 to 65535.
Step 7	**identity-trustpoint** *trustpoint-name* **Example:** `Device(config-destination)# identity-trustpoint trustpoint1`	Configures the specified trustpoint certificate as the TLS identity when connecting securely to a destination.
Step 8	**source-address** *IP-address* **Example:** `Device(config-destination)# source-address 209.165.201.30`	Configures the outgoing source address to use when connecting to the tunnel server or destination.

	Command or Action	Purpose
Step 9	**source-vrf** *VRF-name* **Example:** `Device(config-destination)# source-vrf Mgmt-vrf`	Configures a source Virtual Routing and Forwarding (VRF) instance when connecting to the tunnel server or destination.
Step 10	**end** **Example:** `Device(config-destination)# end`	Exits destination configuration mode and returns to privileged EXEC mode.

Verifying the gNMI Dial-Out Using gRPC Tunnel Service Configuration

Use the following command to verify the state of the gRPC tunnel service interface:

```
Device# show gnxi state detail

Settings
========
Server: Enabled
Server port: 50052
Secure server: Enabled
Secure server port: 9339
Secure client authentication: Disabled
Secure trustpoint: gnoi_pyats
Secure client trustpoint:
Secure password authentication: Disabled

GNMI
====
Admin state: Enabled
Oper status: Up
State: Provisioned
gRPC Server
-----------
Admin state: Enabled
Oper status: Up
Configuration service
---------------------
Admin state: Enabled
Oper status: Up
Telemetry service
-----------------
Admin state: Enabled
Oper status: Up

GNOI
====
Cert Management service
-----------------
Admin state: Enabled
Oper status: Up
OS Image service
----------------
Admin state: Enabled
Oper status: Up
```

```
Supported: Supported
Factory Reset service
---------------------
Admin state: Enabled
Oper status: Up
Supported: Supported

GRPC Tunnel
===========
Admin state: Enabled
Oper status: Up
```

Use the following command to display the statuses of all the currently configured gRPC tunnel servers:

```
Device# show gnxi grpctunnel destinations

All configured destinations
Destination Name: foobar
Target: GNMI_GNOI
Tag: 1
Registered: Yes
Session Started: Yes
Tunnel Active: Yes
Error:
Destination Name: example
Target: GNMI_GNOI
Tag: 1
Registered: Yes
Session Started: Yes
Tunnel Active: Yes
Error:
```

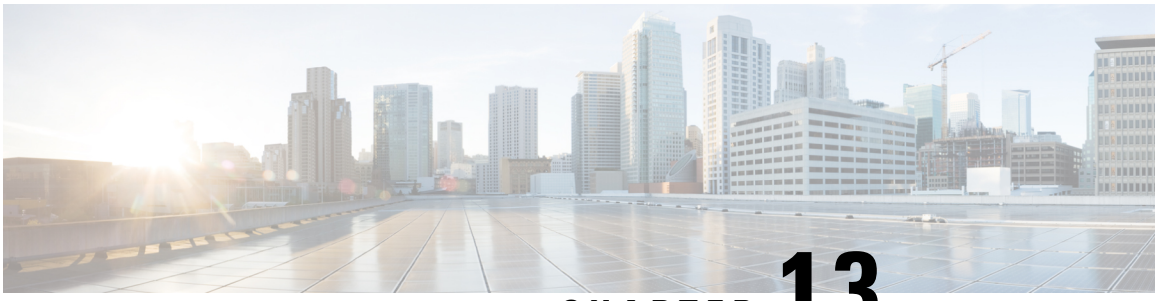
Feature Information for gNMI Dial-Out Using gRPC Tunnel Service

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 22: Feature Information for gNMI Dial-Out Using gRPC Tunnel Service

Feature Name	Release	Feature Information
gNMI Dial-Out Using gRPC Tunnel Service	Cisco IOS XE Dublin 17.11.1	This feature allows you to configure a network device (tunnel client) to register certain targets (preapproved services) with a gRPC tunnel server through the CLI. The following commands were introduced for this feature: • **gnxi grpctunnel destination** • **gnxi grpctunnel target** This feature was introduced on the following platforms: • Cisco Catalyst 9200, 9200L, and 9200CX Series Switches • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 and 9400X Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 and 9600X Series Switches • Cisco Network Convergence System 4200 Series



CHAPTER 13

Model Based AAA

The NETCONF and RESTCONF interfaces implement the NETCONF Access Control Model (NACM). NACM is a form of role-based access control (RBAC) specified in RFC 6536.

- Model Based AAA, on page 303
- Additional References for Model Based AAA, on page 309
- Feature Information for Model-Based AAA, on page 309

Model Based AAA

Prerequisites for Model Based AAA

Working with the model based AAA feature requires prior understanding of the following :

- NETCONF-YANG
- NETCONF-YANG kill-session
- RFC 6536: Network Configuration Protocol (NETCONF) Access Control Model

Initial Operation

Upon enabling the NETCONF and/or RESTCONF services, a device that has no prior configuration of the /nacm subtree will deny read, write, and execute access to all operations and data other than the users of privilege level 15. This is described in the following configuration of the /nacm subtree:

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
      <enable-nacm>true</enable-nacm>
      <read-default>deny</read-default>
      <write-default>deny</write-default>
      <exec-default>deny</exec-default>
      <enable-external-groups>true</enable-external-groups>
      <rule-list>
        <name>admin</name>
        <group>PRIV15</group>
        <rule>
          <name>permit-all</name>
          <module-name>*</module-name>
          <access-operations>*</access-operations>
          <action>permit</action>
```

```
    </rule>
  </rule-list>
</nacm>
```

Group Membership

The group membership of a user can come from two sources- first, from the privilege level of the user as configured on the AAA server used for authorization, and second, from those configured in the /nacm/groups subtree. The names of the groups that correspond to each privilege level are as follows:

Privilege level	NACM group name
0	PRIV00
1	PRIV01
2	PRIV02
3	PRIV03
4	PRIV04
5	PRIV05
6	PRIV06
7	PRIV07
8	PRIV08
9	PRIV09
10	PRIV10
11	PRIV11
12	PRIV12
13	PRIV13
14	PRIV14
15	PRIV15

Note Traditional IOS command authorization, such as those based on privilege level, does not apply to NETCONF or RESTCONF.

Note Access granted to a NACM group based on a privilege level do not inherently apply to NACM groups with higher privilege level. For example, rules that apply to PRIV10 do not automatically apply to PRIV11, PRIV12, PRIV13, PRIV14, and PRIV15 as well.

NACM Privilege Level Dependencies

If the AAA configuration is configured with **no aaa new-model**, the privilege level locally configured for the user is used. If the AAA configuration is configured with **aaa new-model**, the privilege level is determined by the AAA servers associated with the method list **aaa authorization exec default**.

NACM Configuration Management and Persistance

The NACM configuration can be modified using NETCONF or RESTCONF. In order for a user to be able to acccess the NCAM configuration, they must have explicit permission to do so, that is, through a NACM rule. Configuration under the /nacm subtree persists when the **copy running-config startup-config** EXEC command is issued, or the **cisco-ia:save-config** RPC is issued.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save-config xmlns="http://cisco.com/yang/cisco-ia"/>
</rpc>
```

Note The NACM rules that apply to a NETCONF session are those that are configured in the /nacm subtree at the time of session establishment. Modifying the /nacm subtree has no effect on NETCONF sessions as they are already established. The <kill-session> RPC or the **clear netconf-yang session** EXEC command can be used to forcibly end an unwanted NETCONF session. See NETCONF Kill Session, on page 174.

Note Care should be taken when crafting rules to deny access to certain data as the same data may be exposed through multiple YANG modules and data node paths. For example, interface configuration is exposed through both **Cisco-IOS-XE-native** and **ietf-interface**. Rules that may apply to one representation of the same underlying data may not apply to other representations of that data.

Resetting the NACM Configuration

Use the following command to reset the /nacm subtree configuration to the initial configuration (see Initial Operation).

```
Router#request platform software yang-management nacm reset-config
```

Sample NACM Configuration

Note The examples in this section are for illustrative purposes only.

The following is a sample for groups configuration.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
    <groups>
      <group>
        <name>administrators</name>
        <user-name>admin</user-name>
        <user-name>root</user-name>
```

```
    </group>

    <group>
      <name>limited-permission</name>
      <user-name>alice</user-name>
      <user-name>bob</user-name>
    </group>
  </groups>
</nacm>
```

Table 23: Description of the Configuration Paramenters for Groups Configuration

Parameter	Description
<name>administrators</name>	Group name
<user-name>admin</user-name>	User name
<user-name>root</user-name>	User name

The following is a sample for creating module rules.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>only-ietf-interfaces</name>
    <group>limited-permission</group>
    <rule>
      <name>deny-native</name>
      <module-name>Cisco-IOS-XE-native</module-name>
      <access-operations>*</access-operations>
      <action>deny</action>
    </rule>
    <rule>
      <name>allow-ietf-interfaces</name>
      <module-name>ietf-interfaces</module-name>
      <access-operations>*</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
</nacm>
```

Table 24: Description of the Configuration Paramenters for Creating Module Rules

Parameter	Description
<name>only-ietf-interfaces</name>	Unique rule-list name
<group>limited-permission</group>	Groups that rule-list applies to
<name>deny-native</name>	Unique rule name
<module-name>Cisco-IOS-XE-native</module-name>	Name of the YANG module
<access-operations>*</access-operations>	CRUDx operation types
<action>deny</action>	Permit/deny

The following is a sample for creating protocol operation rules.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>only-get</name>
    <group>limited-permission</group>

    <rule>
      <name>deny-edit-config</name>
      <module-name>ietf-netconf</module-name>
      <rpc-name>edit-config</rpc-name>
      <access-operations>exec</access-operations>
      <action>deny</action>
    </rule>
    <rule>
      <name>allow-get</name>
      <module-name>ietf-netconf</module-name>
      <rpc-name>get</rpc-name>
      <access-operations>exec</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
</nacm>
```

Table 25: Description of the Configuration Paramenters for Creating Protocol Operation Rules

Parameter	Description
<name>only-get</name>	Unique rule-list name
<group>limited-permission</group>	Groups that rule-list applies to
<name>deny-edit-config</name>	Unique rule name
<module-name>ietf-netconf</module-name>	Name of module containing the RPC
<rpc-name>edit-config</rpc-name>	Name of the RPC
<access-operations>exec</access-operations>	Execute permission for the RPC
<action>deny</action>	Permit/deny

The following is a sample for creating data node rules.

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
  <rule-list>
    <name>hide-enable-passwords</name>
    <group>limited-permission</group>

    <rule>
      <name>deny-enable-passwords</name>
      <path xmlns:ios="http://cisco.com/ns/yang/Cisco-IOS-XE-native>/ios:native/enable
      </path>
      <access-operations>*</access-operations>
      <action>deny</action>
    </rule>
  </rule-list>
</nacm>
```

Table 26: Description of the Configuration Paramenters for Creating Data Node Rules

Parameter	Description
<name>hide-enable-passwords</name>	Unique rule-list name
<group>limited-permission</group>	Groups that rule-list applies to
<name>deny-enable-passwords</name>	Unique rule name
<path xmlns:ios="http://cisco.com/ns/yang/Cisco-IOS-XE-native">/ios:native/enable</path>	Path to the data node being granted/denied
<access-operations>*</access-operations>	CRUDx operation types
<action>deny</action>	Permit/deny

The following is an example NACM configuration that permits all groups to use the standard NETCONF RPCs <get> and <get-config>, the schema download RPC <get-schema>, and read-only access to the data in the module **ietf-interfaces**:

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
      <rule-list>
        <name>readonly-protocol</name>
        <group>*</group>
        <rule>
          <name>get-permit</name>
          <module-name>ietf-netconf</module-name>
          <rpc-name>get</rpc-name>
          <access-operations>exec</access-operations>
          <action>permit</action>
        </rule>
        <rule>
          <name>get-config-permit</name>
          <module-name>ietf-netconf</module-name>
          <rpc-name>get-config</rpc-name>
          <access-operations>exec</access-operations>
          <action>permit</action>
        </rule>
        <rule>
          <name>get-schema-permit</name>
          <module-name>ietf-netconf-monitoring</module-name>
          <rpc-name>get-schema</rpc-name>
          <access-operations>exec</access-operations>
          <action>permit</action>
        </rule>
      </rule-list>
      <rule-list>
        <name>readonly-data</name>
        <group>*</group>
        <rule>
          <name>ietf-interfaces-permit</name>
          <module-name>ietf-interfaces</module-name>
          <access-operations>read</access-operations>
          <action>permit</action>
        </rule>
      </rule-list>
    </nacm>
```

Additional References for Model Based AAA

Related Documents

Related Topic	Document Title
YANG data models for various release of IOS-XE, IOS-XR, and NX-OS platforms	To access Cisco YANG models in a developer-friendly way, please clone the *GitHub repository*, and navigate to the *vendor/cisco* subdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.

Standards and RFCs

Standard/RFC	Title
RFC 6020	*YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*
RFC 6241	*Network Configuration Protocol (NETCONF)*
RFC 6536	*Network Configuration Protocol (NETCONF) Access Control Model*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

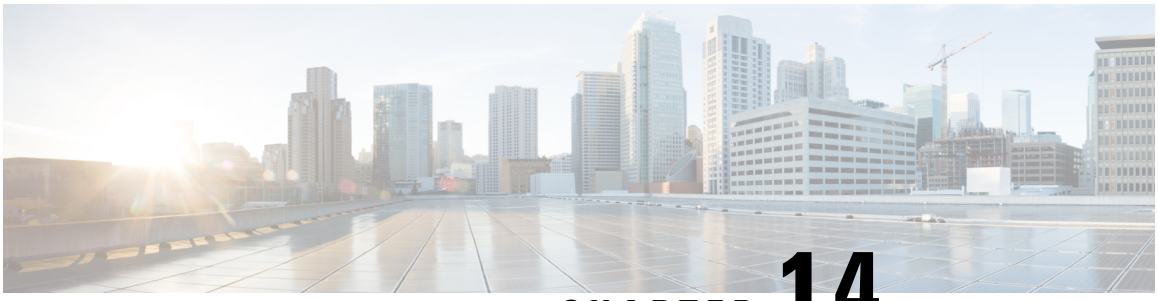
Feature Information for Model-Based AAA

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 27: Feature Information for Programmability: Data Models

Feature Name	Release	Feature Information
Model-Based AAA	Cisco IOS XE Fuji 16.8.1	This feature was implemented on the following platforms: • Cisco ASR 900 Series Aggregated Services Routers • Cisco ASR 920 Series Aggregated Services Routers • Cisco ASR 1000 Series Aggregated Services Routers • Cisco CSR 1000v Switches • Cisco ISR 1100 Series Integrated Services Routers • Cisco ISR 4000 Series Integrated Services Routers • Cisco NCS 4200 Series
	Cisco IOS XE Fuji 16.8.1a	This feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches



CHAPTER 14

Model-Driven Telemetry

- Model-Driven Telemetry, on page 311

Model-Driven Telemetry

Model-driven telemetry provides a mechanism to stream YANG-modelled data to a data collector. This module describes model-driven telemetry and provides sample telemetry remote procedure calls (RPCs).

Prerequisites for Model-Driven Telemetry

- Knowledge of YANG is needed to understand and define the data that is required when using telemetry.
- Knowledge of XML, XML namespaces, and XML XPath.
- Knowledge of standards and principles defined by the IETF telemetry specifications.
- The *urn:ietf:params:netconf:capability:notification:1.1* capability must be listed in hello messages. This capability is advertised only on devices that support IETF telemetry.
- NETCONF-YANG must be configured and running on the device.

Note Either NETCONF-YANG or gNXI must be configured for telemetry to work. If your platform does not support gNXI, you must configure NETCONF, even if NETCONF is not used. For more information on configuring NETCONF-YANG, see the *NETCONF Protocol* module. For more information on gNXI, see the *gNMI Protocol* module.

Verify that the following processes are running, by using the **show platform software yang-management process** command:

```
Device# show platform software yang-management process

confd : Running
nesd : Running
syncfd : Running
ncsshd : Running
dmiauthd : Running
nginx : Running
```

```
ndbmand : Running
pubd : Running
gnmib : Running
```

Note The process *pubd* is the model-driven telemetry process, and if it is not running, model-driven telemetry will not work.

The following table provides details about each of the Device Management Interface (DMI) processes.

Table 28: Field Descriptions

Device Management Interface Process Name	Primary Role
confd	Configuration daemon.
nesd	Network element synchronizer daemon.
syncfd	Sync daemon (maintains synchronization between the running state and corresponding models).
ncsshd	NETCONF Secure Shell (SSH) daemon.
dmiauthd	DMI authentication daemon.
nginx	NGINX web server. Acts as a web server for RESTCONF.
ndbmand	NETCONF database manager.
pubd	Publication manager and publisher used for model-driven telemetry.
gnmib	GNMI protocol server.

NETCONF-Specific Prerequisites

- Knowledge of NETCONF and how to use it, including:
  - Establishing a NETCONF session.
  - Sending and receiving hello and capabilities messages.
  - Sending and receiving YANG XML RPCs over the established NETCONF session. For more information, see the *Configure NETCONF/YANG and Validate Example for Cisco IOS XE 16.x Platforms* document.

Enabling and Validating NETCONF

The NETCONF functionality can be verified by creating an SSH connection to the device using a valid username and password and receiving a hello message, which contains the capability of the device:

```
Device:~ USER1$ ssh -s cisco1@172.16.167.175 -p 830 netconf
cisco1@172.16.167.175's password: cisco1

<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability
.
.
.
</capabilities>
<session-id>2870</session-id></hello>]]>]]>

Use < ^C > to exit
```

NETCONF is ready to use, when a successful reply is received in response to your hello message.

RESTCONF-Specific Prerequisites

- Knowledge of RESTCONF and how to use it (when creating a subscription using RESTCONF).
- RESTCONF must be configured on the device.
- RESTCONF must send correctly-formed Uniform Resource Identifiers (URIs) that adhere to RESTCONF *RFC 8040*.

Enabling and Validating RESTCONF

Validate RESTCONF using appropriate credentials and the following URI:

```
Operation: GET
Headers:
" Accept: application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
" Content-Type: application/yang-data+json
Returned Output (omitted for breverity):
{
    "ietf-restconf:data": {
        "ietf-yang-library:modules-state": {
            "module": [
                {
                    "name": "ATM-FORUM-TC-MIB",
                    "revision": "",
                    "schema":
"https://10.85.116.28:443/restconf/tailf/modules/ATM-FORUM-TC-MIB",
                    "namespace": "urn:ietf:params:xml:ns:yang:smiv2:ATM-FORUM-TC-MIB"
                },
                {
                    "name": "ATM-MIB",
                    "revision": "1998-10-19",
                    "schema":
"https://10.85.116.28:443/restconf/tailf/modules/ATM-MIB/1998-10-19",
                    "namespace": "urn:ietf:params:xml:ns:yang:smiv2:ATM-MIB"
                },
                {
                    "name": "ATM-TC-MIB",
```

```
                    "revision": "1998-10-19",
                    "schema": "https://10.85.116.28:443/restconf/tailf/
..
<snip>
..
}
```

RESTCONF is validated successfully when you receive the above reply with all device capabilities.

gRPC-Specific Prerequisites

- Set up a gRPC collector that understands key-value Google Protocol Buffers (GPB) encoding.

Restrictions for Model-Driven Telemetry

- Automatic hierarchy in selections is not supported for on-change subscriptions when using the *yang-push* stream. This means that when selecting a list, child lists of the list are not automatically included. For example, the subscriber must manually create a subscription for each child list.

  This restriction also applies to periodic subscriptions, if subscribed to the elements in the list below:

  - Cisco-IOS-XE-wireless-access-point-oper
  - Cisco-IOS-XE-wireless-ap-global-oper
  - Cisco-IOS-XE-wireless-awips-oper
  - Cisco-IOS-XE-wireless-client-global-oper
  - Cisco-IOS-XE-wireless-client-oper
  - Cisco-IOS-XE-wireless-general-cfg
  - Cisco-IOS-XE-wireless-general-oper
  - Cisco-IOS-XE-wireless-mesh-cfg
  - Cisco-IOS-XE-wireless-mesh-oper
  - Cisco-IOS-XE-wireless-mobility-oper
  - Cisco-IOS-XE-wireless-rfid-oper
  - Cisco-IOS-XE-wireless-rrm-emul-oper
  - Cisco-IOS-XE-wireless-rrm-global-oper
  - Cisco-IOS-XE-wireless-rrm-oper
  - Cisco-IOS-XE-wireless-site-cfg
  - bootcamp-test-autonomous
  - openconfig-access-points
  - openconfig-ap-manager
  - openconfig-lacp
  - openconfig-platform-psu

- Checking the authorization of data access is not supported. All the data requested by a subscriber is sent.
- Subtree filters are not supported. If subtree filters are specified, the subscription is marked as invalid.
- Defining multiple receivers within subscription parameters is not supported; only the first receiver destination is attempted. Other defined receivers are ignored.
- On Cisco Catalyst 9800 Wireless Controllers, the value of *tx-retries* and *excessive-retries* fields in the /client-oper-data/traffic-stats/ XPath will always display zero. To view the total number of times a clients tries to resend a packet, use the *data-retries* field of the XPath.

gRPC-Specific Restrictions

- Transport Layer Security-based (TLS-based) authentication between a device and a receiver is not supported.

  TLS-based authentication is supported in Cisco IOS XE Amsterdam 17.1.1 and later releases.

yang-push-Specific Restrictions

- Subscription quality of service (QoS) is not supported.

gRPC Telemetry Configured Subscriptions Restrictions

- Multiple subscriptions are dropped during normal operation. This issue was caused by a bug in the gRPC library that kept sending keepalive pings when keepalives were disabled. This issue was fixed through https://github.com/grpc/grpc/pull/22997in the following Cisco IOS XE releases:
  - Cisco IOS XE Cupertino 17.9.6
  - Cisco IOS XE Dublin 17.12.2
  - Cisco IOS XE 17.13.1 and later releases

  However, if you are using an Cisco IOS XE version without the fix, as a workaround, decrease the value of the GRPC_ARG_HTTP2_MIN_RECV_PING_INTERVAL_WITHOUT_DATA_MS parameter on the receiver to below 5 minutes. For more information about this parameter, see https://github.com/grpc/grpc/blob/master/doc/keepalive.md.

  Some gRPC receivers may have a configuration option to decrease the value of the parameter. For example, telegraf has "inputs.cisco_telemetry_mdt.grpc_enforcement_policy.keepalive_minimum_time". See: https://github.com/influxdata/telegraf/tree/master/plugins/inputs/cisco_telemetry_mdt#configuration.

Information About Model-Driven Telemetry

The following sections provide information about the various aspects of model-driven telemetry.

Model-Driven Telemetry Overview

Telemetry is an automated communications process by which measurements and other data are collected at remote or inaccessible points and transmitted to the receiving equipment for monitoring. Model-driven telemetry provides a mechanism to stream YANG-modeled data to a data collector.

Applications can subscribe to specific data items they need, by using standards-based YANG data models over NETCONF, RESTCONF, or gRPC Network Management Interface (gNMI) protocols. Subscriptions can also be created by using CLIs if it is a configured subscription.

Structured data is published at a defined cadence, or on-change, based upon the subscription criteria and data type.

Telemetry Roles

In systems that use telemetry, different roles are involved. In this document the following telemetry roles are described:

- Publisher: Network element that sends the telemetry data.
- Receiver: Receives the telemetry data. This is also called the collector.
- Controller: Network element that creates subscriptions but does not receive the telemetry data. The telemetry data associated with the subscriptions, it creates goes to receivers. This is also called the management agent or management entity.
- Subscriber: Network element that creates subscriptions. Technically, while this does not have to be the receiver too, in this document, both are the same.

Subscription Overview

Subscriptions are items that create associations between telemetry roles, and define the data that is sent between them.

Specifically, a subscription is used to define the set of data that is requested as part of the telemetry data; when the data is required, how the data is to be formatted, and, when not implicit, who (which receivers) should receive the data.

Even though the maximum number of supported subscriptions is platform-dependent, currently 100 subscriptions are supported. The subscriptions can be either configured or dynamic, and use any combination of transport protocols. If too many subscriptions are operating at the same time to allow all the valid configured subscriptions to be active, the removal of an active subscription will cause one of the inactive but valid configured subscriptions to be attempted. Periodic triggered subscriptions (100 centiseconds is the default minimum) and on-change triggered subscriptions are supported.

NETCONF and other northbound programmable interfaces (such as RESTCONF or gNMI) are supported to configure subscriptions.

Two types of subscriptions are used in telemetry on Cisco IOS XE systems: dynamic and configured subscriptions.

Because dynamic subscriptions are created by clients (the subscriber) that connect into the publisher, they are considered dial-in. Configured subscriptions cause the publisher to initiate connections to receivers, and as a result, they are considered dial-out.

Dial-In and Dial-Out Model-Driven Telemetry

The two flavors of model-driven telemetry are, dial-in and dial-out.

Table 29: Dial-in and Dial-Out Model-Driven Telemetry

Dial-In (Dynamic)	Dial-Out (Static or Configured)
Telemetry updates are sent to the initiator or subscriber.	Telemetry updates are sent to the specified receiver or collector.
Life of the subscription is tied to the connection (session) that created it, and over which telemetry updates are sent. No change is observed in the running configuration.	Subscription is created as part of the running configuration; it remains as the device configuration till the configuration is removed.
Dial-in subscriptions need to be reinitiated after a reload, because established connections or sessions are killed during stateful switchover.	Dial-out subscriptions are created as part of the device configuration, and they automatically reconnect to the receiver after a stateful switchover.
Subscription ID is dynamically generated upon successful establishment of a subscription.	Subscription ID is fixed and configured on the device as part of the configuration.

Data Source Specifications

Sources of telemetry data in a subscription are specified by the use of a stream and a filter. The term stream refers to a related set of events. RFC 5277 defines an event stream as a set of event notifications matching some forwarding criteria.

Normally, the set of events from a stream are filtered. Different filter types are used for different stream types.

Cisco IOS XE supports two streams: *yang-push* and *yang-notif-native*.

Update Notifications

As part of a subscription, you can specify when data is required. However this is stream-dependent. Some streams support making data available only when there a change happens, or after an event within the stream. Other streams make data available when there is a change or at a defined time period.

The result of the *when* specification is a series of update notifications that carry the telemetry data of interest. How the data is sent is dependent on the protocol used for the connection between the publisher and the receiver.

Subscription Identifiers

Subscriptions are identified by a 32-bit positive integer value. The IDs for configured subscriptions is set by the controller, and for dynamic subscriptions is set by the publisher.

Controllers must limit the values they use for configured subscriptions in the range 0 to 2147483647 to avoid collisions with the dynamic subscriptions created on the publisher. The dynamic subscription ID space is global, meaning that the subscription IDs for independently-created dynamic subscriptions do not overlap.

Subscription Management

Any form of management operation can be used to create, delete, and modify configured subscriptions. This includes both CLIs and network protocol management operations.

All subscriptions, both configured and dynamic, can be displayed using **show** commands and network protocol management operations.

The following table describes the supported streams and encodings along with the combinations that are supported. While streams-as-inputs is intended to be independent of the protocols-as-outputs, not all combinations are supported.

Table 30: Supported Combination of Protocols

Transport Protocol	**NETCONF**		**gRPC**		**gNMI**	
	Dial-In	Dial-Out	Dial-In	Dial-Out	Dial-In	Dial-Out
Stream						
yang-push	Yes	No	No	Yes	Yes	No
yang-notif-native	Yes	No	No	Yes	No	No
Encodings	XML	No	No	Key-value Google Protocol Buffers (kvGPB)	• JSON_IETF • PROTO PROTO encoding is supported in Cisco IOS XE Dublin 17.11.1 and later releases.	No

RPC Support in Telemetry

You can send and receive YANG XML remote procedure calls (RPCs) in established NETCONF sessions.

The <establish-subscription> and <delete-subscription> RPCs are supported for telemetry.

When an <establish-subscription> RPC is sent, the RPC reply from a publisher contains an <rpc-reply> message with a <subscription-result> element containing a result string.

The following table displays the response and reason for the response in an <rpc-reply> message:

Result String	**RPC**	**Cause**
ok	<establish-subscription> <delete-subscription>	Success
error-no-such-subscription	<delete-subscription>	The specified subscription does not exist.
error-no-such-option	<establish-subscription>	The requested subscription is not supported.
error-insufficient-resources	<establish-subscription>	A subscription cannot be created because of the following reasons:

Result String	RPC	Cause
		• There are too many subscriptions. • The amount of data requested is too large. • The interval for a periodic subscription is too small.
error-other	<establish-subscription>	Some other error.

Service gNMI

The gNMI specification identifies a single top-level service named gNMI that contains high-level RPCs. The following is a service definition that contains the subscribe service RPC:

