



Programmability Configuration Guide, Cisco IOS XE Gibraltar 16.10.x

First Published: 2018-11-15

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



CONTENTS

CHAPTER 1	New and Changed Feature Information	1
------------------	--	----------

PART I	Provisioning	15
---------------	---------------------	-----------

CHAPTER 2	Zero-Touch Provisioning	17
	Information About Zero-Touch Provisioning	17
	Zero-Touch Provisioning Overview	17
	DHCP Server Configuration for Zero-Touch Provisioning	18
	DHCPv6 Support	18
	Sample Zero-Touch Provisioning Configurations	19
	Sample DHCP Server Configuration on a Management Port Using TFTP Copy	19
	Sample DHCP Server Configuration on a Management Port Using HTTP Copy	19
	Sample DHCP Server Configuration on an In-Band Port Using TFTP Copy	19
	Sample DHCP Server Configuration on an In-Band Port Using HTTP Copy	20
	Sample DHCP Server Configuration on a Linux Ubuntu Device	20
	Sample DHCPv6 Server Configuration on a Management Port Using TFTP Copy	21
	Sample Python Provisioning Script	21
	Boot Log for Cisco 4000 Series Integrated Services Routers	22
	Boot Log for Cisco Catalyst 9000 Series Switches	23
	Feature Information for Zero-Touch Provisioning	37

CHAPTER 3	iPXE	39
	Information About iPXE	39
	About iPXE	39
	iPXE Overview	40
	IPv6 iPXE Network Boot	42

IPv6 Address Assignment in Rommon Mode	44
Supported ROMMON Variables	45
iPXE-Supported DHCP Options	45
DHCPv6 Unique Identifiers	47
How to Configure iPXE	47
Configuring iPXE	47
Configuring Device Boot	48
Configuration Examples for iPXE	49
Example: iPXE Configuration	49
Sample iPXE Boot Logs	50
Sample DHCPv6 Server Configuration for iPXE	50
Troubleshooting Tips for iPXE	52
Additional References for iPXE	53
Feature Information for iPXE	53

PART II
Shells and Scripting 55

CHAPTER 4
Guest Shell 57

Information About the Guest Shell	57
Guest Shell Overview	57
Guest Shell Vs Guest Shell Lite	58
Guest Shell Security	58
Hardware Requirements for the Guest Shell	58
Guest Shell Storage Requirements	59
Accessing Guest Shell on a Device	60
Accessing Guest Shell Through the Management Port	60
Stacking with Guest Shell	61
IOx Overview	61
IOx Tracing and Logging Overview	61
IOXMAN Structure	61
Logging and Tracing System Flow	62
Logging and Tracing of Messages	64
Example: Guest Shell Networking Configuration	65
How to Enable the Guest Shell	65

Managing IOx	65
Managing the Guest Shell	67
Enabling and Running the Guest Shell	68
Disabling and Destroying the Guest Shell	69
Managing the Guest Shell Using Application Hosting	69
Accessing the Python Interpreter	70
Configuration Examples for the Guest Shell	71
Example: Managing the Guest Shell	71
Sample VirtualPortGroup Configuration	72
Example: Guest Shell Usage	73
Example: Guest Shell Networking Configuration	73
Sample DNS Configuration for Guest Shell	74
Example: Configuring Proxy Environment Variables	74
Example: Configuring Yum and PIP for Proxy Settings	74
Additional References for Guest Shell	75
Feature Information for Guest Shell	76

CHAPTER 5**Python API** 79

Using Python	79
Cisco Python Module	79
Cisco Python Module to Execute IOS CLI Commands	81

CHAPTER 6**CLI Python Module** 85

Information About Python CLI Module	85
About Python	85
Python Scripts Overview	85
Interactive Python Prompt	85
Python Script	86
Supported Python Versions	87
Updating the Cisco CLI Python Module	88
Additional References for the CLI Python Module	89
Feature Information for the CLI Python Module	89

CHAPTER 7**EEM Python Module** 91

Prerequisites for the EEM Python Module	91
Information About EEM Python Module	91
Python Scripting in EEM	91
EEM Python Package	91
Python-Supported EEM Actions	92
EEM Variables	93
EEM CLI Library Command Extensions	93
How to Configure the EEM Python Policy	94
Registering a Python Policy	94
Running Python Scripts as Part of EEM Applet Actions	95
Adding a Python Script in an EEM Applet	97
Additional References EEM Python Module	99
Feature Information for EEM Python Module	100

PART III**Model-Driven Programmability 103****CHAPTER 8****NETCONF Protocol 105**

Information About the NETCONF Protocol	105
Introduction to Data Models - Programmatic and Standards-Based Configuration	105
NETCONF	106
NETCONF RESTCONF IPv6 Support	106
NETCONF Global Session Lock	106
NETCONF Kill Session	107
Candidate Config Support	107
NETCONF Operations on Candidate	108
Candidate Support Configuration	109
How to Configure the NETCONF Protocol	110
Providing Privilege Access to Use NETCONF	110
Configuring NETCONF-YANG	112
Configuring NETCONF Options	113
Configuring SNMP	113
Verifying the NETCONF Protocol Configuration	114
Additional References for NETCONF Protocol	117
Feature Information for NETCONF Protocol	117

CHAPTER 9	RESTCONF Protocol	123
	Prerequisites for the RESTCONF Protocol	123
	Restrictions for the RESTCONF Protocol	123
	Information About RESTCONF Programmable Interface	124
	Overview of RESTCONF	124
	HTTPs Methods	124
	RESTCONF Root Resource	124
	RESTCONF API Resource	125
	Methods	126
	How to Configure RESTCONF Programmable Interface	126
	Authentication of NETCONF/RESTCONF Using AAA	126
	Enabling Cisco IOS HTTP Services for RESTCONF	128
	Verifying RESTCONF Configuration	129
	Configuration Examples for RESTCONF Programmable Interface	131
	Example: Configuring the RESTCONF Protocol	131
	Additional References for the RESTCONF Protocol	134
	Feature Information for the RESTCONF Protocol	135

CHAPTER 10	gNMI Protocol	137
	Restrictions for the gNMI Protocol	137
	Information About the gNMI Protocol	138
	About GNMI	138
	JSON IETF Encoding for YANG Data Trees	138
	gNMI GET Request	139
	gNMI SetRequest	144
	gNMI Namespace	146
	gNMI Wildcards	147
	gNMI Error Messages	150
	How to Enable the gNMI Protocol	151
	Creating Certs with OpenSSL on Linux	151
	Installing Certs on a Device	151
	Enabling gNMI in Insecure Mode	152
	Enabling gNMI in Secure Mode	154

Connecting the gNMI Client 155

Configuration Examples for the gNMI Protocol 156

 Example: Enabling the gNMI Protocol 156

Additional References for the gNMI Protocol 157

Feature Information for the gNMI Protocol 157

CHAPTER 11

Model Based AAA 159

Model Based AAA 159

 Prerequisites for Model Based AAA 159

 Initial Operation 159

 Group Membership 160

 NACM Privilege Level Dependencies 161

 NACM Configuration Management and Persistence 161

 Resetting the NACM Configuration 161

 Sample NACM Configuration 161

Additional References for Model Based AAA 165

Feature Information for Model-Based AAA 165

CHAPTER 12

Model-Driven Telemetry 167

Model-Driven Telemetry 167

 Prerequisites for Model-Driven Telemetry 167

 Restrictions for Model-Driven Telemetry 170

 Information About Model-Driven Telemetry 171

 Model-Driven Telemetry Overview 171

 Telemetry Roles 171

 Subscription Overview 171

 Subscription Monitoring 178

 Streams 180

 Transport Protocol 183

 High Availability in Telemetry 184

Sample Model-Driven Telemetry RPCs 184

 Managing Configured Subscriptions 184

 Receiving a Response Code 187

 Receiving Subscription Push-Updates 187

Retrieving Subscription Details	188
Additional References for Model-Driven Telemetry	190
Feature Information for Model-Driven Telemetry	191

CHAPTER 13
In-Service Model Update 195

Restrictions for In-Service Model Update	195
Information About In-Service Model Update	195
Overview of In-Service Model Updates	195
Compatibility of In-Service Model Update Packages	195
Update Package Naming Conventions	196
Installing the Update Package	196
Deactivating the Update Package	197
Rollback of the Update Package	197
How to Manage In-Service Model Update	198
Managing the Update Package	198
Configuration Examples for In-Service Model Updates	199
Example: Managing an Update Package	199
Feature Information for In-Service Model Update	203

PART IV
OpenFlow 205

CHAPTER 14
OpenFlow 207

Prerequisites for OpenFlow	207
Restrictions for OpenFlow	207
Information About OpenFlow	208
OpenFlow Overview	208
OpenFlow Controller	208
Flow Management	208
OpenFlow Pipeline	209
Supported Match Fields and Actions	209
Flow Operations	211
OpenFlow Table Pipeline	211
How to Configure OpenFlow	212
Enabling OpenFlow Mode on a Device	212

Configuring OpenFlow	213
Configuring Interfaces in OpenFlow Mode	215
Configuration Examples for OpenFlow	216
Example: Enabling OpenFlow on a Device	216
Example: Configuring OpenFlow	216
Example: Configuring Interfaces in OpenFlow Mode	217
Verifying OpenFlow	217
Additional References for OpenFlow	220
Feature Information for OpenFlow	220



CHAPTER 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	
Zero-Touch Provisioning	<p>Cisco IOS XE Everest 16.5.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>Cisco IOS XE Everest 16.5.1b</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers <p>Cisco IOS XE Everest 16.6.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9400 Series Switches <p>Cisco IOS XE Fuji 16.7.1</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) <p>Cisco IOS XE Fuji 16.8.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches <p>Cisco IOS XE Fuji 16.8.2</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Aggregation Services Routers (ASR1004, ASR1006, ASR1006-X, ASR1009-X, ASR1013)

Feature	Release & Platform
Zero Touch Provisioning: HTTP Download	Cisco IOS XE Fuji 16.8.1 <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches
DHCPv6 Support for Zero-Touch Provisioning	Cisco IOS XE Fuji 16.9.1 <ul style="list-style-type: none"> • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches
iPXE	Cisco IOS XE Denali 16.3.2 and Cisco IOS XE Everest 16.5.1a <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3650 Series Switches Cisco IOS XE Everest 16.6.1 <ul style="list-style-type: none"> • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.6.2 <ul style="list-style-type: none"> • Cisco Catalyst 9400 Series Switches Cisco IOS XE Fuji 16.8.1a <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches Cisco IOS XE Fuji 16.9.2 <ul style="list-style-type: none"> • Cisco Catalyst 9200 Series Switches
Shells and Scripting	

Feature	Release & Platform
Guest Shell	<p>Cisco IOS XE Everest 16.5.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>Cisco IOS XE Everest 16.5.1b</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers <p>Cisco IOS XE Everest 16.6.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9400 Series Switches <p>Cisco IOS XE Fuji 16.7.1</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) • Cisco Cloud Services Router 1000V Series <p>Cisco IOS XE Fuji 16.8.1</p> <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches <p>Cisco IOS XE Fuji 16.9.1</p> <ul style="list-style-type: none"> • Cisco 1000 Series Integrated Services Routers

Feature	Release & Platform
Python APIs	<p>Cisco IOS XE Everest 16.5.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>Cisco IOS XE Everest 16.5.1b</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers <p>Cisco IOS XE Everest 16.6.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9400 Series Switches <p>Cisco IOS XE Fuji 16.7.1</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) • Cisco Cloud Services Router 1000V Series <p>Cisco IOS XE Fuji 16.8.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches

Feature	Release & Platform
Python CLI Module	<p>Cisco IOS XE Everest 16.5.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>Cisco IOS XE Everest 16.5.1b</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers <p>Cisco IOS XE Everest 16.6.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9400 Series Switches <p>Cisco IOS XE Fuji 16.7.1</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) • Cisco Cloud Services Router 1000V Series <p>Cisco IOS XE Fuji 16.8.1</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Router models with a minimum of 4 GB RAM. • Cisco ASR 1000 Aggregation Services Routers (ASR1004, ASR1006, ASR1006-X, ASR1009-X, ASR1013) <p>Cisco IOS XE Fuji 16.8.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches

Feature	Release & Platform
EEM Python Module	<p>Cisco IOS XE Everest 16.5.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>Cisco IOS XE Everest 16.5.1b</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers <p>Cisco IOS XE Everest 16.6.2.</p> <ul style="list-style-type: none"> • Cisco Catalyst 9400 Series Switches <p>Cisco IOS XE Fuji 16.7.1</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) • Cisco Cloud Services Router 1000V Series <p>Cisco IOS XE Fuji 16.8.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches
Model-Driven Programmability	

