Programmability Configuration Guide, Cisco IOS XE Everest 16.6.x

First Published: 2017-07-31

Last Modified: 2017-11-03

C O N T E N T S

New and Changed Information

This chapter provides release-specific information about all features.

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	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches

Feature	Release & Platform
Zero Touch Provisioning: HTTP Copy	Cisco IOS XE Everest 16.6.1 • Cisco 4000 Series Integrated Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches
iPXE	Cisco IOS XE Denali 16.3.2 and Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3650 Series Switches Cisco IOS XE Everest 16.6.1 • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches
Shells and Scripting	
Guest Shell	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches

Feature	Release & Platform
Python APIs	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches
Python CLI Module	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches
EEM Python Module	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.2. • Cisco Catalyst 9400 Series Switches
Model-Driven Programmability	

Feature	Release & Platform
NETCONF Network Management Interface	Cisco IOS XE Denali 16.3.1 • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches
Model-Driven Telemetry NETCONF Dial-In	Cisco IOS XE Everest 16.6.1 • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches
In-Service Model Updates	Cisco IOS XE Everest 16.5.1a • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches Cisco IOS XE Everest 16.5.1b • Cisco 4000 Series Integrated Services Routers Cisco IOS XE Everest 16.6.1 • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches Cisco IOS XE Everest 16.6.2 • Cisco Catalyst 9400 Series Switches

Feature	Release & Platform
RESTCONF Network Management Interface	Cisco IOS XE Everest 16.6.1 • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Aggregation Services Routers (ASR1001-HX and ASR1002-HX) • Cisco Cloud Services Router 1000V Series

PART **I**

Provisioning

C H A P T E R 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.

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

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.

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 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.executep(cli_command)

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

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


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

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

Boot Log for Cisco 4000 Series Integrated Services Routers

The following sample Zero-Touch Provisioning boot log displays that Guest Shell is successfully enabled, the Python script is downloaded to the Guest Shell, and the Guest Shell 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
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/wwl/export/crypto/tool/stqrg.html

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

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

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


%INIT: waited 0 seconds for NVRAM to be available

SETUP: new interface Vlan1 placed in "shutdown" state


Press RETURN to get started!


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

*Sep  4 20:35:13.406: %CRYPTO-4-AUDITWARN: Encryption audit check could not be performed
*Sep  4 20:35:13.480: %SPANTREE-5-EXTENDED_SYSID: Extended SysId enabled for type vlan
*Sep  4 20:35:13.715: %LINK-3-UPDOWN: Interface Lsmpi18/3, changed state to up
*Sep  4 20:35:13.724: %LINK-3-UPDOWN: Interface EOBC18/1, changed state to up
*Sep  4 20:35:13.724: %LINEPROTO-5-UPDOWN: Line protocol on Interface LI-Null0, changed
state to up
*Sep  4 20:35:13.724: %LINK-3-UPDOWN: Interface GigabitEthernet0/0, changed state to down
*Sep  4 20:35:13.725: %LINK-3-UPDOWN: Interface LIIN18/2, changed state to up
*Sep  4 20:35:13.749: WCM-PKI-SHIM: buffer allocation failed for SUDI support check
*Sep  4 20:35:13.749: PKI/SSL unable to send Sudi support to WCM
*Sep  4 20:35:14.622: %IOSXE_MGMTVRF-6-CREATE_SUCCESS_INFO: Management vrf Mgmt-vrf created
 with ID 1,
    ipv4 table-id 0x1, ipv6 table-id 0x1E000001
*Sep  4 20:34:42.022: %STACKMGR-6-STACK_LINK_CHANGE: Switch 1 R0/0: stack_mgr:  Stack port
 1 on Switch 1 is nocable
*Sep  4 20:34:42.022: %STACKMGR-6-STACK_LINK_CHANGE: Switch 1 R0/0: stack_mgr:  Stack port
 2 on Switch 1 is down
*Sep  4 20:34:42.022: %STACKMGR-6-STACK_LINK_CHANGE: Switch 1 R0/0: stack_mgr:  Stack port
```