```
service gNMI{
  .
  .
  .
   rpc Subscribe(stream SubscribeRequest)
       returns (stream SubscribeResponse);
```

The <subscribe RPC> is used by a management agent to request a dynamic subscription. This RPC contains a set of messages. The following section describes the messages supported by the <subscribe RPC>

SubscribeRequest Message

This message is sent by a client to request updates from the target for a specified set of paths. The following is a message definition:

```
message SubscribeRequest {
  oneof request {
    SubscriptionList subscribe = 1;
    PollRequest poll = 3;
    AliasList aliases = 4;
  }
  Repeated gNMI_ext.Extensions = 5;
}
```

Note Only request.subscribe is supported.

SubscribeResponse Message

This message is carried from the target to the client over an established <subscribe RPC>. The following is a message definition:

```
message SubscribeResponse {
  oneof response {
    Notification update = 1;
```

```
    Bool sync_response = 3;
    Error error = 4 [deprecated=true];
  }
}
```

Note Only Notification update is supported.

gNMI sync_response Message

A SubscribeResponse message is transmitted by a target (switch or router) to a gNMI client (collector) over an established gNMI Subscribe RPC. The sync_response is a boolean field that is part of the SubscribeResponse response message, which indicates that all data values that corresponds to the paths in the SubscriptionList has been transmitted at least once. The sync_response message is sent after the first update message.

This field is enabled for both gNMI on-change and periodic notifications.

The following sample SubscribeResponse message displays the sync_response field:

```
message SubscribeResponse {
  oneof response {
    Notification update = 1;  // Changed or sampled value for a path.
    // Indicate target has sent all values associated with the subscription
    // at least once.
    bool sync_response = 3;
    // Deprecated in favour of google.golang.org/genproto/googleapis/rpc/status
    Error error = 4 [deprecated = true];
  }
  // Extension messages associated with the SubscribeResponse. See the
  // gNMI extension specification for further definition.
  repeated gnmi_ext.Extension extension = 5;
}
```

SubscriptionList Message

This message is used to indicate a set of paths for which common subscription behavior are required. Within the specification of the SubscriptionList message, the client can identify one or more subscriptions to a given prefix in the model. The following is a SubscriptionList message defintion:

```
message SubscriptionList {
  Path prefix = 1;
  repeated Subscription subscription = 2;
  bool use_aliases = 3;
  QOSMarking qos = 4;
  enum Mode {
      STREAM = 0;
     ONCE = 1;
     POLL = 2;
  }
  Mode mode = 5;
  bool allow_aggregation = 6;
  repeated ModelData use_models = 7;
  Encoding encoding = 8; // only JSON_IETF supported in R16.12
  Bool updates_only = 9;
```

```
}
```

Note Path prefix (only explicit element names), Subscription subscription, Mode mode STREAM, and Encoding encoding IETF_JSON are supported.

Prefix Message

A valid subscription list may or may not contain a filled in prefix, composed of the shared (across all requested subscriptions) portion of the xPath.

```
message Path {
  repeated string element  = 1; [ deprecated ]
  string origin = 2;
   repeated PathElem elem = 3;
   optional string target = 4;

}
```

Note Origin (supported values are “” and “openconfig”), elem (supported element name is prefix-free), and target are supported.

Subscription Message

This message generically describes a set of data that is to be subscribed to by a client. It contains a path,and attributes used to govern the notification behaviors. The following is a Subscription message definition:

```
message Subscription {
   Path path = 1;
    SubscriptionMode mode = 2;
    uint64 sample_interval = 3;
   bool suppress_redundant = 4;
   uint64 heartbeat_interval = 5;
}
```

Note Path path, SubscriptionMode mode, Uint64 sample_interval, and Uint64 heartbeat_interval (only if the value is set to 0) are supported.

Path Message

A valid subscription contains a filled in path, which when added to the prefix associated with the subscription list constitutes a full qualified path. The following is a Path message definition:

```
message Path {
  repeated string element  = 1; [ deprecated ]
```

```
  string origin = 2;
   repeated PathElem elem = 3;
   optional string target = 4;

}
```

Note Origin (supported values are “” and “openconfig”), elem (supported element name is prefix-free), and target are supported.

SubscriptionMode Message

This message informs the target about how to trigger notifications updates. The following is a SubscriptionMode message definition:

```
enum SubscriptionMode {
   TARGET_DEFINED = 0;
   ON_CHANGE       = 1;
   SAMPLE          = 2;
}
```

Note Only SAMPLE and ON_CHANGE (from Cisco IOS XE Bengaluru 17.6.1) are supported.

ON_CHANGE support is limited to certain model paths. To check whether a path supports ON_CHANGE, query the path in the Cisco-IOS-XE-MDT-capabilities-oper model. For more information about the model, see the section, Displaying On-Change Subscription Capabilities, on page 339.

Notifications Message

This message delivers telemetry data from the subscription target to the collector. The following is a Notifications message definition:

```
message Notification {
  int64 timestamp = 1;
   Path prefix = 2;
  string alias = 3;
  repeated Update update = 4;
  repeated Path delete = 5;
  bool atomic = 6;
}
```

Note Timestamp, prefix, and update are supported.

Dynamic Subscription Management

This section describes how to create and delete dynamic subscriptions.

Creating Dynamic Subscriptions for NETCONF Dial-In

Dynamic subscriptions are created by subscribers who connect to the publisher and call for subscription creation using a mechanism within that connection, usually, an RPC. The lifetime of the subscription is limited to the lifetime of the connection between the subscriber and the publisher, and telemetry data is sent only to that subscriber. These subscriptions do not persist if either the publisher or the subscriber is rebooted. You can create dynamic subscriptions by using the in-band <establish-subscription> RPC. The <establish-subscription> RPC is sent from an IETF telemetry subscriber to the network device. The stream, xpath-filter, and period fields in the RPC are mandatory.

RPCs that are used to create and delete dynamic subscriptions using NETCONF are defined in *Custom Subscription to Event Notifications draft-ietf-netconf-subscribed-notifications-03* and *Subscribing to YANG datastore push updates draft-ietf-netconf-yang-push-07*.

Periodic Dynamic Subscriptions

The following is a sample periodic subscription for NETCONF Dial-In:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <establish-subscription
          xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
          xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <stream>yp:yang-push</stream>
        <yp:xpath-filter>/mdt-oper:mdt-oper-data/mdt-subscriptions</yp:xpath-filter>
        <yp:period>1000</yp:period>
      </establish-subscription>
    </rpc>
```

On-Change Dynamic Subscription

The following is a sample on-change dynamic subscription over NETCONF:

```
<establish-subscription xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <stream>yp:yang-push</stream>

<yp:xpath-filter>/cdp-ios-xe-oper:cdp-neighbor-details/cdp-neighbor-detail</yp:xpath-filter>

    <yp:dampening-period>0</yp:dampening-period>
</establish-subscription>
```

Deleting Dynamic Subscriptions

You can delete dynamic subscriptions by using the in-band <delete subscription> RPC, the **clear telemetry ietf subscription** command, and the <kill-subscription> RPC along with disconnecting the transport session.

For gNMI each subscription in the SubscribeRequest.subscribe.subscription a separate dynamic subscription ID is generated. Killing any of these subscription IDs, either through the <kill-subscription> RPC or clear CLI, will cause all subscriptions specified in the subscribe request to be killed.

Introduced in Cisco IOS XE Gibraltar 16.10.1, the <delete-subscription> RPC can be issued only by a subscriber, and it deletes only the subscriptions owned by that subscriber.

In Cisco IOS XE Gibraltar 16.11.1 and later releases, you can use the **clear telemetry ietf subscription** command to delete a dynamic subscription. Introduced in Cisco IOS XE Gibraltar 16.11.1, the

<kill-subscription> RPC deletes dynamic subscription, the same way as the **clear telemetry ietf subscription** command.

A subscription is also deleted when the parent NETCONF session is torn down or disconnected. If the network connection is interrupted, it may take some time for the SSH or NETCONF session to timeout, and for subsequent subscriptions to be removed.

The <kill-subscription> RPC is similar to the <delete-subscription> RPC. However, the <kill-subscription> RPC uses the *identifier* element that contains the ID of the subscription to be deleted, instead of the *subscription-id* element. The transport session used by the target subscription also differs from the one used by the <delete-subscription> RPC.

Deleting Subscriptions Using the CLI

The following sample output shows all the available subscriptions:

```
Device# show telemetry ietf subscription all

 Telemetry subscription brief

  ID               Type        State       Filter type
  ---------------------------------------------------------
  2147483648       Dynamic     Valid       xpath
  2147483649       Dynamic     Valid       xpath
```

The following example shows how to delete a dynamic subscription:

```
Device# clear telemetry ietf subscription 2147483648
```

Deleting Subscriptions Using NETCONF <delete-Subscription> RPC

The following example shows how to delete a subscription using NETCONF:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <delete-subscription xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
        <subscription-id>2147483650</subscription-id>
    </delete-subscription>
</rpc>
```

Deleting Subscriptions Using NETCONF <kill-Subscription> RPC

The following examples show how to delete subscriptions using the <kill-subscription> RPC:

```
<get>
<filter>
<mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
<mdt-subscriptions/>
</mdt-oper-data>
</filter>
</get>

* Enter a NETCONF operation, end with an empty line

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
```

```
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-subscriptions>
        <subscription-id>2147483652</subscription-id>
        <base>
…
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
…
        </mdt-receivers>
        <last-state-change-time>2018-12-13T21:16:48.848241+00:00</last-state-change-time>
      </mdt-subscriptions>
      <mdt-subscriptions>
        <subscription-id>2147483653</subscription-id>
        <base>
…
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
…
        </mdt-receivers>
        <last-state-change-time>2018-12-13T21:16:51.319279+00:00</last-state-change-time>
      </mdt-subscriptions>
      <mdt-subscriptions>
        <subscription-id>2147483654</subscription-id>
        <base>
…
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
…
        </mdt-receivers>
        <last-state-change-time>2018-12-13T21:16:55.302809+00:00</last-state-change-time>
      </mdt-subscriptions>
      <mdt-subscriptions>
        <subscription-id>2147483655</subscription-id>
        <base>
…
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
…
        </mdt-receivers>
        <last-state-change-time>2018-12-13T21:16:57.440936+00:00</last-state-change-time>
      </mdt-subscriptions>
    </mdt-oper-data>
  </data>
</rpc-reply>
<kill-subscription xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
 xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <identifier>2147483653</identifier>
</kill-subscription>

* Enter a NETCONF operation, end with an empty line

<?xml version="1.0" encoding="UTF-8"?>
```

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <subscription-result xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"

xmlns:notif-bis="urn:ietf:params:xml:ns:yang:ietf-event-notifications">notif-bis:ok</subscription-result>
</rpc-reply>
<get>
<filter>
<mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
<mdt-subscriptions/>
</mdt-oper-data>
</filter>
</get>

* Enter a NETCONF operation, end with an empty line

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-subscriptions>
        <subscription-id>2147483652</subscription-id>
        <base>
…
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
…
        </mdt-receivers>
        <last-state-change-time>2018-12-13T21:16:48.848241+00:00</last-state-change-time>
      </mdt-subscriptions>
      <mdt-subscriptions>
        <subscription-id>2147483654</subscription-id>
        <base>
…
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
…
        </mdt-receivers>
        <last-state-change-time>2018-12-13T21:16:55.302809+00:00</last-state-change-time>
      </mdt-subscriptions>
      <mdt-subscriptions>
        <subscription-id>2147483655</subscription-id>
        <base>
…
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
…
        </mdt-receivers>
        <last-state-change-time>2018-12-13T21:16:57.440936+00:00</last-state-change-time>
      </mdt-subscriptions>
   </mdt-oper-data>
  </data>
</rpc-reply>
```

Configured Subscription Management

This section describes how to create, modifiy, and delete configured subscriptions.

Creating Configured Subscriptions

Configured subscriptions are created by management operations on the publisher by controllers, and explicitly include the specification of the receiver of the telemetry data defined by a subscription. These subscriptions persist across reboots of the publisher.

Configured subscriptions can be configured with multiple receivers, however; only the first valid receiver is used. Connection to other receivers is not attempted, if a receiver is already connected, or is in the process of being connected. If that receiver is deleted, another receiver is connected.

Configured dial-out subscriptions are configured on the device by the following methods:

- Using configuration CLIs to change to device configuration through console/VTY.
- Using NETCONF/RESTCONF to configure the desired subscription.

In Cisco IOS XE Dublin 17.11.1, the enum values, *sub-upd-trig-on-change* was replaced by *sub-upd-trig-on-change-v2*, and *no-synch-on-start* was replaced by *no-synch-on-start-v2*, for the dampening period to work.

This section displays sample RPCs to create configured subscriptions.

Periodic Subscription

The following example shows how to configure gRPC as the transport protocol for configured subscriptions using the CLI:

```
telemetry ietf subscription 101
  encoding encode-kvgpb
  filter xpath /memory-ios-xe-oper:memory-statistics/memory-statistic
  stream yang-push
  update-policy periodic 6000
  source-vrf Mgmt-intf
  receiver ip address 10.28.35.45 57555 protocol grpc-tcp
```

The following sample RPC shows how to create a periodic subscription using NETCONF that sends telemetry updates to the receiver every 60 seconds:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><edit-config>
 <target>
  <running/>
 </target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
   <mdt-subscription>
    <subscription-id>200</subscription-id>
    <base>
     <stream>yang-push</stream>
     <encoding>encode-kvgpb</encoding>
     <period>6000</period>
     <xpath>/memory-ios-xe-oper:memory-statistics/memory-statistic</xpath>
    </base>
    <mdt-receivers>
     <address>10.22.23.48</address>
     <port>57555</port>
     <protocol>grpc-tcp</protocol>
```

```
    </mdt-receivers>
   </mdt-subscription>
  </mdt-config-data>
 </config>
</edit-config>
</rpc>
```

The following sample RPC creates a periodic subscription using RESTCONF:

```
URI:https://10.85.116.28:443/restconf/data/Cisco-IOS-XE-mdt-cfg:mdt-config-data
Headers:
application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
Content-Type:
application/yang-data+json
BODY:
{
"mdt-config-data": {
 "mdt-subscription":[
 {
  "subscription-id": "102",
  "base": {
   "stream": "yang-push",
   "encoding": "encode-kvgpb",
                    "period": "6000",
   "xpath": "/memory-ios-xe-oper:memory-statistics/memory-statistic"
  }
        "mdt-receivers": {
            "address": "10.22.23.48"
            "port": "57555"
        }
 }
 ]
}
}
```

On-Change Subscription

The following sample RPC shows how to create an on-change subscription using NETCONF that sends updates only when there is a change in the target database:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><edit-config>
 <target>
  <running/>
 </target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
   <mdt-subscription>
    <subscription-id>200</subscription-id>
    <base>
     <stream>yang-push</stream>
     <encoding>encode-kvgpb</encoding>
     <no-synch-on-start-v2>false</no-synch-on-start-v2>
     <xpath>/cdp-ios-xe-oper:cdp-neighbor-details/cdp-neighbor-detail</xpath>
    </base>
    <mdt-receivers>
     <address>10.22.23.48</address>
     <port>57555</port>
     <protocol>grpc-tcp</protocol>
    </mdt-receivers>
   </mdt-subscription>
  </mdt-config-data>
 </config>
```

```
</edit-config>
</rpc>
```

The following sample RPC shows how to create an on-change subscription using RESTCONF:

```
URI:
https://10.85.116.28:443/restconf/data/Cisco-IOS-XE-mdt-cfg:mdt-config-data
Headers:
application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
Content-Type:
application/yang-data+json
BODY:
{
"mdt-config-data": {
 "mdt-subscription":[
 {
  "subscription-id": "102",
  "base": {
   "stream": "yang-push",
   "encoding": "encode-kvgpb",
                     "dampening period": "0",
   "xpath": "/cdp-ios-xe-oper:cdp-neighbor-details/cdp
                           -neighbor-detail "
  }
        "mdt-receivers": {
            "address": "10.22.23.48"
            "port": "57555"
        }
 }
 ]
}
}
```

gNMI Dial-In Subscription

The following is a sample gNMI dial-in subscription:

```
subscribe: <
  prefix: <>
  subscription: <
    path: <
      origin: "openconfig"
      elem: <name: "routing-policy">
    >
    mode: SAMPLE
    sample_interval: 10000000000
  >
  mode: STREAM
  encoding: JSON_IETF
>'

subscribe: <
  prefix: <>
  subscription: <
    path: <
      origin: "legacy"
      elem: <name: "oc-platform:components">
      elem: <
        name: "component"
        key: <
          key: "name"
          value: "PowerSupply8/A"
```

```
        >
       >
       elem: <name: "power-supply">
       elem: <name: "state">
     >
     mode: SAMPLE
     sample_interval: 10000000000
   >
   mode: STREAM
   encoding: JSON_IETF
>'
```

Modifying Configured Subscriptions

There are two ways to modify configured subscriptions:

- Management protocol configuration operations, such as NETCONF <edit-config> RPC
- CLI (same process as creating a subscription)

Subscription receivers are identified by the address and port number. Receivers cannot be modified. To change the characteristics (protocol, profile, and so on) of a receiver, it must be deleted first and a new receiver created.

If a valid receiver configuration on a valid subscription is in the disconnected state, and the management wants to force a new attempt at setting up the connection to the receiver, it must rewrite the receiver with the exact same characteristics.

Deleting Configured Subscriptions

You can use the CLI or management operation to delete configured subscriptions. The **no telemetry ietf subscription** command removes the configured subscriptions. Note that configured subscriptions cannot be deleted using RPCs, only through the configuration interface.

Deleting Subscriptions Using the CLI

```
Device# configure terminal
Device(config)# no telemetry ietf subscription 101
Device(config)# end
```

Deleting Subscriptions Using NETCONF

The following sample RPC shows how to delete a configured subscription:

```
<edit-config>
   <target>
     <running/>
   </target>
   <config>
     <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
       <mdt-subscription  operation="delete">
       <subscription-id>102</subscription-id>
      </mdt-subscription>
     </mdt-config-data>
   </config>
 </edit-config>
```

Named Receivers

With FQDN support, a new method of configuring receivers is introduced, called the named-receiver configuration. Named receivers are top-level configuration entities that can exist independent of subscriptions. Named receivers are identified by a name. The name is an arbitrary string, and is the index or key of the named receiver records in the system. The named receiver configuration contains all configurations associated with the receiver that is not subscription-dependent.

The advantages of using named receivers are as follows:

- Capable of supporting different types of receivers.
- Better state and diagnostics information.
- Hostname can be used instead of an IP address to specify the host for protocol receivers.
- Parameters of a receiver that is used by multiple subscriptions can be changed at a single place.

Only protocol-type named receivers are supported, and these are:

- cloud-native: Cloud native protocol
- cntp-tcp: Civil Network Time Protocol (CNTP) TCP protocol
- cntp-tls: CNTP TLS protocol
- grpc-tcp: gRPC TCP protocol
- grpc-tls: gRPC TLS protocol
- native: Native protocol
- tls-native: Native TLS protocol

Named Protocol Receivers

Named protocol receivers are used to specify telemetry transports that use protocols. In addition to the name that identifies a receiver, named protocol receivers also use a host specification. The host specification takes a hostname or IP address, and a destination port number. Secure protocol transports also use a profile string.

Note When a valid named protocol receiver is created, it is not automatically connected to the receiver. The named protocol receiver must be requested by at least one subscription to create a connection to the receiver.

You can configure a named protocol receiver by using the CLI or YANG models.

Configuring the Named Protocol Receiver Using YANG Models

The YANG model, Cisco-IOS-XE-mdt-cfg, contains the named protocol receiver. The container mdt-named-protocol-rcvrs inside the top level mdt-config-data container has a list of mdt-named-protocol-rcvr structures. This group has five members:

- Name, which is the index in the list
- Protocol
- Profile

- Hostname
- Port number

The following is a sample NETCONF RPC that shows how to create a named protocol receiver:

```
<edit-config>
 <target>
  <running/>
 </target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
   <mdt-named-protocol-rcvrs>
    <mdt-named-protocol-rcvr>
     <name>my-receiver</name>
     <protocol>grpc-tls</protocol>
     <profile>tls-profile</profile>
     <host>
      <hostname>collector.cisco.com</hostname>
     </host>
     <port>57500</port>
    </mdt-named-protocol-rcvr>
   </mdt-named-protocol-rcvrs>
  </mdt-config-data>
 </config>
</edit-config>
```

Subscription Configuration Using Named Receivers

To use a named receiver with a subscription, both the receiver type and receiver name must be specified. No additional receiver configuration is required, since all receiver-specific configuration is part of the named receiver configuration. However; named protocol receivers still use the source VRF and source address of the subscriptions as part of the connection resolution process.

The only supported name receiver type is *protocol*.

Subscriptions can use either named receivers or legacy receivers, but cannot use both. If the legacy receiver is configured, setting the subscription receiver type and a named-receiver name is blocked. Similarly, if a subscription receiver type or a named receiver is specified, you cannot configure legacy receivers.

Note that subscriptions use only one receiver, even if more than one receiver is configured.

Subscriptions using legacy receivers and subscriptions using named receivers are permitted to use the same connection; however, it is not recommended.

Configuring a Named-Receiver Subscription Configuration Using YANG Model

The only value supported for rcvr-type is rcvr-type-protocol, when named receivers are used. When legacy receivers are used, the value is the default rcvr-type-unspecified.

The following is a sample NETCONF RPC that shows how to create a subscription using a named protocol-receiver:

```
<edit-config>
 <target>
  <running/>
 </target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
   <mdt-subscription>
```

```
    <subscription-id>1</subscription-id>
    <base>
     <rcvr-type>rcvr-type-protocol</rcvr-type>
    </base>
    <mdt-receiver-names>
     <mdt-receiver-name>
      <name>receiver1</name>
     </mdt-receiver-name>
    </mdt-receiver-names>
   </mdt-subscription>
  </mdt-config-data>
 </config>
</edit-config>
```

Named Receiver Operation and Operational State

Named receiver objects and subscription receiver objects (that refer to the named receiver) have two different operational states. The operational states are valid or invalid. The most common reason for a named receiver to be invalid is incomplete configuration, however; it could also be due to other reasons. The operational state view of a named receiver has a field that provides a text explanation on why the receiver is invalid. When the receiver state is valid, this field is empty.

Displaying Named Receiver State Using the CLI

To view the state of named receivers of all types, use the **show telemetry receiver** command. The **all** keyword displays information about all named receivers in a brief format, and the **name** keyword displays detailed information about the specified named receiver.

The following is sample output from the **show telemetry receiver all** command:

```
Device# show telemetry receiver all

Telemetry receivers

  Name       <…>     Type        Profile               State       Explanation
  -----------<…>-------------------------------------------------------------------
  receiver1  <…>     protocol    tls-trustpoint        Valid
```

The following is sample output from the **show telemetry receiver name** command:

```
Device# show telemetry receiver name my-receiver

  Name: my-receiver
  Profile: tls-profile
  State: Valid
  State Description:
  Last State Change: 09/18/24 15:50:54
  Type: protocol
  Protocol: grpc-tls
  Host: collector.cisco.com
  Port: 45000 (edited)
```

Named Receiver State Using YANG Models

The state of the named receivers can be retrieved using the Cisco-IOS-XE-mdt-oper-v2 YANG model. The mdt-oper-v2-data container contains an mdt-named-receivers list that contains the operational state of all named receivers.

The following is a sample NETCONF reply to retrieve the state of named receivers:

```
<get>
 <filter>
  <mdt-oper-v2-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper-v2">
   <mdt-named-receivers/>
  </mdt-oper-v2-data>
 </filter>
</get>

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
 <data>
  <mdt-oper-v2-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper-v2">
   <mdt-named-receivers>
    <name>my-receiver</name>
    <profile>tls-profile</profile>
    <params>
     <protocol>grpc-tls</protocol>
<host>
      <hostname>rcvr.test.com </hostname>
     </host>
     <port>45000</port>
    </params>
    <state>named-rcvr-state-valid</state>
    <last-state-change-time>2020-…:00</last-state-change-time>
   </mdt-named-receivers>
  </mdt-oper-v2-data>
 </data>
</rpc-reply>
```

Subscription Receiver Operation and Operational States

Subscription receivers are the subscription-related objects that connects to the actual subscription receiver or collector. While the mechanism needed to reach the collector is specific to the receiver type, a connection is the entity that is used to allow the subscription to reach its receiver or collector.

Subscription receiver state is based on its ability to request and use the connection to the receiver and has a number of states that are associated with the control of other resources required to allow the subscription to send updates to the receiver or collector.

Subscription Receiver States

The operational state of a subscription receiver consists of the configured name (that is the index of the connection), the state of the receiver, an explanation or note about the state, and the time of the last state change. The explanation string is not always used.

The possible states of a subscription receiver are shown in the table below.

Table 31: Subscription Receiver States

Subscription Receiver State		**Description**
CLI Value	**YANG Value**	
Disconnected	rcvr-state-disconnected	The receiver is disconnected and no attempt is made to reconnect it.

Subscription Receiver State		Description
Resolving	rcvr-state-resolving	Resolving the connection parameters required to reach the receiver.
Transport requested	rcvr-state-transport-requested	A request for a connection to reach the receiver was using the connection parameters determined from the resolving state.
Connecting	rcvr-state-connecting	Resources needed to connect the subscription to the receiver are being allocated.
Connected	rcvr-state-connected	The subscription is connected to the receiver, and updates can flow to the receiver.
Disconnecting	rcvr-state-disconnecting	Resources used on the connection are being re-allocated.

The YANG value rcvr-state-invalid is used only by legacy receivers. Subscription receivers that are invalid cannot be connected, so the subscription receiver state is set to disconnected when it is invalid. The explanation string provides the distinction between invalid subscription receivers and disconnected subscription receivers.