Feature	Release & Platform
NETCONF Protocol	<p>Cisco IOS XE Denali 16.3.1</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco Cloud Services Router 1000V Series <p>Cisco IOS XE Everest 16.5.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches <p>Cisco IOS XE Everest 16.6.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches <p>Cisco IOS XE Fuji 16.8.1a</p> <ul style="list-style-type: none"> • 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 Converged Broadband Router • Cisco Network Convergence System 4200 Series <p>Cisco IOS XE Fuji 16.9.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs <p>Cisco IOS XE Gibraltar 16.10.1</p> <ul style="list-style-type: none"> • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco Network Convergence System 520 Series

Feature	Release & Platform
NETCONF and RESTCONF IPv6 Support	Cisco IOS XE Fuji 16.8.1a <ul style="list-style-type: none"> • 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
NETCONF Global Lock and Kill Session	Cisco IOS XE Fuji 16.8.1a <ul style="list-style-type: none"> • 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
RESTCONF Protocol	<p>Cisco IOS XE Everest 16.6.1</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Router • Cisco ASR 1000 Aggregation Services Routers • Cisco Cloud Services Router 1000V Series <p>Cisco IOS XE Fuji 16.8.1a</p> <ul style="list-style-type: none"> • 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 <p>Cisco IOS XE Fuji 16.9.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9200 Series Switches
gNMI Protocol	<p>Cisco IOS XE Fuji 16.8.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches <p>Cisco IOS XE Gibraltar 16.10.1</p> <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches

Feature	Release & Platform
Model-Based AAA	<p data-bbox="776 289 1052 321">Cisco IOS XE Fuji 16.8.1</p> <ul data-bbox="813 338 1406 678" style="list-style-type: none"><li data-bbox="813 338 1321 369">• Cisco 1000 Series Integrated Services Routers<li data-bbox="813 390 1321 422">• Cisco 4000 Series Integrated Services Routers<li data-bbox="813 443 1406 474">• Cisco ASR 1000 Series Aggregation Services Routers<li data-bbox="813 495 1393 527">• Cisco ASR 900 Series Aggregation Services Routers<li data-bbox="813 548 1393 579">• Cisco ASR 920 Series Aggregation Services Routers<li data-bbox="813 600 1284 632">• Cisco Cloud Services Router 1000v Series<li data-bbox="813 653 1284 684">• Cisco Network Convergence System 4200 <p data-bbox="776 709 1065 741">Cisco IOS XE Fuji 16.8.1a</p> <ul data-bbox="813 758 1219 993" style="list-style-type: none"><li data-bbox="813 758 1219 789">• Cisco Catalyst 3650 Series Switches<li data-bbox="813 810 1219 842">• Cisco Catalyst 3850 Series Switches<li data-bbox="813 863 1219 894">• Cisco Catalyst 9300 Series Switches<li data-bbox="813 915 1219 947">• Cisco Catalyst 9400 Series Switches<li data-bbox="813 968 1219 999">• Cisco Catalyst 9500 Series Switches

Feature	Release & Platform
Model-Driven Telemetry NETCONF Dial-In	<p>Cisco IOS XE Everest 16.6.1</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>Cisco IOS XE Everest 16.6.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9400 Series Switches <p>Cisco IOS XE Fuji 16.7.1</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Aggregation Services Routers (ASR1001-HX, ASR1001-X, ASR1002-HX, ASR1002-X) <p>Cisco IOS XE Fuji 16.8.1</p> <ul style="list-style-type: none"> • Cisco 1000 Series Integrated Services Routers • Cisco ASR 1000 RP2 and RP3 Series Aggregation Services Routers <p>Cisco IOS XE Fuji 16.8.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches <p>Cisco IOS XE Fuji 16.9.1</p> <ul style="list-style-type: none"> • 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 <p>Cisco IOS XE Fuji 16.9.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs <p>Cisco IOS XE Gibraltar 16.10.1</p> <ul style="list-style-type: none"> • Cisco Cloud Services Router 1000v • Cisco Network Convergence System 520 Series

Feature	Release & Platform
Model-Driven Telemetry gRPC Dial-Out	<p>Cisco IOS XE Gibraltar 16.10.1</p> <ul style="list-style-type: none"> • 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
In-Service Model Updates	<p>Cisco IOS XE Everest 16.5.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>Cisco IOS XE Everest 16.5.1b</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers <p>Cisco IOS XE Everest 16.6.1</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches <p>Cisco IOS XE Everest 16.6.2</p> <ul style="list-style-type: none"> • Cisco Catalyst 9400 Series Switches <p>Cisco IOS XE Fuji 16.7.1</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Aggregation Services Routers <p>Cisco IOS XE Fuji 16.8.1a</p> <ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches.

Feature	Release & Platform
OpenFlow	
OpenFlow	Cisco IOS XE Fuji 16.9.1 <ul style="list-style-type: none">• Catalyst 9300 Series Switches• Catalyst 9400 Series Switches• Catalyst 9500 Series Switches• Catalyst 9500 Series High Performance Switches



PART I

Provisioning

- [Zero-Touch Provisioning, on page 17](#)
- [iPXE, on page 39](#)



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.

- [Information About Zero-Touch Provisioning, on page 17](#)
- [Sample Zero-Touch Provisioning Configurations, on page 19](#)
- [Feature Information for Zero-Touch Provisioning, on page 37](#)

Information About Zero-Touch Provisioning

Zero-Touch Provisioning Overview

Zero-Touch Provisioning provides open bootstrap interfaces to automate network device provisioning in heterogeneous network environments.

When a device that supports Zero-Touch Provisioning boots up, and does not find the startup configuration (during initial installation), the device enters the Zero-Touch Provisioning mode. The device searches for a Dynamic Host Control Protocol (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/TFTP server, and downloads the Python script from an HTTP/TFTP server to configure the device.

Guest Shell provides the environment for the Python script to run. Guest Shell executes 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 In case Zero-Touch Provisioning fails, the device falls back to AutoInstall to load configuration files. For more information, see [Using AutoInstall and Setup](#).

DHCP Server Configuration for Zero-Touch Provisioning

In Zero-Touch Provisioning, a DHCP server must be running on the same network as the new device that is being provisioned. Zero-Touch Provisioning 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/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* and *Python CLI Module* chapters.

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

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

After receiving these DHCP options, the device connects to the HTTP/TFTP server, and downloads the Python script. The device, at this point does not have any route to reach the HTTP/TFTP server, so it uses the default route provided by the DHCP server.

DHCPv6 Support

In Cisco IOS XE Fuji 16.9.1, Dynamic Host Control Protocol Version 6 (DHCPv6) 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 the HTTP or TFTP download of Python scripts fail, the device will revert to the start (without any configuration). For both DHCPv4, and DHCPv6 to work, the correct HTTP 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 the 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 "xxxxxxxxxxxxxxxx";
    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";
```

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 connected via 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 connected via 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 connected via 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 connected via 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 displays that the server is either connected to the management port or in-band port on a device, and 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";
  }
}

```

```
}
```

Once the DHCP server is running, boot a management-network connected device, and the rest of the configuration is automatic.

Sample DHCPv6 Server Configuration on a Management Port Using TFTP Copy

The following is a sample DHCPv6 server configuration using TFTP copy, when connected via 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 can be used from either an HTTP 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.execute(cli_command)

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

print "\n\n *** Configuring a Loopback Interface *** \n\n"
cli.configure(["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.execute(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 executes the downloaded Python script and configures 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
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 IOS prompt is accessible.

Boot Log for Cisco Catalyst 9000 Series Switches

The following sections displays 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

This section displays 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 section 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/wl/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 Lsmpl18/3, 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 Lsmpil8/3, 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 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

*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

Switch#	Role	Priority	Current 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:.
OK 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

This section displays 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
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 section 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

*** 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

Switch#	Role	Priority	Current 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	Type	Technology-package
Current		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
Tel1/0/37         unassigned      YES unset  down      down
Tel1/0/38         unassigned      YES unset  down      down
Tel1/0/39         unassigned      YES unset  down      down
Tel1/0/40         unassigned      YES unset  down      down
Tel1/0/41         unassigned      YES unset  down      down
Tel1/0/42         unassigned      YES unset  down      down
Tel1/0/43         unassigned      YES unset  down      down
Tel1/0/44         unassigned      YES unset  down      down
Tel1/0/45         unassigned      YES unset  down      down
Tel1/0/46         unassigned      YES unset  down      down
Tel1/0/47         unassigned      YES unset  down      down
Tel1/0/48         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
Tel1/1/1          unassigned      YES unset  down      down
Tel1/1/2          unassigned      YES unset  down      down
Tel1/1/3          unassigned      YES unset  down      down
Tel1/1/4          unassigned      YES unset  down      down
Tel1/1/5          unassigned      YES unset  down      down
Tel1/1/6          unassigned      YES unset  down      down
Tel1/1/7          unassigned      YES unset  down      down
Tel1/1/8          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!

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 2: 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	<p>To address network provisioning challenges, Cisco introduces a zero-touch provisioning model.</p> <p>In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platform:</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Router models with a minimum of 8 GB RAM to support Guest Shell. <p>In Cisco IOS XE Fuji 16.7.1, this feature was implemented on the following platform:</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Aggregation Services Routers (ASR1001-X, ASR1001-HX, ASR1002-X, ASR1002-HX) <p>In Cisco IOS XE Fuji 16.8.2, this feature was implemented on the following platform:</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Series Aggregation Services Routers (ASR1004, ASR1006, ASR1006-X, ASR1009-X, ASR1013)

Feature Name	Release	Feature Information
Zero-Touch Provisioning: HTTP Download	Cisco IOS XE Fuji 16.8.1 Cisco IOS XE Fuji 16.8.1a	<p>Zero-Touch Provisioning supports HTTP and TFTP file download.</p> <p>In Cisco IOS XE Everest 16.8.1, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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 <p>In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches</p>
DHCPv6 Support for Zero-Touch Provisioning	Cisco IOS XE Fuji 16.9.1	<p>In Cisco IOS XE Fuji 16.9.1, this feature was implemented on the following platforms</p> <ul style="list-style-type: none"> • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 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 39](#)
- [How to Configure iPXE, on page 47](#)
- [Configuration Examples for iPXE, on page 49](#)
- [Troubleshooting Tips for iPXE, on page 52](#)
- [Additional References for iPXE, on page 53](#)
- [Feature Information for iPXE, on page 53](#)

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.

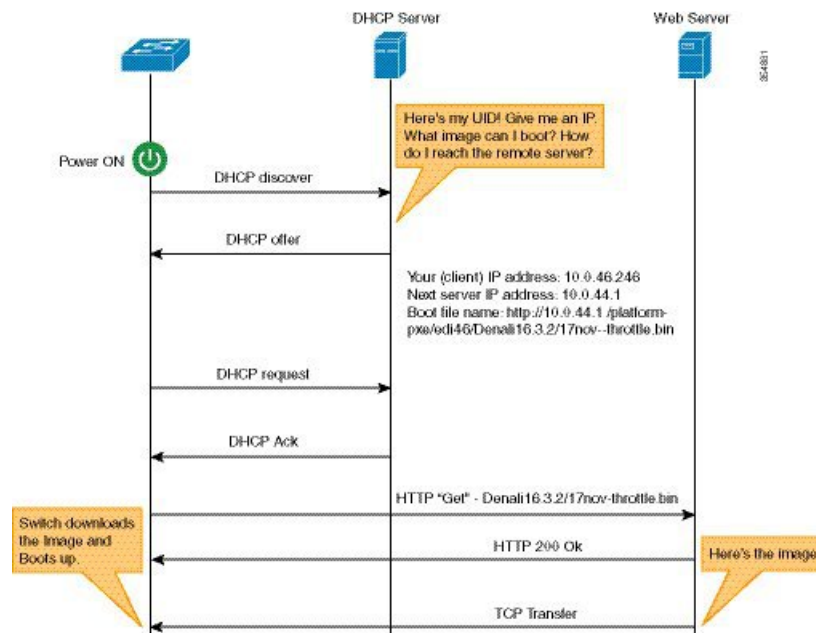


Note Manual boot is another term used in this document. Manual boot is a flag that determines whether to do a rommon reload or not. When the device is in rommon mode, you have to manually issue the **boot** command.