```
 2 on Switch 1 is nocable
*Sep  4 20:34:42.022: %STACKMGR-6-SWITCH_ADDED: Switch 1 R0/0: stack_mgr:  Switch 1 has
been added to the stack.
*Sep  4 20:34:42.022: %STACKMGR-6-SWITCH_ADDED: Switch 1 R0/0: stack_mgr:  Switch 1 has
been added to the stack.
*Sep  4 20:34:42.022: %STACKMGR-6-SWITCH_ADDED: Switch 1 R0/0: stack_mgr:  Switch 1 has
been added to the stack.
*Sep  4 20:34:42.022: %STACKMGR-6-ACTIVE_ELECTED: Switch 1 R0/0: stack_mgr:  Switch 1 has
been elected ACTIVE.
*Sep  4 20:35:14.728: %LINEPROTO-5-UPDOWN: Line protocol on Interface Lsmpi18/3, changed
state to up
*Sep  4 20:35:14.728: %LINEPROTO-5-UPDOWN: Line protocol on Interface EOBC18/1, changed
state to up
*Sep  4 20:35:15.506: %HMANRP-6-HMAN_IOS_CHANNEL_INFO: HMAN-IOS channel event for switch
1: EMP_RELAY: Channel UP!
*Sep  4 20:35:15.510: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan1, changed state
 to down
*Sep  4 20:35:34.501: %LINK-5-CHANGED: Interface Vlan1, changed state to administratively
down
*Sep  4 20:35:34.717: %SYS-5-RESTART: System restarted --
Cisco IOS Software [Everest], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.6.5,
RELEASE SOFTWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
Compiled Mon 10-Dec-18 12:52 by mcpre
*Sep  4 20:35:34.796: %LINK-3-UPDOWN: Interface GigabitEthernet0/0, changed state to up
*Sep  4 20:35:35.266: %SYS-6-BOOTTIME: Time taken to reboot after reload =  283 seconds
*Sep  4 20:35:35.796: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0,
changed state to up
*Sep  4 20:35:36.607: %LINK-3-UPDOWN: Interface GigabitEthernet1/1/1, changed state to down
*Sep  4 20:35:36.607: %LINK-3-UPDOWN: Interface GigabitEthernet1/1/2, changed state to down
*Sep  4 20:35:36.607: %LINK-3-UPDOWN: Interface GigabitEthernet1/1/3, changed state to down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface GigabitEthernet1/1/4, changed state to down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/1, changed state to
down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/2, changed state to
down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/3, changed state to
down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/4, changed state to
down
*Sep  4 20:35:36.608: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/5, changed state to
down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/6, changed state to
down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/7, changed state to
down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/1/8, changed state to
down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface FortyGigabitEthernet1/1/1, changed state
to down
*Sep  4 20:35:36.609: %LINK-3-UPDOWN: Interface FortyGigabitEthernet1/1/2, changed state
to down
*Sep  4 20:35:37.607: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/1/1,
 changed state to down
*Sep  4 20:35:37.608: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/1/2,
 changed state to down
*Sep  4 20:35:37.608: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/1/3,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/1/4,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/1/1,
 changed state to down
*Sep  4 20:35:37.609: %LINEPROTO-5-UPDOWN: Line protocol on Interface TenGigabitEthernet1/1/2,
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    app-hosting guestshell enable

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

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

*Sep  4 20:36:39.416: %LINK-3-UPDOWN: Interface TenGigabitEthernet1/0/48, changed state to
```

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

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

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

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

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


 *** Sample ZTP Day0 Python Script ***



 *** Executing show platform  ***


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

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


 *** Executing show version ***


Cisco IOS XE Software, Version 16.06.05
Cisco IOS Software [Everest], Catalyst L3 Switch Software (CAT9K_IOSXE), Version 16.6.5,
RELEASE SOFTWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2018 by Cisco Systems, Inc.
Compiled Mon 10-Dec-18 12:52 by mcpre
Cisco IOS-XE software, Copyright (c) 2005-2018 by cisco Systems, Inc.
All rights reserved.  Certain components of Cisco IOS-XE software are
licensed under the GNU General Public License ("GPL") Version 2.0.  The
software code licensed under GPL Version 2.0 is free software that comes
with ABSOLUTELY NO WARRANTY.  You can redistribute and/or modify such
GPL code under the terms of GPL Version 2.0.  For more details, see the
documentation or "License Notice" file accompanying the IOS-XE software,
or the applicable URL provided on the flyer accompanying the IOS-XE
software.
ROM: IOS-XE ROMMON
BOOTLDR: System Bootstrap, Version 17.2.1r[FC1], RELEASE SOFTWARE (P)
Switch uptime is 2 minutes
Uptime for this control processor is 4 minutes
System returned to ROM by Reload Command
System image file is "flash:cat9k_iosxe.16.06.05.SPA.bin"
Last reload reason: Reload Command
This product contains cryptographic features and is subject to United
```