A subscription receiver may be disconnected due to the following reasons:

- Another receiver on the subscription is not disconnected.
- Connection setup failed permanently.
- Named receiver does not exist.
- Named receiver is not the type specified in the subscription.
- Named receiver is not valid.
- Subscription is invalid.
- The requested connection is in use by a different receiver.

Subscription Receiver Connections

This section provides information on how subscription receivers use connections.

Telemetry Connections

Telemetry connections represent the transport instances used by subscriptions to reach the receivers and are purely operational. Telemetry connections are identified by an integer index value. Other information about the connections is specific to the type of connection, which is based on the type of receiver that the subscription is configured to use.

For the secure dial-out transports, the host part of the configured named receiver must match the distinguished name (DN) of the certificate provided by the receiver, when the connection is set up. For this reason, it is not permitted to have more than one receiver using the same connection.

While all the states discussed in this section are available to all types of connections, not all have to be used.

Table 32: Telemetry Connection States

Connection State		Description
CLI Value	**YANG Value**	
Pending	con-state-pending	The connection has been created, but not yet initiated.
Connecting	con-state-connecting	A request to set up the connection is in progress.
Active	con-state-active	The connection is up and is available for use by subscription receivers.
Disconnecting	con-state -disconnecting	The connection has been torn down and is waiting to be released by subscription receivers.

Additional operational state associated with a connection includes the identity of the remote receiver (the peer, when available), and the time of the last state change.

Telemetry Protocol Connections

This section discusses protocol type connections and how these are used by subscription receivers that are assigned to named protocol receivers.

Table 33: Parameters of a Protocol-Type Connection

Parameter	Origin	Comments
Destination IP address	Named receiver host	Because hosts use domain names, domain name resolution may be required.
Destination port number	Named receiver port	Must be explicitly configured.
Source VRF	Subscription, if specified	Default VRF is used, if not specified. Otherwise the VRF name is resolved to an internal identifier.
Source IP address	Subscription, if specified	If not specified, the source IP address is determined based on the VRF and destination IP address.

Some of these parameters are based on the configuration of the subscription receiver's parent subscription.

When resolving the connection parameters from the configuration, the VRF is determined first, followed by the destination IP address, and finally the source IP address, if an order is not specified. If a given step in the resolution fails non-permanently, there are infinite retries at 5 second intervals.

A connection is instantiated as soon as it is requested. That is, as soon as the first subscription receiver goes from the resolving state to the transport requested state, a connection instance with the parameters that were resolved by the subscription receiver is created.

If the requested connection is successfully setup and used by telemetry, the connection state changes to connected, which means that a connection exists between the Cisco IOS XE device and the receiver device. To reallocate the resources used by the receiver, the subscription receivers that want to use the resources are informed that the connection is set up. These subscription receivers then transition to the connecting state to set up the resources required to connect the subscription to the receiver. Once these resources are in place, the subscription receiver's state changes to connected, and update notifications are received by the receiver.

The following are some of the reasons why a telemetry connection cannot become active:

- Destination unreachable.
- No listener at the remote host port.
- Listener at the remote host port is of the wrong type.
- Authentication failures.

Note When a connection setup is in progress, any subscription receiver using this connection is in the connecting state because it has successfully resolved the parameters needed to initiate the connection setup.

The action taken when a connection setup fails is specific to the protocol. The following table shows the retry behaviors for connections within a single setup request and for re-resolution requests when the connection setup request fails. This behavior is the same for connections requested by the legacy receivers as well.

Table 34: Protocol-Specific Retry Intervals

Protocol	Connection Retries	Re-resolution Requests
• grpc-tcp • grpc-tls	5 retries at 1, 3, 4, and 7 seconds in between them	No limit; continuously requests re-resolution when connection retries fail. (14 seconds per try.)
• cloud-native • cntp-tcp • cntp-tls • native • tls-native		5, 10, 15, 20, 25, and 30 seconds.

Troubleshooting Named Receiver Connections

When a subscription is set up, one of the common problems is that no telemetry update messages are received. Possible reasons could be that there are no events to send, or the subscription is not valid. This section describes how to troubleshoot some of the common problems that occur in named receiver connections.

The logs from the telemetry process, and the output of some of the **show** commands provide information that can be used for troubleshooting the named receiver configuration.

Table 35: Troubleshooting Named Receiver Connections

Problem	How to Check/Symptom	What to Do
Subscription is not valid.	**show telemetry ietf subscription** *id* **details**	Fix the subscription configuration.
Subscription receiver is not valid.	**show telemetry ietf subscription** *id* **receiver**	Fix the named receiver configuration.
Subscription receiver's connection parameters cannot be resolved.	**show telemetry ietf subscription** *id* **receiver** Subscription receiver state appears to never leave the resolving state.	Verify the receiver, the network configuration, or the interface state.
Subscription receiver connection does not come up.	**show telemetry ietf subscription** *id* **receiver** Subscription receiver state constantly changes from resolving to connecting.	Verify that the resolved connection is valid, and the receiver or collector is reachable and able to accept inbound connections using the specified transport.
Subscription receiver connections are rejected.	**show telemetry ietf subscription** *id* **receiver** Subscription receiver state constantly changes through all states except disconnected.	Verify that the collector is of the correct type, and that the configured authentication and authorization is valid.
Subscription receiver is connected, but no updates are received.	**show telemetry internal subscription** *id* **stats** Message drop count is incrementing, but the records sent is not.	Verify that the collector is able to keep up with the flow of update notifications.
Subscription receiver is connected, but no updates are received.	**show telemetry internal subscription** No change in the count.	If the subscription is on-change, ensure that there really have been no events. If the subscription is periodic, ensure that the update period is small, that the time is specified in hundredths of a second.

show telemetry internal connection: This command takes an optional connection index value. When no index is specified, it displays the basic connection parameter information for all connections that are being used. When a connection index is specified in the command, it shows low-level details about the connection. The command output is transport-specific, and might not be available for all transports. The output from this command is subject to change.

show telemetry internal diagnostics: This command attempts to dump all telemetry logs and operational state. When reporting problems, it may be helpful to use this command as close to the problem time as possible and provide the output of the **show running-config | section telemetry** command as well.

FQDN Support for gRPC Subscriptions

gRPC telemetry subscriptions are configuration-based, which means that users must specify the receiving host and other subscription parameters as part of the device configuration. This receiver configuration is used to determine the connection details for sending telemetry updates. With the introduction of the FQDN Support for gRPC Subscriptions feature, along with IP addresses, Fully Qualified Domain Names (FQDNs) can also be used for gRPC subscriptions. This feature cannot be disabled.

To use FQDN, a subscription must use a named receiver, as described earlier. This example shows an FQDN used for gRPC subscriptions.

```
telemetry receiver protocol my-receiver
 host name receiver.cisco.com 12345
 protocol grpc-tcp

telemetry ietf subscription 101

 encoding encode-kvgpb
 filter xpath /mdt-oper-v2-data/mdt-subscriptions

 stream yang-push
 update-policy periodic 1000

 receiver name my-receiver

telemetry ietf subscription 102

 encoding encode-kvgpb
 filter xpath /mdt-oper-v2-data/mdt-connections

 stream yang-push
 update-policy periodic 1000

 receiver name my-receiver
```

Displaying On-Change Subscription Capabilities

From Cisco IOS XE 17.6.1, Cisco IOS XE software supports the capabilities model, *Cisco-IOS-XE-mdt-capabilities-oper.yang* that can be queried to determine the paths that support on-change subscriptions and their transports. This model is patterned after the IETF models, *ietf-system-capabilities* and *ietf-notification-capabilities*, which are described in RFC 9196.

Within a single YANG model, some nodes may support on-change, while others may not. Because of this reason, capabilities cannot be specified at the model level. Instead of listing every individual leaf and container node that supports on-change, the *Cisco-IOS-XE-mdt-capabilities-oper.yang* model essentially lists only the inflection points, that is, the paths to the nodes whose support differs from that of their parent node. This makes the capabilities model very compact.

The model has two nodes.

- System-level capabilities: At the top of the model are the system-level capabilities that are similar to the system default; if no matching entry is found in the path of interest, the capabilities are given by the system-level container. The system-level capabilities for Cisco IOS XE support only periodic subscriptions; on-change subscriptions are not supported.
- mdt-datastore: The model specifies capabilities on a per-datastore basis, through the *mdt-datastore* list. Currently, only the *running* datastore is supported.

The mdt-datastore indicates which datastore a given set of capabilities apply to. Datastores are described in RFC 8342.

- mdt-per-node: Below the *mdt-datastore* list is the per-node capabilities list, *mdt-per-node*. Each entry in this list represents an inflection point, and each of these entries contains an XPath and a set of capabilities.

Determining the Capabilities of a Node

To determine the capabilities of a particular node in a YANG model, the *mdt-per-node* list is searched to verify if any entry has an XPath that selects this node. The order of the entries is important. The first entry that selects a given node is what determines the capabilities of that node. During a search, more-specific XPaths typically appear before less-specific ones. Note that the XPath in the *mdt-per-node* list may not be an exact match for the node being checked.

In this example, the *mdt-per-node* list contains the following XPaths, in the order:

1. /interfaces/interface/state/counters
2. /interfaces/interface/state
3. /interfaces/interface

The capabilities of the node */interfaces/interface/state* will be entry 2, the capabilities of the node */interfaces/interface/config*, will be entry 3, and the capabilities of the node */interfaces* will be given by the system capabilities, because this node is not selected by any entry.

Transport-Specific Capabilities

The capabilities model has another level—the transport-dependent, per-node capabilities list, *mdt-transport-per-node*. This node works the same as the *mdt-per-node*; but it gives capabilities for a particular transport protocol (gNMI, gRPC, or NETCONF). This node is required because in some cases, on-change support can differ depending on the transport that is used to create subscriptions. For example, gRPC supports on-change on some paths that NETCONF does not support.

To determine the capabilities of a node for a particular transport, first check the *mdt-transport-per-node* list for that transport. If no match is found, check the *mdt-per-node* list. If again no match is found, then fall back to the system capabilities.

To determine the general on-change support without regard to a particular transport, skip the *mdt-transport-per-node* list, and start checking from the *mdt-per-node* list.

Capability Specification

A capability specification, either at the system or per-node level consists of two fields, *is-valid* and a bit field.

The *is-valid* field is an empty leaf node. The presence of the *is-valid* field indicates that the capability applies to that entry. Not all entries will specify all capabilities. For example, entries in the *mdt-per-node* list only specify the on-change capabilities, and not the periodic capabilities. As a result, these entries will have on-change-is-valid set, and periodic-is-valid unset.

The bit field indicates the actual support. There are two bit values, *mdt-cap-notif-config* and *mdt-cap-notif-state*, which indicate that the capability is supported for config or operational nodes, respectively.

This sample capability specification indicates that on-change is supported for both config and operational nodes; however it does not indicate periodic support.

```
<sub-caps>
  <on-change-is-valid/>
  <on-change-supported>mdt-cap-notif-config mdt-cap-notif-state</on-change-supported>
</sub-caps>
```

This sample capability specification indicates that on-change is not supported, because neither of the bits is set.

```
<sub-caps>
  <on-change-is-valid/>
  <on-change-supported/>
</sub-caps>
```

Subscription Dampening Period for On-Change Telemetry

This feature introduces a dampening period as defined by RFC 8641 for on-change subscriptions. When a dampening period is configured, the publisher streams the latest version of all changed records at the end of the dampening period. The dampening period is supported only for native TDL telemetry.

Without a dampening period, the receiver may be flooded with repeated updates that contains fast-changing objects. Because of these repeated updates, the publisher may also need to spend more processing time to process the on-change events, exhausting resources in the publisher and/or receiver. To mitigate this impact, the dampening period is defined for on-change subscriptions.

A value of zero indicates that no dampening is configured, and the dampening period defaults to zero, if it is not explicitly set. Based on the platform, the maximum and minimum values that can be set for the dampening period changes.

To configure the dampening period through the CLI, use the **dampening-period** *interval* command in update on-change configuration mode. You can also configure the dampening period through the Cisco-IOS-XE-mdt-common-defs.yang model.

Note Dampening period is not supported for some specific TDL filters or URI nodes. You need to configure subscriptions to determine, if dampening period is supported. If dampening period is not supported, the following error message is displayed when the **show telemetry ietf subscription** *subscription_id*> **detail** command is executed:

```
Value 'x' for parameter 'dampening-period' is not supported by the specified filter (supported
 values include '0')
```

Here 'x' is the interval configured for the dampening-period command.

Subscription Monitoring

Subscriptions of all types can be monitored by using CLIs and management protocol operations.

CLI

Use the **show telemetry ietf subscription** command to display information about telemetry subscriptions. The following is sample output from the command:

```
Device#  show telemetry ietf subscription 2147483667 detail

Telemetry subscription detail:
```

```
  Subscription ID: 2147483667
  State: Valid
  Stream: yang-push
  Encoding: encode-xml
  Filter:
    Filter type: xpath
    XPath: /mdt-oper:mdt-oper-data/mdt-subscriptions
  Update policy:
    Update Trigger: periodic
    Period: 1000
  Notes:
```

NETCONF

The following is a sample NETCONF message that displays information about telemetry subscriptions:

```
<get>
<filter>
<mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
<mdt-subscriptions/>
</mdt-oper-data>
</filter>
</get>


* Enter a NETCONF operation, end with an empty line
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-subscriptions>
        <subscription-id>101</subscription-id>
        <base>
          <stream>yang-push</stream>
          <encoding>encode-kvgpb</encoding>
          <source-vrf>RED</source-vrf>
          <period>10000</period>
          <xpath>/ios:native/interface/Loopback[name="1"]</xpath>
        </base>
        <type>sub-type-static</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
          <address>5.22.22.45</address>
          <port>57500</port>
          <protocol>grpc-tcp</protocol>
          <state>rcvr-state-connecting</state>
          <comments/>
          <profile/>
          <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
        </mdt-receivers>
        <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
      </mdt-subscriptions>
      <mdt-subscriptions>
        <subscription-id>2147483648</subscription-id>
        <base>
          <stream>yang-push</stream>
          <encoding>encode-xml</encoding>
          <source-vrf/>
          <period>1000</period>
         <xpath>/if:interfaces-state/interface[name="GigabitEthernet0/0"]/oper-status</xpath>

        </base>
```

```
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
          <address>5.22.22.45</address>
          <port>51259</port>
          <protocol>netconf</protocol>
          <state>rcvr-state-connected</state>
          <comments/>
          <profile/>
          <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
        </mdt-receivers>
        <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
      </mdt-subscriptions>
    </mdt-oper-data>
  </data>
</rpc-reply>
```

Streams

A stream defines a set of events that can be subscribed to, and this set of events can be almost anything. However, as per the definition of each stream, all possible events are related in some way. This section describes the supported streams.

To view the set of streams that are supported, use management protocol operations to retrieve the *streams* table from the Cisco-IOS-XE-mdt-oper module (from the YANG model Cisco-IOS-XE-mdt-oper.yang) in the *mdt-streams* container.

The following example shows how to use NETCONF to retrieve supported streams:

```
<get>
<filter>
<mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
<mdt-streams/>
</mdt-oper-data>
</filter>
</get>

* Enter a NETCONF operation, end with an empty line

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-streams>
        <stream>native</stream>
          <stream>yang-notif-native</stream>
          <stream>yang-push</stream>
      </mdt-streams>
    </mdt-oper-data>
  </data>
</rpc-reply>
```

The example shows that three streams are supported: *native*, *yang-notif-native*, and *yang-push*. The stream *native* is not available for general use and can be ignored.

Note Currently there are no CLIs to return the list of supported streams.

The yang-push Stream

The *yang-push* stream is the data in configuration and operational databases that is described by a supported YANG model. This stream supports an XPath filter to specify what data is of interest within the stream, and where the XPath expression is based on the YANG model that defines the data of interest.

Update notifications for this stream can be sent either when data changes or during fixed periods, but not for both, for a given subscription. Subscriptions for data that does not currently exist are permitted, and these run as normal subscriptions.

The only target database that is supported is *running*.

Determining On-Change Capability

Currently, there is *no* indication within YANG models about the type of data that can be subscribed to, by using an on-change subscription. Attempts to subscribe to data that cannot be subscribed to by using on-change subscription results in a failure (dynamic) or an invalid subscription (configured). For more information on On-Change Publication, see the section, *On-Change Publication for yang-push*.

IETF Draft Compliance

Telemetry using the *yang-push* stream is based on the IETF NETCONF working group's early drafts for telemetry. These are:

- *Custom Subscription to Event Notifications, Version 03*
- *Subscribing to YANG datastore push updates, Version 07*

Note The following features that are described in the corresponding drafts are not supported:

- Subtree filters
- Out-of-band notifications
- Any subscription parameter not explicitly stated as supported

X-Path Filter for yang-push

The dataset within the *yang-push* stream to be subscribed to should be specified by the use of an XPath filter. The following guidelines apply to the XPath expression:

- XPath expressions can have keys to specify a single entry in a list or container. The supported key specification syntax is

  ```
  [{key name}={key value}]
  ```

  The following is an example of an XPath expression:

  ```
  filter xpath
  /rt:routing-state/routing-instance[name="default"]/ribs/rib[name="ipv4-default"]/routes/route

  # VALID!
  ```

  Compound keys are supported by the use of multiple key specifications. Key names and values must be exact; no ranges or wildcard values are supported.

- In XPath expressions, select multiple keys using [] between the keys, and encapsulate the string with “. The following is an example of an XPath expression:

```
filter xpath
/environment-ios-xe-oper:environment-sensors/environment-sensor[location=\"Switch\ 1\"]
[name=\"Inlet\ Temp\ Sens\"]/current-reading
```

- XPath expressions support the use of the union operator (|) to allow a single subscription to support multiple objects. The union operator only works for NETCONF transport and not for gRPC.

XPath Expressions Supported on Cisco Catalyst 9800 Wireless Controllers

In Cisco IOS XE Bengaluru, 17.4.1, the following set of OpenConfig XPath expressions are supported on the Cisco Catalyst 9800 Series Wireless Controllers.

Ensure that you run the following RPC using any of the programmability interfaces, such as NETCONF, RESTCONF, or gNMI protocol, to enable telemetry subscription:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <provision-aps xmlns="http://openconfig.net/yang/wifi/ap-manager">
        <provision-ap>
          <mac>eth_mac_of_the_AP</mac>
          <config>
            <mac>eth_mac_of_the_AP</mac>
            <hostname>AP_NAME</hostname>
          </config>
        </provision-ap>
      </provision-aps>
    </config>
  </edit-config>
</rpc>
```

All of the XPath expressions listed below are a part of the *openconfig-access-points* YANG model, except the last one, which is a part of the *openconfig-ap-manager* YANG model. For the telemetry operation to work correctly, ensure that configurations are done based on the OpenConfig model.

- /access-points/access-point/radios/radio/state
- /access-points/access-point/radios/radio/neighbors/neighbor
- /access-points/access-point/radios/radio/neighbors/neighbor/state
- /access-points/access-point/ssids/ssid/bssids/bssid/state/counters
- /access-points/access-point/ssids/ssid/clients/client/state/counters
- /access-points/access-point/ssids/ssid/clients/client/client-rf/state
- /access-points/access-point/ssids/ssid/clients/client/client-connection/state
- /access-points/access-point/system/aaa/server-groups/server-group/servers/server/radius/state
- /joined-aps/joined-ap/state/opstate

When you subscribe to an XPath, you receive data for the subscribed XPath and all the XPaths under it in the hierarchy. For example, subscribing to */access-points/access-point/radios/radio/state* delivers data for all the leaves associated with it, as well as the subcontainers under it.

If you require only a subset of information, set filters in the XPath expressions to limit the updates. To filter the data of a specific access point (AP), use a key after the node. For example, to receive data for an AP with hostname 'my_hostname', use the subscription *XPath: access-point[hostname='my_hostname']*. Note that the data updates will contain data objects from all the leaves, and not just from the limited subset that is defined.

Scale Information

The following tables show the minimum recommended intervals for each of the gathering points under three different scale scenarios.

Scenario1: Full Scale with four SSIDs

Table 36: Setup

APs	2,000
Clients	30,000
SSIDs per AP	4
BSSIDs per AP	8
Physical neighbors per AP	12
Neighbors per AP	96

Table 37: Recommended Intervals

Gathering Point	Records	Recommended Interval (Seconds) One Collector	Recommended Interval (Seconds) Two Collectors
Joined	2000	30	60
AAA	2000	30	60
Radio	4000	30	60
Client RF	30,000	30	60
Client CNTR	30,000	30	60
Client CONN	30,000	60	120
BSSID	16,000	90	180
Neighbor	192,000	180	360

Scenario2: Full Scale with six SSIDs

Table 38: Setup

APs	2,000
Clients	30,000
SSIDs per AP	6
BSSIDs per AP	12
Physical neighbors per AP	12
Neighbors per AP	144

Table 39: Recommended Intervals

Gathering point	Records	Recommended Interval (Seconds) One Collector	Recommended Interval (Seconds) Two Collectors
Joined	2000	30	60
AAA	2000	30	60
Radio	4000	30	60
Client RF	30,000	30	60
Client CNTR	30,000	30	60
Client CONN	30,000	60	120
BSSID	24,000	120	240
Neighbor	288,000	240	420

Scenario3: Reduced Scale with six SSIDs

Table 40: Setup

APs	1,000
Clients	15,000
SSIDs per AP	6
BSSIDs per AP	12
Physical neighbors per AP	12
Neighbors per AP	144

Table 41: Recommended Intervals

Gathering Point	Records	Recommended Interval (Seconds) One Collector	Recommended Interval (Seconds) Two Collectors
Joined	1000	NA	30
AAA	1000	NA	30
Radio	2000	NA	30
Client RF	15,000	NA	30
Client CNTR	15,000	NA	30
Client CONN	15,000	NA	30
BSSID	12,000	NA	120
Neighbor	144,000	NA	180

XPath Values and Corresponding Rates on Cisco Catalyst 9800 Wireless Controllers

In the Cisco-IOS-XE-wireless-mesh-rpc, following are the permitted values and corresponding rates for XPath `/exec-linktest-ap/data-rate-idx`:

```
ewlc-mesh-linktest-rate-idx-1 1 Mbps
ewlc-mesh-linktest-rate-idx-2 2 Mbps
ewlc-mesh-linktest-rate-idx-3 5 Mbps
ewlc-mesh-linktest-rate-idx-4 6 Mbps
ewlc-mesh-linktest-rate-idx-5 9 Mbps
ewlc-mesh-linktest-rate-idx-6 11 Mbps
ewlc-mesh-linktest-rate-idx-7 12 Mbps
ewlc-mesh-linktest-rate-idx-8 18 Mbps
ewlc-mesh-linktest-rate-idx-9 24 Mbps
ewlc-mesh-linktest-rate-idx-10 36 Mbps
ewlc-mesh-linktest-rate-idx-11 48 Mbps
ewlc-mesh-linktest-rate-idx-12 54 Mbps
ewlc-mesh-linktest-rate-idx-13 108 Mbps
ewlc-mesh-linktest-rate-idx-14 m0
ewlc-mesh-linktest-rate-idx-15 m1
ewlc-mesh-linktest-rate-idx-16 m2
ewlc-mesh-linktest-rate-idx-17 m3
ewlc-mesh-linktest-rate-idx-18 m4
ewlc-mesh-linktest-rate-idx-19 m5
ewlc-mesh-linktest-rate-idx-20 m6
ewlc-mesh-linktest-rate-idx-21 m7
ewlc-mesh-linktest-rate-idx-22 m8
ewlc-mesh-linktest-rate-idx-23 m9
ewlc-mesh-linktest-rate-idx-24 m10
ewlc-mesh-linktest-rate-idx-25 m11
ewlc-mesh-linktest-rate-idx-26 m12
ewlc-mesh-linktest-rate-idx-27 m13
ewlc-mesh-linktest-rate-idx-28 m14
ewlc-mesh-linktest-rate-idx-295 m15
```

Periodic Publication for yang-push

With periodic subscriptions, the first push-update with the subscribed information is sent immediately; but it can be delayed if the device is busy or due to network congestion. Updates are then sent at the expiry of the

configured periodic timer. For example, if the period is configured as 10 minutes, the first update is sent immediately after the subscription is created and every 10 minutes thereafter.

The period is time, in centiseconds (1/100 of a second), between periodic push updates. A period of 1000 will result in getting updates to the subscribed information every 10 seconds. The minimum period that can be configured is 100, or one second. There is no default value. This value must be explicitly set in the <establish-subscription> RPC for dynamic subscriptions and in the configuration for configured subscriptions.

Periodic updates contain a full copy of the subscribed data element or table for all supported transport protocols.

When subscribing for empty data using a periodic subscription, empty update notifications are sent at the requested period. If data comes into existence, its values at the next period are sent as a normal update notification.

On-Change Publication for yang-push

When creating an on-change subscription, the dampening period must be set to 0 to indicate that there is no dampening period; no other value is supported.

With on-change subscriptions, the first push update is the entire set of subscribed to data (the initial sychronization as defined in the IETF documents). This is not controllable. Subsequent updates are sent when the data changes, and consist of only the changed data. However, the minimum data resolution for a change is a row. So, if an on-change subscription is to a leaf within a row, if any item in that row changes, an update notification is sent. The exact contents of the update notification depend on the transport protocol.

In addition, on-change subscriptions are not hierarchical. That is, when subscribing to a container that has child containers, changes in the child container are not seen by the subscription.

Subscriptions for data that does not currently exist are permitted and run as normal subscriptions. The initial synchronization update notification is empty and there are no further updates until data is available.

XPath expressions must specify a single object. That object can be a container, a leaf, a leaf list or a list.

The yang-notif-native Stream

The *yang-notif-native* stream is any YANG notification in the publisher where the underlying source of events for the notification uses Cisco IOS XE native technology. This stream also supports an XPath filter that specifies which notifications are of interest. Update notifications for this stream are sent only when events that the notifications are for occur.

Since this stream supports only on-change subscriptions, the dampening interval must be specified with a value of 0.

XPath Filter for yang-notif-native

The dataset within the *yang-notif-native* stream to be subscribed to is specified by the use of an XPath filter. The following guideline applies to the XPath expression:

- XPath expressions must specify an entire YANG notification; attribute filtering is not supported.
- The union operator (|) is not supported.

Transport Protocol

The protocol that is used for the connection between a publisher and a receiver decides how the data is sent. This protocol is referred to as the transport protocol, and is independent of the management protocol for

configured subscriptions. The transport protocol affects both the encoding of the data, for example XML, Google Protocol Buffers (GPB) and the format of the update notification itself.

Note The stream that is chosen may also affect the format of the update notification.

Supported transport protocols are gNMI, gRPC, and NETCONF.

NETCONF Protocol

The NETCONF protocol is available only for the transport of dynamic subscriptions, and can be used with *yang-push* and *yang-notif-native* streams.

Three update notification formats are used when using NETCONF as the transport protocol:

- When the subscription uses the *yang-push* stream, and if it is periodic or when the initial synchronization update notification is sent on an on-change subscription.
- When the subscription uses the *yang-push* stream and it is an on-change subscription, other than the initial synchronization update notification.
- When the subscription uses the *yang-notif-native* stream.

The yang-push Format

When the *yang-push* source stream is sent over NETCONF as a transport with XML encoding, two update notification formats are defined. These update notification formats are based on the *draft-ietf-netconf-yang-push-07*. For more information, see section 3.7 of the IETF draft.

The yang-notif-native Format

When the source stream is *yang-notif-native*, the format of the update notification when encoded in XML over NETCONF is as defined by *RFC 7950*. For more information, see section 7.16.2 of the RFC.

Unlike the formats for the *yang-push* stream, the subscription ID is not found in the update notification.

gRPC Protocol

The gRPC protocol is available only for the transport of configured subscriptions, and can be used with *yang-push* and *yang-notif-native* streams. Only kvGPB encoding is supported with gRPC transport protocol.

Receiver connection retries based on gRPC protocol (exponential back-off) are supported.

For telemetry messages defined in .proto files, see: *mdt_grpc_dialout.proto* and telemetry.proto.

Mutual Authentication for gRPC Telemetry

gRPC is one of the supported dial-out protocols used to transmit telemetry data. For dial-out protocols, the device is considered the *client* and the collector, the *server*. gRPC supports both unencrypted TCP and encrypted TLS-based connections.

A new gRPC-TLS profile that contains a pair of trustpoints is added to the telemetry configuration, so that a client ID certificate can be used for mutual authentication. The profile contains two trustpoints, one is the Certificate Authority (CA) certificate for server validation, and the other is the ID certificate for client validation.

When a device connects to a receiver for the first time, based on the server configuration, client or mutual authentication may be required. The device will receive the receiver's identity certificate and validate whether the certificate is signed by the CA identified in the certificate associated with the trustpoint configured in the receiver profile. If the receiver then requests for the certificate ID of the device, the device sends the client ID certificate previously installed in the profile's ID-trustpoint field.

If the server is configured to require mutual authentication, and there is no client ID trustpoint in the profile, the client authentication will not happen, nor will the connection succeed.

The same *trustpoint* label can be configured for multiple profiles, and the same profile can be configured for multiple receivers.

Note The trustpoint with the client ID is not mandatory in the profile configuration, as mutual authentication is not required for gRPC over TLS. As in prior releases, gRPC over TLS can be configured only with server validation.

To add the client ID trustpoint, use the **telemetry protocol grpc profile <name>** command.

This feature cannot be disabled; but it can be left unused by not configuring the receivers to use the gRPC-TLS protocol, or by removing or not configuring the client ID trustpoint field in the receiver configuration.

This example shows how to configure a gRPC-TLS profile for mutual authentication.