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

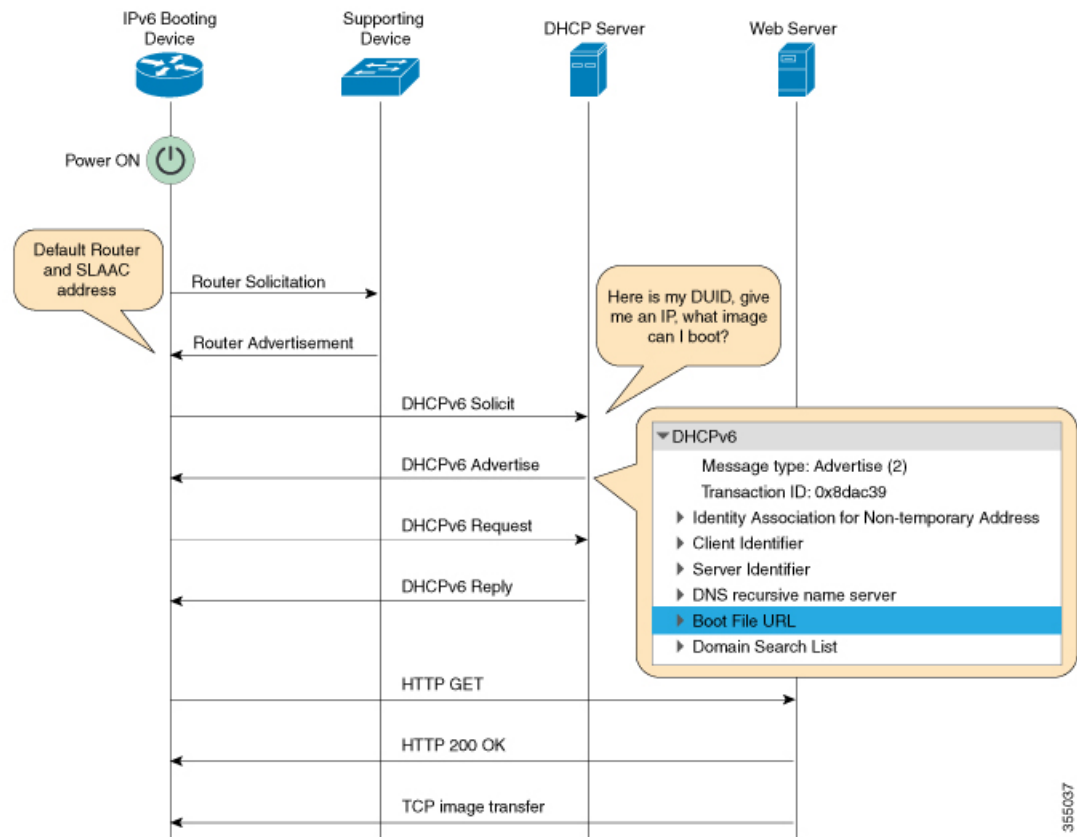
Switch#   Role   Mac Address           Priority  H/W   Current
State
-----
1         Member  046c.9dle.1a00        1
2         Standby 046c.9d01.7d80        1
*3        Active  f8b7.e24e.9a00        1      P2B   Ready
```

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.

- 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
- 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.
 - The client then sends a DHCPv6 request packet to the multicast group address ff02::1:2, requesting for advertised parameters.
 - 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
```

- The device then sends an HTTP GET request to the web server.
- If the requested image is available at the specified path, the web server responds with an OK for the HTTP GET request.
- 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:

- 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 the ROMMON 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 3: 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

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	<ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • 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

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	<ul style="list-style-type: none"> • no boot ipxe • default boot ipxe Example: Device(config)# no boot ipxe Example: Device(config)# default boot ipxe	Configures device boot. The default boot 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** command 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	<i>Dynamic Host Configuration Protocol for IPv6 (DHCPv6)</i>
RFC 3986	<i>Uniform Resource Identifier (URI): Generic Syntax</i>

Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	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 4: Feature Information for iPXE

Feature Name	Release	Feature Information
iPXE	Cisco IOS XE Denali 16.5.1a	<p>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.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Catalyst 3650 Series Switches • Catalyst 3850 Series Switches
	Cisco IOS XE Denali 16.6.1	<p>In Cisco IOS XE Denali 16.6.1, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Catalyst 9300 Series Switches • Catalyst 9500 Series Switches
	Cisco IOS XE Everest 16.6.2	<p>In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.</p>
	Cisco IOS XE Fuji 16.9.2	<p>In Cisco IOS XE Fuji 16.9.2, this feature was implemented on the following platforms.</p> <ul style="list-style-type: none"> • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300L SKUs
IPXE IPv6 Support	Cisco IOS XE 16.8.1a	<p>IPXE supports the IPv6 protocol.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Catalyst 9300 Series Switches • Catalyst 9400 Series Switches • Catalyst 9500 Series Switches



PART II

Shells and Scripting

- [Guest Shell, on page 57](#)
- [Python API, on page 79](#)
- [CLI Python Module, on page 85](#)
- [EEM Python Module, on page 91](#)



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.

- [Information About the Guest Shell, on page 57](#)
- [How to Enable the Guest Shell, on page 65](#)
- [Configuration Examples for the Guest Shell, on page 71](#)
- [Additional References for Guest Shell, on page 75](#)
- [Feature Information for Guest Shell, on page 76](#)

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 rootfs. 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.

This table provides information about the various Guest Shell capabilities and the supported platforms.

Table 5: Cisco Guest Shell Capabilities

	Guest Shell Lite (Limited LXC Container)	Guest Shell (LXC Container)
Operating System	Cisco IOS XE	Cisco IOS XE
Supported Platforms	<ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches (all models) • Cisco Catalyst 3850 Series Switches (all models) 	<ul style="list-style-type: none"> • Cisco ISR 4000 Series Integrated Services Routers (Models with a minimum of 8 GB RAM.)
Guest Shell Environment	Montavista CGE7	CentOS 7
Python 2.7	Supported (Python V2.7.11)	Supported (Python V2.7.5)
Custom Python Libraries	<ul style="list-style-type: none"> • Cisco Embedded Event Manager • Cisco IOS XE CLIs • Neclient 	<ul style="list-style-type: none"> • Cisco Embedded Event Manager • Cisco IOS XE CLIs
Supported Rootfs	Busybox, SSH, and Python PIP install	SSH, Yum install, and Python PIP install
GNU C Compiler	Not supported	Not supported
RPM Install	Not supported	Supported
Architecture	MIPS	x86

Guest Shell Vs Guest Shell Lite

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 7.0 minimal rootfs. You can install other Python libraries such as, Python Version 3.0 during runtime using the Yum utility in CentOS 7.0. You can also install or update python packages using PIP.

The Guest Shell Lite container on MIPS platforms such as, Catalyst 3650 and Catalyst 3850 Series Switches have the Montavista Carrier Grade Edition (CGE) 7.0 rootfs. You can only install or run scripts in Guest Shell Lite. Yum install is not supported on these devices.

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. The Cisco CSR 1000v and Cisco ISRv (virtual platforms) implement these requirements in the software.

Table 6: Guest Shell Support on Catalyst Switches

Platforms	Default DRAM	Guest Shell Support
WS-3650-xxx (all)	4 GB	Supported
WS-3850-xxx (all)	4 GB	Supported
C9300-xx-x (all)	8 GB	Supported
C9500-24Q-x (all)	16 GB	Supported

The minimum system requirement for Catalyst 3850 Series Switches is 4 GB DRAM.

Table 7: Guest Shell Support on ISR 4000 Series Integrated Services Routers

Platform	Default DRAM	Guest Shell Support
ISR 4221	4GB	Not Supported
ISR 4321	4 GB	Not Supported
	8 GB	Supported
ISR 4331	8 GB	Supported
	16 GB	Supported
ISR 4351	8 GB	Supported
	16 GB	Supported
ISR 4431	8 GB	Supported
	16 GB	Supported
ISR 4451	8 GB	Supported
	16 GB	Supported

The minimum system requirement for ISR 4000 Series Integrated Services Routers is 8 GB DRAM.



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

The minimum system requirement for CSR 1000v and ISRv is 4GB RAM.

Guest Shell Storage Requirements

On Catalyst 3650 and Catalyst 3850 Series Switches, Guest Shell can only be installed on the flash filesystem. Bootflash of Catalyst 3850 Series Switches require 75 MB free 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)-Service Set Identifier (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-SSID) space for Guest Shell to install successfully.

For Cisco 4000 Series Integrated Services Routers and ASR 1000 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 ISR 4000 Series 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 Catalyst 3850 Series Switches, Guest Shell has 18 MB of storage space available and 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 CSR 1000v)	256/256 MB 512/512 MB (Cisco CSR 1000v)

Accessing Guest Shell on a Device

Network administrators can use 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 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 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 IOS configuration. For more information, see the *Sample VirtualPortGroup Configuration* section.

Stacking with Guest Shell

When Guest Shell is installed, a directory is automatically created in the flash filesystem. This directory is synchronized across stack members. During a switchover, only contents of the this directory are synchronized across all stack members. To preserve data during high availability switchover, place data in this directory.

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 filesystem is not maintained. Guestshell state is internally synchronized across all stack members.

IOx Overview

IOx is a Cisco-developed 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/use.

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

App hosting provides the following features:

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

IOx Tracing and Logging Overview

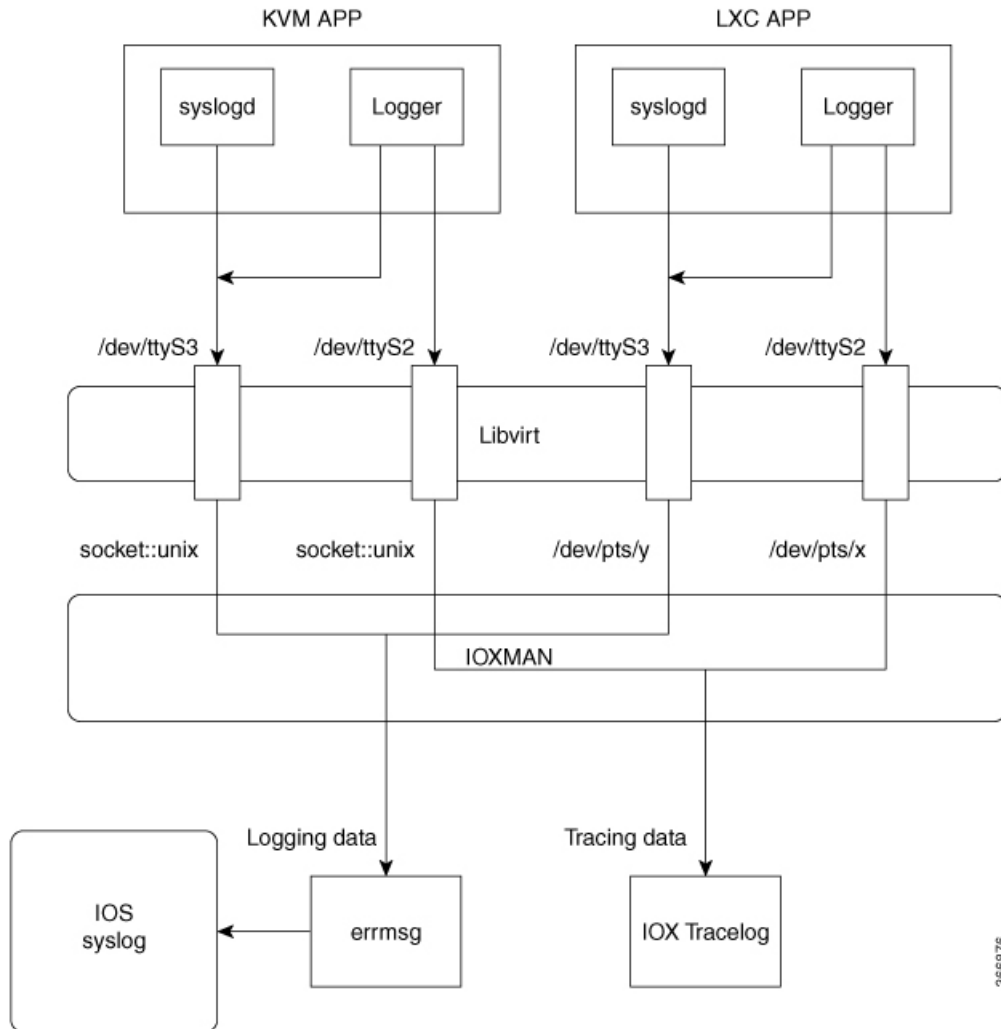
IOx tracing and logging feature allows guest application to run separately on the host device that can help reporting the logging and tracing of the data to the host. The tracing data is saved into IOx tracelog, and the logging data is saved into IOS syslog on the host device.

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 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 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 IOS syslog, to save tracing data to IOx tracelog, and to maintain IOx tracelog for each guest application.

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 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 IOS syslog.

KVM Logging

1. Guest OS enables **/dev/ttyS2** on 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. Start timer to send the message to **/dev/log** device on the host using **errmsg**.
8. Data is saved to IOS syslog.

LXC Tracing

1. Guest OS enables **/dev/ttyS3** on 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 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 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 available data from the socket, sets the severity 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 explain the logging and tracing of messages in to IOS syslog.

Logging Messages in IOS Syslog

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

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

In order to comply with IOS syslog, IOXMAN does support severity for logging message. To report logging message with severity, Guest Application needs to append the 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
```

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 Guest Application to IOS syslog:

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

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

Tracing Message to IOx Tracelog

Perform the following steps to report tracing messages from Guest Application to IOx tracelog:

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

```
#define SYSLOG_TEST      "tracelog test"

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

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

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

Example: Guest Shell Networking Configuration

For Guest Shell networking, the following configurations are required.

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

How to Enable the Guest Shell

Managing IOx

Before you begin

IOx takes upto two minutes to start. CAF, IOXman, and Libirdtd 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

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • 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.
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.