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


 *** Configuring a Loopback Interface ***


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


 *** Executing show ip interface brief  ***


Interface           IP-Address      OK? Method Status                Protocol
Vlan1               unassigned      YES unset  administratively down down
GigabitEthernet0/0  10.127.128.3    YES DHCP   up                    up
Tw1/0/1             unassigned      YES unset  down                  down
Tw1/0/2             unassigned      YES unset  down                  down
Tw1/0/3             unassigned      YES unset  down                  down
Tw1/0/4             unassigned      YES unset  down                  down
Tw1/0/5             unassigned      YES unset  down                  down
Tw1/0/6             unassigned      YES unset  down                  down
Tw1/0/7             unassigned      YES unset  down                  down
Tw1/0/8             unassigned      YES unset  down                  down
```

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


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



Line 1 SUCCESS: username cisco privilege 15 password cisco
Line 2 SUCCESS: ip domain name domain
Line 3 SUCCESS: line vty 0 15
Line 4 SUCCESS: login local
```

```
Line 5 SUCCESS: transport input all
Line 6 SUCCESS: end


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

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	To address network provisioning challenges, Cisco introduces a zero-touch provisioning model. In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platform: • Cisco 4000 Series Integrated Services Router models with a minimum of 8 GB RAM to support Guestshell.
Zero-Touch Provisioning: HTTP Download	Cisco IOS XE Everest 16.6.1	Zero-Touch Provisioning supports HTTP and TFTP file download. In Cisco IOS XE Everest 16.6.1, this feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches
	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.

C H A P T E R 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

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 address auto-configuration (SLAAC) and stateful IP auto-configuration variants for Dynamic Host Configuration Protocol Version 6 (DHCPv6), boot URI, and parameters for DHCPv6 options depending on the IPv6 router advertisement.

Note IPv6 is not supported on Catalyst 9000 Series Switches.

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 iPXE 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. 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 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.

The following section describes how an iPXE bootloader works:

Figure 1: iPXE Bootloader Workflow
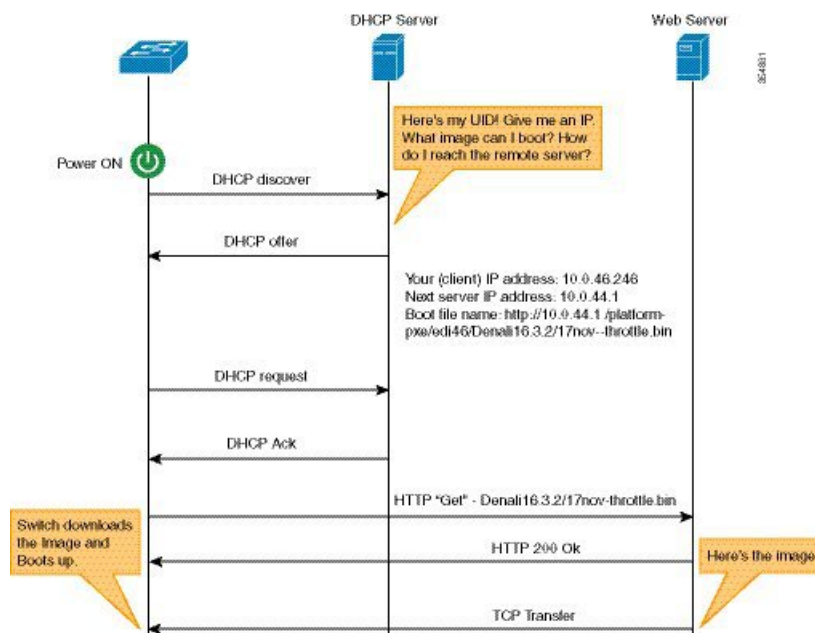
1. Bootloader sends a DHCP request.

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

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.

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 device IP address is taken from the IP_ADDRESS variable. 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 rommon variables should be configured for iPXE:

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

IPv6 iPXE Network Boot

IPv6 is not supported on Catalyst 9000 Series Switches.

This illustration displays how IPv6 iPXE network boot works on a Cisco device:
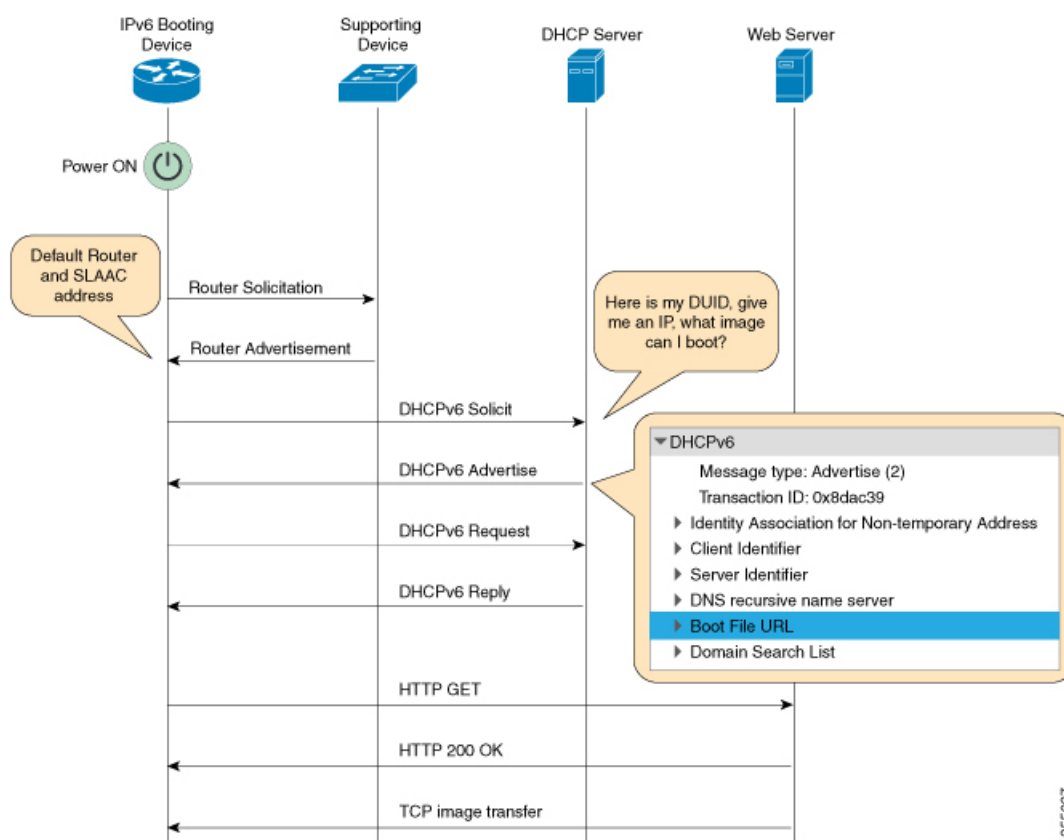
The four elements in the above illustration are described below:

• 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 open source DHCP server.

• Web server—Any open source 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 an RA, ICMPv6 type 134 packet. The device that sent the router solicitation message, gets the default router and prefix information for Stateless Address AutoConfiguration (SLAAC) address completion from the RA packet.

3. The device sends a DHCP Version 6 (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.

   • DHCPv6 Option 3

   • DHCPv6 Option 6

   • DHCPv6 Option 15

   • DHCPv6 Option 16

   • DHCPv6 Option 17

4. If the DHCPv6 server is configured, it responds with a DHCPv6 advertise packet that contains the 128 Bit IPv6 address, the boot file Uniform Resource Identifier (URI), the Domain Name System (DNS) server

and domain search list, and the client and server IDs. The client ID contains the DUID of the client (In this illustration, the IPv6 Booting Device), and the Server ID contains the DUID of the DHCPv6 server.

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

6. The server responds with a unicast DHCPv6 reply to the Link Local (FE80::) IPv6 address of the client. The following sample displays the fields in a DHCPv6 reply packet:

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

7. The device then sends an HTTP GET request to the web server.

8. If the requested image is available at the specified path, the web server responds with an OK for the HTTP GET request.

9. The TCP image transfer copies the image, and the device boots up.

IPv6 Address Assignment in Rommon Mode

Note   IPv6 is not supported on Catalyst 9000 Series Switches.

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

1. DHCP Server-assigned address

2. Stateless Address Auto-Configuration (SLAAC) address

3. Link-local address

4. Site-local address

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

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

iPXE-Supported DHCP Options

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

> **Note** Except for DHCP Option 77, the other options are not supported on Catalyst 9000 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 15—User Class Option. This option is the IPv6 User Class option in a DHCPv6 solicit message. 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).
N/A	DHCPv6 Option 3 to request for a non-temporary address.
N/A	DHCPv6 Option 17, the Vendor-Specific option that contains the reserved Enterprise ID *9* for Cisco Systems.
dhcp6.bootfile-url	DHCPv6 Option 6 to request for the Boot File URI

DHCPv6 Unique Identifiers

IPv6 is not supported on Catalyst 9000 Series Switches.

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 use the DUID-EN (DUID Type 2) to identify the DHCP client (that is the device in the DHCPv6 Solicit packet).

How to Configure iPXE

Configuring iPXE

Procedure