```
telemetry receiver protocol grpc-mtls
 host name collector.cisco.com 57500
 protocol grpc-tls profile different-ca

telemetry ietf subscription 100
 encoding encode-kvgpb
 filter xpath /memory-ios-xe-oper:memory-statistics/memory-statistic
 receiver-type protocol
 stream yang-push
 update-policy periodic 5000
 receiver name grpc-mtls
```

High Availability in Telemetry

Dynamic telemetry connections are established over a NETCONF session through SSH to the active switch or a member in a switch stack, or the active route processor in a high-availability-capable device. After switchover, you must destroy and re-establish all the sessions that use crypto, including NETCONF sessions that carry telemetry subscriptions. You must also re-create all the dynamic subscriptions after a switchover. gNMI dial-in subscriptions also work the same as a NETCONF session through SSH.

gRPC dial-out subscriptions are configured on the device as part of the running configuration of the active switch or member of the stack. When switchover occurs, the existing connections to the telemetry receivers are torn down and reconnected (as long as there is still a route to the receiver). Subscriptions need not be reconfigured.

Note In the event of a device reload, subscription configurations must be synchronized to the start-up configuration of a device. This ensures that after the device reboots, subscription configurations remain intact on the device. When the necessary processes are up and running, the device attempts to connect to the telemetry receiver and resume normal operations.

Pubd Restartability

In Cisco IOS XE Cupertino 17.9.1, the pubd process is restartable on all platforms. Prior to this release, pubd was restartable only on certain platforms. On other platforms, to restart the pubd process, the whole device had to be restarted.

Pubd can be restarted by removing and re-adding the NETCONF-YANG or gNXI configuration, as applicable. Note that this will also restart the other NETCONF-YANG or gNXI processes.

Sample Model-Driven Telemetry RPCs

The following section provides a list of sample RPCs, and describes how to configure subscriptions.

Managing Configured Subscriptions

Note Currently, you can only use the gRPC protocol for managing configured subscriptions.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **telemetry ietf subscription** *id*
4. **stream yang-push**
5. **filter xpath** *path*
6. **update-policy** {**on-change** | **periodic**} *period*
7. **encoding encode-kvgpb**
8. **source-vrf** *vrf-id*
9. **source-address** *source-address*
10. **receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name*
11. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**telemetry ietf subscription** *id* **Example:** `Device(config)# telemetry ietf subscription 101`	Creates a telemetry subscription and enters telemetry-subscription mode.
Step 4	**stream yang-push** **Example:** `Device(config-mdt-subs)# stream yang-push`	Configures a stream for the subscription.
Step 5	**filter xpath** *path* **Example:** `Device(config-mdt-subs)# filter xpath /memory-ios-xe-oper:memory-statistics/memory-statistic`	Specifies the XPath filter for the subscription.
Step 6	**update-policy** {**on-change** | **periodic**} *period* **Example:** `Device(config-mdt-subs)# update-policy periodic 6000`	Configures a periodic update policy for the subscription.
Step 7	**encoding encode-kvgpb** **Example:** `Device(config-mdt-subs)# encoding encode-kvgpb`	Specifies kvGPB encoding.
Step 8	**source-vrf** *vrf-id* **Example:** `Device(config-mdt-subs)# source-address Mgmt-intf`	Configures the source VRF instance.
Step 9	**source-address** *source-address* **Example:** `Device(config-mdt-subs)# source-vrf 192.0.2.1`	Configures the source address.
Step 10	**receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name* **Example:** `Device(config-mdt-subs)# receiver ip address 10.28.35.45 57555 protocol grpc-tcp`	Configures the receiver IP address, protocol, and profile for notifications.
Step 11	**end** **Example:** `Device(config-mdt-subs)# end`	Exits telemetry-subscription configuration mode and returns to privileged EXEC mode.

Configuring On-Change gRPC Subscriptions

SUMMARY STEPS

1. **enable**
2. **configure terminal**

3. **telemetry ietf subscription** *id*
4. **stream yang-push**
5. **filter xpath** *path*
6. **update-policy** {**on-change** | **periodic** *period*}
7. **encoding encode-kvgpb**
8. **receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name*
9. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**telemetry ietf subscription** *id* **Example:** `Device(config)# telemetry ietf subscription 8`	Creates a telemetry subscription and enters telemetry-subscription mode.
Step 4	**stream yang-push** **Example:** `Device(config-mdt-subs)# stream yang-push`	Configures a stream for the subscription.
Step 5	**filter xpath** *path* **Example:** `Device(config-mdt-subs)# filter xpath /iosxe-oper:ios-oper-db/hwidb-table`	Specifies the XPath filter for the subscription.
Step 6	**update-policy** {**on-change**	**periodic** *period*} **Example:** `Device(config-mdt-subs)# update-policy on-change`	Configures an on-change update policy for the subscription.
Step 7	**encoding encode-kvgpb** **Example:** `Device(config-mdt-subs)# encoding encode-kvgpb`	Specifies kvGPB encoding.
Step 8	**receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name* **Example:**	Configures the receiver IP address, protocol, and profile for notifications.

	Command or Action	Purpose
	`Device(config-mdt-subs)# receiver ip address 10.22.22.45 45000 protocol grpc_tls profile secure_profile`	
Step 9	**end** **Example:** `Device(config-mdt-subs)# end`	Exits telemetry-subscription configuration mode and returns to privileged EXEC mode.

Receiving a Response Code

When a subscription is successfully created, the device responds with a subscription result of notif-bis:ok and a subscription ID. The following is a sample response RPC message for a dynamic subscription:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
<subscription-result xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
xmlns:notif-bis="urn:ietf:params:xml:ns:yang:ietf-event-notifications">notif-bis:
ok</subscription-result>
<subscription-id
xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications">2147484201</subscription-id>
</rpc-reply>
```

Receiving Subscription Push Updates for NETCONF Dial-In

Subscription updates pushed from the device are in the form of an XML RPC and are sent over the same NETCONF session on which these are created. The subscribed information element or tree is returned within the *datastore-contents-xml* tag. The following is a sample RPC message that provides the subscribed information:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-05-09T21:34:51.74Z</eventTime>
    <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <subscription-id>2147483650</subscription-id>
        <datastore-contents-xml>
            <cpu-usage
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-process-cpu-oper"><cpu-utilization>
            <five-minutes>5</five-minutes></cpu-utilization></cpu-usage>
        </datastore-contents-xml>
    </push-update>
</notification>
```

If the information element to which a subscription is made is empty, or if it is dynamic, for example, a named access list, and does not exist, the periodic update will be empty and will have a self-closing *datastore-contents-xml* tag. The following is a sample RPC message in which the periodic update is empty:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-05-09T21:34:09.74Z</eventTime>
    <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <subscription-id>2147483649</subscription-id>
        <datastore-contents-xml />
    </push-update>
</notification>
```

Retrieving Subscription Details

You can retrieve the list of current subscriptions by sending a <get> RPC to the Cisco-IOS-XE-mdt-oper model. You can also use the **show telemetry ietf subscription** command to display the list of current subscriptions.

The following is a sample <get> RPC message:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
        <mdt-subscriptions/>
      </mdt-oper-data>
    </filter>
  </get>
</rpc>
```

The following is a sample RPC reply:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-subscriptions>
        <subscription-id>2147485164</subscription-id>
        <base>
          <stream>yang-push</stream>
          <encoding>encode-xml</encoding>
          <period>100</period>
          <xpath>/ios:native/router/ios-rip:rip/ios-rip:version</xpath>
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <updates-in>0</updates-in>
        <updates-dampened>0</updates-dampened>
        <updates-dropped>0</updates-dropped>
      </mdt-subscriptions>
    </mdt-oper-data>
  </data>
</rpc-reply>
```

The following is sample output from the **show telemetry ietf subscription dynamic brief** command:

```
Device# show telemetry ietf subscription dynamic brief

Telemetry subscription brief

  ID               Type        State       Filter type
  ------------------------------------------------------
  2147483667       Dynamic     Valid       xpath
  2147483668       Dynamic     Valid       xpath
  2147483669       Dynamic     Valid       xpath
```

The following is sample output from the **show telemetry ietf subscription** *subscription-ID* **detail** command:

```
Device# show telemetry ietf subscription 2147483667 detail

Telemetry subscription detail:

  Subscription ID: 2147483667
  State: Valid
  Stream: yang-push
  Encoding: encode-xml
  Filter:
    Filter type: xpath
    XPath: /mdt-oper:mdt-oper-data/mdt-subscriptions
  Update policy:
    Update Trigger: periodic
    Period: 1000
  Notes:
```

The following is sample output from the **show telemetry ietf subscription all detail** command:

```
Device# show telemetry ietf subscription all detail

Telemetry subscription detail:

  Subscription ID: 101
  Type: Configured
  State: Valid
  Stream: yang-push
  Encoding: encode-kvgpb
  Filter:
    Filter type: xpath
    XPath: /iosxe-oper:ios-oper-db/hwidb-table
  Update policy:
    Update Trigger: on-change
    Synch on start: Yes
    Dampening period: 0
  Notes:
```

The following sample RPC shows how to retrieve subscription details uisng RESTCONF:

```
Subscription details can also be retrieved through a RESTCONF GET request to the
Cisco-IOS-XE-mdt-oper database:
URI:
https://10.85.116.28:443/restconf/data/Cisco-IOS-XE-mdt-oper: mdt-oper-data/mdt-subscriptions
Headers:
application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
Content-Type:
application/yang-data+json
Returned output:
{
  "Cisco-IOS-XE-mdt-oper:mdt-subscriptions": [
    {
      "subscription-id": 101,
      "base": {
        "stream": "yang-push",
        "encoding": "encode-kvgpb",
        "source-vrf": "",
        "no-synch-on-start-v2": false,
```

```
        "xpath": "/iosxe-oper:ios-oper-db/hwidb-table"
      },
      "type": "sub-type-static",
      "state": "sub-state-valid",
      "comments": "",
      "updates-in": "0",
      "updates-dampened": "0",
      "updates-dropped": "0",
      "mdt-receivers": [
        {
          "address": "5.28.35.35",
          "port": 57555,
          "protocol": "grpc-tcp",
          "state": "rcvr-state-connecting",
          "comments": "Connection retries in progress",
          "profile": ""
        }
      ]
    }
  ]
}
```


Configuring Named Protocol Receiver Using the CLI

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **telemetry receiver protocol** *receiver-name*
4. **protocol** {**cloud-native** | **cntp-tcp** | **cntp-tls profile** *profile-name* | **grpc-tcp** | **grpc-tls profile** *profile-name* | **native** | **tls-native profile** *profile-name*}
5. **host** {**ip** *ip-address* | **name** *hostname*} *receiver-port*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**telemetry receiver protocol** *receiver-name* **Example:** `Device(config)# telemetry receiver protocol receiver1`	Configures a named protocol receiver, and enters telemetry protocol-receiver configuration mode.

	Command or Action	Purpose
Step 4	**protocol** {**cloud-native** | **cntp-tcp** | **cntp-tls profile** *profile-name* | **grpc-tcp** | **grpc-tls profile** *profile-name* | **native** | **tls-native profile** *profile-name*} **Example:** `Device(config-mdt-protocol-receiver)# protocol grpc-tcp`	Configures a protocol for the named protocol receiver connection.
Step 5	**host** {**ip** *ip-address* | **name** *hostname*} *receiver-port* **Example:** `Device(config-mdt-protocol-receiver)# host name rcvr.test.com 45000`	Configures the name protocol receiver hostname.
Step 6	**end** **Example:** `Device(config-mdt-protocol-receiver)# end`	Exits telemetry protocol-receiver configuration mode and returns to privileged EXEC mode.

Subscription Configuration Using Named Receivers Using CLI

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **telemetry ietf subscription** *id*
4. **receiver-type protocol**
5. **receiver name** *name*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**telemetry ietf subscription** *id* **Example:** `Device(config)# telemetry ietf subscription 101`	Creates a telemetry subscription and enters telemetry-subscription mode.

	Command or Action	Purpose
Step 4	**receiver-type protocol** **Example:** `Device(config-mdt-subs)# receiver-type protocol`	Configures a protocol-type receiver.
Step 5	**receiver name** *name* **Example:** `Device(config-mdt-subs)# receiver name receiver1`	Configures a name for the receiver for notifications.
Step 6	**end** **Example:** `Device(config-mdt-subs)# end`	Exits telemetry telemetry-subscription mode and returns to privileged EXEC mode.

Additional References for Model-Driven Telemetry

Standards and RFCs

Standard/RFC	Title
Custom Subscription to Event Notifications draft-ietf-netconf-subscribed-notifications-03	https://tools.ietf.org/id/draft-ietf-netconf-subscribed-notifications-03.txt
NETCONF Support for Event Notifications	draft-ietf-netconf-netconf-event-notifications-01
RFC 5277	NETCONF Event Notifications
RFC 6241	Network Configuration Protocol (NETCONF)
RFC 7950	The YANG 1.1 Data Modeling Language
RFC 8040	RESTCONF Protocol
RFC 8641	Subscription to YANG Notifications for Datastore Update
Subscribing to Event Notifications	draft-ietf-netconf-rfc5277bis-01
Subscribing to YANG Datastore Push Updates	draft-ietf-netconf-yang-push-04
Subscribing to YANG datastore push updates draft-ietf-netconf-yang-push-07	https://tools.ietf.org/id/draft-ietf-netconf-yang-push-07.txt

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for Model-Driven Telemetry

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 42: Feature Information for Model-Driven Telemetry

Feature Name	Release	Feature Information
Model-Driven Telemetry NETCONF Dial-In	Cisco IOS XE Everest 16.6.1	Model-driven telemetry allows network devices to continuously stream real time configuration and operating state information to subscribers. • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches
	Cisco IOS XE Everest 16.6.2	• Cisco Catalyst 9400 Series Switches
	Cisco IOS XE Fuji 16.7.1	• Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers (ASR1001-HX, ASR1001-X, ASR1002-HX, ASR1002-X)
	Cisco IOS XE Fuji 16.8.1	• Cisco 1000 Series Integrated Services Routers • Cisco ASR 1000 RP2 and RP3 Series Aggregation Services Routers
	Cisco IOS XE Fuji 16.8.1a	• Cisco Catalyst 9500-High Performance Series Switches
	Cisco IOS XE Fuji 16.9.1	• Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Router • Cisco cBR-8 Converged Broadband Router • Cisco Network Convergence System 4200 Series

Feature Name	Release	Feature Information
	Cisco IOS XE Gibraltar 16.9.2	• Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs
	Cisco IOS XE Gibraltar 16.10.1	• Cisco Cloud Services Router 1000v • Cisco Network Convergence System 520 Series
	Cisco IOS XE Gibraltar 16.11.1	• Cisco Catalyst 9600 Series Switches
	Cisco IOS XE Cupertino 17.9.1	• Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller

Feature Name	Release	Feature Information
Model-Driven Telemetry gNMI Dial-In	Cisco IOS XE Gibraltar 16.12.1	Telemetry updates that are sent to the initiator/subscriber are called Dial-in. This feature was implemented on the following platforms: • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco cBR-8 Converged Broadband Router
	Cisco IOS XE Amsterdam 17.1.1	• Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Amsterdam 17.2.1	Cisco ASR 1000 Series Aggregation Services Routers
	Cisco IOS XE Cupertino 17.9.1	• Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller

Feature Name	Release	Feature Information
Model-Driven Telemetry gRPC Dial-Out	Cisco IOS XE Gibraltar 16.10.1	Configured subscriptions cause the publisher to initiate connections to receivers, and these connections are considered as dial-out. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Gibraltar 16.11.1	• Cisco Catalyst 9600 Series Switches
	Cisco IOS XE Cupertino 17.9.1	

Feature Name	Release	Feature Information
		• Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller

Feature Name	Release	Feature Information
Model-Driven Telemetry: Kill Subscription	Cisco IOS XE Gibraltar 16.11.1	To delete dynamic subscriptions, you can use the CLI and the kill-subscription RPC. • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router (RSP2) • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Cupertino 17.9.1	• Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller

Feature Name	Release	Feature Information
TLS for gRPC Dial-Out	Cisco IOS XE Amsterdam 17.1.1	Transport-Layer Security is supported for gRPC dial-out. This feature is supported on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Cupertino 17.9.1	

Feature Name	Release	Feature Information
		• Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller

Feature Name	Release	Feature Information
gNMI sync_response Message	Cisco IOS XE 17.14.1	

Feature Name	Release	Feature Information
		The sync_response is a boolean field that is part of the SubscribeResponse response message. The sync_response message is sent after the first update message. This feature was implemented on the following platforms: • Cisco ASR 1000 Series Aggregation Services Routers • Cisco Catalyst IE3200 Rugged Series • Cisco Catalyst IE3300 Rugged Series • Cisco Catalyst IE3400 Rugged Series • Cisco Catalyst IR1101 Rugged Series • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller • Cisco Cloud Services Router 1000V Series

Feature Name	Release	Feature Information
		• Cisco Embedded Services 3300 Series

Feature Name	Release	Feature Information
gNMI: Stream Subscriptions with On-Change Mode	Cisco IOS XE 17.14.1	gNMI telemetry supports on-change subscriptions on the same set of models as other telemetry protocols. This feature was implemented on the following platforms: • Cisco ASR 1000 Series Aggregation Services Routers • Cisco Catalyst IE3200 Rugged Series • Cisco Catalyst IE3300 Rugged Series • Cisco Catalyst IE3400 Rugged Series • Cisco Catalyst IR1101 Rugged Series • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller • Cisco Cloud Services Router 1000V Series • Cisco Embedded Services 3300 Series

Feature Name	Release	Feature Information
FQDN Support for gRPC Subscriptions	Cisco IOS XE Bengaluru 17.6.1	With the introduction of the FQDN Support for gRPC Subscriptions feature, along with IP addresses, FQDN can also be used for gRPC subscriptions. • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature Name	Release	Feature Information
	Cisco IOS XE Cupertino 17.9.1	• Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller
Leaf-Level Filtering	Cisco IOS XE Cupertino 17.7.1	The Leaf-Level Filtering for Telemetry feature allows filtering below the gatherpoint level for the optimized code paths. • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches

Feature Name	Release	Feature Information
Mutual Authentication for gRPC Telemetry	Cisco IOS XE Cupertino 17.9.1	

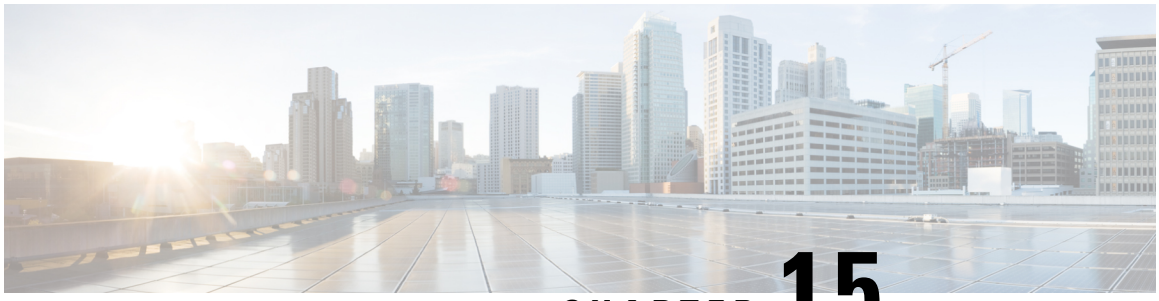
Feature Name	Release	Feature Information
		A new gRPC TLS profile that contains a pair of trustpoints was added to the telemetry configuration, so that a client ID certificate can be specified for mutual authentication. This new profile can be used instead of the trustpoint containing the server CA certificate when configuring the receiver profile. The trustpoint containing the server CA certificate is now configured as part of the gRPC TLS profile. This feature is supported on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller • Cisco Catalyst 9800-40 Series

Feature Name	Release	Feature Information
		Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
On-Change Notification	Cisco IOS XE Dublin 17.10.1	The TDL URI string supports on-change notifications. This feature was implemented on Cisco ASR 1000 Series Aggregation Services Routers

Feature Name	Release	Feature Information
Pubd Restartability	Cisco IOS XE Cupertino 17.9.1	

Feature Name	Release	Feature Information
		The pubd process is made restartable from this release onwards. This feature was introduced on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-CL Wireless Controller • Cisco Catalyst 9800-L Wireless Controller • Cisco Catalyst 9800-40 Series Wireless Controller • Cisco Catalyst 9800-80 Series Wireless Controller • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series

Feature Name	Release	Feature Information
		• Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
Subscription Dampening Period for On-Change Telemetry	Cisco IOS XE Dublin 17.11.1	This feature introduces a dampening period for on-change subscriptions. All record updates are sent at the end of the dampening period, even when there are multiple updates of the same record during that period. This feature was introduced on all platforms that support Telemetry.



CHAPTER 15

In-Service Model Update

This module describes how to update the YANG data models on a device through an In-Service Model Update.

- Restrictions for In-Service Model Update, on page 383
- Information About In-Service Model Update, on page 383
- How to Manage In-Service Model Update, on page 386
- Configuration Examples for In-Service Model Updates, on page 387
- Feature Information for In-Service Model Update, on page 391

Restrictions for In-Service Model Update

- High availability or In-Service Software Upgrade (ISSU) is not supported. After a switchover, users must install the Software Maintenance Update (SMU) on standby device.

Information About In-Service Model Update

Overview of In-Service Model Updates

In-Service Model Update adds new data models or extend functionality to existing data models. The In-Service Model Update provides YANG model enhancements outside of a release cycle. The update package is a superset of all existing models; it includes all existing models as well as updated YANG models.

The data model infrastructure implements the YANG model-defined management interfaces for Cisco IOS XE devices. The data model infrastructure exposes the NETCONF interface northbound from Cisco IOS XE devices. The supported data models include industry standard models such as IETF, and Cisco IOS XE device-specific models.

The functionality provided by the In-Service Model Update is integrated into the subsequent Cisco IOS XE software maintenance release. Data model update packages can be downloaded from the Cisco Download Software Center.

Compatibility of In-Service Model Update Packages

An update package is built on a per release basis and is specific to a platform. This means that an update package for Cisco ASR 1000 Series Aggregation Services Routers cannot be installed on Cisco CSR 1000V

Series Cloud Services Routers. Similarly, an update package built for Cisco IOS XE Fuji 16.7.1 cannot be applied on a device that runs the Cisco IOS XE Everest 16.5.2 version.

All contents of an update package will be part of future mainline or maintenance release images. The image and platform versions are checked by the In-Service Model Update commands during the package add and activate. If an image or platform mismatch occurs, the package install fails.

Update Package Naming Conventions

In-Service Model Updates are packaged as a .bin files. This file includes all updates for a specific release and platform and the Readme file. These files have a release date and are updated periodically with additional model updates.

The naming convention of the data model update package follows the format—platform type-license level.release version.DDTS ID-file. The following is an example of a data model update file:

- isr4300-universalk9.16.05.01.CSCxxxxxx.dmp.bin
- asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxx.SSA.dmp.bin

The readme file provides the following information:

- Console and error messages during data model activation or deactivation
- Data model installation impact
- Side effects and possible workarounds
- Package(s) that the In-Service Model Update impacts
- Restart type

Installing the Update Package

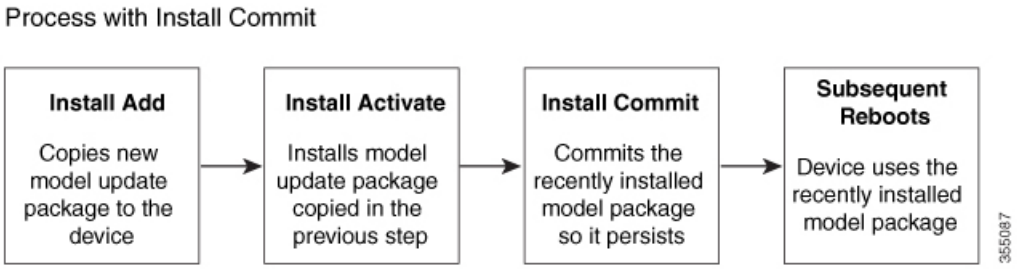
You can install the In-Service Model Update package on a device by using the **install add**, **install activate**, and **install commit** commands in privileged EXEC mode.

The **install add** command copies the update package from a remote location to the device. You can also use other methods to copy the package; however, you must still enable the **install add** command for the installation to work. For the **install activate** command to work, the package must be available in the device bootflash. Enable the **install commit** command to make updates persistent over reloads.

Installing an update replaces any previously installed data models. At any time, only one update is installed on the device. A data model package includes all updated YANG models and all existing YANG models previously installed on the device.

The following flow chart explains how the model update package works:

Figure 11: Committing a Model Update Package



If NETCONG-YANG is enabled during package activation, NETCONF processes are restarted. All active NETCONF sessions are killed during package activation. Failure during a package verification terminates the activation process.

Deactivating the Update Package

You can deactivate an update package by using the **install deactivate** command. Enable the **install commit** command to make changes persistent.