What to do next

The following is sample output from the **show iox-service** command on an ISR 4000 Series Router:

```
Device# show iox-service

Virtual Service Global State and Virtualization Limits:

Infrastructure version : 1.7
Total virtual services installed : 0
Total virtual services activated : 0

Machine types supported   : KVM, LXC
Machine types disabled    : none

Maximum VCPUs per virtual service : 6
Resource virtualization limits:
Name                       Quota      Committed   Available
-----
system CPU (%)             75         0           75
memory (MB)                10240     0           10240
bootflash (MB)             1000      0           1000
harddisk (MB)              20000     0           18109
volume-group (MB)         190768    0           170288
```

```
IOx Infrastructure Summary:
-----
IOx service (CAF)      : Running
IOx service (HA)      : Not Running
IOx service (IOxman)  : Running
LibvirtD              : Running
```

The following is truncated sample output from the **show iox-service** command on a Catalyst 3850 Series Switch:

```
Device# show iox-service

IOx Infrastructure Summary:
-----
IOx service (CAF)      : Running
IOx service (HA)      : Running
IOx service (IOxman)  : Running
LibvirtD              : Running
```

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, a message to configure IOx is displayed. 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

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none">• Enter your password if prompted.
Step 2	guestshell enable Example: Device# guestshell enable	Enables the Guest Shell service. Note <ul style="list-style-type: none">• 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 <i>linux-executable</i> 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: Device# guestshell disable	Disables the Guest Shell service.
Step 6	guestshell destroy Example: Device# guestshell destroy	Deactivates and uninstalls the Guest Shell service.

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.

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, a message to configure IOx is displayed. 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 67](#).

```
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
```

```

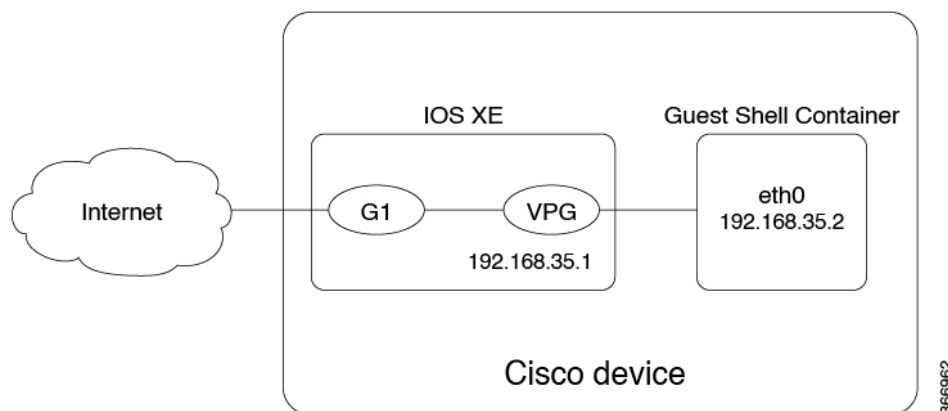
Device(config)# ip access-list standard GS_NAT_ACL
Device(config)# permit 192.168.0.0 0.0.255.255

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.

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

The following example shows how to enable Guest Shell on a Catalyst 3850 Series Switch:

```
Device> enable
Device# guestshell enable

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

```

Device# guestshell run python

Python 2.7.11 (default, Feb 21 2017, 03:39:40)
[GCC 5.3.0] 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

```

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: 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 3.10.101.cge-rt110 #1 SMP Sat Feb 11 00:33:02
PST 2017 mips64 GNU/Linux

```

Catalyst 3650 and Catalyst 3850 Series Switches have a defined set of Linux executables that are provided by BusyBox and Cisco 4000 Series Integrated Services Routers have commands provided by CentOS Linux release 7.1.1503.

The following example shows the usage of the **dohost** command on a Catalyst 3850 Series Switch.

```

[guestshell@guestshell ~]$ dohost "show version"

Cisco IOS Software [Everest], Catalyst L3 Switch Software [CAT3K_CAA-UNIVERSALK9-M],
Experimental Version 16.5.2017200014[v165_throttle-BLD-
BLD_V165_THROTTLE_LATEST_20170531_192849 132]

```



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

Example: Guest Shell Networking Configuration

For Guest Shell networking, the following configurations are required.

- 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-host 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
	Programmability Command Reference, Cisco IOS XE Everest 16.6.1
Python module	CLI Python Module
Zero-Touch Provisioning	Zero-Touch Provisioning

MIBs

MB	MIBs Link
	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	<p>http://www.cisco.com/support</p>

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	<p>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.</p> <p>In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>In Cisco IOS Everest 16.5.1b, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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
		<p>In Cisco IOS XE Fuji 16.7.1, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Series Aggregation Services Routers • Cisco Cloud Services Router 1000v Series <p>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.</p>
	Cisco IOS XE Fuji 16.8.1	<p>In Cisco IOS XE Fuji 16.8.1, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches.</p>
	Cisco IOS XE Fuji 16.9.1	<p>In Cisco IOS XE Fuji 16.9.1, this feature was implemented on Cisco 1000 Series Integrated Services Routers.</p>



CHAPTER 5

Python API

Python programmability supports Python APIs.

- [Using Python, on page 79](#)

Using Python

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(cli)
```

```
Help on function cli in module cli:
```

```
cli(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.
```

```
cli("show version")
```

```
cli("show version ; show ip interface brief")
```

```
cli("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.cli('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.cli('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
```




CHAPTER 6

CLI Python Module

Python Programmability provides a Python module that allows users to interact with IOS using CLIs.

- [Information About Python CLI Module, on page 85](#)
- [Additional References for the CLI Python Module, on page 89](#)
- [Feature Information for the CLI Python Module, on page 89](#)

Information About Python CLI Module

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.

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 /flash/script.py command at the device prompt. The following example shows how to run a Python script from the Guest Shell:

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

```
Device# guestshell run python /flash/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
:
:
Tel1/1/4              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	<ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches

Python Version	Platform
Python Version 3.6	<p>Supported in Cisco IOS XE Amsterdam 17.1.1 and later releases.</p> <p>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.</p> <p>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.</p> <p>Note Not supported by Cisco Catalyst 3650 Series Switches and Cisco Catalyst 3850 Series Switches.</p>

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 the CLI Python Module

Related Documents

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

Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	http://www.cisco.com/support

Feature Information for the CLI 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 11: Feature Information for the CLI Python Module

Feature Name	Release	Feature Information
CLI Python Module	Cisco IOS XE Everest 16.5.1a	<p>Python programmability provides a Python module that allows users to interact with IOS using CLIs.</p> <p>In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco ASR 1000 Aggregation Services Routers • Cisco CSR 1000v Series Cloud Services Routers



CHAPTER 7

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 91](#)
- [Information About EEM Python Module, on page 91](#)
- [How to Configure the EEM Python Policy, on page 94](#)
- [Additional References EEM Python Module, on page 99](#)
- [Feature Information for EEM Python Module, on page 100](#)

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 predefined 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

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	event manager directory user policy <i>path</i> 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 <code>eem_script.py</code> policy is available in the <code>flash:/user_library</code> folder or path.
Step 4	event manager policy <i>policy-filename</i> Example: Device(config)# event manager policy eem_script.py	Registers a policy with EEM. <ul style="list-style-type: none"> • The policy is parsed based on the file extension. If the file extension is <code>.py</code>, 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 invoked, EEM examines the policy and registers it to be run when the specified event occurs.

	Command or Action	Purpose
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
Tel1/1/1 unassigned YES unset down down
Tel1/1/2 unassigned YES unset down down
Tel1/1/3 unassigned YES unset down down
```

```
Tel1/1/4 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

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	event manager applet <i>applet-name</i> 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 <i>event-tag</i>] syslog pattern <i>regular-expression</i> 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 <i>label</i> cli command <i>cli-string</i> 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 <i>label</i> cli command <i>cli-string</i> [pattern <i>pattern-string</i>] 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. <ul style="list-style-type: none"> • 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: Device# show event manager policy active	Displays EEM policies that are executing.
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
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	<p>http://www.cisco.com/support</p>

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	This feature supports Python scripts as EEM policies.
	Cisco IOS XE Everest 16.5.1b	No new commands were introduced. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: <ul style="list-style-type: none"> • 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: <ul style="list-style-type: none"> • 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 105](#)
- [RESTCONF Protocol, on page 123](#)
- [gNMI Protocol, on page 137](#)
- [Model Based AAA, on page 159](#)
- [Model-Driven Telemetry, on page 167](#)
- [In-Service Model Update, on page 195](#)



CHAPTER 8

NETCONF Protocol

- [Information About the NETCONF Protocol, on page 105](#)
- [How to Configure the NETCONF Protocol, on page 110](#)
- [Verifying the NETCONF Protocol Configuration, on page 114](#)
- [Additional References for NETCONF Protocol, on page 117](#)
- [Feature Information for NETCONF Protocol, on page 117](#)

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.



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](#) repository, to view **-oper* in the naming convention.

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.

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.

Candidate Config Support

The Candidate Config Support feature enables support for candidate capability by implementing RFC 6241 with a simple commit option.

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 Candidate Datastore. This is a shared data store which enables the user to create, add, delete and make changes

to the device configuration without effecting the running configuration on the device. A commit operation pushes the configuration from “candidate” to the “running” configuration on the device. When “candidate” data store is enabled, the Running data store is NOT writable through NETCONF sessions, all configurations get committed ONLY through Candidate. In other words, the writable-running NETCONF capability is not enabled with Candidate.



Note It must be kept in mind that candidate datastore is a shared data store. Multiple NETCONF sessions can modify its contents simultaneously. Therefore, it is important to lock the datastore before modifying its contents, to prevent conflicting commits which can eventually lead to losing any configuration changes.

NETCONF Operations on Candidate

The following operations can be performed on Candidate data store.



Note The information in this section has been referenced from section 8.3.4 of RFC6241. 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 other users from modifying the configuration in the “locked” data store. We can lock both candidate and running data through the lock operation.



Note “Locking” candidate datastore does not affect Cisco IOS config lock or running config 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 “Running” or “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” config 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

Candidate configuration can be used as a target for “edit-config” operation to modify a configuration. This provides you the ability to stage configuration changes in candidate without affecting the running configuration on the device.

Discard

To remove the changes made to candidate configuration, perform a discard operation to revert the candidate configuration to running configuration.

If the contents of candidate datastore have already been modified by NETCONF session A, and session B tries to lock the Candidate, the lock will fail. NETCONF session B must perform a <discard> operation to remove any outstanding configurations on candidate 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 data store, by specifying “candidate” as target in the Unlock RPC. Candidate is now available for all operations in other sessions.

If failure occurs with outstanding changes to candidate datastore, it can be challenging to recover the configuration and creates problems for other sessions. To avoid any issues, outstanding changes are discarded in candidate when the lock is released—either implicitly on “NETCONF session failure” or explicitly on “Unlock” operation.

Get-config, Copy-config, Validate

Candidate 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 changes in candidate to the device but only to validate the configuration, you can do so by <validate> RPC followed by a “discard”.

How to use Candidate?

The following diagram explains the recommended best practice when modifying the device configuration through candidate datastore:

Figure 4: Modifying Candidate Datastore Steps



1. Lock running datastore.
2. Lock candidate datastore.
3. Make modifications to candidate configuration through edit-config RPCs with target candidate.
4. Commit candidate configuration to running.
5. Unlock candidate and running.

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 will be displayed notifying the user that a restart of netconf-yang or restconf will occur in order for the change to take effect. When candidate is enabled, the running data store is *not* writable through Netconf sessions, all configurations get committed *only* through candidate. In other words, the writable-running Netconf capability is not enabled with candidate.

If the selection of candidate or running datastore, is specified in the configuration when a netconf-yang or restconf confd process starts, a warning 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 candidate or running is made after 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.



Note Candidate data store is a shared data store, that is, multiple Netconf sessions can modify the contents simultaneously. Therefore, it is important for a user to lock the data store before modifying its contents, to prevent conflicting commits which can eventually lead to losing any configuration changes; wherein another user overwrites the configuration by modifying the configuration and issuing a commit.

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

	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: <ul style="list-style-type: none"> • 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 Only the default AAA authentication login method is supported for the NETCONF protocol. <ul style="list-style-type: none"> • For a remote AAA server, replace <i>local</i> 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 Only the default AAA authorization method is supported for the NETCONF protocol. <ul style="list-style-type: none"> • For a remote AAA server, replace <i>local</i> with your AAA server. • The default keyword applies the local user database authentication to all ports.

	Command or Action	Purpose
Step 7	end Example: Device(config)# <code>end</code>	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

	Command or Action	Purpose
Step 1	enable Example: Device> <code>enable</code>	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# <code>configure terminal</code>	Enters global configuration mode.
Step 3	netconf-yang Example: Device (config)# <code>netconf-yang</code>	Enables the NETCONF interface on your network device. <p>Note After the initial enablement through the CLI, network devices can be managed subsequently through a model based interface. The complete activation of model-based interface processes may require up to 90 seconds.</p>
Step 4	netconf-yang feature candidate-datastore Example: Device(config)# <code>netconf-yang feature candidate-datastore</code>	Enables candidate datastore.
Step 5	exit Example: Device (config)# <code>exit</code>	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

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>