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

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

Step 4	**boot system** {**switch** *switch-number* | **all**} {**flash:** | **ftp:** | **http:** | **tftp:**}
	
	Example:
	`Device(config)# boot system switch 1 http://192.0.2.42/image-filename`
	or
	`Device(config)# boot system switch 1 http://[2001:db8::1]/image-filename`
		Boots an image from the specified location.
		• You can either use an IPv4 or an IPv6 address for the remote FTP/HTTP/TFTP servers.
		• You must enter the IPv6 address inside the square brackets (as per RFC 2732); if not the device will not boot.
		Note IPv6 is not supported on Catalyst 9000 Series Switches.

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.

Procedure

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

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

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

Configuration Examples for iPXE

Example: iPXE Configuration

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

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

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

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

Note IPv6 is not supported on Catalyst 9000 Series Switches.

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 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. By default, this DHCP option is not supported by the ISC DHCP Server, and to define it 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.

- 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

- 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.

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

- Enable HTTP debug logs to view the HTTP response code from the web server.

- If SLAAC addresses are not generated, there is no router that is providing IPv6 RA messages. iPXE boot for IPv6 can still work but only with link or site-local addresses.

Additional References for iPXE

Related Documents

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

Standards and RFCs

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

Technical Assistance

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

Feature Information for iPXE

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

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

Table 4: Feature Information for iPXE

Feature Name	Release	Feature Information
iPXE	Cisco IOS XE Denali 16.5.1a	Network Bootloaders support booting from an IPv4/IPv6 device-based or network-based source. A network boot source must be detected automatically by using an iPXE-like solution. This feature was implemented on the following platforms: • Catalyst 3650 Series Switches • Catalyst 3850 Series Switches
	Cisco IOS XE Denali 16.6.1	iPXE IPv6 is not supported on Catalyst 9000 Series Switches. This feature was implemented on the following platforms: • Catalyst 9300 Series Switches • Catalyst 9500 Series Switches
	Cisco IOS XE Everest 16.6.2	In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.

P A R T II

Shells and Scripting

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 Guest Shell

Guest Shell Overview

Guestshell is a virtualized Linux-based environment, designed to run custom Linux applications, including Python for automated control and management of Cisco devices. Using Guest Shell, customers can also install, update, and operate third-party Linux applications. It is bundled with the system image and can be installed using the **guestshell enable** 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	• Cisco Catalyst 3650 Series Switches (all models) • Cisco Catalyst 3850 Series Switches (all models)	• 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	• Cisco Embedded Event Manager • Cisco IOS XE CLIs • Ncclient	• 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 Guestshell

This section provides information about the hardware requirements for supported platforms.

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.

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, 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.

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	256/256 MB

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 g 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.

Only the *guest-share* directory under bootflash is shared with Guest Shell. All other file access to bootflash from Guest Shell is prohibited.

During a high availability switchover, the new active device creates its own Guest Shell installation; the old filesystem is not maintained. The Guest Shell state is maintained during a switchover.

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.

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 Guest Shell

Managing IOx

Before you begin

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

Procedure

	Command or Action	**Purpose**
Step 1	**enable**	Enables privileged EXEC mode.
	Example:	• Enter your password if prompted.
	`Device> enable`	
Step 2	**configure terminal**	Enters global configuration mode.
	Example:	
	`Device# configure terminal`	
Step 3	**iox**	Configures IOx services.
	Example:	
	`Device(config)# iox`	
Step 4	**exit**	Exits global configuration mode and returns to
	Example:	privileged EXEC mode.
	`Device(config)# exit`	
Step 5	**show iox-service**	Displays the status of the IOx service
	Example:	
	`Device# show iox-service`	
Step 6	**show app-hosting list**	Displays the list of app-hosting services enabled
	Example:	on the device.
	`Device# show app-hosting list`	

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.

Procedure

	Command or Action	**Purpose**
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	• **guestshell enable**	Enables the Guest Shell service.

	Command or Action	Purpose
	Example: Device# guestshell enable	**Note** • The **guestshell enable** command uses the management virtual routing and forwarding (VRF) instance for networking. • When using VirtualPortGroups (VPGs) for front panel networking, the VPG must be configured first. • The guest IP address and the gateway IP address must be in the same subnet.
Step 3	**guestshell run** *linux-executable* **Example:** Device# guestshell run python	Executes or runs a Linux program in the Guest Shell.
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
```