Table 43: Deactivating a Model Update Package

Action	Command to Use
To remove a package.	Use the **install remove** command. **Note** Deactivate a package before removing it.
To deactivate a package	Use the **install deactivate** command, followed by the **install commit** command. **Note** The **install commit** command must be used to ensure that the deactivation of the model package is persistent across reloads. Subsequent attempts at removal of the package will fail, if the deactivation is not committed.

When you deactivate an update, if more than one model update package is installed, the most recently committed model update package becomes the model package used by the device. If there are no other previously committed model packages, then the base version of data models included with the standard image is used.

Rollback of the Update Package

Rollback provides a mechanism to move a device back to the state in which it was operating prior to an update. After a rollback, NETCONF-YANG processes are restarted before changes are visible.

You can roll back an update to the base version, the last committed version, or a known commit ID by using the **install rollback** command.

How to Manage In-Service Model Update

Managing the Update Package

SUMMARY STEPS

1. **enable**
2. **install add file tftp:** *filename*
3. **install activate file bootflash:** *filename*
4. **install commit**
5. **install deactivate file bootflash:** *filename*
6. **install commit**
7. **install rollback to** {**base** | **committed** | **id** *commit-ID*}
8. **install remove** {**file bootflash:** *filename* | **inactive**}
9. **show install summary**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

Step 2	**install add file tftp:** *filename*
	Example:
	`Device# install add file tftp://172.16.0.1//tftpboot/folder1/ isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`
	`Device# install add file tftp://172.16.0.1//tftpboot/folder1/ asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`
		Copies the model update package from a remote location (via FTP, TFTP) to the device, and performs a compatibility check for the platform and image versions.
		• You can use other methods to copy the update package from the remote location to the device, however; you still have to execute the **install add** command before the package is activated.

Step 3	**install activate file bootflash:** *filename*
	Example:
	`Device# install activate file bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`
	`Device# install activate file bootflash: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`
		Validates whether the update package is added through the **install add** command, and restarts the NETCONF processes.
		• Perform the **install add** operation prior to activating an update package.

Step 4	**install commit** **Example:** `Device# install commit`	Makes the changes persistent over reload. • NETCONF processes are not restarted.

	Command or Action	Purpose

Step 5	**install deactivate file bootflash:** *filename*
	Example:
	`Device# install deactivate file bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`
	`Device# install deactivate file bootflash: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`
		Deactivates the specified update package, and restarts the NETCONF processes.

Step 6	**install commit** **Example:** `Device# install commit`	Makes the changes persistent over reload. • NETCONF processes are not restarted.
Step 7	**install rollback to** {**base** | **committed** | **id** *commit-ID*} **Example:** `Device# install rollback to base`	Rollbacks the update to the base version, the last committed version, or a known commit ID, and restarts NETCONF processes. • Valid values for the *commit-id* argument are from 1 to 4294967295. • Older versions of data models updates are available for use.

Step 8	**install remove** {**file bootflash:** *filename* | **inactive**}
	Example:
	`Device# install remove file bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`
	`Device# install remove file bootflash: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin`
		Removes the specified update package from the bootflash.
		• A package must be deactivated before it is removed.

Step 9	**show install summary** **Example:** `Device# show install summary`	Displays information about the active package. • The output of this command varies according to the **install** commands that are configured.

Configuration Examples for In-Service Model Updates

Example: Managing an Update Package

The sample image used in the following examples are a Cisco 4000 Series Integrated Services Router image.

The following example shows how to add a model update package file:

```
Device# install add file tftp://172.16.0.1//tftpboot/folder1/
isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin

install_add: START Sun Feb 26 05:57:04 UTC 2017
Downloading file
```

```
tftp://172.16.0.1//tftpboot/folder1/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Finished downloading file
tftp://172.16.0.1//tftpboot/folder1/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
to bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
SUCCESS: install_add /bootflash/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Sun Feb 26 05:57:22 UTC 2017
Device#
```

The sample image used in the following examples are a Cisco ASR1000 Series Aggregated Services Router image.

The following example shows how to add a model update package file:

```
Device# install add file tftp://172.16.0.1//tftpboot/folder1/
asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin

install_add: START Sun Feb 26 05:57:04 UTC 2017
Downloading file
tftp://172.16.0.1//tftpboot/folder1/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Finished downloading file
tftp://172.16.0.1//tftpboot/folder1/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
to bootflash: asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
SUCCESS: install_add /bootflash/asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.SSA.dmp.bin
Sun Feb 26 05:57:22 UTC 2017
Device#
```

The following is sample output from the **show install summary** command after adding an update package file to the device:

```
Device# show install summary

Active Packages:
No packages
Inactive Packages:
bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Committed Packages:
No packages
Uncommitted Packages:
No packages
Device#
```

The following example shows how to activate an added update package file:

```
Device# install activate file bootflash:
isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin

install_activate: START Sun Feb 26 05:58:41 UTC 2017
DMP package.
Netconf processes stopped
SUCCESS: install_activate /bootflash/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Sun Feb 26 05:58:58 UTC 2017*Feb 26 05:58:47.655: %DMI-4-CONTROL_SOCKET_CLOSED:
SIP0: nesd: Confd control socket closed Lost connection to ConfD (45): EOF on socket to
ConfD.
*Feb 26 05:58:47.661: %DMI-4-SUB_READ_FAIL: SIP0: vtyserverutild:
Confd subscription socket read failed Lost connection to ConfD (45):
EOF on socket to ConfD.
*Feb 26 05:58:47.667: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: syncfd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 05:59:43.269: %DMI-5-SYNC_START: SIP0: syncfd:
External change to running configuration detected.
The running configuration will be synchronized to the NETCONF running data store.
*Feb 26 05:59:44.624: %DMI-5-SYNC_COMPLETE: SIP0: syncfd:
The running configuration has been synchronized to the NETCONF running data store.
```

```
Device#
```

The following sample output from the **show install summary** command displays the status of the model package as active and uncommitted:

```
Device# show install summary

Active Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Inactive Packages:
No packages
Committed Packages:
No packages
Uncommitted Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Device#
```

The following example shows how to execute the **install commit** command:

```
Device# install commit

install_commit: START Sun Feb 26 06:46:48 UTC 2017
SUCCESS: install_commit Sun Feb 26 06:46:52 UTC 2017
Device#
```

The following sample output from the **show install summary** command displays that the update package is now committed, and that it will be persistent across reloads:

```
Device# show install summary

Active Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Inactive Packages:
No packages
Committed Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Uncommitted Packages:
No packages
Device#
```

The following example shows how to rollback an update package to the base package:

```
Device# install rollback to base

install_rollback: START Sun Feb 26 06:50:29 UTC 2017
7 install_rollback: Restarting impacted processes to take effect
7 install_rollback: restarting confd
*Feb 26 06:50:34.957: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: syncfd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 06:50:34.962: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: nesd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 06:50:34.963: %DMI-4-SUB_READ_FAIL: SIP0: vtyserverutild:
Confd subscription socket read failed Lost connection to ConfD (45):
EOF on socket to ConfD.Netconf processes stopped
7 install_rollback: DMP activate complete
SUCCESS: install_rollback Sun Feb 26 06:50:41 UTC 2017
*Feb 26 06:51:28.901: %DMI-5-SYNC_START: SIP0: syncfd:
External change to running configuration detected.
The running configuration will be synchronized to the NETCONF running data store.
*Feb 26 06:51:30.339: %DMI-5-SYNC_COMPLETE: SIP0: syncfd:
The running configuration has been synchronized to the NETCONF running data store.
```

```
Device#
```

The following is sample output from the **show install package** command:

```
Device# show install package bootflash:
isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin

Name: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Version: 16.5.1.0.199.1484082952..Everest
Platform: ISR4300
Package Type: dmp
Defect ID: CSCxxxxxxx
Package State: Added
Supersedes List: {}
Smu ID: 1
Device#
```

The following sample NETCONF hello message verifies the new data model package version:

```
Getting Capabilities: (admin @ 172.16.0.1:830)
PROTOCOL netconf
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
<capability>http://tail-f.com/ns/netconf/extensions</capability>
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=
explicit&amp;also-supported=report-all-tagged</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?
revision=2011-06-01&amp;module=ietf-netconf-with-defaults</capability>
<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-aaa?module=
Cisco-IOS-XE-aaa&amp;revision=2017-02-07</capability>
<<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-native?module=
Cisco-IOS-XE-native&amp;revision=2017-01-07&amp;features=virtual-
template,punt-num,multilink,eth-evc,esmc,efp,dot1x</capability>
Device#
```

The following is sample output from the **show install log** command:

```
Device# show install log

[0|install_op_boot]: START Fri Feb 24 19:20:19 Universal 2017
[0|install_op_boot]: END SUCCESS Fri Feb 24 19:20:23 Universal 2017
[3|install_add]: START Sun Feb 26 05:55:31 UTC 2017
[3|install_add( FATAL)]: File path (scp) is not yet supported for this command
[4|install_add]: START Sun Feb 26 05:57:04 UTC 2017
[4|install_add]: END SUCCESS /bootflash/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Sun Feb 26 05:57:22 UTC 2017
[5|install_activate]: START Sun Feb 26 05:58:41 UTC 2017
Device#
```

The sample image used in the following examples are a Cisco Catalyst 3000 Series Switch image.

The following example shows how to add a model update package file:

```
Device# install add file tftp://172.16.0.1//tftpboot/folder1/
cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin

install_add: START Sat Jul 29 05:57:04 UTC 2017
Downloading file tftp://172.16.0.1//tftpboot/folder1/
cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Finished downloading file tftp://172.16.0.1//tftpboot/folder1/
cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.SPA.smu.bin
to bootflash:cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
SUCCESS: install_add /bootflash/cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Sat Jul 29 05:57:22 UTC 2017
Device#
```

The following sample output from the **show install summary** command displays that the update package is now committed, and that it will be persistent across reloads:

```
Device# show install summary

Active Packages:
bootflash:cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Inactive Packages:
No packages
Committed Packages:
bootflash:cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Uncommitted Packages:
No packages
Device#
```

Feature Information for In-Service Model Update

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

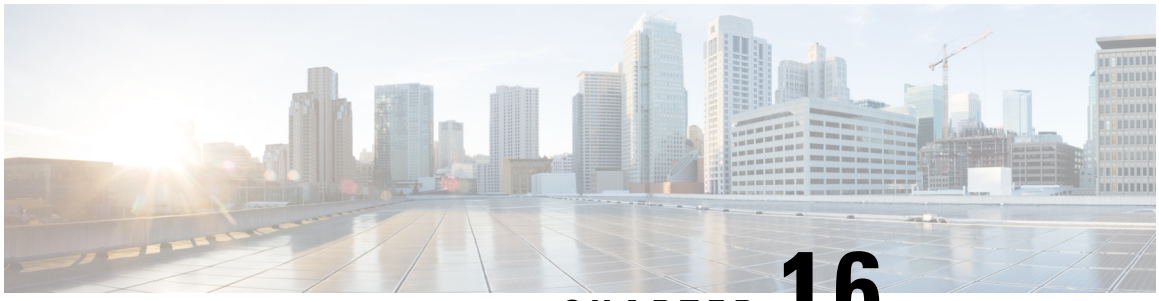
Table 44: Feature Information for In-Service Model Update

Feature Name	Release	Feature Information
In-Service Model Update	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	This module describes how to update YANG data models through In-Service Model Update. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco Cloud Services Router 1000v • Cisco Integrated Services Virtual Routers (ISRv) The following commands were introduced or updated: **install (Programmability)**, **show install (Programmability)**.
	Cisco IOS XE Everest 16.6.1	In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches
	Cisco IOS XE Fuji 16.7.x	In Cisco IOS XE Fuji 16.7.x, this feature was implemented on the following platform: • Cisco 1000 Series Aggregated Services Routers
	Cisco IOS XE Fuji 16.8.1a	In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches

PART IV

Application Hosting

- Application Hosting, on page 395
- ThousandEyes Enterprise Agent, on page 441



CHAPTER 16

Application Hosting

A hosted application is a software as a service (SaaS) solution, and it can be run remotely using commands. Application hosting gives administrators a platform for leveraging their own tools and utilities.

Note Application hosting supports only Docker applications.

This module describes the Application Hosting feature and how to enable it.

- Prerequisites for Application Hosting, on page 395
- Restrictions for Application Hosting, on page 395
- Information About Application Hosting, on page 396
- How to Configure Application Hosting, on page 408
- Verifying the Application-Hosting Configuration, on page 426
- Verifying the Application-Hosting Configuration, on page 429
- Configuration Examples for Application Hosting, on page 432
- Additional References, on page 436
- Feature Information for Application Hosting, on page 437

Prerequisites for Application Hosting

- Applications hosted by Catalyst 9000 Series Switches must be configured in the underlay Switch Virtual Interface (SVI). This applies to Cisco Software-Defined Access deployments as well.

Restrictions for Application Hosting

- Application hosting is not virtual routing and forwarding aware (VRF-aware).
- Application hosting is not supported on Cisco Catalyst 9200, 9200CX, and 9200L platforms.
- In releases prior to Cisco IOS XE Amsterdam 17.3.3, application hosting requires dedicated storage allocations, and is disabled on the bootflash.

  In Cisco IOS XE Amsterdam 17.3.3 and later releases, application hosting is enabled on the bootflash, however, only Cisco-signed applications are hosted.

- The front-panel Universal Serial Bus (USB) stick is not supported.

  Cisco Catalyst 9300 Series Switches support only back-panel Cisco-certified USB.

- Cisco Catalyst 9500-High Performance Series Switches and Cisco Catalyst 9600 Series Switches do not support front-panel USB for application hosting.
- Cisco Catalyst 9500 and 9500-High Performance Series Switches and Cisco Catalyst 9600 Series Switches do not support AppGigabitEthernet interfaces.
- Cisco Catalyst 9410R Switches do not support application-hosting in release prior to Cisco IOS XE Bengaluru 17.5.1.

  Configure the **enable** command on the AppGigabitEthernet interfaces to enable application hosting on Cisco Catalyst 9410R Switches. This restriction is applicable only to Cisco Catalyst 9400 Series Supervisor 1, 1XL, and 1XL-Y 25G Modules (C9400-SUP-1, C9400-SUP-1XL, and C9400-SUP-1XL-Y).

  You do not need to configure the **enable** command on Cisco Catalyst 9400 Series Supervisor 2 and 2XL Modules (C9400X-SUP-2 and C9400X-SUP-2XL).

Information About Application Hosting

This section provides information about Application Hosting.

Need for Application Hosting

The move to virtual environments has given rise to the need to build applications that are reusable, portable, and scalable. Application hosting gives administrators a platform for leveraging their own tools and utilities. An application, hosted on a network device, can serve a variety of purposes. This ranges from automation, configuration management monitoring, and integration with existing tool chains.

Note In this document, *container* refers to Docker applications.

Cisco IOx Overview

Cisco IOx (IOs + linuX) is an end-to-end application framework that provides application-hosting capabilities for different application types on Cisco network platforms. The Cisco Guest Shell, a special container deployment, is one such application, that is useful in system deployment.

Cisco IOx facilitates the life cycle management of applications and data exchange by providing a set of services that helps developers to package prebuilt applications, and host them on a target device. IOx life cycle management includes distribution, deployment, hosting, starting, stopping (management), and monitoring of applications and data. IOx services also include application distribution and management tools that help users discover and deploy applications to the IOx framework.

Cisco IOx application hosting provides the following features:

- Hides network heterogeneity.

- Cisco IOx application programming interfaces (APIs) remotely manage the life cycle of applications hosted on a device.
- Centralized application life cycle management.
- Cloud-based developer experience.

Application Hosting Overview

The Cisco application-hosting framework is an IOx Python process that manages virtualized and container applications that run on devices.

Application hosting provides the following services:

- Launches designated applications in containers.
- Checks available resources (memory, CPU, and storage), and allocates and manages them.
- Provides support for console logging.
- Provides access to services through REST APIs.
- Provides a CLI endpoint.
- Provides an application-hosting infrastructure referred to as Cisco Application Framework (CAF).
- Helps setup platform-specific networking (packet-path) through management interfaces.

  Data ports are supported on platforms that have AppGigabitEthernet port functionality.

The application-hosting container that is referred to as the virtualization environment is provided to run a guest application on the host operating system. The Cisco IOS-XE virtualization services provide manageability and networking models for running a guest application. The virtualization infrastructure allows an administrator to define a logical interface that specifies the connectivity between the host and the guest. Cisco IOx maps the logical interface into a Virtual Network Interface Card (vNIC) that the guest application uses.

Applications that are to be deployed in the containers are packaged as TAR files. The configuration that is specific to these applications is also packaged as part of the TAR files.

The management interface on the device connects the application-hosting network to the Cisco IOS management interface. The Layer 3 interface of the guest application receives the Layer 2-bridged traffic from the Cisco IOS management interface. The management interface connects to the container interface through the management bridge. The IP address of the application must be on the same subnet as the management interface IP address.

Note On all Cisco Catalyst stack and stackwise virtual models (all software versions), Guest Shell and the AppGigabitEthernet interface operate only on the active switch in the stack. Therefore, the AppGigabitEthernet interface configuration must be applied to the AppGigabitEthernet interface on all the switches in the stack. If the configuration is not applied to all the switches, the AppGigabitEthernet interface on the switch will not work after a switchover.

Cisco Catalyst 9000 Series Switches support multiple applications when hosted on the SSD. The applications must meet the following criteria:

- Cisco-signed
- Meet the switching infrastructure requirements:
  - Network configuration on AppGigabitEthernet ports does not create a conflict between the applications.
  - Enough resources are available to run the applications.

Multiple applications cannot be deployed if an application consumes all the available App-hosting resources. For example, if one application consumes all the compute and run time resources, other applications are prevented from getting installed on the device.

Application Hosting on Front-Panel Trunk and VLAN Ports

Front-panel VLAN and trunk ports are supported for application hosting. Layer 2 traffic is delivered through these ports to software components that run outside of the Cisco IOS daemon.

For application hosting, you can configure the front-panel port as either a trunk interface or a VLAN-specific interface. When using as a trunk interface, the front-panel port is extended to work as a Layer 2 trunk port, and all the traffic received by the port is available to the application. When using the port as a VLAN interface, the application is connected to a specific VLAN network.

Note When using a back-panel USB or an M2 SATA drive for application hosting, the storage medium should be formatted as an *ext4* file system.

Application Hosting on Cisco Catalyst 9300 Series Switches

This section describes application-hosting on Cisco Catalyst 9300 Series Switches.

For application hosting, Cisco Catalyst 9300 Series Switches support the management interface and front-panel ports.

The USB 3.0 SSD is enabled on Cisco Catalyst 9300 Series Switches. The USB 3.0 SSD provides an extra 120 GB storage for application hosting. For more information, see the "Configuring USB 3.0 SSD" chapter in the *Interfaces and Hardware Configuration Guide*.

The following two types of networking applications are supported:

- Control plane: Applications that access the management interface.
- Data plane: Applications that access the front-panel ports.

Front-Panel App Hosting on Cisco Catalyst 9300X Series Switches

Front-panel application hosting is enabled on Cisco Catalyst 9300X Series Switches in Cisco IOS XE Bengaluru 17.6.1.

Applications can use dedicated front-panel ports for hosting. Use the **app-vnic AppGigabitEthernet port** command to specify the port to be used for application hosting. Both the front-panel ports can be attached to the same Layer 2 application.

These switches support application hosting in both access mode and trunk mode. Application hosting can be enabled on both modes simultaneously.

Note Any configuration done under the **app-vnic** command can be rejected during activation.

Table 45: Sample Configuration Scenarios for App Hosting in Access and Trunk Modes

Scenario	Supported/Unsupported
Single application with two front-panel ports in access mode.	Supported. No overlapping VLANs.
Single application with two front-panel ports in trunk mode.	Supported. No overlapping VLANs.
Single application with two front-panel ports in trunk and access modes.	Supported. No overlapping VLANs.
Single application with two front-panel ports in trunk mode with the default app-gateway configured.	Supported. The same application with two interfaces is configured in different subnets; but the default gateway is connected to one VLAN, which has external connectivity.
Single application with two front-panel ports in trunk and access modes with an overlapping VLAN.	Not a valid configuration. VLAN overlapping with both ports.
Single application in access mode and two front-panel ports configured on the same VLAN.	Not a valid configuration.
Single application in trunk mode and two front-panel ports configured on an overlapping VLAN range.	Not a valid configuration. The traffic is not isolated, and the VLAN range is overlapping.
Single application in trunk mode and two front-panel ports configured on an overlapping VLAN range.	Not a valid configuration. This configuration will be rejected during activation. Both the front-panel ports are in trunk mode, so any VLAN can be used. However, the same VLAN is configured for both the ports, and as a result, the VLAN overlaps with both the ports. **Note** The same scenario applies for access mode.

Scenario	Supported/Unsupported
Single application in trunk and access modes, and front-panel ports with an overlapping VLAN.	Not a valid configuration. The same VLAN is configured in trunk mode and access mode. Because of the configuration, the VLAN overlaps with both ports.
Multiple application in trunk mode.	Not a valid configuration. The traffic is not isolated.
Two applications, one in trunk mode and the other in access mode.	Not a valid configuration. Overlapping VLAN.

High Availability on Cisco Catalyst 9300X Series Switches

With mixed mode stacking available on Cisco Catalyst 9300X Series Switches, the active and standby devices use the 1+1 redundancy for application hosting. Mixed mode support is when different model variants and different network modules are used in a stack.

When Cisco Catalyst 9300X Series Switches and Cisco Catalyst 9300 Series Switches are stacked, one of the two front-panel ports on Cisco Catalyst 9300X Series Switches are dynamically disabled. Only the AppGigabitEthernet 1/0/1 interface is displayed as enabled.

This section describes some of the high availability scenarios:

Stack Mode	Apps	Ports Used	Behavior
Cisco Catalyst 9300X Series Switch active + Cisco Catalyst 9300X Series Switch standby	2	1	Supported
Cisco Catalyst 9300X Series Switch active + Cisco Catalyst 9300X Series Switch standby	1	1	Supported
Cisco Catalyst 9300X Series Switch active + Cisco Catalyst 9300 Series Switch standby	1	1 Only if port 1 is used.	Supported. This configuration is supported, if port 1 is configured by using the **app-vnic Appgigabitethernet port 1 trunk** or the **app-vnic AppgigabitEthernet trunk** commands. If a port number is not specified, the default port 1 is used, when a switchover happens.

Stack Mode	Apps	Ports Used	Behavior
Cisco Catalyst 9300X Series Switch active + Cisco Catalyst 9300 Series Switch standby	1	2	Not supported. In this scenario, when a switchover happens, the new active will not have two front-panel ports, and the app configuration fails. After the switchover, the application is not restarted on Cisco Catalyst 9300 Series Switch, because only one front-panel port is configured, and this configuration fails. The application must be reconfigured using the available front-panel port.
Cisco Catalyst 9300X Series Switch active + Cisco Catalyst 9300 Series Switch standby	2	2	Not supported. **Note** Two applications, for example, app1 and app2 are running, with each application using a different front-panel port, for example, port1 and port2 respectively. After a switchover, app1 on front-panel port1 starts on Cisco Catalyst 9300 Series Switch in a running state. However, app2 is not started on Cisco Catalyst 9300 Series Switch as there is no front-panel port2.
Catalyst 9300 Series Switch active + Catalyst 9300X Series Switch standby	1 or more	1	Supported. **Note** After a switchover, the application is restarted on Catalyst 9300X Series Switch using the front-panel port.

Application Hosting on Cisco Catalyst 9400 Series Switches

This section describes application-hosting on Cisco Catalyst 9400 Series Switches.

Cisco Catalyst 9400 Series Switches support the management interface and front-panel ports for application hosting. Applications can be hosted on C9400-SSD-240GB, C9400-SSD-480GB, and C9400-SSD-960GB solid state drives (SSDs).

These switches use the M2 SATA module for application hosting. For more information, see the "M2 SATA Module" chapter in the *Interfaces and Hardware Configuration Guide*.

On Cisco Catalyst 9400 Series Switches, applications can be hosted only on active supervisors. After a switchover, the AppGigbitEthernet interface on the newly active supervisor becomes active and can be used for application hosting.

Application Hosting on Cisco Catalyst 9410 Series Switches

In Cisco IOS XE Bengaluru 17.5.1, application hosting is supported on Cisco Catalyst 9410 Series Switches. To enable the AppGigabitEthernet interface for application hosting, configure the **enable** command in interface configuration mode.

Note The **enable** command is available only on Cisco Catalyst 9410 Series Switches.

When using slot 4 of the 48-port linecard for application hosting, the port must be in the default shutdown mode. If slot 4 of the 48-port linecard is active, application hosting is rejected. If the linecard port is disabled, slot 4 of the 48-port linecard is marked as *inactive*.

If slot 4 of the 48-port linecard is populated, the port 4/0/48 will not come up. If linecard 4 is empty or if it is a 24-port linecard, no ports are disabled.

To enable the port (4/0/48), disable application hosting by using the **no iox** command. No system messages are displayed on the console when the port is enabled or disabled.

During an In-Service Software Upgrade (ISSU), the linecard port is not automatically disabled, because the AppGigabitEthernet interface has to be enabled. Before a software downgrade, the AppGigabitEthernet interface must be disabled to disable the front-panel port.

Online Insertion and Removal

Table 46: Online Insertion and Removal (OIR) Scenarios

OIR Scenario	Action
The linecard on slot 4 is empty, and the AppGigabitEthernet interface is enabled.	No port is disabled.
The linecard on slot 4 is a 48-port linecard, and the AppGigabitEthernet interface is enabled.	Port 48 on slot 4 is disabled. After the port is disabled, no configuration is applied to the port. Port 48 is marked as inactive.
The linecard on slot 4 is a 24-port linecard.	No port on slot 4 is disabled.
The linecard on slot 4 is a 48-port linecard that is replaced by a 24-port linecard, and the AppGigabitEthernet interface is enabled.	No port on slot 4 is disabled.
The linecard on slot 4 is a 24-port linecard that is replaced by a 48-port linecard, and the AppGigabitEthernet interface is enabled.	Port 48 on slot 4 is disabled.
During OIR, the standby Supervisor becomes the new active, and the front-panel port on the new active is used for app hosting.	No state change will happen to port 48 on slot 4. The standby Supervisor OIR has no effect on the active Supervisor front-panel port.

Cisco StackWise Virtual

This section describes the scenarios when uplink ports in a dual Supervisor are used as StackWise Virtual links:

- When application hosting is enabled, and port 48 on linecard 4 is not up, it is disabled on both the active and standby chassis.
- If the link is up on either the active or standby chassis on port 48 linecard 4, then the **enable** command is rejected.
- If port 48 on linecard 4 is used as a dual-active detection (DAD) link, remove the DAD link, and configure it on another port.
- If port 48 on linecard 4 is used as a StackWise Virtual link, and the front-panel port must be enabled, remove the StackWise Virtual link on port 48 and use another port as the StackWise Virtual link. Port 48 on linecard 4 cannot be used as a StackWise Virtual or DAD link.

Application Hosting on Cisco Catalyst 9500 Series Switches

Cisco Catalyst 9500-High Performance Series Switches support only M2 SATA modules, SSD-240G, SSD-480G, and SSD-960 (C9k-F1-SSD-240GB). Front-panel USB is not supported.