```

Verifying the NETCONF Protocol Configuration

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 statistics**
5. **show platform software yang-management process**

DETAILED STEPS

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 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 5 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
vtyserverutil : 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 interface (DMI) authentication daemon
vtyserverutil	VTY server util daemon
opdatamgrd	Operational Data Manager daemon
nginx	NGINX web server
ndbmand	NETCONF database manager

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	<i>YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)</i>
RFC 6241	<i>Network Configuration Protocol (NETCONF)</i>
RFC 6536	<i>Network Configuration Protocol (NETCONF) Access Control Model</i>
RFC 8040	<i>RESTCONF Protocol</i>

Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	http://www.cisco.com/support

Feature Information for NETCONF 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 14: Feature Information for NETCONF Protocol

Feature Name	Release	Feature Information
NETCONF Protocol	Cisco IOS XE Denali 16.3.1	<p>The NETCONF Protocol feature facilitates a programmatic and standards-based way of writing configurations and reading operational data from network devices.</p> <p>The following command was introduced: netconf-yang.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches
	Cisco IOS XE Everest 16.6.2	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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
		<p>In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs
	Cisco IOS XE Gibraltar 16.10.1	<p>In Cisco IOS XE Gibraltar 16.10.1, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco Network Convergence System 520 Series

Feature Name	Release	Feature Information
NETCONF and RESTCONF IPv6 Support	Cisco IOS XE Fuji 16.8.1a	<p>IPv6 support for the NETCONF and RESTCONF protocols. This feature was implemented on the following platforms:</p> <ul style="list-style-type: none">• 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 Global Lock and Kill Session	Cisco IOS XE Fuji 16.8.1a	<p>The NETCONF protocol supports a global lock, and the ability to kill non-responsive sessions. This feature is implemented on the following platforms:</p> <ul style="list-style-type: none">• 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	<p>The Candidate Config Support feature enables support for candidate capability by implementing RFC 6241 with a simple commit option.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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 <p>The following command was introduced: netconf-yang feature candidate-datastore.</p>



CHAPTER 9

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 123](#)
- [Restrictions for the RESTCONF Protocol, on page 123](#)
- [Information About RESTCONF Programmable Interface, on page 124](#)
- [How to Configure RESTCONF Programmable Interface, on page 126](#)
- [Configuration Examples for RESTCONF Programmable Interface, on page 131](#)
- [Additional References for the RESTCONF Protocol, on page 134](#)
- [Feature Information for the RESTCONF Protocol, on page 135](#)

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 RESTCONF Programmable Interface

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](#) repository, 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\(mtu\)](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>

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 formatted RPC that is sent to the RESTCONF 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-formatted 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*

How to Configure RESTCONF Programmable Interface

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

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode <ul style="list-style-type: none"> • 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 <i>server-name</i> Example: Device(config)# aaa group server radius ISE	Adds the RADIUS server and enters server group RADIUS configuration mode. <ul style="list-style-type: none"> • The <i>server-name</i> argument specifies the RADIUS server group name.
Step 5	server-private <i>ip-address</i> key <i>key-name</i> 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 <i>vrf-name</i> 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.
Step 8	aaa authentication login default group <i>group-name</i> local Example:	Sets the specified group name as the default local AAA authentication during login.

	Command or Action	Purpose
	Device(config)# aaa authentication login default group ISE local	
Step 9	aaa authentication login <i>list-name</i> 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 <i>group-name</i> 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 <i>number</i> Example: Device(config)# line console 0	Identifies a specific line for configuration and enter line configuration mode.
Step 13	login authentication <i>authentication-list</i> 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

	Command or Action	Purpose
Step 1	enable	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: Device> enable	<ul style="list-style-type: none"> 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 processes 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 server, 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 restarted 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 RESTCONF Programmable Interface

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 available 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/ciscosubdirectory. 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	Representational State Transfer Configuration Protocol (RESTCONF)

Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	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 www.cisco.com/go/cfn. 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	<p>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.</p> <p>This feature was introduced on the following platforms:</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Router • Cisco ASR 1000 Aggregation Services Routers • Cisco Cloud Services Router 1000V Series <p>The following commands were introduced or modified: ip http server and restconf</p>
	Cisco IOS XE Fuji 16.8.1a	<p>In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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	<p>In Cisco IOS XE Fuji 16.9.2, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs



CHAPTER 10

gNMI Protocol

This feature describes the model-driven configuration and retrieval of operational data using the gRPC Network Management Interface (gNMI) CAPABILITIES, Get and Set remote procedure calls (RPCs). gNMI version 0.4.0 is supported.

- [Restrictions for the gNMI Protocol, on page 137](#)
- [Information About the gNMI Protocol, on page 138](#)
- [How to Enable the gNMI Protocol, on page 151](#)
- [Configuration Examples for the gNMI Protocol, on page 156](#)
- [Additional References for the gNMI Protocol, on page 157](#)
- [Feature Information for the gNMI Protocol, on page 157](#)

Restrictions for the gNMI Protocol

The following restrictions apply to the feature:

- Subscribe RPC services are not supported.
- JSON, BYTES, PROTO, and ASCII encoding options are not supported.
JSON IETF keys must contain a YANG-prefix where the namespace of the following elements differs from 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 is not supported. It is a string that provides an alias for a prefix specified within the notification message.
 - Delete is not supported. It is a set of paths that are to be removed from a data tree.

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.

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 (leaves, 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
```

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 data 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.

Table 16: GetRequest JSON Structure

GetRequest	GetResponse
<p>The following is a path for the openconfig-interfaces model</p> <pre>+++++++ Sending get request: +++++++ path { elem { name: "interfaces" } elem { name: "interface" key { key: "name" value: "Loopback111" } } }</pre>	

GetRequest	GetResponse
	<pre> 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\t \"Loopback111\", \n\t\t \"openconfig-interfaces:config\":\t{\n\t\t\t\t \"openconfig-interfaces:type\":\t\t\"ianaift: softwareLoopback\", \n\t\t\t\t \"openconfig-interfaces:name\":\t\t\"Loopback111\", \n\t\t\t\t \"openconfig-interfaces:enabled\":\t\t\"true\"\n\t\t}, \n\t\t\t\t \"openconfig-interfaces:state\":\t{\n\t\t\t\t\t\t \"openconfig-interfaces:type\":\t\t\"ianaift: softwareLoopback\", \n\t\t\t\t\t\t \"openconfig-interfaces:name\":\t\t\"Loopback111\", \n\t\t\t\t\t\t \"openconfig-interfaces:enabled\":\t\t\"true\", \n\t\t\t\t\t\t \"openconfig-interfaces:ifindex\":\t\t52, \n\t\t\t\t\t\t \"openconfig-interfaces:admin-status\":\t\t\"UP\", \n\t\t\t\t\t\t \"openconfig-interfaces:oper-status\":\t\t\"UP\", \n\t\t\t\t\t\t \"openconfig-interfaces:last-change\":\t\t2018, \n\t\t\t\t\t\t \"openconfig-interfaces:counters\":\t{\n\t\t\t\t\t\t\t\t </pre>

GetRequest	GetResponse
	<pre> "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:in-unknown-protos\":\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),\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\t\ "openconfig-interfaces:config\":\t{\n\t\t\t\t\ "openconfig-interfaces:index\":\t0,\n\t\t\t\t\ "openconfig-interfaces:name\":\t"Loopback111",\n\t\t\t\t\ "openconfig-interfaces:enabled\":\t"true"\n\t\t),\n\t\t\ </pre>

GetRequest	GetResponse
	<pre> "openconfig-interfaces:state\":"\t{\n\t\t\t\t\ "openconfig-interfaces:index\":"\t0,\n\t\t\t\t\ "openconfig-interfaces:name\":"\t"Loopback11.0\",\n\t\t\t\t\ "openconfig-interfaces:enabled\":"\t"true","\n\t\t\t\t\ "openconfig-interfaces:admin-status\":"\t"UP","\n\t\t\t\t\ "openconfig-interfaces:oper-status\":"\t"UP","\n\t\t\t\t\ "openconfig-interfaces:last-change\":"\t2018,\n\t\t\t\t\ "openconfig-interfaces:counters\":"\t{\n\t\t\t\t\t\t\ "openconfig-interfaces:in-octets\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:in-unicast-pkts\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:in-broadcast-pkts\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:in-multicast-pkts\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:in-discards\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:in-errors\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:out-octets\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:out-unicast-pkts\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:out-broadcast-pkts\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:out-multicast-pkts\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:out-discards\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:out-errors\":"\t0,\n\t\t\t\t\t\t\ "openconfig-interfaces:last-clear\":"\ t2018\n\t\t\t\t}\n\t\t\t\t},\n\t\t\t\t\ "openconfig-if-ip:ipv6\":"\t{\n\t\t\t\t\t\t\ </pre>

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 will be disallowed and rolled back. A SetResponse is sent back for a SetRequest.

Table 18: Example of a SetRequest JSON Structure

SetRequest	SetResponse
<pre> +++++++ 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\"}" } } </pre>	<pre> +++++++ Received set response: ++++++ response { path { elem { name: "interfaces" } elem { name: "interface" key { key: "name" value: "Loopback111" } } elem { name: "config" } } op: UPDATE } timestamp: 1521699342123890045 </pre>

Table 19: Example of a SetRequest on Leaf Value

SetRequest	SetResponse
<pre> +++++++ 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\"" } } </pre>	<pre> +++++++ 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 </pre>

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 example 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 example 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 example shows a path ellipsis wildcard. This wildcard will return all description fields available under /interfaces.

```
path {
  elem {
    name: "interfaces"
  }
  elem {
    name: "..."/>

```

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 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
# Encrypt device key - needed for input to IOS
openssl rsa -des3 -in device.key -out device.des3.key -passout pass:<password - remember
this for later>

# 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

The following example show how to install certs on a device:

```
# Send:
Device# configure terminal
Device(config)# crypto pki import trustpoint1 pem terminal password password1

# Receive:
% Enter PEM-formatted CA certificate.
% End with a blank line or "quit" on a line by itself.

# Send:
# Contents of rootCA.pem, followed by newline + 'quit' + newline:
```

```

-----BEGIN CERTIFICATE-----
<snip>
-----END CERTIFICATE-----
quit

# Receive:
% Enter PEM-formatted encrypted private General Purpose key.
% End with "quit" on a line by itself.

# Send:
# Contents of device.des3.key, followed by newline + 'quit' + newline:
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,D954FF9E43F1BA20
<snip>
-----END RSA PRIVATE KEY-----
quit

# Receive:
% Enter PEM-formatted General Purpose certificate.
% End with a blank line or "quit" on a line by itself.

# Send:
# Contents of device.crt, followed by newline + 'quit' + newline:
-----BEGIN CERTIFICATE-----
<snip>
-----END CERTIFICATE-----
quit

# Receive:
% PEM files import succeeded.
Device(config)#

# Send:
Device(config)# crypto pki trustpoint trustpoint1
Device(ca-trustpoint)# revocation-check none
Device(ca-trustpoint)# end
Device#

```

Enabling gNMI in Insecure Mode



Note This task is applicable in Cisco IOS XE Fuji 16.8.1 through Amsterdam 17.2.x.

In a Day Zero setup, first enable the device in insecure mode, then disable it, and enable the secure mode. To stop gNMI in insecure mode, use the **no gnmi-yang server** command.



Note gNMI insecure and secure servers can run simultaneously.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **gnmi-yang**

4. **gnmi-yang server**
5. **gnmi-yang port *port-number***
6. **end**
7. **show gnmi-yang state**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none">• Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	gnmi-yang Example: Device(config)# gnmi-yang	Starts the gNMI process.
Step 4	gnmi-yang server Example: Device(config)# gnmi-yang server	Enables the gNMI server in insecure mode.
Step 5	gnmi-yang port <i>port-number</i> Example: (Optional) Device(config)# gnmi-yang port 50000	Sets the gNMI port to listen to. <ul style="list-style-type: none">• The default insecure gNMI port is 9339.
Step 6	end Example: Device(config)# end	Exits global configuration mode and returns to privileged EXEC mode.
Step 7	show gnmi-yang state Example: Device# show gnmi-yang state	Displays the status of gNMI servers.

Example

The following is sample output from the **show gnmi-yang state** command:

```
Device# show gnmi-yang state
State Status
-----
Enabled Up
```

Enabling gNMI in Secure Mode



Note This task is applicable in Cisco IOS XE Fuji 16.8.1 through Amsterdam 17.2.x.

To stop gNMI in secure mode, use the **no gnmi-yang secure-server** command.



Note gNMI insecure and secure servers can run simultaneously.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **gnmi-yang**
4. **gnmi-yang secure-server**
5. **gnmi-yang secure-trustpoint** *trustpoint-name*
6. **gnmi-yang secure-client-auth**
7. **gnmi-yang secure-port**
8. **end**
9. **show gnmi-yang state**

DETAILED STEPS

	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	gnmi-yang Example: Device(config)# gnmi-yang	Starts the gNMI process.
Step 4	gnmi-yang secure-server Example: Device(config)# gnmi-yang secure-server	Enables the gNMI server in secure mode.
Step 5	gnmi-yang secure-trustpoint <i>trustpoint-name</i> Example: Device(config)# gnmi-yang secure-trustpoint trustpoint1	Specifies the trustpoint and cert set that gNMI uses for authentication.

	Command or Action	Purpose
Step 6	gnmi-yang secure-client-auth Example: Device(config)# gnmi-yang secure-client-auth	(Optional) The gNMI process authenticates the client certificate against the root certificate.
Step 7	gnmi-yang secure-port Example: Device(config)# gnmi-yang secure-port	(Optional) Sets the gNMI port to listen to. <ul style="list-style-type: none"> • The default insecure gNMI port is 9339.
Step 8	end Example: Device(config)# end	Exits global configuration mode and returns to privileged EXEC mode.
Step 9	show gnmi-yang state Example: Device# show gnmi-yang state	Displays the status of gNMI servers.

Example

The following is sample output from the **show gnmi-yang state** command:

```
Device# show gnmi-yang 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 50052, can be changed on ios device with 'gnmi-yang secure-port ####'
>>> port = 50052
>>> host = <HOSTNAME FQDN>
>>> secure_channel = grpc.secure_channel("%s:%d" % (host, port), credentials)

# Create secure stub:
>>> secure_stub = gnmi_pb2_grpc.gNMISub(stub=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 the gNMI Protocol



Note This example is applicable in Cisco IOS XE Fuji 16.8.1 through Amsterdam 17.2.x.