Disabling and Destroying the Guest Shell

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

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

Accessing 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 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

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
Device(config)# exit
Device#
```

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-bost pypi.example.com --index-url
http://pypi.example.com/simple requests
```

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

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

Additional References for Guest Shell

Related Documents

Related Topic	Document Title
	Programmability Command Reference, Cisco IOS XE Everest 16.6.1
Python module	CLI Python Module
Zero-Touch Provisioning	Zero-Touch Provisioning

MIBs

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

Feature Information for Guest Shell

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

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

Table 9: Feature Information for Guest Shell

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

C H A P T E R 5

Python API

Python programmabililty supports Python APIs.

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(clip)**
```
Help on function clip in module cli:

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

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

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

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

Cisco Python Module to Execute IOS CLI Commands

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

The Python programming language uses six functions that can execute CLI commands. These functions are available from the Python CLI module. To use these functions, execute the **import cli** command. The **ip http server** command must be enabled for these functions to work.

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

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

```
>>> import cli
>>> cli.clip('configure terminal; interface loopback 10; ip address
10.10.10.10 255.255.255.255')
*Mar 13 18:39:48.518: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback10, changed
 state to up
>>> cli.clip('show clock')
'\n*18:11:53.989 UTC Mon Mar 13 2017\n'
>>> output=cli.cli('show clock')
>>> print(output)
```