For more information, see the "M2 SATA Module" of the *Interface and Hardware Components Configuration Guide, Cisco IOS XE Amsterdam 17.2.x (Catalyst 9500 Switches)*.

In Cisco IOS XE Cupertino 17.7.1, Cisco Catalyst 9500X Series Switches support application hosting on AppGigabitEthernet interfaces. Application Hosting is supported on the M2 SATA modules: SSD-240G, SSD-480G, and SSD-960 (C9k-F1-SSD-240GB).

Application Hosting on Cisco Catalyst 9600 Series Switches

Cisco Catalyst 9600 Series Switches support only M2 SATA modules for application hosting; front-panel USB is not supported. The following M2 SATA modules are supported: SSD-240G, SSD-480G, and SSD-960 (C9k-F2-SSD-240GB)

For more information, see the "M2 SATA Module" of the *Interface and Hardware Components Configuration Guide, Cisco IOS XE Amsterdam 17.2.x (Catalyst 9600 Switches)*.

Autotransfer and Auto-Install of Apps from Internal Flash to SSD

When IOx is enabled, it chooses the best available media, and starts the IOx service using that media. IOx also selects the media to run the applications at startup.

When IOx is restarted and a different media is selected, all applications (only Docker applications are supported.) must be migrated to the new media, and the containers must be restored to the same state as before the change. All persistent data and volumes attached to an application must also be migrated.

During a restart, IOx selects the media in the following order of precedence:

1. Harddisk
2. Flash

   Flash only supports Guest Shell; no other applications are allowed.

Use Cases

This section describes a couple of use cases during autotransfer and auto-install of applications.

Table 47: Use Cases for the AutoTransfer and Auto-Install of Applications

Use Case	Result
SSD is plugged in while IOx is running on flash.	If the SSD is plugged in while IOx is already running, there is no impact to the running applications or to IOx. IOx is migrated to the SSD only when IOx is restarted by disabling and then enabling IOx through the CLI, or due to a system restart.
System reboots, while IOx data is being copied to the new media.	While IOx data is getting migrated from one media to another, and the system reboots, the migration process will continue, when the system restarts. The data from the old media is deleted only when the copy operation is complete.

Native Docker Container: Application Auto-Restart

The Application Auto-Restart feature helps applications deployed on platforms to retain the last configured operational state in the event of a system switchover or restart. The underlying hosting framework is also retained during switchovers. This feature is enabled by default, and cannot be disabled by users.

The persistent data of applications is not synchronized; only secure data storage and persistent data that is known to Cisco Application Framework (CAF) is synchronized.

IOx media present on the active and standby devices must be in-sync to restart IOx in the same state upon a switchover or system restart.

Cisco Catalyst 9300 Series Switches only support Solid State Drive (SSD) for application hosting. When a new SSD is inserted, it needs to be brought up to the same sync state as the others. The standby device must have an SSD that is compatible with IOx for application auto-restart synchronization to work.

The output of the **show iox-service** command displays the status of the synchronization.

The Application Auto-Restart feature is supported only on Cisco Catalyst 9300 Series Switches.

Application Auto-Restart Scenarios

This section describes various application auto-restart scenarios:

Table 48: Application Auto-Restart Scenarios

Scenario	Single Media in the Active Device	Media in the Active and Standby Devices
System bootup	Starts IOx and the application at system bootup. The USB SSD is visible immediately because it is a local device. No synchronization happens at this time.	Starts IOx and the application on system bootup. Does a bulk synchronization of the existing information to the standby device.

Scenario	Single Media in the Active Device	Media in the Active and Standby Devices
Switchover	Media is not found on the new active device. IOx starts on the system flash with no previously installed applications and with minimum capabilities.	Starts IOx and the application in the previous state on the new active device after the system switchover (SSO). Does a bulk synchronization of the information to the new standby device after it boots up.
Bootup or switchover: USB SSD is present on a member device.	No synchronization of the SSD present in member devices. The member SSD is not used to host IOx and applications.	No synchronization of the SSD present in member devices. The member SSD is not used to host IOx and applications.
Device removal: Local USB SSD is removed from the active device.	When the local USB SSD is removed, IOx takes care of the graceful exit. User-triggered IOx restart is required once SSD is plugged back in the active device.	IOx takes care of the graceful exit. Since IOx operates only on the local disk, the standby SSD is not used to start IOx. User-triggered IOx restart is required once SSD is plugged back in the active device.
Device removal: USB SSD is removed from the standby device.	NA	IOx synchronization operation fails. IOx is no longer SSO ready.
Device removal: Remote USB SSD is removed from a remote member device.	IOx does not use any member SSD, and hence, there is no impact.	IOx does not use any member SSD, and hence, there is no impact.
Device going down: The active device on which IOx is running goes down.	Media is not found on the new active device. IOx starts up on the system flash with no previously installed applications and with minimum capabilities.	Starts IOx and applications in the state before the SSO on the new active device. Does a bulk synchronization of the information to the new standby device once it boots up.
Designated active-standby device change (stack environment 1:1)	The change is reflected after the reboot. IOx starts from the new active device after the reboot.	The change is reflected after the reboot. IOx starts from the new active device after the reboot.

Application Auto-Restart on Cisco Catalyst 9300 Series Switches

This section describes how application auto-restart works on Cisco Catalyst 9300 Series Switches in a multimember stack:

On Cisco Catalyst 9300 Series Switches, application auto-restart is supported in 1+1 switch redundancy or StackWise Virtual modes that assign the active and standby roles to specific devices in the stack.

Application auto-restart is not supported when the switch stack in is N+1 mode. If the device is in N+1 mode, the following log message is displayed on the console:

```
Feb 5 20:29:17.022: %IOX-3-IOX_RESTARTABITLITY: Switch 1 R0/0: run_ioxn_caf:Stack is in N+1
 mode,
disabling sync for IOx restartability
```

IOx uses a Cisco-certified USB3.0 flash drive in the back-panel USB port as storage for application hosting. This media may not be present in all the stack members.

Data is synced using the rsync utility from the active to the standby device.

Supported Network Types

This section lists the types of networks supported on Cisco Catalyst Switches.

Table 49: Supported Network Types

Network Type	Supported Platform and Release
Management port	• Catalyst 9300 Series Switches and C9300L in Cisco IOS XE Gibraltar 16.12.1 • Catalyst 9400 Series Switches in Cisco IOS XE Amsterdam 17.1.1 • Catalyst 9500 Series Switches and Catalyst 9500-High Performance Series Switches in Cisco IOS XE Amsterdam 17.2.1 • Catalyst 9600 Series Switches in Cisco IOS XE Amsterdam 17.2.1
Front-panel port (trunk and VLAN)	• Catalyst 9300 Series Switches and C9300L in Cisco IOS XE Gibraltar 16.12.1 • Catalyst 9400 Series Switches in Cisco IOS XE Amsterdam 17.1.1 • Catalyst 9500- High Performance Series Switches in Cisco IOS XE Amsterdam 17.5.1 • Catalyst 9600 Series Switches in Cisco IOS XE Amsterdam 17.5.1 • Catalyst 9300X Series Switches in Cisco IOS XE Bengaluru 17.6.1 **Note** Catalyst 9300X Series Switches support multiple AppGigabitEthernet ports.

Network Type	Supported Platform and Release
Cisco IOS Network Address Translation (NAT)	• Catalyst 9300 Series Switches and C9300L in Cisco IOS XE Gibraltar 16.12.1 • Catalyst 9400 Series Switches in Cisco IOS XE Amsterdam 17.1.1 On both these platforms, NAT is supported through the hardware data-port features applied on the front-panel data ports and on the AppGigabitEthernet port.
Cisco IOx NAT	Not supported

Virtual Network Interface Card

To manage the life cycle of an application container, the Layer 3 routing model that supports one container per internal logical interface is used. This means that a virtual Ethernet pair is created for each application, and one interface of this pair, called the Virtual Network Interface Card (vNIC) is part of the application container.

NIC is the standard Ethernet interface inside the container that connects to the platform data plane for the sending and receiving of packets. Cisco IOx is responsible for assigning the IP address and unique MAC address for each vNIC in the container.

The vNICs inside a container are considered as standard Ethernet interfaces.

ERSPAN Support on the AppGigabitEthernet Port

Encapsulated Remote Switch Port Analyzer (ERSPAN) support on an AppGigabitEthernet port enables the mirroring of data traffic from a device to an application that runs on the AppGigabitEthernet port by using IOx.

Note The Cisco IOx process must be running before the IOx virtual application can be hosted on a Cisco device.

Multicast Routing on the AppGigabitEthernet Interface

Multicast traffic forwarding is supported on the AppGigabitEthernet interface. Applications can select the networks that allow multicast traffic. Multicast traffic forwarding is enabled through the IOS CLI and the package.yaml file.

If a platform supports multicast routing; but the network does not support multicast, an activation error message is displayed. On some platforms, to enable multicast routing, you must disable IGMP snooping.

If multicast traffic forwarding is enabled on an app in a network, you cannot activate another app on the same network, without enabling multicast on that app.

Note Multicast traffic forwarding is not supported on the management interface. However; when the management interface is used as an external AppGigabitEthernet interface, multicast traffic forwarding can be enabled on the interface.

How to Configure Application Hosting

The following sections provide information about the various tasks that comprise the configuration of application hosting.

Enabling Cisco IOx

Perform this task to enable access to Cisco IOx, which provides a CLI-based user interface that you can use to manage, administer, monitor, and troubleshoot the apps on the host system, and to perform a variety of related activities.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **iox**
4. **username** *name* **privilege** *level* **password** {**0** | **7** | *user-password*}*encrypted-password*
5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**iox** **Example:** `Device(config)# iox`	Enables Cisco IOx.
Step 4	**username** *name* **privilege** *level* **password** {**0** | **7** | *user-password*}*encrypted-password* **Example:**	Establishes a username-based authentication system and privilege level for the user.

	Command or Action	Purpose
	`Device(config)# username cisco privilege 15 password 0 ciscoI`	• The username privilege level must be configured as 15.
Step 5	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC configuration mode.

Configuring Application Hosting on Front-Panel VLAN Ports

Note This task is applicable to Cisco IOS XE Amsterdam 17.1.1 and later releases.

In application-hosting trunk-configuration mode, all the allowed AppGigabitEthernet VLAN ports are connected to a container. Native and VLAN-tagged frames are transmitted and received by the container guest interface. Only one container guest interface can be mapped to the AppGigabitEthernet trunk port.

Concurrent configuration of both *trunk* and *vlan-access* ports are supported.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *AppGigabitEthernet number*
4. **switchport trunk allowed vlan** *vlan-ID*
5. **switchport mode trunk**
6. **exit**
7. **app-hosting appid** *name*
8. **app-vnic AppGigabitEthernet trunk**
9. **vlan** *vlan-ID* **guest-interface** *guest-interface-number*
10. **guest-ipaddress** *ip-address* **netmask** *netmask*
11. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**interface** *AppGigabitEthernet number* **Example:** `Device(config)# interface AppGigabitEthernet 1/0/1`	Configures the AppGigabitEthernet and enters interface configuration mode. • For stackable switches, the *number* argument is *switch-number/0/1*.
Step 4	**switchport trunk allowed vlan** *vlan-ID* **Example:** `Device(config-if)# switchport trunk allowed vlan 10-12,20`	Configures the list of VLANs allowed on the trunk.
Step 5	**switchport mode trunk** **Example:** `Device(config-if)# switchport mode trunk`	Sets the interface into permanent trunking mode and negotiates to convert the neighboring link into a trunk link.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 7	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid iox_app`	Configures an application and enters application-hosting configuration mode.
Step 8	**app-vnic AppGigabitEthernet trunk** **Example:** `Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk`	Configures a trunk port as the front-panel port for an application, and enters application-hosting trunk-configuration mode.
Step 9	**vlan** *vlan-ID* **guest-interface** *guest-interface-number* **Example:** `Device(config-config-app-hosting-trunk)# vlan 10 guest-interface 2`	Configures a VLAN guest interface and enters application-hosting VLAN-access IP configuration mode. • Multiple VLAN-to-guest interface mapping is supported.
Step 10	**guest-ipaddress** *ip-address* **netmask** *netmask* **Example:** `Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.0.2 netmask 255.255.255.0`	(Optional) Configures a static IP address.
Step 11	**end** **Example:** `Device(config-config-app-hosting-vlan-access-ip)# end`	Exits application-hosting VLAN-access IP configuration mode and returns to privileged EXEC mode.

Configuring Application Hosting on Front-Panel Trunk Ports

In application-hosting trunk-configuration mode, all the allowed AppGigabitEthernet VLAN ports are connected to a container. Native and VLAN-tagged frames are transmitted and received by the container guest interface. Only one container guest interface can be mapped to the AppGigabitEthernet trunk port.

In Cisco IOS XE Gibraltar 16.2.1, you can configure an app-ID in either application-hosting trunk configuration mode or application-hosting VLAN-access configuration mode; but not in both modes.

In Cisco IOS XE Amsterdam 17.1.1 and later releases, concurrent configuration of both *trunk* and *vlan-access* ports is supported.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *AppGigabitEthernet number*
4. **switchport trunk allowed vlan** *vlan-ID*
5. **switchport mode trunk**
6. **exit**
7. **app-hosting appid** *name*
8. **app-vnic AppGigabitEthernet trunk**
9. **guest-interface** *guest-interface-number*
10. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *AppGigabitEthernet number* **Example:** `Device(config)# interface AppGigabitEthernet 1/0/1`	Configures the AppGigabitEthernet and enters interface configuration mode. • For stackable switches, the *number* argument is *switch-number/0/1*.
Step 4	**switchport trunk allowed vlan** *vlan-ID* **Example:** `Device(config-if)# switchport trunk allowed vlan 10-12,20`	Configures the list of VLANs allowed on the trunk.

	Command or Action	Purpose
Step 5	**switchport mode trunk** **Example:** `Device(config-if)# switchport mode trunk`	Sets the interface into permanent trunking mode and negotiates to convert the neighboring link into a trunk link.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 7	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid iox_app`	Configures an application and enters application-hosting configuration mode.
Step 8	**app-vnic AppGigabitEthernet trunk** **Example:** `Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk`	Configures a trunk port as the front-panel port for an application, and enters application-hosting trunk-configuration mode.
Step 9	**guest-interface** *guest-interface-number* **Example:** `Device(config-config-app-hosting-trunk)# guest-interface 2`	Configures an application's interface that is connected to the AppGigabitEthernet interface trunk.
Step 10	**end** **Example:** `Deviceconfig-config-app-hosting-trunk)# end`	Exits application-hosting trunk-configuration mode and returns to privileged EXEC mode.

Starting an Application in Configuration Mode

The **start** command in application-hosting configuration mode is equivalent to the **app-hosting activate appid** and **app-hosting start appid** commands.

The **no start** command in application-hosting configuration mode is equivalent to the **app-hosting stop appid** and **app-hosting deactivate appid** commands.

Note If the **start** command is configured before an application is installed, and then the **install** command is configured, Cisco IOx automatically performs internal **activate** and **start** actions. This allows the application to be automatically started by configuring the **install** command.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *application-name*
4. **start**

5. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**app-hosting appid** *application-name* **Example:** `Device(config)# app-hosting appid iox_app`	Configures an application and enters application-hosting configuration mode.
Step 4	**start** **Example:** `Device(config-app-hosting)# start`	(Optional) Starts and runs an application. • Use the **no start** command to stop the application.
Step 5	**end** **Example:** `Device(config-app-hosting)# end`	Exits application-hosting configuration mode and returns to privileged EXEC mode.

Lifecycle of an Application

The following EXEC commands take you through an application's lifecycle.

Note If any configuration changes are made after an application is installed, the application in the running state will not reflect these changes. The application must be explicitly stopped and deactivated, and then activated and started again for the configuration changes to take effect.

SUMMARY STEPS

1. **enable**
2. **app-hosting install appid** *application-name* **package** *package-path*
3. **app-hosting activate appid** *application-name*
4. **app-hosting start appid** *application-name*
5. **app-hosting stop appid** *application-name*
6. **app-hosting deactivate appid** *application-name*

7. **app-hosting uninstall appid** *application-name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**app-hosting install appid** *application-name* **package** *package-path* **Example:** `Device# app-hosting install appid iox_app package usbflash1:my_iox_app.tar`	Installs an application from the specified location. • An application can be installed from a local storage location such as, flash, bootflash, usbflash0, usbflash1, and harddisk.
Step 3	**app-hosting activate appid** *application-name* **Example:** `Device# app-hosting activate appid iox_app`	Activates the application. • This command validates all the application resource requests, and if all the resources are available, the application is activated; if not, the activation fails.
Step 4	**app-hosting start appid** *application-name* **Example:** `Device# app-hosting start appid iox_app`	Starts the application. • Application start-up scripts are activated.
Step 5	**app-hosting stop appid** *application-name* **Example:** `Device# app-hosting stop appid iox_app`	(Optional) Stops the application.
Step 6	**app-hosting deactivate appid** *application-name* **Example:** `Device# app-hosting deactivate appid iox_app`	(Optional) Deactivates all the resources allocated for the application.
Step 7	**app-hosting uninstall appid** *application-name* **Example:** `Device# app-hosting uninstall appid iox_app`	(Optional) Uninstalls the application. • Uninstalls all the packaging and images stored. All the changes and updates to the application are also removed.

Configuring Docker Run Time Options

You can add a maximum of 30 lines of run time options. The system generates a concatenated string from line 1 though line 30. A string can have more than one Docker run time option.

When a run time option is changed, stop, deactivate, activate, and start the application for the new run time options to take effect.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *application-name*
4. **app-resource docker**
5. **run-opts** *options*
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**app-hosting appid** *application-name* **Example:** `Device(config)# app-hosting appid iox_app`	Configures an application and enters application-hosting configuration mode.
Step 4	**app-resource docker** **Example:** `Device(config-app-hosting)# app-resource docker`	Enters application-hosting docker-configuration mode to specify application resource updates.
Step 5	**run-opts** *options* **Example:** `Device(config-app-hosting-docker)# run-opts 1 "-v $(APP_DATA):/data"`	Specifies the Docker run time options.
Step 6	**end** **Example:** `Device(config-app-hosting-docker)# end`	Exits application-hosting docker-configuration mode and returns to privileged EXEC mode.

Configuring a Static IP Address in a Container

When configuring a static IP address in a container, the following guidelines apply:

- Only the last configured default gateway configuration is used.
- Only the last configured name server configuration is used.

You can configure the IP address of a container through Cisco IOS CLIs.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *name*
4. **name-server#** *ip-address*
5. **app-vnic management guest-interface** *interface-number*
6. **guest-ipaddress** *ip-address* **netmask** *netmask*
7. **exit**
8. **app-default-gateway** *ip-address* **guest-interface** *network-interface*
9. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid iox_app`	Configures an application and enters application-hosting configuration mode.
Step 4	**name-server#** *ip-address* **Example:** `Device(config-app-hosting)# name-server0 10.2.2.2`	Configures the Domain Name System (DNS) server.
Step 5	**app-vnic management guest-interface** *interface-number* **Example:** `Device(config-app-hosting)# app-vnic management guest-interface 0`	Configures the management gateway of the virtual network interface and guest interface, and enters application-hosting management-gateway configuration mode.
Step 6	**guest-ipaddress** *ip-address* **netmask** *netmask* **Example:**	Configures the management guest interface details.

	Command or Action	Purpose
	`Device(config-app-hosting-mgmt-gateway)# guest-ipaddress 172.19.0.24 netmask 255.255.255.0`	
Step 7	**exit** **Example:** `Device(config-app-hosting-mgmt-gateway)# exit`	Exits application-hosting management-gateway configuration mode and returns to application-hosting configuration mode.
Step 8	**app-default-gateway** *ip-address* **guest-interface** *network-interface* **Example:** `Device(config-app-hosting)# app-default-gateway 172.19.0.23 guest-interface 0`	Configures the default management gateway.
Step 9	**end** **Example:** `Device(config-app-hosting)# end`	Exits application-hosting configuration mode and returns to privileged EXEC mode.

Configuring Application Hosting on the Management Port

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface gigabitethernet0/0**
4. **vrf forwarding** *vrf-name*
5. **ip address** *ip-address mask*
6. **exit**
7. **app-hosting appid** *name*
8. **app-vnic management guest-interface** *network-interface*
9. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`Device# configure terminal`	
Step 3	**interface gigabitethernet0/0** **Example:** `Device(config)# interface gigabitethernet0/0`	Configures an interface and enters interface configuration mode. • On Cisco Catalyst 9000 Series Switches, the management interface is GigabitEthernet0/0.
Step 4	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding Mgmt-vrf`	Associates a Virtual Routing and Forwarding (VRF) instance or a virtual network with an interface or subinterface. • *Mgmt-vrf* is automatically set for the management interface on the Cisco Catalyst 9000 Series Switch.
Step 5	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 198.51.100.1 255.255.255.254`	Configures an IP address for the interface.
Step 6	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 7	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid iox_app`	Configures an application and enters application-hosting configuration mode.
Step 8	**app-vnic management guest-interface** *network-interface* **Example:** `Device(config-app-hosting)# app-vnic management guest-interface 1`	Connects the guest interface to the management port, and enters application-hosting management-gateway configuration mode. • The **management** keyword specifies the Cisco IOS management GigabitEthernet0/0 interface that is connected to the container. • The **guest-interface** *network-interface* keyword-argument pair specifies the container's internal Ethernet interface number that is connected to the Cisco IOS management interface. The example provided here uses *guest-interface 1* for the container's Ethernet 1 interface.
Step 9	**end** **Example:** `Device(config-app-hosting-mgmt-gateway)# end`	Exits application-hosting management-gateway configuration mode and returns to privileged EXEC mode.

Manually Configuring the IP Address for an Application

You can set up the IP address of a container using the following methods:

- Log into the container, and configure the **ifconfig** Linux command.
  1. Log in to the application by using the following command:

     ```
     app-hosting connect appid APPID {session | console}
     ```

  2. Based on the application's Linux support, use the standard Linux interface configuration commands:

     ```
     - ifconfig dev IFADDR/subnet-mask-length
     ```

     Or

     ```
     - ip address {add|change|replace} IFADDR dev IFNAME [ LIFETIME ] [ CONFFLAG-LIST ]
     ```

- Enable the Dynamic Host Configuration Protocol (DHCP) in the container, and configure the DHCP server and relay agent in the Cisco IOS configuration.
  - Cisco IOx provides a DHCP client to run within the application container that is used for an application DHCP interface.

Overriding App Resource Configuration

For resource changes to take effect, you must first stop and deactivate an app using the **app-hosting stop** and **app-hosting deactivate** commands, and then restart the app using the **app-hosting activate** and **app-hosting start** commands.

If you are using the **start** command in application-hosting configuration mode, configure the **no start** and **start** commands.

You can use these commands to reset both resources and the app-hosting appid iox_app configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *name*
4. **app-resource profile** *name*
5. **cpu** *unit*
6. **memory** *memory*
7. **vcpu** *number*
8. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid iox_app`	Enables application hosting and enters application-hosting configuration mode.
Step 4	**app-resource profile** *name* **Example:** `Device(config-app-hosting)# app-resource profile custom`	Configures the custom application resource profile, and enters custom application resource profile configuration mode. • Only the custom profile name is supported.
Step 5	**cpu** *unit* **Example:** `Device(config-app-resource-profile-custom)# cpu 7400`	Changes the default CPU allocation for the application. • Resource values are application specific, and any adjustment to these values must ensure that the application can run reliably with the changes.
Step 6	**memory** *memory* **Example:** `Device(config-app-resource-profile-custom)# memory 2048`	Changes the default memory allocation.
Step 7	**vcpu** *number* **Example:** `Device(config-app-resource-profile-custom)# vcpu 2`	Changes the virtual CPU (vCPU) allocation for the application.
Step 8	**end** **Example:** `Device(config-app-resource-profile-custom)# end`	Exits custom application resource profile configuration mode and returns to privileged EXEC mode.

Configuring ERSPAN Support on the AppGigabitEthernet Port

Perform the following tasks to configure ERSPAN through an AppGigabitEthernet interface.

Note The IOx process must be running before the IOx virtual application can be hosted on a Cisco device.

Configuring an ERSPAN Source Session

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **monitor session** *span-session-number* **type erspan-source**
4. **source interface** *interface-type interface-id*
5. **no shutdown**
6. **ip address** *ip-address*
7. **origin ip address** *ip-address*
8. **erspan-id** *erspan-flow-id*
9. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**monitor session** *span-session-number* **type erspan-source** **Example:** `Device(config)# monitor session 2 type erspan-source`	Defines an ERSPAN source session using the session ID and the session type, and enters ERSPAN monitor source session configuration mode.
Step 4	**source interface** *interface-type interface-id* **Example:** `Device(config-mon-erspan-src)# source interface gigabitethernet 1/0/3`	Configures the source interface and the traffic direction to be monitored.
Step 5	**no shutdown** **Example:** `Device(config-mon-erspan-src)# no shutdown`	Enables the configured sessions on an interface.
Step 6	**ip address** *ip-address* **Example:** `Device(config-mon-erspan-src-dst)# ip address 10.1.1.5`	Configures the IP address that is used as the destination of the ERSPAN traffic.

	Command or Action	Purpose
Step 7	**origin ip address** *ip-address* **Example:** `Device(config-mon-erspan-src-dst)# origin ip address 10.1.1.2`	Configures the IP address used as the source of the ERSPAN traffic.
Step 8	**erspan-id** *erspan-flow-id* **Example:** `Device(config-mon-erspan-src-dst)# erspan-id 5`	Configures the ID used by the source and destination sessions to identify the ERSPAN traffic, which must also be entered in the ERSPAN destination session configuration.
Step 9	**end** **Example:** `Device(config-mon-erspan-src-dst)# end`	Exits ERSPAN monitor source session configuration mode and returns to privileged EXEC mode.

Configuring the AppGigabitEthernet Interface for ERSPAN

Note You can either use a Layer 2 port or a Layer 3 port for the ERSPAN traffic. Use the **no switchport mode** command to change the port from a Layer 2 interface to a Layer 3 interface.

Before you begin

1. Step 1 to Step 9 show how to configure a VLAN for traffic mirroring.
2. Step 10 to Step 14 show how to configure the AppGigabitEthernet interface to transport ERSPAN-mirrored data traffic to the IOx virtual application.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **vtp mode off**
4. **vlan** {*vlan-ID* | *vlan-range*}
5. **exit**
6. **interface vlan** *vlan-ID*
7. **ip address** *ip-address mask*
8. **no shutdown**
9. **exit**
10. **interface AppGigabitEthernet** *number*
11. (Optional) **no switchport mode**
12. (Optional) **ip address** *ip-address mask*
13. (Optional) **switchport mode trunk**
14. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**vtp mode off** **Example:** `Device(config)#vtp mode off`	Sets the VTP-device mode to Off for VLANs.
Step 4	**vlan** {*vlan-ID* | *vlan-range*} **Example:** `Device(config)# vlan 2508`	Adds a VLAN and enters config-VLAN configuration mode.
Step 5	**exit** **Example:** `Device(config-vlan)# exit`	Exits config-vlan configuration mode and returns to global configuration mode.
Step 6	**interface vlan** *vlan-ID* **Example:** `Device(config)# interface vlan 2508`	Creates a dynamic Switch Virtual Interface (SVI) and enters interface configuration mode.
Step 7	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 192.0.2.1 255.255.255.252`	Configures an IP address.
Step 8	**no shutdown** **Example:** `Device(config-if)# no shutdown`	Restarts a disabled interface.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 10	**interface AppGigabitEthernet** *number* **Example:** `Device(config)# interface AppGigabitEthernet 1/1`	Configures the AppGigabitEthernet and enters interface configuration mode. For stackable switches, the number argument is *switch-number/0/1*.