Example: Enabling gNMI in Insecure Mode

The following example shows how to enable the gNMI server in insecure mode:

```

Device# configure terminal
Device(config)# gnmi-yang
Device(config)# gnmi-yang server
Device(config)# gnmi-yang port 50000 <The default port is 9339.>
Device(config)# end
Device#

```

Example: Enabling gNMI in Secure Mode

The following example shows how to enable the gNMI server in secure mode:

```

Device# configure terminal
Device(config)# gnmi-yang server
Device(config)# gnmi-yang secure-server
Device(config)# gnmi-yang secure-trustpoint trustpoint1
Device(config)# gnmi-yang secure-client-auth
Device(config)# gnmi-yang 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
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
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	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 20: Feature Information for the gNMI Protocol

Feature Name	Release	Feature Information
gNMI Protocol	Cisco IOS XE Fuji 16.8.1a	<p>This feature describes the model-driven configuration and retrieval of operational data using the gNMI capabilities, GET and SET RPCs.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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.



CHAPTER 11

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 159](#)
- [Additional References for Model Based AAA, on page 165](#)
- [Feature Information for Model-Based AAA, on page 165](#)

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>
```

```

    </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 Persistence

The NACM configuration can be modified using NETCONF or RESTCONF. In order for a user to be able to access the NACM 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 107](#).



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>
  </groups>
</nacm>
```

```

    </group>

    <group>
      <name>limited-permission</name>
      <user-name>alice</user-name>
      <user-name>bob</user-name>
    </group>
  </groups>
</nacm>

```

Table 21: Description of the Configuration Parameters 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 22: Description of the Configuration Parameters 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 23: Description of the Configuration Parameters 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 24: Description of the Configuration Parameters for Creating Data Node Rules

Parameter	Description
<code><name>hide-enable-passwords</name></code>	Unique rule-list name
<code><group>limited-permission</group></code>	Groups that rule-list applies to
<code><name>deny-enable-passwords</name></code>	Unique rule name
<code><path xmlns="http://cisco.com/yang/Cisco-IOS-XE-ntf-ns:ietf-netconf"></code>	Path to the data node being granted/denied
<code><access-operations>*</access-operations></code>	CRUDx operation types
<code><action>deny</action></code>	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	<i>YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)</i>
RFC 6241	<i>Network Configuration Protocol (NETCONF)</i>
RFC 6536	<i>Network Configuration Protocol (NETCONF) Access Control Model</i>

Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	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 25: Feature Information for Programmability: Data Models

Feature Name	Release	Feature Information
Model-Based AAA	Cisco IOS XE Fuji 16.8.1	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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 12

Model-Driven Telemetry

- [Model-Driven Telemetry](#), on page 167

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 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 NETCONF-YANG must be configured for telemetry to work, even if NETCONF is not used. For more information on configuring NETCONF-YANG, see the [NETCONF 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 26: 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 brevity):
{
  "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.

gRPC-Specific Restrictions

- Transport Layer Security-based (TLS-based) authentication between a device and receiver is not supported. TLS-based authentication is supported in Cisco IOS XE Amsterdam 17.1.1 and later releases.

yang-push-Specific Restriction

- Subscription quality of service (QoS) is not supported.

Information About 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 standard-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, there are different roles involved. This document uses the following telemetry roles:

- Publisher: Network element that sends the telemetry data.
- Receiver: Network element that receives the telemetry data. 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. Also called the management agent or management entity.
- Subscriber: Network element that creates subscriptions. While it technically does not have to also be the receiver, for the purposes of 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 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 north-bound programmable interfaces (such as RESTCONF or gNMI) are supported to configure subscriptions.

There are two types of subscriptions 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

There are two flavors of model-driven telemetry: dial-in and dial-out.

Table 27: Dial-in and Dial-Out Model-Driven Telemetry

Dial-in (Dynamic)	Dial-out (Static or Configured)
Telemetry updates are sent to the initiator/subscriber.	Telemetry updates are sent to the specified receiver/collector.
Life of the subscription is tied to the connection (session) that created it, and over which telemetry updates are sent. No change in the running configuration is observed.	Subscription is created as part of the running configuration; it remains the device configuration till the configuration is removed.
Dial-in subscriptions need to be re-initiated 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 the data is required; however this is stream-dependent. Some streams support making data available only when there is a change or 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 subscription ID 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 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 28: 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	No	No	No
Encodings	XML	No	No	Key-value Google Protocol Buffers (kvGPB)	JSON_IETF	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 the reason for the response in an `<rpc-reply>` message:

Result String	RPC	Cause
ok	<code><establish-subscription></code> <code><delete-subscription></code>	Success
error-no-such-subscription	<code><delete-subscription></code>	The specified subscription does not exist.
error-no-such-option	<code><establish-subscription></code>	The requested subscription is not supported.
error-insufficient-resources	<code><establish-subscription></code>	A subscription cannot be created because of the following reasons: <ul style="list-style-type: none"> • There are too many subscriptions. • The amount of data requested is too large. • The interval for a periodic subscription is too small.
error-other	<code><establish-subscription></code>	Some other error.

Dynamic Subscription Control

This section describes how to create and delete dynamic subscriptions

Creating Dynamic Subscriptions

Dynamic subscriptions are created by subscribers that connect to the publisher and call for subscription creation using a mechanism within that connection, usually, a remote procedure call (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.

Periodic Dynamic Subscriptions

The following is a sample periodic subscription:

```
<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 in-band `<delete subscription>` RPC, and disconnecting the transport session.

The `<delete-subscription>` RPC can be issued only by the subscriber, and it deletes only the subscriptions owned by that subscriber.

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/NETCONF session to timeout, and subsequent subscriptions to be removed.

RPCs 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](#).

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>

```

Configured Subscription Management

This section describes how to create, modify, 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 the 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 are not attempted, if a receiver is already connected or in the process of being connected. If that receiver is deleted, another receiver is connected.

This section displays sample RPCs to create configured subscriptions.

Periodic Subscription

The following sample RPC creates 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 creates 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>false</no-synch-on-start>
          <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 creates 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"
        }
      }
    ]
  }
}
```

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. Configured subscriptions cannot be deleted using RPCs. These subscriptions are deleted 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>
```

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 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>1000</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-IOX-XE-mdt-oper module (from the YANG model Cisco-IOX-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-IOX-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-IOX-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.

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 may be sent either when data changes or at 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 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).

IETF Draft Compliance

Telemetry using the *yang-push* stream is based on the IETF NETCONF working group's early drafts for telemetry. These are the following:

- [Custom Subscription to Event Notifications, Version 03](#)
- [Subscribing to YANG datastore push updates, Version 07](#)

The following features described in the 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.

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 the 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 synchronization 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 chosen may also affect the format of the update notification.

Supported transport protocols are NETCONF and gRPC.

NETCONF Protocol

The NETCONF protocol is available for the transport of dynamic subscriptions only, 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.

yang-push Format

This format defines two formats of the update notifications, when the *yang-push* stream is sent over NETCONF as a transport with XML encoding is as defined in *draft-ietf-netconf-yang-push-07*. For more information, see section 3.7 of the IETF draft.

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](#).

High Availability in Telemetry

Dynamic telemetry connections are established over a NETCONF session via SSH to the active switch or a member in a switch stack, or the active route-processor in an high-availability capable device. After switchover, you must destroy and re-establish all sessions that use Crypto, including NETCONF sessions that carry telemetry subscriptions. You must also recreate all dynamic subscriptions after a switchover.

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, 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 synced to the start-up configuration of the device. This ensures that after the device reboots, subscription configurations remain intact on the device. Once the necessary processes are up and running, the device attempts to connect to the telemetry receiver and resume normal operations.

Sample Model-Driven Telemetry RPCs

Managing Configured Subscriptions

Use the **show platform software ndbman switch** *{switch-number | active | standby}* **models** command to display the list of YANG models that support on-change subscription.



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

	Command or Action	Purpose
Step 1	enable	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: Device> enable	<ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	telemetry ietf subscription <i>id</i> 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 <i>path</i> 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} <i>period</i> 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 <i>vrf-id</i> Example: Device(config-mdt-subs)# source-address Mgmt-intf	Configures the source VRF instance.
Step 9	source-address <i>source-address</i> Example: Device(config-mdt-subs)# source-vrf 192.0.2.1	Configures the source address.
Step 10	receiver ip address <i>ip-address receiver-port protocol protocol profile name</i> 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

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	telemetry ietf subscription <i>id</i> 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 <i>path</i> 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 <i>period</i>} 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.

	Command or Action	Purpose
Step 8	receiver ip address <i>ip-address receiver-port protocol protocol profile name</i> Example: Device(config-mdt-subs)# receiver ip address 10.22.22.45 45000 protocol grpc_tls profile secure_profile	Configures the receiver IP address, protocol, and profile for notifications.
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 with 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

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/hwldb-table
Update policy:
  Update Trigger: on-change
  Synch on start: Yes
  Dampening period: 0
Notes:
```

Retrieving Subscription Details Using 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": {
```


Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	<p>http://www.cisco.com/support</p>

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 29: Feature Information for Model-Driven Telemetry

Feature Name	Release	Feature Information
Model-Driven Telemetry NETCONF Dial-In	Cisco IOS XE Everest 16.6.1	<p>Model-driven telemetry allows network devices to continuously stream real time configuration and operating state information to subscribers.</p> <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Cisco Catalyst 9400 Series Switches
	Cisco IOS XE Fuji 16.7.1	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Cisco 1000 Series Integrated Services Routers • Cisco ASR 1000 RP2 and RP3 Series Aggregation Services Routers
	Cisco IOS XE Fuji 16.8.1a	<ul style="list-style-type: none"> • Cisco Catalyst 9500-High Performance Series Switches
	Cisco IOS XE Fuji 16.9.1	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none">• Cisco Catalyst 9200 and 9200L Series Switches• Cisco Catalyst 9300L SKUs
	Cisco IOS XE Gibraltar 16.10.1	<ul style="list-style-type: none">• Cisco Cloud Services Router 1000v• Cisco Network Convergence System 520 Series

Feature Name	Release	Feature Information
Model-Driven Telemetry gRPC Dial-Out	Cisco IOS XE Gibraltar 16.10.1	<p>Configured subscriptions cause the publisher to initiate connections to receivers, and these connections are considered as dial-out.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • 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



CHAPTER 13

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 195
- [Information About In-Service Model Update](#), on page 195
- [How to Manage In-Service Model Update](#), on page 198
- [Configuration Examples for In-Service Model Updates](#), on page 199
- [Feature Information for In-Service Model Update](#), on page 203

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.CSCxxxxxxx.dmp.bin
- asr1000-universalk9.2017-08-23_17.48.0.CSCxxxxxxx.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.

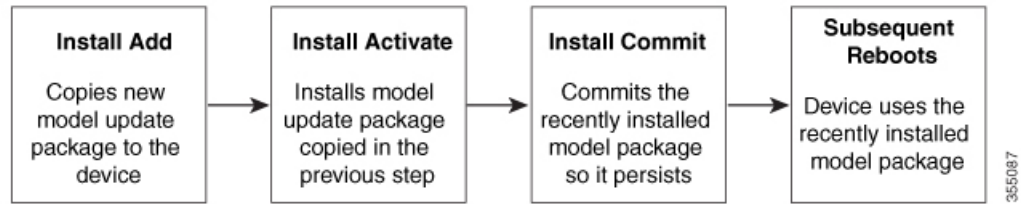
The **install add** command copies the update package from a remote location to the device. You can also use other methods to copy the package; however, you must still enable the **install add** command for the installation to work. For the **install activate** command to work, the package must be available 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 5: Committing a Model Update Package

Process with Install Commit



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 30: 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

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 2	install add file tftp: <i>filename</i> 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. <ul style="list-style-type: none"> • 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: <i>filename</i> 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. <ul style="list-style-type: none"> • 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. <ul style="list-style-type: none"> • NETCONF processes are not restarted.
Step 5	install deactivate file bootflash: <i>filename</i> Example:	Deactivates the specified update package, and restarts the NETCONF processes.