```
*18:12:04.705 UTC Mon Mar 13 2017
```

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

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

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

  function:

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

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

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

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

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

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

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

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

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

C H A P T E R 6

CLI Python Module

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

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.

✎

Note The **ip http server** command must be configured for the **import cli** in Python to work.

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
     :
          :
          :
Te1/1/4               unassigned      YES unset  down                  down
Loopback55            10.55.55.55     YES TFTP   up                    up
Loopback66            unassigned      YES manual up                    up


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

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

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

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

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

Supported Python Versions

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

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

Table 10: Python Version Support

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

Python Version	Platform
Python Version 2.7.11	• Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches
Python Version 3.6	Supported in Cisco IOS XE Amsterdam 17.1.1 and later releases. In Cisco IOS XE Amsterdam 17.1.1 and Cisco IOS XE Amsterdam 17.2.1, Python V2 is the default. However, in Cisco IOS XE Amsterdam 17.3.1 and later releases, Python V3 is the default. **Note** Cisco Catalyst 9200 Series Switches do not support Python Version 3.6 in Cisco IOS XE Amsterdam 17.1.1 and Cisco IOS XE Amsterdam 17.2.1. Cisco Catalyst 9200 Series Switches support Python V3 in Cisco IOS XE Amsterdam 17.3.1 and later releases. **Note** Not supported by Cisco Catalyst 3650 Series Switches and Cisco Catalyst 3850 Series Switches.

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

Updating the Cisco CLI Python Module

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

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

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

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

- Standalone tarball installation
- PIP install for the CLI module

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

Feature Information for the 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	Python programmabilty provides a Python module that allows users to interact with IOS using CLIs.
		In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms:
		• Cisco Catalyst 3650 Series Switches
		• Cisco Catalyst 3850 Series Switches
		• Cisco Catalyst 9300 Series Switches
		• Cisco Catalyst 9500 Series Switches
		In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms:
		• Cisco 4000 Series Integrated Services Routers
	Cisco IOS XE Everest 16.6.2	In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.

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

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

Information About EEM Python Module

Python Scripting in EEM

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

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

EEM Python Package

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

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

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

- Action APIs—Perform EEM actions and have default parameters.

- CLI-execution APIs—Run IOS commands, and return the output. The following are the list of CLI-execution APIs:

    - eem_cli_open()

    - eem_cli_exec()

    - eem_cli_read()

    - eem_cli_read_line()

    - eem_cli_run()

    - eem_cli_run_interactive()

    - eem_cli_read_pattern()

    - eem_cli_write()

    - eem_cli_close()

- Environment variables-accessing APIs—Get the list of built-in or user-defined variables. The following are the environment variables-accessing APIs:

    - eem_event_reqinfo ()-Returns the built-in variables list.

    - eem_user_variables()-Returns the current value of an argument.

Python-Supported EEM Actions

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

- Syslog message printing

- Send SNMP traps

- Reload the box

- Switchover to the standby device

- Run a policy

- Track Object read

- Track Object Set

- Cisco Networking Services event generation

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

EEM Variables

An EEM policy can have the following types of variables:

- Event-specific built-in variables—A set of predefinied variables that are populated with details about the event that triggered the policy. The eem_event_reqinfo () API returns the builtin variables list. These variables can be stored in the local machine and used as local variables. Changes to local variables do not reflect in builtin variables.

- User-defined variables—Variables that can be defined and used in policies. The value of these variables can be referred in the Python script. While executing the script, ensure that the latest value of the variable is available. The eem_user_variables() API returns the current value of the argument that is provided in the API.

EEM CLI Library Command Extensions

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

- eem_cli_close()—Closes the EXEC process and releases the VTY and the specified channel handler connected to the command.

- eem_cli_exec—Writes the command to the specified channel handler to execute the command. Then reads the output of the command from the channel and returns the output.

- eem_cli_open—Allocates a VTY, creates an EXEC CLI session, and connects the VTY to a channel handler. Returns an array including the channel handler.

- eem_cli_read()—Reads the command output from the specified CLI channel handler until the pattern of the device prompt occurs in the contents read. Returns all the contents read up to the match.

- eem_cli_read_line()—Reads one line of the command output from the specified CLI channel handler. Returns the line read.

- eem_cli_read_pattern()—Reads the command output from the specified CLI channel handler until the pattern that is to be matched occurs in the contents read. Returns all the contents read up to the match.

- eem_cli_run()—Iterates over the items in the *clist* and assumes that each one is a command to be executed in the enable mode. On success, returns the output of all executed commands and on failure, returns error.

- eem_cli_run_interactive()—Provides a sublist to the *clist* which has three items. On success, returns the output of all executed commands and on failure, returns the error. Also uses arrays when possible as a way of making things easier to read later by keeping expect and reply separated.

- eem_cli_write()—Writes the command that is to be executed to the specified CLI channel handler. The CLI channel handler executes the command.

How to Configure the EEM Python Policy

For the Python script to work, you must enable the Guest Shell. For more information, see the *Guest Shell* chapter.

Registering a Python Policy

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**event manager directory user policy** *path* **Example:** `Device(config)# event manager directory user policy flash:/user_library`	Specifies a directory to use for storing user library files or user-defined EEM policies. **Note** You must have a policy in the specified path. For example, in this step, the eem_script.py policy is available in the flash:/user_library folder or path.
Step 4	**event manager policy** *policy-filename* **Example:** `Device(config)# event manager policy eem_script.py`	Registers a policy with EEM. • The policy is parsed based on the file extension. If the file extension is .py, the policy is registered as Python policy. • EEM schedules and runs policies on the basis of an event specification that is contained within the policy itself. When the **event manager policy** command is invoked, EEM examines the policy and registers it to be run when the specified event occurs.
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.
Step 6	**show event manager policy registered** **Example:**	Displays the registered EEM policies.

	Command or Action	Purpose
	`Device# show event manager policy registered`	
Step 7	**show event manager history events** **Example:** `Device# show event manager history events`	Displays EEM events that have been triggered.

Example

The following is sample output from the **show event manager policy registered** command:

```
Device# show event manager policy registered

No.  Class    Type    Event Type          Trap  Time Registered        Name
1    script   user    multiple            Off   Tue Aug 2 22:12:15 2016   multi_1.py
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi_1.py} sync {yes}
 trigger delay 10.000
  correlate event 1 or event 2
  attribute tag 1 occurs 1
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none


2    script    user    multiple            Off   Tue Aug 2 22:12:20 2016   multi_2.py
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi_2.py} sync {yes}
 trigger
  correlate event 1 or event 2
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none


3    script    user    multiple            Off   Tue Aug 2 22:13:31 2016   multi.tcl
 1: syslog: pattern {COUNTER}
 2: none: policyname {multi.tcl} sync {yes}
 trigger
  correlate event 1 or event 2
  attribute tag 1 occurs 1
 nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none
```

Running Python Scripts as Part of EEM Applet Actions

Python Script: eem_script.py

An EEM applet can include a Python script with an action command. In this example, an user is trying to run a standard Python script as part of the EEM action, however; EEM Python package is not available in the standard Python script. The standard Python script in IOS has a package named *from cli import cli,clip* and this package can be used to execute IOS commands.

```
import sys
from cli import cli,clip,execute,executep,configure,configurep

intf= sys.argv[1:]
```

```
intf = ''.join(intf[0])


print ('This script is going to unshut interface %s and then print show ip interface
brief'%intf)

if intf == 'loopback55':
configurep(["interface loopback55","no shutdown","end"])
else :
cmd='int %s,no shut ,end' % intf
configurep(cmd.split(','))

executep('show ip interface brief')
```

This following is sample output from the **guestshell run python** command.