	Command or Action	Purpose
		Note You can use either a Layer 2 or a Layer 3 port.
Step 11	(Optional) **no switchport mode** **Example:** `Device(config-if)# no switchport mode`	Changes the port from a Layer 2 interface to a Layer 3 interface.
Step 12	(Optional) **ip address** *ip-address mask* **Example:** `Device(config-if)# 10.1.1.2 255.255.255.0`	Configures an IP address for a Layer 3 port.
Step 13	(Optional) **switchport mode trunk** **Example:** `Device(config-if)# switchport mode  trunk`	Sets the interface into permanent trunking mode and negotiates to convert the neighboring link into a trunk link for a Layer 2 port.
Step 14	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Enabling Multicast Routing on the AppGigabitEthernet Interface

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *name*
4. **app-vnic AppGigabitEthernet trunk**
5. **vlan** *vlan-ID* **guest-interface** *guest-interface-number*
6. **guest-ipaddress***ip-address* **netmask** *netmask*
7. **multicast**
8. **exit**
9. **exit**
10. **app-default-gateway** *ip-address* **guest-interface** *network-interface*
11. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**app-hosting appid** *name* **Example:** `Device(config)# app-hosting appid iox_app`	Configures an application and enters application-hosting configuration mode.
Step 4	**app-vnic AppGigabitEthernet trunk** **Example:** `Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk`	Configures a trunk port as the front-panel port for an application, and enters application-hosting trunk-configuration mode.
Step 5	**vlan** *vlan-ID* **guest-interface** *guest-interface-number* **Example:** `Device(config-config-app-hosting-trunk)# vlan 10 guest-interface 2`	Configures a VLAN guest interface and enters application-hosting VLAN-access IP configuration mode.
Step 6	**guest-ipaddress***ip-address* **netmask** *netmask* **Example:** `Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.0.2 netmask 255.255.255.0`	Configures a static IP address.
Step 7	**multicast** **Example:** `Device(config-config-app-hosting-vlan-access-ip)# multicast`	Enables multicast traffic forwarding on the AppGigabitEthernet interface.
Step 8	**exit** **Example:** `Device(config-config-app-hosting-vlan-access-ip)# exit`	Exits application-hosting VLAN-access IP configuration mode and returns to application-hosting trunk-configuration mode
Step 9	**exit** **Example:** `Device(config-config-app-hosting-trunk)# exit`	Exits application-hosting trunk-configuration mode and returns to application-hosting configuration mode.
Step 10	**app-default-gateway** *ip-address* **guest-interface** *network-interface* **Example:** `Device(config-app-hosting)# app-default-gateway 172.19.0.23 guest-interface 0`	Configures the default management gateway.
Step 11	**end** **Example:**	Exits application-hosting configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
	`Device(config-app-hosting)# end`	

Verifying the Application-Hosting Configuration

Use these **show** commands to verify the configuration. These commands can be used in any order.

SUMMARY STEPS

1. **enable**
2. **show iox-service**
3. **show app-hosting detail**
4. **show app-hosting device**
5. **show app-hosting list**
6. **show interfaces trunk**
7. **show controller ethernet-controller AppGigabitEthernet** *interface-number*

DETAILED STEPS

Procedure

Step 1 **enable**

Enables privileged EXEC mode.

- Enter your password if prompted.

Example:

```
Device> enable
```

Step 2 **show iox-service**

Displays the status of all the Cisco IOx services.

Example:

```
Device# show iox-service

IOx Infrastructure Summary:
---------------------------
IOx service (CAF)          : Not Running
IOx service (HA)           : Not Running
IOx service (IOxman)       : Not Running
IOx service (Sec storage)  : Not Running
Libvirtd                   : Running
Dockerd                    : Not Running
Application DB Sync Info   : Not available
```

Step 3 **show app-hosting detail**

Displays detailed information about the application.

Example:

```
Device# show app-hosting detail

State                       : Running
Author                      : Cisco Systems, Inc
Application
 Type                       : vm
 App id                     : Wireshark
 Name                       : Wireshark
 Version                    : 3.4
 Activated Profile Name     : custom
 Description                : Ubuntu based Wireshark
Resource Reservation
 Memory                     : 1900 MB
 Disk                       : 10 MB
 CPU                        : 4000 units
 VCPU                       : 2
Attached devices
Type          Name         Alias
---------------------------------------------------------------------
Serial/shell
Serial/aux
Serial/Syslog              serial2
Serial/Trace               serial3
Network Interfaces
---------------------------------------------------------------
eth0:
 MAC address               : 52:54:dd:80:bd:59
 IPv4 address
eth1:
 MAC address               : 52:54:dd:c7:7c:aa
 IPv4 address
```

Step 4 **show app-hosting device**

Displays information about the USB device.

Example:

```
Device# show app-hosting device

USB port Device name Available
1 Front_USB_1 true


app-hosting appid testvm
app-vnic management guest-interface 0
app-device usb-port 1
```

Step 5 **show app-hosting list**

Displays the list of applications and their status.

Example:

```
Device# show app-hosting list

App id              State
---------------------------------------
Wireshark           Running
```

Step 6 **show interfaces trunk**

Displays trunk interface information.

Example:

```
Device# show interfaces trunk

Port Mode Encapsulation Status Native vlan
Gi3/0/1 on 802.1q trunking 1
Ap3/0/1 on 802.1q trunking 1

Port Vlans allowed on trunk
Gi3/0/1 1-4094
Ap3/0/1 1-4094

Port Vlans allowed and active in management domain
Gi3/0/1 1,8,10,100
Ap3/0/1 1,8,10,100

Port Vlans in spanning tree forwarding state and not pruned
Gi3/0/1 1,8,10,100
Ap3/0/1 1,8,10,100

Device# show runnning-config interface AppGigabitEthernet 3/0/1

Building configuration...

Current configuration : 64 bytes
!
interface AppGigabitEthernet3/0/1
switchport mode trunk
end
```

Step 7 **show controller ethernet-controller AppGigabitEthernet** *interface-number*

Displays the send and receive statistics for the AppGigabitEthernet interface that is read from the hardware.

Example:

```
Device# show controller ethernet-controller AppGigabitEthernet 1/0/1

Transmit                  AppGigabitEthernet1/0/1   Receive
            0 Total bytes                           0 Total bytes
            0 Unicast frames                        0 Unicast frames
            0 Unicast bytes                         0 Unicast bytes
            0 Multicast frames                      0 Multicast frames
            0 Multicast bytes                       0 Multicast bytes
            0 Broadcast frames                      0 Broadcast frames
            0 Broadcast bytes                       0 Broadcast bytes
            0 System FCS error frames               0 IpgViolation frames
            0 MacUnderrun frames                    0 MacOverrun frames
            0 Pause frames                          0 Pause frames
            0 Cos 0 Pause frames                    0 Cos 0 Pause frames
            0 Cos 1 Pause frames                    0 Cos 1 Pause frames
            0 Cos 2 Pause frames                    0 Cos 2 Pause frames
            0 Cos 3 Pause frames                    0 Cos 3 Pause frames
            0 Cos 4 Pause frames                    0 Cos 4 Pause frames
            0 Cos 5 Pause frames                    0 Cos 5 Pause frames
            0 Cos 6 Pause frames                    0 Cos 6 Pause frames
            0 Cos 7 Pause frames                    0 Cos 7 Pause frames
            0 Oam frames                            0 OamProcessed frames
            0 Oam frames                            0 OamDropped frames
            0 Minimum size frames                   0 Minimum size frames
            0 65 to 127 byte frames                 0 65 to 127 byte frames
            0 128 to 255 byte frames                0 128 to 255 byte frames
```

```
0 256 to 511 byte frames
0 512 to 1023 byte frames
0 1024 to 1518 byte frames
0 1519 to 2047 byte frames
0 2048 to 4095 byte frames
0 4096 to 8191 byte frames
0 8192 to 16383 byte frames
0 16384 to 32767 byte frame
0 > 32768 byte frames
0 Late collision frames
0 Excess Defer frames
0 Good (1 coll) frames
0 Good (>1 coll) frames
0 Deferred frames
0 Gold frames dropped
0 Gold frames truncated
0 Gold frames successful
0 1 collision frames
0 2 collision frames
0 3 collision frames
0 4 collision frames
0 5 collision frames
0 6 collision frames
0 7 collision frames
0 8 collision frames
0 9 collision frames
0 10 collision frames
0 11 collision frames
0 12 collision frames
0 13 collision frames
0 14 collision frames
0 15 collision frames
0 Excess collision frame
```

```
0 256 to 511 byte frames
0 512 to 1023 byte frames
0 1024 to 1518 byte frames
0 1519 to 2047 byte frames
0 2048 to 4095 byte frames
0 4096 to 8191 byte frames
0 8192 to 16383 byte frames
0 16384 to 32767 byte frame
0 > 32768 byte frames
0 SymbolErr frames
0 Collision fragments
0 ValidUnderSize frames
0 InvalidOverSize frames
0 ValidOverSize frames
0 FcsErr frames
```

Verifying the Application-Hosting Configuration

Use these **show** commands to verify the configuration. These commands can be used in any order.

- **show iox-service**

  Displays the status of all the Cisco IOx services.

```
Device# show iox-service

IOx Infrastructure Summary:
---------------------------
IOx service (CAF)          : Not Running
IOx service (HA)           : Not Running
IOx service (IOxman)       : Not Running
IOx service (Sec storage)  : Not Running
Libvirtd                   : Running
Dockerd                    : Not Running
Application DB Sync Info   : Not available
```

- **show app-hosting detail**

  Displays detailed information about the application.

```
Device# show app-hosting detail

State                       : Running
Author                      : Cisco Systems, Inc
Application
 Type                       : vm
 App id                     : Wireshark
 Name                       : Wireshark
 Version                    : 3.4
 Activated Profile Name     : custom
 Description                : Ubuntu based Wireshark
Resource Reservation
 Memory                     : 1900 MB
 Disk                       : 10 MB
 CPU                        : 4000 units
 VCPU                       : 2
Attached devices
Type          Name          Alias
---------------------------------------------------------------------
Serial/shell
Serial/aux
Serial/Syslog               serial2
Serial/Trace                serial3
Network Interfaces
---------------------------------------------------------------
eth0:
 MAC address                : 52:54:dd:80:bd:59
 IPv4 address
eth1:
 MAC address                : 52:54:dd:c7:7c:aa
 IPv4 address
```

- **show app-hosting device**

  Displays information about the USB device.

```
Device# show app-hosting device

USB port Device name Available
1 Front_USB_1 true


app-hosting appid testvm
app-vnic management guest-interface 0
app-device usb-port 1
```

- **show app-hosting list**

  Displays the list of applications and their status.

```
Device# show app-hosting list

App id              State
---------------------------------------
Wireshark           Running
```

- **show interfaces trunk**

  Displays trunk interface information.

```
Device# show interfaces trunk

Port Mode Encapsulation Status Native vlan
```

```
Gi3/0/1 on 802.1q trunking 1
Ap3/0/1 on 802.1q trunking 1

Port Vlans allowed on trunk
Gi3/0/1 1-4094
Ap3/0/1 1-4094

Port Vlans allowed and active in management domain
Gi3/0/1 1,8,10,100
Ap3/0/1 1,8,10,100

Port Vlans in spanning tree forwarding state and not pruned
Gi3/0/1 1,8,10,100
Ap3/0/1 1,8,10,100

Device# show runnning-config interface AppGigabitEthernet 3/0/1

Building configuration...

Current configuration : 64 bytes
!
interface AppGigabitEthernet3/0/1
switchport mode trunk
end
```

- **show controller ethernet-controller AppGigabitEthernet** *interface-number*

  Displays the send and receive statistics for the AppGigabitEthernet interface that is read from the hardware.

```
Device# show controller ethernet-controller AppGigabitEthernet 1/0/1

Transmit                  AppGigabitEthernet1/0/1   Receive
            0 Total bytes                           0 Total bytes
            0 Unicast frames                        0 Unicast frames
            0 Unicast bytes                         0 Unicast bytes
            0 Multicast frames                      0 Multicast frames
            0 Multicast bytes                       0 Multicast bytes
            0 Broadcast frames                      0 Broadcast frames
            0 Broadcast bytes                       0 Broadcast bytes
            0 System FCS error frames               0 IpgViolation frames
            0 MacUnderrun frames                    0 MacOverrun frames
            0 Pause frames                          0 Pause frames
            0 Cos 0 Pause frames                    0 Cos 0 Pause frames
            0 Cos 1 Pause frames                    0 Cos 1 Pause frames
            0 Cos 2 Pause frames                    0 Cos 2 Pause frames
            0 Cos 3 Pause frames                    0 Cos 3 Pause frames
            0 Cos 4 Pause frames                    0 Cos 4 Pause frames
            0 Cos 5 Pause frames                    0 Cos 5 Pause frames
            0 Cos 6 Pause frames                    0 Cos 6 Pause frames
            0 Cos 7 Pause frames                    0 Cos 7 Pause frames
            0 Oam frames                            0 OamProcessed frames
            0 Oam frames                            0 OamDropped frames
            0 Minimum size frames                   0 Minimum size frames
            0 65 to 127 byte frames                 0 65 to 127 byte frames
            0 128 to 255 byte frames                0 128 to 255 byte frames
            0 256 to 511 byte frames                0 256 to 511 byte frames
            0 512 to 1023 byte frames               0 512 to 1023 byte frames
            0 1024 to 1518 byte frames              0 1024 to 1518 byte frames
            0 1519 to 2047 byte frames              0 1519 to 2047 byte frames
            0 2048 to 4095 byte frames              0 2048 to 4095 byte frames
            0 4096 to 8191 byte frames              0 4096 to 8191 byte frames
            0 8192 to 16383 byte frames             0 8192 to 16383 byte frames
            0 16384 to 32767 byte frame             0 16384 to 32767 byte frame
            0 > 32768 byte frames                   0 > 32768 byte frames
```

```
0 Late collision frames
0 Excess Defer frames
0 Good (1 coll) frames
0 Good (>1 coll) frames
0 Deferred frames
0 Gold frames dropped
0 Gold frames truncated
0 Gold frames successful
0 1 collision frames
0 2 collision frames
0 3 collision frames
0 4 collision frames
0 5 collision frames
0 6 collision frames
0 7 collision frames
0 8 collision frames
0 9 collision frames
0 10 collision frames
0 11 collision frames
0 12 collision frames
0 13 collision frames
0 14 collision frames
0 15 collision frames
0 Excess collision frame
```

```
0 SymbolErr frames
0 Collision fragments
0 ValidUnderSize frames
0 InvalidOverSize frames
0 ValidOverSize frames
0 FcsErr frames
```

Configuration Examples for Application Hosting

The following are the various examples pertaining to the configuration of the Application Hosting feature.

Example: Enabling Cisco IOx

This example shows how to enable Cisco IOx.

```
Device> enable
Device# configure terminal
Device(config)# iox
Device(config)# username cisco privilege 15 password 0 ciscoI
Device(config)# end
```

Example: Configuring Application Hosting on Front-Panel VLAN Ports

Note This section is applicable to Cisco IOS XE Amsterdam 17.1.1 and later releases.

This example shows how to configure application hosting on front-panel VLAN ports.

```
Device# configure terminal
Device(config)# interface AppGigabitEthernet 1/0/1
Device(config-if)# switchport trunk allowed vlan 10-12,20
Device(config-if)# switchport mode trunk
```

```
Device(config-if)# exit
Device(config)# app-hosting appid iox_app
Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk
Device(config-config-app-hosting-trunk)# vlan 10 guest-interface 2
Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.0.1
netmask 255.255.255.0
Device(config-config-app-hosting-vlan access-ip)# end
```

Example: Configuring Application Hosting on Front-Panel Trunk Ports

This example shows how to configure application hosting on front-panel trunk ports.

```
Device# configure terminal
Device(config)# interface AppGigabitEthernet 3/0/1
Device(config-if)# switchport trunk allowed vlan 10-12,20
Device(config-if)# switchport mode trunk
Device(config-if)# exit
Device(config)# app-hosting appid iox_app
Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk
Device(config-config-app-hosting-trunk)# guest-interface 2
Device(config-config-app-hosting-trunk)# end
```

Example: Installing an Application from disk0:

The following example shows how to install an application from disk0:

```
Device> enable
Device# app-hosting install appid iperf3 package disk0:iperf3.tar

Installing package 'disk0:iperf3.tar' for 'iperf3'. Use 'show app-hosting list' for progress.


Device# show app-hosting list
App id                                   State
---------------------------------------------------------
iperf3                                   DEPLOYED

Switch#app-hosting activate appid iperf3
iperf3 activated successfully
Current state is: ACTIVATED
Switch#
Switch#show app-hosting list
App id                                   State
---------------------------------------------------------
iperf3                                   ACTIVATED

Switch#app-hosting start appid iperf3
iperf3 started successfully
Current state is: RUNNING
Switch#show app-hosting list
App id                                   State
---------------------------------------------------------
iperf3                                   RUNNING

Device#
```

Example: Starting an Application

This example shows how to start an application.

```
Device> enable
Device# configure terminal
Device(config)# app-hosting appid iox_app
Device(config-app-hosting)# start
Device(config-app-hosting)# end
```

Example: Lifecycle for an Application

This example shows how to install and uninstall an application:

```
Device> enable
Device# app-hosting install appid iox_app package usbflash1:my_iox_app.tar.tar
Device# app-hosting activate appid iox_app
Device# app-hosting start appid iox_app
Device# app-hosting stop appid iox_app
Device# app-hosting deactivate appid iox_app
Device# app-hosting uninstall appid iox_app
```

Example: Configuring Docker Run Time Options

This example shows how to configure Docker run time options.

```
Device> enable
Device# configure terminal
Device(config)# app-hosting appid iox_app
Device(config-app-hosting)# app-resource docker
Device(config-app-hosting-docker)# run-opts 1 "-v $(APP_DATA):/data"
Device(config-app-hosting-docker)# run-opts 3 "--entrypoint '/bin/sleep 1000000'"
Device(config-app-hosting-docker)# end
```

Example: Configuring a Static IP Address in a Container

This example shows how to configure a static IP address in a container.

```
Device> enable
Device# configure terminal
Device(config)# app-hosting appid iox_app
Device(config-app-hosting)# name-server0 10.2.2.2
Device(config-app-hosting)# app-vnic management guest-interface 0
Device(config-app-hosting-mgmt-gateway)# guest-ipaddress 172.19.0.24 netmask 255.255.255.0
Device(config-app-hosting-mgmt-gateway)# exit
Device(config-app-hosting)# app-default-gateway 172.19.0.23 guest-interface 0
Device(config-app-hosting)# end
```

Example: Configuring Application Hosting on the Management Port

This example shows how to manually configure the IP address for an application.

```
Device# configure terminal
Device(config)# interface gigabitethernet 0/0
Device(config-if)# vrf forwarding Mgmt-vrf
Device(config-if)# ip address 198.51.100.1 255.255.255.254
Device(config-if)# exit
Device(config)# app-hosting appid iox_app
Device(config-app-hosting)# app-vnic management guest-interface 1
Device(config-app-hosting-mgmt-gateway)# end
```

Example: Overriding App Resource Configuration

This example shows how to override an app resource configuration.

```
Device# configure terminal
Device(config)# app-hosting appid iox_app
Device(config-app-hosting)# app-resource profile custom
Device(config-app-resource-profile-custom)# cpu 7400
Device(config-app-resource-profile-custom)# memory 2048
Device(config-app-resource-profile-custom)# vcpu 2
Device(config-app-resource-profile-custom)# end
```

Example: Configuring ERSPAN Support on an AppGigabitEthernet Port

These examples show how to configure ERSPAN on an AppGigabitEthernet port.

Example: Configuring an ERSPAN Source Session

This example shows how to configure an ERSPAN source session:

```
Device> enable
Device# configure terminal
Device(config)# monitor session 2 type erspan-source
Device(config-mon-erspan-src)# source interface gigabitethernet 1/0/3
Device(config-mon-erspan-src)# no shutdown
Device(config-mon-erspan-src-dst)# ip address 10.1.1.5
Device(config-mon-erspan-src-dst)# origin ip address 10.1.1.2
Device(config-mon-erspan-src-dst)# erspan-id 5
Device(config-mon-erspan-src-dst)# end
```

Examples: Configuring ERSPAN Through an AppGigabitEthernet Interface

This example shows how to configure ERSPAN through an AppGigabitEthernet interface:

Note Layer 3 port used for ERSPAN traffic:

```
Device> enable
Device# configure terminal
Device(config)# vtp mode off
Device(config)# vlan 2508
Device(config-vlan)# exit
Device(config)# interface vlan 2508
Device(config-if)# ip address 192.0.2.1 255.255.255.252
Device(config-if)# no shutdown
Device(config-if)# exit
Device(config)# interface AppGigabitEthernet 1/1
Device(config-if)# no switchport mode
Device(config-if)# ip address 10.1.1.2 255.255.255.0
Device(config-if)# end
```

This example shows a Layer 2 port used for ERSPAN traffic:

```
Device> enable
Device# configure terminal
Device(config)# vtp mode off
Device(config)# vlan 2508
Device(config-vlan)# exit
Device(config)# interface vlan 2508
Device(config-if)# ip address 192.0.2.1 255.255.255.252
Device(config-if)# no shutdown
Device(config-if)# exit
Device(config)# interface AppGigabitEthernet 1/1
Device(config-if)# switchport mode trunk
Device(config-if)# end
```

Example: Enabling Multicast Routing on the AppGigabitEthernet Interface

```
Device> enable
Device# configure terminal
Device(config)# app-hosting appid iox_app
Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk
Device(config-config-app-hosting-trunk)# vlan 10 guest-interface 2
Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.0.2 netmask
255.255.255.0
Device(config-config-app-hosting-vlan-access-ip)# multicast
Device(config-config-app-hosting-vlan-access-ip)# exit
Device(config-config-app-hosting-trunk)# exit
Device(config-app-hosting)# app-default-gateway 172.19.0.23 guest-interface 0
Device(config-app-hosting)# end
```

Additional References

Related Documents

Related Topic	Document Title
Programmability commands	*Programmability Command Reference*
DevNet	https://developer.cisco.com/docs/app-hosting/
M2 SATA on Cisco Catalyst 9400 Series Switches	*M2 SATA Module*

Related Topic	Document Title
M2 SATA on Cisco Catalyst 9500-High Performance Series Switches	*M2 SATA Module*
M2 SATA on Cisco Catalyst 9600 Series Switches	*M2 SATA Module*
USB3.0 SSD on Cisco Catalyst 9300 Series Switches	*Configuring USB 3.0 SSD*
USB3.0 SSD on Cisco Catalyst 9500 Series Switches	*Configuring USB 3.0 SSD*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for Application Hosting

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

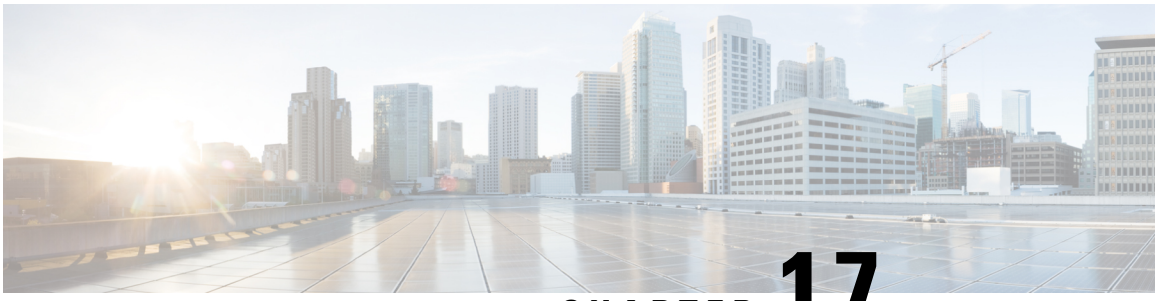
Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 50: Feature Information for Application Hosting

Feature Name	Release	Feature Information
Application Hosting	Cisco IOS XE Gibraltar 16.12.1 Cisco IOS XE Amsterdam 17.1.1 Cisco IOS XE Amsterdam 17.2.1 Cisco IOS XE Bengaluru 17.5.1 Cisco IOS XE Cupertino 17.7.1	A hosted application is a software as a service (SaaS) solution, and users can execute and operate this solution entirely from the cloud. This module describes the Application Hosting feature and how to enable it. • In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on Cisco Catalyst 9300 Series Switches. • In Cisco IOS XE Amsterdam 17.1.1, this feature was implemented on Cisco Catalyst 9400 Series Switches. • In Cisco IOS XE Amsterdam 17.2.1, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches, and Cisco Catalyst 9600 Series Switches. • In Cisco IOS XE Bengaluru 17.5.1, this feature was implemented on Cisco Catalyst 9410 Series Switches. • In Cisco IOS XE Cupertino 17.7.1, this feature was implemented on Cisco Catalyst 9500X Series Switches.
Application Hosting: Autotransfer and Auto-Install of Apps from Internal Flash to SSD	Cisco IOS XE Bengaluru 17.6.1	When IOx is restarted and a different media is selected, all applications must be migrated to the new media, and containers must be restored to the same state as before the change. In Cisco IOS XE Bengaluru 17.6.1, this feature was introduced on the following platforms: • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches

Feature Name	Release	Feature Information
Application Hosting: Front-Panel Network Port Access	Cisco IOS XE Gibraltar 16.12.1 Cisco IOS XE Amsterdam 17.1.1	Introduces datapath connectivity between the Application Hosting container and the front-panel network ports. Also enables ZTP functionality on the front-panel network. • In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on Cisco Catalyst 9300 Series Switches. • In Cisco IOS XE Amsterdam 17.1.1, this feature was implemented on Cisco Catalyst 9400 Series Switches.
Application Hosting: Front-Panel USB Port Access	Cisco IOS XE Gibraltar 16.12.1 Cisco IOS XE Amsterdam 17.1.1	Introduces datapath connectivity between the Application Hosting container and the front-panel USB port. • In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on Cisco Catalyst 9300 Series Switches. • In Cisco IOS XE Amsterdam 17.1.1, this feature was implemented on Cisco Catalyst 9400 Series Switches.
ERSPAN Support on the AppGigabitEthernet Port	Cisco IOS XE Dublin 17.10.1	ERSPAN support on the AppGigabitEthernet port enables the mirroring of data traffic from the device to an application that runs on the AppGigabitEthernet port by using Cisco IOx.
Multicast Routing on the AppGigabitEthernet Port	Cisco IOS XE Dublin 17.11.1	Multicast traffic forwarding is supported on the AppGigabitEthernet interface. Applications can select thenetworks that allow multicast traffic. In Cisco IOS XE Dublin 17.11.1, this feature was introduced on the following platforms: • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 and 9400X Series Switches • Cisco Catalyst 9500-High Performance Series Switches • Cisco Catalyst 9600 and 9600X Series Switches

Feature Name	Release	Feature Information
Native Docker Container: Application Auto-Restart	Cisco IOS XE Amsterdam 17.2.1 Cisco IOS XE Bengaluru 17.5.1	The Application Auto-Restart feature helps applications deployed on platforms to retain the last configured operational state in the event of a system switchover or restart. This feature is enabled by default, and cannot be disabled by users. • In Cisco IOS XE Amsterdam 17.2.1, this feature was implemented on Cisco Catalyst 9300 Series Switches. • In Cisco IOS XE Bengaluru 17.5.1, this feature was implemented on Cisco Catalyst 9410 Series Switches.