	Command or Action	Purpose
	<pre>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</pre>	
Step 6	<p>install commit</p> <p>Example:</p> <pre>Device# install commit</pre>	<p>Makes the changes persistent over reload.</p> <ul style="list-style-type: none"> NETCONF processes are not restarted.
Step 7	<p>install rollback to {base committed id <i>commit-ID</i>}</p> <p>Example:</p> <pre>Device# install rollback to base</pre>	<p>Rollbacks the update to the base version, the last committed version, or a known commit ID, and restarts NETCONF processes.</p> <ul style="list-style-type: none"> Valid values for the <i>commit-id</i> argument are from 1 to 4294967295. Older versions of data models updates are available for use.
Step 8	<p>install remove {file bootflash: <i>filename</i> inactive}</p> <p>Example:</p> <pre>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</pre>	<p>Removes the specified update package from the bootflash.</p> <ul style="list-style-type: none"> A package must be deactivated before it is removed.
Step 9	<p>show install summary</p> <p>Example:</p> <pre>Device# show install summary</pre>	<p>Displays information about the active package.</p> <ul style="list-style-type: none"> 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: nesc: 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: vtyserverutil:
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: ned:
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&also-supported=report-all-tagged</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?
revision=2011-06-01&module=ietf-netconf-with-defaults</capability>
<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-aaa?module=
Cisco-IOS-XE-aaa&revision=2017-02-07</capability>
<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-native?module=
Cisco-IOS-XE-native&revision=2017-01-07&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 31: 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	<p>This module describes how to update YANG data models through In-Service Model Update.</p> <p>In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches <p>In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco 4000 Series Integrated Services Routers • Cisco Cloud Services Router 1000v • Cisco Integrated Services Virtual Routers (ISRv) <p>The following commands were introduced or updated: install (Programmability), show install (Programmability).</p>
	Cisco IOS XE Everest 16.6.1	<p>In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches
	Cisco IOS XE Fuji 16.7.x	<p>In Cisco IOS XE Fuji 16.7.x, this feature was implemented on the following platform:</p> <ul style="list-style-type: none"> • Cisco 1000 Series Aggregated Services Routers
	Cisco IOS XE Fuji 16.8.1a	<p>In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches</p>



PART **IV**

OpenFlow

- [OpenFlow](#) , on page 207



CHAPTER 14

OpenFlow

This module describes how to enable and configure OpenFlow on a device.

- [Prerequisites for OpenFlow, on page 207](#)
- [Restrictions for OpenFlow, on page 207](#)
- [Information About OpenFlow, on page 208](#)
- [How to Configure OpenFlow, on page 212](#)
- [Configuration Examples for OpenFlow, on page 216](#)
- [Verifying OpenFlow, on page 217](#)
- [Additional References for OpenFlow, on page 220](#)
- [Feature Information for OpenFlow, on page 220](#)

Prerequisites for OpenFlow

The device must be booted up in OpenFlow mode.

OpenFlow mode is enabled, when you configure the **boot mode openflow** command on a switch. All ports will be in this mode, and the switch will not support any regular Cisco IOS XE features.

Restrictions for OpenFlow

- When enabling OpenFlow mode on a device, erase all prior configurations, and delete the `vlan.dat` and `stby-vlan.dat` files from the flash filesystem.

When the device is in Openflow mode, do not enable other control plane protocols (protocols such as Border Gateway Protocol [BGP], Spanning Tree Protocol [STP], Port Channels, StackWise and so on) that work when the device is in normal mode.

Information About OpenFlow

OpenFlow Overview

OpenFlow is a specification from the Open Networking Foundation (ONF) that defines a flow-based forwarding infrastructure and a standardized application-programmatic interface. OpenFlow allows a controller to direct the forwarding functions of a device through a secure channel.

OpenFlow is the protocol between the controller (control plane) and the Ethernet switch (data plane). The switch/device has flow tables arranged into a pipeline. Flows are rules to examine packets that reach these tables.

An OpenFlow agent on the switch communicates with the controller using the OpenFlow protocol. The agent supports both OpenFlow 1.0 (wire protocol 0x1) and OpenFlow 1.3 (wire protocol 0x4). It can have up to 8 controller connections. These connections are not preserved across switchover, and the controller will have to reconnect to the agent.

The OpenFlow implementation on Cisco Catalyst 9400 Series Switches is stateless; and non-stop forwarding (NSF) is not supported. The standby supervisor does not sync to the flow database.

OpenFlow Controller

The OpenFlow controller is an entity that interacts with the OpenFlow switch using the OpenFlow protocol. In most cases, the controller is a software that manages many OpenFlow logical switches. Controllers offer a centralized view of the network, and enable administrators to dictate to the underlying systems (switches and routers) on how to handle the network traffic. A controller typically runs on a Linux server, and must have IP connectivity to OpenFlow-capable switches.

The controller manages the switch, and inserts and deletes flows on the switch. These flows support a subset of OpenFlow 1.3 and 1.0 *match* and *action* criteria.

The switch connects to the controller using the management port. The management port is in the management virtual routing and forwarding (VRF) instance, and hence provides a secure connection to the controller. To connect a controller to the switch, configure the IP address and port number on which the controller can be reached.

Flow Management

A flow entry is an element in a flow table that is used to match and process packets. It contains a priority for matching precedence, a set of match fields for matching packets, a set of instructions to apply, and packet and byte counters. A timeout is also associated with each flow (a hard timeout or an inactivity timeout), which is used to automatically remove flows.

Cisco Catalyst 9000 Series Switches support a maximum of 9 flow tables.

Each flow provides the following information:

- **Priority:** High priority flows are matched first. Any flow update requires all flows to be prioritized based on the configured priority.
- **Match fields:** A part of a flow entry against which a packet is matched. Match fields can match the various packet header fields. If no match information is provided for a field; a wildcard is used.

- Action: An operation that acts on a packet.

OpenFlow Pipeline

An OpenFlow pipeline is a set of linked flow tables that provide matching, forwarding, and packet modification in an OpenFlow switch. A port is where packets enter and exits the pipeline.

Packets are received on an ingress port and processed by the OpenFlow pipeline that forwards it to output ports. The packet ingress port is owned by the packet throughout the pipeline, and represents the port on which the packet was received into the switch. The ingress port can also be used as a match field in a flow.

Flow actions allow packets to be sent to subsequent tables in the pipeline for further processing, and allow information to be communicated between tables. Pipeline processing stops when the action associated with a matching flow entry does not specify the next table. At this point, the packet is usually modified and forwarded. The packet can also be dropped.

Flow tables of an OpenFlow switch are sequentially numbered, starting from 0. Pipeline processing always starts by matching the packet against flow entries of flow table 0. Other flow tables may be used depending on the outcome of the match and actions in the first table, which could result in matching the packet against flow entries in subsequent tables.

Supported Match Fields and Actions

Match Field is a field against which a packet is matched, including packet headers, and the ingress port. A match field can be a wildcard (match any value) and have a bit mask to match selected bits of the field.

Action is an operation that forwards a packet to a port or subsequent tables, or modifies a packet field. Actions may be specified as part of the instructions associated with a flow entry, or an action bucket associated with a group entry. A group entry is a collection of actions that can be shared by multiple flows.

The action specified in one or more flow entries can direct packets to a base action called a group action. The purpose of the group action is to share a set of actions among multiple flows. A group consist of one or more buckets, and in turn, a bucket can have a set of actions (set, pop, or output). Cisco Catalyst 9000 Series Switches support group types, *all* and *indirect*.

This section lists the supported match fields and actions:

Table 32: Supported Match Fields

Header Fields	Prerequisites	Maskable Entries	Example Value
Ethernet destination MAC address		Yes	01:80:c2:00:00:00/ ff:ff:ff:00:00:00 (with mask) de:f3:50:c7:e2:b2 (without mask)
Ethernet source MAC address		Yes	0e:00:00:00:00:019 (without mask)
Ethernet type			ARP (0x0806), IPv4 (0x0800), IPv6 (0x86dd), and so on

Header Fields	Prerequisites	Maskable Entries	Example Value
VLAN ID			0x13f
ARP target protocol address	Ethernet type should be set to 0x0806	Yes	
IP protocol	Ethernet type should be set to 0x0800 or 0x86dd		ICMP (0x01), TCP (0x06), UDP (0x11), and so on
IPv4 source address	Ethernet type should be set to 0x0800	Yes	10.0.0.0/24 (with mask)
IPv4 destination address	Ethernet type should be set to 0x0800	Yes	10.0.0.254 (without mask)
IPv6 source address	Ethernet type should be set to 0x08dd	Yes	2001:DB8::1 (without mask)
IPv6 destination address	Ethernet type should be set to 0x08dd	Yes	2001:DB8:0:ABCD::1/48 (with mask)
Neighbor Discovery Target	Ethernet type should be set to 0x08dd and IP protocol should be set to 0x01		ND target
ICMPv6 type	Ethernet type should be set to 0x08dd and IP protocol should be set to 0x01		
UDP/TCP source port	Ethernet type should be set to 0x0800 or 0x86dd and protocol should be set to 0x06 or 0x11		
UDP/TCP destination port	Ethernet type should be set to 0x0800 or 0x86dd and protocol should be set to 0x06 or 0x11		
Incoming interface			

Supported Actions

A flow can send a packet to:

- The controller
- Any interface of the switch (including the incoming interface)
- A subsequent flow table (after table 0) for another lookup

- To a group

A flow can also modify the following header fields of a packet:

- Ethernet destination MAC address
- Ethernet source MAC address
- VLAN ID

A flow can add (push) or remove (pop) a VLAN tag. If the packet is an IP packet, the flow can decrement the Time to Live (TTL) header field.

Flow Operations

This section describes the operations that take place when a flow is sent by the controller to be programmed in the OpenFlow device.

The device has flow tables arranged into a pipeline. The pipeline capabilities information specifies the structure of the pipeline, such as the number of tables/stages, what each stage is capable of doing (match/actions), and the size of each table.

When the controller sends a flow request, the OpenFlow agent verifies whether the flow can be handled by the hardware. It compares the flow against the capabilities of the hardware that are defined when the switch is booted up. If the flow is valid, it is then programmed in the appropriate flow table.

If the new pipeline is validated (whether the hardware can support the pipeline), it becomes the new set of capabilities that is used to check if a flow can be installed or not.

Once the pipeline is instantiated and flows are installed, packets are forwarded by the switch. Ingress packets are matched against flows in each flow table, until the highest-priority matching flow entry is found. Packet matching may be exact (match all fields of the table exactly), or partial (match some or all fields, and fields with bit masks may be partially matched). Packets may be modified or forwarded based on the configured actions. Actions can be applied at any time in the pipeline. The action may determine the next flow table to match, the set of egress ports for the packet, and whether the packet should be routed to the controller.

OpenFlow Table Pipeline

OpenFlow table feature request messages allow an OpenFlow controller to query the capabilities of existing flow tables of an OpenFlow-managed device, or configure these tables to match the supplied configuration.

All tables may be configured with any subset of the match and action capabilities. Table sizes can also be modified at runtime. When a new flow table configuration is successfully applied, flow entries from old flow tables are removed without any notification. Dynamically configured flow tables are not persistent across reboot. The default pipeline comes up when the device boots up.

While configuring a new flow table based on a request from the OpenFlow controller, any ongoing traffic flowing via the existing flows are dropped.

How to Configure OpenFlow

Enabling OpenFlow Mode on a Device

If the switch is operating in normal mode, we recommend that you configure the **write erase** command to delete the previous configuration.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **boot mode openflow**
4. **exit**
5. **write erase**
6.
 - **delete flash:vlan.dat**
 - **delete flash:stby-vlan.dat**
7. **reload**
8. **enable**
9. **show boot mode**

DETAILED STEPS

	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 mode openflow Example: Device(config)# boot mode openflow	Enables OpenFlow forwarding mode.
Step 4	exit Example: Device(config)# exit	Exits global configuration mode and enters privileged EXEC mode.
Step 5	write erase Example: Device# write erase	Erases all files in the NVRAM. • This is recommended if the device was operating in normal mode previously.
Step 6	• delete flash:vlan.dat	Deletes the vlan.dat file that stores the VLAN information.