CHAPTER 17

ThousandEyes Enterprise Agent

ThousandEyes Enterprise Agent is an enterprise network-monitoring tool that provides you an end-to-end view across networks and services that impact your business. This module describes how to download and install the Enterprise Agent.

- Prerequisites for the ThousandEyes Enterprise Agent, on page 441
- Information About ThousandEyes Enterprise Agent, on page 442
- How to Install the ThousandEyes Enterprise Agent, on page 445
- Configuration Examples for ThousandEyes Enterprise Agent, on page 450
- Additional References, on page 454
- Feature Information for ThousandEyes Enterprise Agent, on page 454

Prerequisites for the ThousandEyes Enterprise Agent

- The ThousandEyes Enterprise Agent image available at the ThousandEyes site must be signed by the same certificate authority (CA) that is used by www.cisco.com for HTTPS downloads; without an username or a password.
- Installation of the Enterprise Agent requires Internet connectivity, or a proxy server. For more information, see the *ThousandEyes documentation* at: https://docs.thousandeyes.com/product-documentation/enterprise-agents.
- The Enterprise Agent application can only be used after the user's license privileges are validated.
- Only Docker-based applications are supported.
- 1:1 stack mode is a must for ThousandEyes Stateful Switchover (SSO) support.

  1:1 mode is when the active and standby roles are assigned to specific devices in a stack. This overrides the traditional N+1 role selection algorithm, where any device in the stack can be the active or the standby.

Information About ThousandEyes Enterprise Agent

ThousandEyes Enterprise Agent Overview

ThousandEyes Enterprise Agent is an enterprise network-monitoring tool that provides you an end-to-end view across networks and services that impact your business. It monitors the network traffic paths across internal, external, carrier, and Internet networks in real time, to provide network performance data. Enterprise Agents are commonly installed in branch sites and data centers to provide a detailed understanding of WAN and Internet connectivity.

In previous Cisco IOS XE releases, ThousandEyes was supported as a third-party Kernel-based Virtual Machine (KVM) appliance on the SSD.

In Cisco IOS XE Amsterdam 17.3.3, a new version of the ThousandEyes Enterprise Agent, Version 3.0 is introduced. This is an embedded Docker-based application that runs on Cisco devices using the application-hosting capability. The Enterprise Agent is available on both the SSD and bootflash, and it supports all tests except browser tests (page load and transaction). The browser tests are available in Cisco IOS XE Bengaluru 17.6.1 and later releases with Enterprise Agent Version 4.0.

The ThousandEyes Enterprise Agent provides the following:

- Benchmarking the performance of networks and applications.
- Detailed hop-by-hop metrics.
- End-to-end path visualization from branch or campus to data center or cloud.
- Outage detection and resolution.
- User-experience analysis.
- Visualization of the traffic-flow pattern.

ThousandEyes Enterprise Agent Version 4.0 available in Cisco IOS XE Bengaluru 17.6.1, supports the following additional features that are not available in the ThousandEyes Agent Version 3.0:

- BrowserBot support when back-panel SSD is available.
- DNAC app icon and description.
- Docker health monitoring.
- The **app-hosting upgrade URL** command to upgrade the ThousandEyes Enterprise Agent.

Resources Required for the ThousandEyes Enterprise Agent

This table describes the required resources for installing the ThousandEyes Enterprise Agent:

Table 51: Resources Required for the ThousandEyes Enterprise Agent

App Media	Maximum Resource	Supported Release
SSD **Note** Only 120G SSD is supported.	• CPU: 2 vCPU • Memory: 2G RAM • Storage: No limit on SSD	Cisco IOS XE Amsterdam 17.3.3 • Cisco Catalyst 9300 and 9300L Series Switches Cisco IOS XE Bengaluru 17.5.1 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Bengaluru 17.6.1 • Cisco Catalyst 9300X Series Switches
Flash	• CPU: 2 vCPU • Memory: 2G RAM • Storage: 1G for persistent logging by applications, out of the 4G partition in the flash file system. The storage is shared with the IOx metadata.	Cisco IOS XE Amsterdam 17.3.3 • Cisco Catalyst 9300 and 9300L Series Switches Cisco IOS XE Bengaluru 17.5.1 • Cisco Catalyst 9400 Series Switches Cisco IOS XE Bengaluru 17.6.1 • Cisco Catalyst 9300X Series Switches

In Cisco IOS XE Bengaluru 17.6.1, add-on mode is supported on Cisco Catalyst 9300, 9300L, and 9300X Series Switches, and Cisco Catalyst 9400 Series Switches.

ThousandEyes Enterprise Agent Download

BrownField and GreenField are two types of ThousandEyes Enterprise Agents. For existing devices, you can download the Brownfield version from the ThousandEyes website. However, new devices are shipped with the Greenfield application loaded in the bootflash.

This table lists the download options available for the agents.

Table 52: ThousandEyes Enterprise Agent Download Options

BrownField	GreenField
• Download the file from the *Installing Enterprise Agents on Cisco Switches with Docker* page. The file is signed by the same certificate authority (CA) that is used by www.cisco.com for HTTPS downloads; without an username or a password. • Use the **install** command to download and deploy the application.	• Available in the bootflash under the */apps* folder. Shipped with the device. • Use the **install** command to download and deploy the application.

This section describes the maximum resources required for the agent to run:

- CPU: 2 vCPUs
- Memory: 2G
- Storage: 1G for persistent logging by applications, out of the 4G partition in the flash file system. This storage is shared by the IOx metadata.
- Media storage:
  - 120G SSD for Cisco Catalyst 9300 and Cat9300 L Series Switches in Cisco IOS XE Amsterdam 17.3.3.
  - 240/480/960GB M2-SATA-HDD for Cisco Catalyst 9400 Series Switches in Cisco IOS XE Bengaluru 17.5.1.

After the download of the Enterprise Agent, it initiates a call to create a secure channel to the ThousandEyes cloud-based portal that provides the required application configuration, and gathers application data. The link to the TE portal is https://app.thousandeyes.com.

ThousandEyes BrowserBot

ThousandEyes Enterprise Agent Version 4.0 provides a BrowserBot for transaction scripting test. The BrowserBot is a component of the Enterprise Agent that manages page load and transaction tests. The BrowserBot allows you to enable customized JavaScript tests which mimic the actions of your web browser on the ThousandEyes Cloud Portal. To protect the host operating system from any errant JavaScript operations, the ThousandEyes agent creates sandbox containers to run your JavaScript.

If an unrestricted disk is used by the application, the ThousandEyes agent will dynamically install the BrowserBot package during initialization that permits portal transaction scripting tests to be configured.

Note The BrowserBot support is not available in ThousandEyes Agent Version 3.0.

BrowserBot consumes a large amount of hardware resources. 2GB system memory and 2 VCPU loads are the maximum IOx system memory and CPU load allocated for all IOx apps. To allow multiple apps to concurrently run in the bootflash, lower the default package.yaml BrowserBot resources before activating the agent. Use the **app-resource profile custom** command to override the default package.yaml settings:

- CPU:1850 CPU units (1/4 VCPU)
- Memory: 500MB

For more information on transaction scripting, see the following links:

- https://docs.thousandeyes.com/product-documentation/tests/transaction-scripting-guide
- https://docs.thousandeyes.com/product-documentation/tests/transaction-scripting-reference

For examples of transaction scripting, see https://github.com/thousandeyes/transaction-scripting-examples.

ThousandEyes Agent Upgrade and Downgrade

ThousandEyes Agent Upgrade

The ThousandEyes Enterprise Agent 3.0 available in Cisco IOS XE Amsterdam 17.3.3 and Bengaluru 17.5.1 can be upgraded to Agent 3.0 or Agent 4.0 that is available in Cisco IOS XE Bengaluru17.6.1. Agent 3.0 is operationally restored after an upgrade.

Agent 4.0 is available in Cisco IOS XE Bengaluru 17.6.1, and the agent auto-upgrade updates to the latest Agent 4.0 binary on startup. No upgrade is available for Agent 4.0 at present.

Application upgrades can be done using the following methods:

- ThousandEyes agent auto-upgrade: Happens automatically when an application starts up. The agent binary within the running container is upgraded, but the application package is not upgraded.
- Using the **app-hosting upgrade** command.
- DNAC app upgrades.

ThousandEyes Agent Downgrade

Agent 3.0 available in Cisco IOS XE Amsterdam 17.3.3, Cisco IOS XE Bengaluru 17.5.1, and Cisco IOS XE 17.6.1 cannot be downgraded.

Agent 4.0 available in Cisco IOS XE Bengaluru 17.6.1 can be downgraded to Agent 3.0 available in Cisco IOS XE Bengaluru 17.6.1. No other downgrade is possible.

When downgrading, if the application does not come to the same state as the previous release, deactivate or uninstall the application, and install or restart it.

How to Install the ThousandEyes Enterprise Agent

To install the Enterprise Agent, follow these steps:

1. Configure IOx. For more information, see the "Enabling Ciso IOx" section.
2. Configure AppHosting.
3. Configure the AppGigabitEthernet port.
4. Install the ThousandEyes Enterprise Agent.

Configuring AppHosting for the ThousandEyes Enterprise Agent

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *application-name*
4. **app-vnic AppGigabitEthernet trunk**
5. **vlan** *vlan-ID* **guest-interface** *guest-interface-number*
6. **guest-ip** *ip-address* **netmask** *netmask*
7. **exit**
8. **exit**
9. **app-default-gateway** *ip-address* **guest-interface** *network-interface*
10. **nameserver#** *ip-address*
11. **app-resource docker**
12. **run-opts** *options*
13. **prepend-pkg-opts**
14. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**app-hosting appid** *application-name* **Example:** `Device(config)# app-hosting appid appid 1keyes`	Configures an application and enters application-hosting configuration mode.
Step 4	**app-vnic AppGigabitEthernet trunk** **Example:** `Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk`	Configures a trunk port as the front-panel port for an application, and enters application-hosting trunk-configuration mode.
Step 5	**vlan** *vlan-ID* **guest-interface** *guest-interface-number* **Example:** `Device(config-config-app-hosting-trunk)# vlan 10 guest-interface 2`	Configures a VLAN guest interface and enters application-hosting VLAN-access IP configuration mode.

	Command or Action	Purpose
Step 6	**guest-ip** *ip-address* **netmask** *netmask* **Example:** `Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 172.19.0.24 netmask 255.255.255.0`	Configures a static IP address for the guest interface.
Step 7	**exit** **Example:** `Device(config-config-app-hosting-vlan-access-ip)# exit`	Exits application hosting VLAN-access IP configuration mode and returns to application-hosting trunk-configuration mode.
Step 8	**exit** **Example:** `Device(config-config-app-hosting-trunk)# exit`	Exits application-hosting trunk-configuration mode and returns to application hosting configuration mode.
Step 9	**app-default-gateway** *ip-address* **guest-interface** *network-interface* **Example:** `Device(config-app-hosting)# app-default-gateway 172.19.0.23 guest-interface 0`	Configures the default management gateway.
Step 10	**nameserver#** *ip-address* **Example:** `Device(config-app-hosting)# name-server0 10.2.2.2`	Configures the DNS server.
Step 11	**app-resource docker** **Example:** `Device(config-app-hosting)# app-resource docker`	Enters application-hosting docker-configuration mode to specify application resource updates.
Step 12	**run-opts** *options* **Example:** `Device(config-app-hosting-docker)# run-opts 1 "-e TEAGENT_ACCOUNT_TOKEN=[account-token]"`	Specifies the Docker run time options.
Step 13	**prepend-pkg-opts** **Example:** `Device(config-app-hosting-docker)# prepend-pkg-opts`	Merges the package options with the Docker runtime options. • Any duplicate variable is overwritten.
Step 14	**end** **Example:** `Device(config-app-hosting-docker)# end`	Exits application-hosting docker-configuration mode and returns to privileged EXEC mode.

Configuring AppGigabitEthernet Interface for the ThousandEyes Enterprise Agent

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface appgigabitethernet** *number*
4. **switchport trunk allowed vlan** *vlan-ID*
5. **switchport mode trunk**
6. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface appgigabitethernet** *number* **Example:** `Device(config)# interface AppGigabitEthernet 1/0/1`	Configures the AppGigabitEthernet and enters interface configuration mode. • For stackable switches, the *number* argument is *switch-number/0/1*.
Step 4	**switchport trunk allowed vlan** *vlan-ID* **Example:** `Device(config-if)# switchport trunk allowed vlan 10-12,20`	Configures the list of VLANs allowed on the trunk.
Step 5	**switchport mode trunk** **Example:** `Device(config-if)# switchport mode trunk`	Sets the interface into permanent trunking mode and negotiates to convert the neighboring link into a trunk link.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Installing the ThousandEyes Enterprise Agent

Before you begin

You can install the ThousandEyes Enterprise Agent either from the URL given below or from the flash filesystem.

SUMMARY STEPS

1. **enable**
2. **app-hosting install appid** *application-name* **package** *package-path*
3. **app-hosting start appid** *application-name*
4. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enters privileged EXEC mode.

Step 2	**app-hosting install appid** *application-name* **package** *package-path*
	Example:
	`Device# app-hosting install 1keyes https://downloads.thousandeyes.com/enterprise-agent/thousandeyes-enterprise-agent-3.0.cat9k.tar`
	Or
	`Device# app-hosting install appid 1keyes package flash:/apps/[greenfield-app-tar]`
		Installs an application from the specified location.

Step 3	**app-hosting start appid** *application-name* **Example:** `Device# app-hosting start appid 1keyes`	(Optional) Starts the application.
Step 4	**end** **Example:** `Device# end`	Exits application hosting configuration mode and returns to privileged EXEC mode.

The following is sample output from the **show app-hosting list** command:

```
Device# show app-hosting list

App id                                   State
---------------------------------------------------------
```

```
1keyes                                    RUNNING
```

Configuration Examples for ThousandEyes Enterprise Agent

Example: Installing ThousandEyes Enterprise Agent

This example shows how to:

- Enable IOx.
- Configure AppHosting.
- Configure the AppGigabitEthernet port.
- Install the ThousandEyes Enterprise Agent.

The following example shows how to enable IOx:

```
Device> enable
Device# configure terminal
Device(config)# iox
Device(config)# username cisco privilege 15 password 0 ciscoI
Device(config)# end
```

The following example shows how to configure AppHosting:

```
Device> enable
Device# configure terminal
Device(config)# app-hosting appid appid 1keyes
Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk
Device(config-config-app-hosting-trunk)# vlan 10 guest-interface 2
Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 172.19.0.24
netmask 255.255.255.0
Device(config-config-app-hosting-vlan-access-ip)# exit
Device(config-config-app-hosting-trunk)# exit
Device(config-app-hosting)# app-default-gateway 172.19.0.23
guest-interface 0
Device(config-app-hosting)# name-server0 10.2.2.2
Device(config-app-hosting)# app-resource docker
Device(config-app-hosting-docker)# run-opts 1
"-e TEAGENT_ACCOUNT_TOKEN=[account-token]"
Device(config-app-hosting-docker)# prepend-pkg-opts
Device(config-app-hosting-docker)# end
```

The following example shows how to configure the Appgigabitethernet interface:

```
Device> enable
Device# configure terminal
Device(config)# interface AppGigabitEthernet 1/0/1
Device(config-if)# switchport trunk allowed vlan 10-12,20
Device(config-if)# switchport mode trunk
Device(config-if)# end
```

The following example shows how to install the ThousandEyes Enterprise Agent.

Note You can either download the BrownField application from the ThousandEyes website or install the prepackaged Greenfield application from the flash filesystem.

```
Device> enable
Device# Device# app-hosting install 1keyes https://downloads.thousandeyes.com/
enterprise-agent/thousandeyes-enterprise-agent-3.0.cat9k.tar
OR
Device# app-hosting install appid 1keyes package flash:/apps/[greenfield-app-tar]
Device# app-hosting start appid 1keyes
Device# end
```

Sample Configuration for ThousandEyes Enterprise Agent

The following is sample output from the **show app-hosting detail** command:

```
Device# show app-hosting detail

App id                 : 1keyes
Owner                  : iox
State                  : RUNNING
Application
  Type                 : docker
  Name                 : thousandeyes/enterprise-agent
  Version              : 3.0
  Description          :
  Path                 : flash:thousandeyes-enterprise-agent-3.0.cat9k.tar
  URL Path             :
Activated profile name : custom

Resource reservation
  Memory               : 0 MB
  Disk                 : 1 MB
  CPU                  : 1850 units
  CPU-percent          : 25 %
  VCPU                 : 1

Attached devices
  Type              Name               Alias
  ---------------------------------------------
  serial/shell     iox_console_shell   serial0
  serial/aux       iox_console_aux     serial1
  serial/syslog    iox_syslog          serial2
  serial/trace     iox_trace           serial3

Network interfaces
   ---------------------------------------
eth0:
   MAC address         : 52:54:dd:c0:a2:ab
   IPv4 address        : 10.0.0.110
   IPv6 address        : ::
   Network name        : mgmt-bridge-v14


Docker
------
Run-time information
  Command              :
  Entry-point          : /sbin/my_init
```

```
  Run options in use   : -e TEAGENT_ACCOUNT_TOKEN=TOKEN_NOT_SET --hostname=$(SYSTEM_NAME)
--cap-add=NET_ADMIN
                         --mount type=tmpfs,destination=/var/log/agent,tmpfs-size=140m
                       --mount type=tmpfs,destination=/var/lib/te-agent/data,tmpfs-size=200m

                          -v $(APP_DATA)/data:/var/lib/te-agent -e TEAGENT_PROXY_TYPE=DIRECT

                         -e TEAGENT_PROXY_LOCATION= -e TEAGENT_PROXY_USER= -e
TEAGENT_PROXY_AUTH_TYPE=
                         -e TEAGENT_PROXY_PASS= -e TEAGENT_PROXY_BYPASS_LIST= -e
TEAGENT_KDC_USER=
                         -e TEAGENT_KDC_PASS= -e TEAGENT_KDC_REALM= -e TEAGENT_KDC_HOST=
-e TEAGENT_KDC_PORT=88
                         -e TEAGENT_KERBEROS_WHITELIST= -e TEAGENT_KERBEROS_RDNS=1 -e
PROXY_APT=
                         -e APT_PROXY_USER= -e APT_PROXY_PASS= -e APT_PROXY_LOCATION= -e
TEAGENT_AUTO_UPDATES=1
                         -e TEAGENT_ACCOUNT_TOKEN=r3d29srpebr4j845lvnamwhswlori2xs
                         --hostname=cat9k-9300-usb --memory=1g
  Package run options  : -e TEAGENT_ACCOUNT_TOKEN=TOKEN_NOT_SET --hostname=$(SYSTEM_NAME)
--cap-add=NET_ADMIN
                         --mount type=tmpfs,destination=/var/log/agent,tmpfs-size=140m
                       --mount type=tmpfs,destination=/var/lib/te-agent/data,tmpfs-size=200m

                          -v $(APP_DATA)/data:/var/lib/te-agent -e TEAGENT_PROXY_TYPE=DIRECT

                         -e TEAGENT_PROXY_LOCATION= -e TEAGENT_PROXY_USER= -e
TEAGENT_PROXY_AUTH_TYPE=
                         -e TEAGENT_PROXY_PASS= -e TEAGENT_PROXY_BYPASS_LIST= -e
TEAGENT_KDC_USER=
                         -e TEAGENT_KDC_PASS= -e TEAGENT_KDC_REALM= -e TEAGENT_KDC_HOST=
                         -e TEAGENT_KDC_PORT=88 -e TEAGENT_KERBEROS_WHITELIST= -e
TEAGENT_KERBEROS_RDNS=1
                         -e PROXY_APT= -e APT_PROXY_USER= -e APT_PROXY_PASS= -e
APT_PROXY_LOCATION=
                         -e TEAGENT_AUTO_UPDATES=1
Application health information
  Status               : 0
  Last probe error     :
  Last probe output    :
```

The following sample output from the **show running-configuration** command displays the static IP address configuration:

```
Device# show running-config | section app-hosting

app-hosting appid 1keyes
 app-vnic AppGigabitEthernet trunk
  vlan 14 guest-interface 0
   guest-ipaddress 10.0.0.110 netmask 255.255.255.0
app-default-gateway 10.0.0.1 guest-interface 0
app-resource docker
  prepend-pkg-opts
  run-opts 1 "-e TEAGENT_ACCOUNT_TOKEN=r3d29srpebr4j845lvnamwhswlori2xs"
  run-opts 2 "--hostname=cat9k-9300-usb --memory=1g"
name-server0 10.0.0.1
start
```

The following sample output from the **show running-configuration** command displays the static IP address configuration and the proxy server information:

```
Device# show running-config | section app-hosting

app-hosting appid 1keyes
 app-vnic AppGigabitEthernet trunk
  vlan 14 guest-interface 0
   guest-ipaddress 172.27.0.137 netmask 255.240.0.0
 app-default-gateway 172.27.0.129 guest-interface 0
 app-resource docker
  run-opts 1 "-e TEAGENT_ACCOUNT_TOKEN=r3d29srpebr4j845lvnamwhswlori2xs"
 run-opts 3 "-e TEAGENT_PROXY_TYPE=STATIC"
 run-opts 4 "-e TEAGENT_PROXY_LOCATION='proxy-wsa.esl.cisco.com:80'"
 prepend-pkg-opts
 name-server0 172.16.0.2
 start
```

The following is sample output from running the app-resource Docker package merged with the Docker runtime options:

```
// Example of "prepend-package-opts" merging
app-hosting appid TEST
app-vnic management guest-interface 3
app-resource docker
prepend-package-opts !!!
run-opts 1 "--entrypoint '/bin/sleep 1000000'"
run-opts 2 "-e TEST=1 "

# Specify runtime and startup
startup:
runtime_options: "--env MYVAR2=foo --cap-add=NET_ADMIN"

Merged docker run-opts passed to CAF's activation payload:
{"auto_deactivate": false, "resources": {"profile": "custom", "cpu":
"1000", "memory": "1024", "rootfs_size": "0", "vcpu": 1, "disk": 10,"network":
[{"interface-name": "eth3", "network-name": "mgmt-bridge100"}, {"interface-name":
"eth4", "network-type": "vlan", "mode": "static", "ipv4": {"ip": "10.2.0.100",
"prefix": "24", "default": false, "gateway": "" },"network-info": { "vlan-id": "10" },
"mac_forwarding": "no", "mirroring": "no"}, {"interface-name": "eth0",
"network-type": "vlan", "network-info": { "vlan-id": "12" }, "mac_forwarding": "no",
"mirroring": "no"}, {"interface-name": "eth2", "network-type": "vlan", "networkinfo":
{"vlan-id": "22" }, "mac_forwarding": "no", "mirroring": "no"},
{"interface-name
": "eth1", "network-type": "vlan", "network-info": {"vlan-id": "all" },
"mac_forwarding": "no", "mirroring": "no"}]},

"startup":{"runtime_options":"--env MYVAR2=foo --cap-add=NET_ADMIN --
entrypoint'/bin/sleep 1000000' -e TEST=1"}}

// Example of no "prepend-package-opts" which is the current behavior since
16.12 where pkg.yml default runoptions are ignored.
app-hosting appid TEST
app-vnic management guest-interface 3
app-resource docker !!!
run-opts 1 "--entrypoint '/bin/sleep 1000000'"
run-opts 2 "-e TEST=1 "

# Specify runtime and startup
startup:
runtime_options: "--env MYVAR2=foo --cap-add=NET_ADMIN"

Merged docker run-opts passed to CAF's activation payload:
{"auto_deactivate": false, "resources": {"profile": "custom", "cpu":
"1000", "memory": "1024", "rootfs_size": "0", "vcpu": 1, "disk": 10,"network":
[{"interface-name": "eth3", "network-name": "mgmt-bridge100"}, {"interface-name":
"eth4", "network-type": "vlan", "mode": "static", "ipv4": {"ip": "10.2.0.100",
```

```
"prefix": "24", "default": false, "gateway": "" },"network-info": { "vlan-id": "10" },
"mac_forwarding": "no", "mirroring": "no"}, {"interface-name": "eth0",
"network-type": "vlan", "network-info": { "vlan-id": "12" }, "mac_forwarding": "no",
"mirroring": "no"}, {"interface-name": "eth2", "network-type": "vlan", "networkinfo":
{"vlan-id": "22" }, "mac_forwarding": "no", "mirroring": "no"},
{"interface-name": "eth1", "network-type": "vlan", "network-info": {"vlan-id": "all" },
"mac_forwarding": "no", "mirroring": "no"}]},

"startup":{"runtime_options":"--entrypoint '/bin/sleep 1000000' -e
TEST=1"}}


// Config 1 : default behavior when "app-resource docker" is not
configured.
app-hosting appid TEST
app-vnic management guest-interface 3

// Config 2: no docker run-opts specified
app-hosting appid TEST
app-vnic management guest-interface 3
app-resource docker
prepend-package-opts
```

Additional References

Related Topic	Document Title
ThousandEyes URL	https://app.thousandeyes.com

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for ThousandEyes Enterprise Agent

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 53: Feature Information for Application Hosting

Feature Name	Release	Feature Information
ThousandEyes Integration	Cisco IOS XE Amsterdam 17.3.3 Cisco IOS XE Bengaluru 17.5.1 Cisco IOS XE Bengaluru 17.6.1	ThousandEyes is a cloud-ready, enterprise network-monitoring tool that provides an end-to-end view across networks and services. • In Cisco IOS XE Amsterdam 17.3.3, this feature was implemented on Cisco Catalyst 9300 and 9300L Series Switches. • In Cisco IOS XE Bengaluru 17.5.1, this feature was implemented on Cisco Catalyst 9400 Series Switches. • In Cisco IOS XE Bengaluru 17.6.1, this feature was implemented on Cisco Catalyst 9300X Series Switches. **Note** The ThousandEyes Integration feature is not supported in Cisco IOS XE Bengaluru 17.4.x release.
ThousandEyes BrowserBot	Cisco IOS XE Bengaluru 17.6.1	ThousandEyes add-on agent mode is supported. Add-on mode provides a BrowserBot for transaction scripting test. In Cisco IOS XE Bengaluru 17.6.1, this feature was introduced on the following platforms: • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 Series Switches
ThousandEyes 5.0 Support	Cisco IOS XE 17.16.1	From Cisco IOS XE 17.16.1 onwards, all Cisco Catalyst 9000 Series Switches will have ThousandEyes Enterprise Agent 5.0 as the default version. ThousandEyes 5.0 supports the Alpine-Linux based image version that reduces the image size drastically from 1GB to 20MB. This feature was introduced on the following platforms: • Cisco Catalyst 9300, 9300L, and 9300X Series Switches • Cisco Catalyst 9400 Series Switches