	Command or Action	Purpose
	<ul style="list-style-type: none"> • delete flash:stby-vlan.dat Example: Device# delete flash:vlan.dat Device# delete flash:stby-vlan.dat	<ul style="list-style-type: none"> • Delete the stby-vlan.dat file, if you have a standby device.
Step 7	reload Example: Device# reload	Reloads the switch and enables OpenFlow forwarding mode for the switch.
Step 8	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted.
Step 9	show boot mode Example: Device# show boot mode	Displays information about the device forwarding mode.

Example

The following sample output from the **show boot mode** command shows that the device is in OpenFlow mode:

```
Device# show boot mode

System initialized in openflow forwarding mode
System configured to boot in openflow forwarding mode
```

What to do next

To go back to normal mode, configure the **no boot mode openflow** command and then reload the device.

Configuring OpenFlow

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **feature openflow**
4. **openflow**
5. **switch 1 pipeline 1**
6. **controller ipv4 ip-address port port-number vrf vrf-name security {none | tls}**
7. **datapath-id ID**
8. **tls trustpoint local name remote name**
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> • Enter your password if prompted
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	feature openflow Example: Device(config)# feature openflow	Enables the OpenFlow feature.
Step 4	openflow Example: Device(config)# openflow	Enables OpenFlow configuration and enters OpenFlow configuration mode.
Step 5	switch 1 pipeline 1 Example: Device(config-openflow)# switch 1 pipeline 1	Configures a logical switch and pipeline, and enters OpenFlow switch configuration mode.
Step 6	controller ipv4 ip-address port port-number vrf vrf-name security {none tls} Example: Device(config-openflow-switch)# controller ipv4 10.2.2.2 port 6633 vrf Mgmt-vrf security tls	Connects to a controller. <ul style="list-style-type: none"> • You must configure the tls trustpoint command, if you have configured TLS as the OpenFlow controller connection security option. • You do not need to configure tls trustpoint command, if you have not configured any security option for the OpenFlow controller.
Step 7	datapath-id ID Example: Device(config-openflow-switch)# datapath-id 0x12345678	(Optional) Sets the OpenFlow logical switch ID. <ul style="list-style-type: none"> • The <i>ID</i> argument specifies the Switch ID, which is a hexadecimal value.
Step 8	tls trustpoint local name remote name Example: Device(config-openflow-switch)# tls trustpoint local trustpoint1 remote trustpoint1	(Optional) Configures an OpenFlow switch Transport Layer Security (TLS) trustpoint.
Step 9	end Example: Device(config-openflow-switch)# end	Exits OpenFlow switch configuration mode and returns to privileged EXEC mode.

Configuring Interfaces in OpenFlow Mode

You can either configure a Layer 2 or Layer 3 interface in OpenFlow mode. When using a Layer 3 interface, configure the **no switchport** command in interface configuration mode. Perform the following task when using a Layer 2 interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **feature openflow**
4. **interface** *type number*
5. **switchport mode trunk**
6. **switchport nonnegotiate**
7. **no keepalive**
8. **spanning-tree bpdupfilter enable**
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	enable Example: Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none">• Enter your password if prompted
Step 2	configure terminal Example: Device# configure terminal	Enters global configuration mode.
Step 3	feature openflow Example: Device(config)# feature openflow	Enables the OpenFlow feature.
Step 4	interface <i>type number</i> Example: Device(config)# interface gigabitethernet 1/0/3	Configures an interface and enters interface configuration mode.
Step 5	switchport mode trunk Example: Device(config-if)# switchport mode trunk	Sets the trunking mode of the Layer 2-switched interface to trunk.
Step 6	switchport nonnegotiate Example: Device(config-if)# switchport nonnegotiate	Specifies that the device will not engage in negotiation protocol on this interface.

	Command or Action	Purpose
Step 7	no keepalive Example: Device(config-if)# no keepalive	Disables keepalive packets.
Step 8	spanning-tree bpdudfilter enable Example: Device(config-if)# spanning-tree bpdudfilter enable	Enables bridge protocol data unit (BPDU) filtering on the interface.
Step 9	end Example: Device(config-if)# end	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for OpenFlow

Example: Enabling OpenFlow on a Device

```

Device> enable
Device# configure terminal
Device(config)# boot mode openflow
Device(config)# exit
Device# write erase
Device# delete flash:vlan.dat
Device# reload
Device> enable
Device# show boot mode

```

Example: Configuring OpenFlow

```

Device# configure terminal
Device(config)# feature openflow
Device(config)# openflow
Device(config-openflow)# switch 1 pipeline 1
Device(config-openflow-switch)# controller ipv4 10.2.2.2 port 6633 vrf Mgmt-vrf security
tls
Device(config-openflow-switch)# datapath-id 0x12345678
Device(config-openflow-switch)# tls trustpoint local trustpoint1 remote trustpoint1
Device(config-openflow-switch)# end

```

Example: Configuring Interfaces in OpenFlow Mode

You can either configure a Layer 2 or Layer 3 interface in OpenFlow mode. When using a Layer 3 interface, configure the **no switchport** command in interface configuration mode. Perform the following task when using a Layer 2 interface.

```
Device> enable
Device# configure terminal
Device(config)# feature openflow
Device(config)# interface gigabitethernet 1/0/3
Device(config-if)# switchport mode trunk
Device(config-if)# switchport nonnegotiate
Device(config-if)# no keepalive
Device(config-if)# spanning-tree bpdufilter enable
Device(config-if)# end
```

Verifying OpenFlow

SUMMARY STEPS

1. **enable**
2. **show openflow hardware capabilities**
3. **show openflow switch 1 controller**
4. **show openflow switch 1 ports**
5. **show openflow switch 1 flows list**

DETAILED STEPS

Step 1 **enable**

Enables privileged EXEC mode.

- Enter your password if prompted.

Example:

```
Device> enable
```

Step 2 **show openflow hardware capabilities**

Displays the hardware capabilities of an OpenFlow device.

Example:

```
Device# show openflow hardware capabilities
```

```
Max Interfaces: 1000
Aggregated Statistics: YES
```

```
Pipeline ID: 1
Pipeline Max Flows: 2322
Max Flow Batch Size: 100
```

```

Statistics Max Polling Rate (flows/sec): 10000
Pipeline Default Statistics Collect Interval: 5

```

```
Flow table ID: 0
```

```

Max Flow Batch Size: 100
Max Flows: 1022
Bind Subintfs: FALSE
Primary Table: TRUE
Table Programmable: TRUE
Miss Programmable: TRUE
Number of goto tables: 1
Goto table id: 1
Number of miss goto tables: 1
Miss Goto table id: 1
Stats collection time for full table (sec): 1

```

```

!
!
!

```

Step 3 show openflow switch 1 controller

Displays information about the controller connected to the switch.

Example:

```
Device# show openflow switch 1 controller
```

```

Logical Switch Id: 1
Total Controllers: 1
Controller: 1
10.10.23.200:6633
Protocol: tcp
VRF: Mgmt-vrf
Connected: Yes
Role: Equal
Negotiated Protocol Version: OpenFlow 1.3
Last Alive Ping: 2018-06-04 17:59:20 PDT
state: ACTIVE
sec_since_connect: 50

```

Step 4 show openflow switch 1 ports

Displays information about the ports on an OpenFlow switch.

Example:

```
Device# show openflow switch 1 ports
```

```

Logical Switch Id: 1

```

Port	Interface Name	Config-State	Link-State	Features
1	Gi1/0/1	PORT_UP	LINK_UP	1GB-FD
2	Gi1/0/2	PORT_UP	LINK_UP	1GB-FD
3	Gi1/0/3	PORT_UP	LINK_UP	1GB-FD
4	Gi1/0/4	PORT_UP	LINK_UP	1GB-FD
5	Gi1/0/5	PORT_UP	LINK_DOWN	1GB-HD
6	Gi1/0/6	PORT_UP	LINK_DOWN	1GB-HD
7	Gi1/0/7	PORT_UP	LINK_DOWN	1GB-HD
8	Gi1/0/8	PORT_UP	LINK_DOWN	1GB-HD
9	Gi1/0/9	PORT_UP	LINK_UP	1GB-FD
10	Gi1/0/10	PORT_UP	LINK_UP	1GB-FD
11	Gi1/0/11	PORT_UP	LINK_UP	1GB-FD
12	Gi1/0/12	PORT_UP	LINK_UP	1GB-FD
13	Gi1/0/13	PORT_UP	LINK_DOWN	1GB-HD

14	Gi1/0/14	PORT_UP	LINK_DOWN	1GB-HD
15	Gi1/0/15	PORT_UP	LINK_DOWN	1GB-HD
16	Gi1/0/16	PORT_UP	LINK_DOWN	1GB-HD
17	Gi1/0/17	PORT_UP	LINK_DOWN	1GB-HD
18	Gi1/0/18	PORT_UP	LINK_DOWN	1GB-HD
19	Gi1/0/19	PORT_UP	LINK_UP	1GB-FD
20	Gi1/0/20	PORT_UP	LINK_UP	1GB-FD
21	Gi1/0/21	PORT_UP	LINK_UP	1GB-FD
22	Gi1/0/22	PORT_UP	LINK_UP	1GB-FD
23	Gi1/0/23	PORT_UP	LINK_DOWN	1GB-HD
24	Gi1/0/24	PORT_UP	LINK_DOWN	1GB-HD
25	Gi1/1/1	PORT_UP	LINK_DOWN	1GB-HD
26	Gi1/1/2	PORT_UP	LINK_DOWN	1GB-HD
27	Gi1/1/3	PORT_UP	LINK_DOWN	1GB-HD
28	Gi1/1/4	PORT_UP	LINK_DOWN	1GB-HD
29	Te1/1/1	PORT_UP	LINK_DOWN	10GB-FD
30	Te1/1/2	PORT_UP	LINK_DOWN	10GB-FD
31	Te1/1/3	PORT_UP	LINK_DOWN	10GB-FD
32	Te1/1/4	PORT_UP	LINK_DOWN	10GB-FD
33	Te1/1/5	PORT_UP	LINK_DOWN	10GB-FD
34	Te1/1/6	PORT_UP	LINK_DOWN	10GB-FD
35	Te1/1/7	PORT_UP	LINK_DOWN	10GB-FD
36	Te1/1/8	PORT_UP	LINK_DOWN	10GB-FD
37	Fo1/1/1	PORT_UP	LINK_DOWN	40GB-FD
38	Fo1/1/2	PORT_UP	LINK_DOWN	40GB-FD
39	Twe1/1/1	PORT_UP	LINK_DOWN	10GB-FD
40	Twe1/1/2	PORT_UP	LINK_DOWN	10GB-FD

Step 5 show openflow switch 1 flows list

Displays OpenFlow entries.

The following sample output displays a flow that is installed in table 0, where *match any* goes to Table 1. (Match any means that all packets go to Table 1.) In table 1, the destination MAC address 00:00:01:00:00:01 is matched, and the output port is set to 36.

Example:

```
Device# show openflow switch 1 flows list
```

```
Logical Switch Id: 1
Total flows: 8
```

```
Flow: 1 Match: any Actions: goto_table:1, Priority: 9000, Table: 0, Cookie: 0x1,
Duration: 2382.117s, Packets: 34443, Bytes: 3359315
```

```
Flow: 2 Match: any Actions: drop, Priority: 0, Table: 0, Cookie: 0x0,
Duration: 2382.118s, Packets: 294137, Bytes: 28806211
```

```
Flow: 3 Match: any Actions: drop, Priority: 0, Table: 1, Cookie: 0x0,
Duration: 2382.118s, Packets: 34443, Bytes: 3359315
```

```
Flow: 4 Match: dl_dst=00:00:01:00:00:01 Actions: output:36, Priority: 9000,
Table: 1, Cookie: 0x1, Duration: 2382.116s, Packets: 0, Bytes: 0
```

Additional References for OpenFlow

Related Documents

Related Topic	Document Title
OpenFlow commands	<i>Programmability Command Reference</i>
Open Network Foundation	https://www.opennetworking.org/
Faucet OpenFlow controller	<ul style="list-style-type: none"> • https://faucet.nz/ • https://docs.faucet.nz/en/latest/

MIBs

MIB	MIBs Link
	<p>To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL:</p> <p>http://www.cisco.com/go/mibs</p>

Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	http://www.cisco.com/support

Feature Information for OpenFlow

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 33: Feature Information for OpenFlow

Feature Name	Release	Feature Information
OpenFlow	Cisco IOS XE Fuji 16.9.1	<p>OpenFlow is a Software-Defined Network (SDN) standard. It defines a communication protocol in SDN environments that enables an SDN controller to directly interact with the forwarding plane of network devices such as switches and routers.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Catalyst 9300 Series Switches • Catalyst 9400 Series Switches • Catalyst 9500 Series Switches • Catalyst 9500 Series High Performance Switches
	Cisco IOS XE Gibraltar 16.10.1	Table feature message support on Catalyst 9500 Series High Performance Switches was introduced.
OpenFlow Power over Ethernet	Cisco IOS XE Gibraltar 16.12.1	<p>PoE is supported on OpenFlow ports.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> • Catalyst 9300 Series Switches • Catalyst 9400 Series Switches