```
Device#  guestshell run python /flash/eem_script.py loop55

This script is going to unshut interface loop55 and then print show ip interface brief
Line 1 SUCCESS: int loop55
Line 2 SUCCESS: no shut
Line 3 SUCCESS: end
Interface IP-Address OK? Method Status Protocol
Vlan1 unassigned YES NVRAM administratively down down
GigabitEthernet0/0 5.30.15.37 YES NVRAM up up
GigabitEthernet1/0/1 unassigned YES unset down down
GigabitEthernet1/0/2 unassigned YES unset down down
GigabitEthernet1/0/3 unassigned YES unset down down
GigabitEthernet1/0/4 unassigned YES unset up up
GigabitEthernet1/0/5 unassigned YES unset down down
GigabitEthernet1/0/6 unassigned YES unset down down
GigabitEthernet1/0/7 unassigned YES unset down down
GigabitEthernet1/0/8 unassigned YES unset down down
GigabitEthernet1/0/9 unassigned YES unset down down
GigabitEthernet1/0/10 unassigned YES unset down down
GigabitEthernet1/0/11 unassigned YES unset down down
GigabitEthernet1/0/12 unassigned YES unset down down
GigabitEthernet1/0/13 unassigned YES unset down down
GigabitEthernet1/0/14 unassigned YES unset down down
GigabitEthernet1/0/15 unassigned YES unset down down
GigabitEthernet1/0/16 unassigned YES unset down down
GigabitEthernet1/0/17 unassigned YES unset down down
GigabitEthernet1/0/18 unassigned YES unset down down
GigabitEthernet1/0/19 unassigned YES unset down down
GigabitEthernet1/0/20 unassigned YES unset down down
GigabitEthernet1/0/21 unassigned YES unset down down
GigabitEthernet1/0/22 unassigned YES unset down down
GigabitEthernet1/0/23 unassigned YES unset up up
GigabitEthernet1/0/24 unassigned YES unset down down
GigabitEthernet1/1/1 unassigned YES unset down down
GigabitEthernet1/1/2 unassigned YES unset down down
GigabitEthernet1/1/3 unassigned YES unset down down
GigabitEthernet1/1/4 unassigned YES unset down down
Te1/1/1 unassigned YES unset down down
Te1/1/2 unassigned YES unset down down
Te1/1/3 unassigned YES unset down down
Te1/1/4 unassigned YES unset down down
Loopback55 10.55.55.55 YES manual up up

Device#
Jun 7 12:51:20.549: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback55,
changed state to up
Jun 7 12:51:20.549: %LINK-3-UPDOWN: Interface Loopback55, changed state to up
```

The following is a sample script for printing messages to the syslog. This script must be stored in a file, copied to the file system on the device, and registered using the event manager policy file.

```
::cisco::eem::event_register_syslog tag "1"    pattern    COUNTER    maxrun 200

import eem
import time

eem.action_syslog("SAMPLE SYSLOG MESSAGE","6","TEST")
```

The following is sample script to print EEM environment variables. This script must be stored in a file, copied to the file system on the device, and registered using the event manager policy file.

```
::cisco::eem::event_register_syslog tag "1"    pattern    COUNTER    maxrun 200

import eem
import time

c = eem.env_reqinfo()


print "EEM Environment Variables"
for k,v in c.iteritems():
      print "KEY : " + k + str(" ---> ") + v

print "Built in Variables"
for i,j in a.iteritems() :
      print "KEY : " + i + str(" ---> ") + j
```

Adding a Python Script in an EEM Applet

Procedure

	Command or Action	**Purpose**
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**event manager applet** *applet-name* **Example:** `Device(config)# event manager applet interface_Shutdown`	Registers an applet with the Embedded Event Manager (EEM) and enters applet configuration mode.
Step 4	**event** [**tag** *event-tag*] **syslog pattern** *regular-expression* **Example:**	Specifies a regular expression to perform the syslog message pattern match.

	Command or Action	Purpose
	`Device(config-applet)# event syslog pattern "Interface Loopback55, changed state to administratively down"`	
Step 5	**action** *label* **cli command** *cli-string* **Example:** `Device(config-applet)# action 0.0 cli command "en"`	Specifies the IOS command to be executed when an EEM applet is triggered.
Step 6	**action** *label* **cli command** *cli-string* [**pattern** *pattern-string*] **Example:** `Device(config-applet)# action 1.0 cli command "guestshell run python3 /bootflash/eem_script.py loop55"`	Specifies the action to be specified with the **pattern** keyword. • Specify a regular expression pattern string that will match the next solicited prompt.
Step 7	**end** **Example:** `Device(config-applet)# end`	Exits applet configuration mode and returns to privileged EXEC mode.
Step 8	**show event manager policy active** **Example:** `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
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for EEM Python Module

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

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

Table 12: Feature Information for EEM Python Module

Feature Name	Release	Feature Information
EEM Python Module	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	This feature supports Python scripts as EEM policies. No new commands were introduced. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms: • Cisco ISR 4000 Series Integrated Service Routers
	Cisco IOS XE Everest 16.6.2	In Cisco IOS XE Everest 16.6.2, this feature was implemented on Cisco Catalyst 9400 Series Switches.

PART III

Model-Driven Programmability

NETCONF Protocol

Restrictions for the NETCONF Protocol

The NETCONF feature is not supported on a device running dual IOSd configuration or software redundancy.

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 respository, 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.

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. To provide this, perform the following task:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**username** *name* **privilege** *level* **password** *password* **Example:** `Device(config)# username example-name privilege 15 password example_password`	Establishes a user name-based authentication system. Configure the following keywords: • **privilege** *level* —Sets the privilege level for the user. For the programmability feature, it must be 15. • **password** *password*—Sets a password to access the CLI view.

Step 4	**aaa authentication login default local** and **aaa authorization exec default local**
	
	Example:
	
	`Device (config)# aaa authentication login default local`
	`Device (config)# aaa authorization exec default local`
		[Optional] If **aaa new-model** is configured, AAA authentication and authorization is required. For a remote AAA server, replace *local* with your AAA server.

Step 5	**end** **Example:** `Device (config)# end`	Exits global configuration 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.

Procedure

	Command or Action	**Purpose**
Step 1	**enable** **Example:** Device> **enable**	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** Device# **configure terminal**	Enters global configuration mode.
Step 3	**netconf-yang** **Example:** Device (config)# **netconf-yang**	Enables the NETCONF interface on your network device. **Note** After the initial enablement through the CLI, network devices can be managed subsequently through a model based interface. The complete activation of model-based interface processes may require up to 90 seconds.
Step 4	**netconf-yang feature candidate-datastore** **Example:** Device(config)# netconf-yang feature candidate-datastore	Enables candidate datastore.
Step 5	**exit** **Example:** Device (config)# **exit**	Exits global configuration mode.

Configuring NETCONF Options

Configuring SNMP

Enable the SNMP Server in IOS to enable NETCONF to access SNMP MIB data using YANG models generated from supported MIBs, and to enable supported SNMP traps in IOS to receive NETCONF notifications from the supported traps.

Perform the following steps:

Procedure

Step 1   Enable SNMP features in IOS.

Example:

```
configure terminal
logging history debugging
logging snmp-trap emergencies
logging snmp-trap alerts
logging snmp-trap critical
logging snmp-trap errors
logging snmp-trap warnings
logging snmp-trap notifications
logging snmp-trap informational
logging snmp-trap debugging
!
snmp-server community public RW
snmp-server trap link ietf
snmp-server enable traps snmp authentication linkdown linkup
snmp-server enable traps syslog
snmp-server manager
exit
```

Step 2   After NETCONF-YANG starts, enable SNMP Trap support by sending the following RPC <edit-config> message to the NETCONF-YANG port.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <netconf-yang xmlns="http://cisco.com/yang/cisco-self-mgmt">
        <cisco-ia xmlns="http://cisco.com/yang/cisco-ia">
          <snmp-trap-control>
            <trap-list>
              <trap-oid>1.3.6.1.4.1.9.9.41.2.0.1</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.3</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.4</trap-oid>
            </trap-list>
          </snmp-trap-control>
        </cisco-ia>
      </netconf-yang>
    </config>
  </edit-config>
</rpc>
```

Step 3   Send the following RPC message to the NETCONF-YANG port to save the running configuration to the startup configuration.

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
```

```
        <cisco-ia:save-config xmlns:cisco-ia="http://cisco.com/yang/cisco-ia"/>
</rpc>
```

Verifying the NETCONF Protocol Configuration

Use the following commands to verify your NETCONF configuration.

Procedure

Step 1 **show netconf-yang datastores**

Displays information about NETCONF-YANG datastores.

Example:

```
Device# show netconf-yang datastores

Device# show netconf-yang datastores
Datastore Name : running
Globally Locked By Session : 42
Globally Locked Time : 2018-01-15T14:25:14-05:00
```

Step 2 **show netconf-yang sessions**

Displays information about NETCONF-YANG sessions.

Example:

```
Device# show netconf-yang sessions

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore
Number of sessions : 10
session-id transport username source-host global-lock
-------------------------------------------------------------------------------
40 netconf-ssh admin 10.85.70.224 None
42 netconf-ssh admin 10.85.70.224 None
44 netconf-ssh admin 10.85.70.224 None
46 netconf-ssh admin 10.85.70.224 None
48 netconf-ssh admin 10.85.70.224 None
50 netconf-ssh admin 10.85.70.224 None
52 netconf-ssh admin 10.85.70.224 None
54 netconf-ssh admin 10.85.70.224 None
56 netconf-ssh admin 10.85.70.224 None
58 netconf-ssh admin 10.85.70.224 None
```

Step 3 **show netconf-yang sessions detail**

Displays detailed information about NETCONF-YANG sessions.

Example:

```
Device# show netconf-yang sessions detail

R: Global-lock on running datastore
```

```
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions     : 1

session-id             : 19
transport              : netconf-ssh
username               : admin
source-host            : 2001:db8::1
login-time             : 2018-10-26T12:37:22+00:00
in-rpcs                : 0
in-bad-rpcs            : 0
out-rpc-errors         : 0
out-notifications      : 0
global-lock            : None
```

Step 4   **show netconf-yang 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
vtyserverutild  : Running
opdatamgrd      : Running
nginx           : Running
ndbmand         : Running
```

Note   The process *nginx* runs if **ip http secure-server** or **ip http server** is configured on the device. This process is not required to be in the *running* state for NETCONF to function properly. However, the *nginx* process is required for RESTCONF.

Table 13: show platform software yang-management process Field Descriptions

Field	Description
confd	Configuration daemon
nesd	Network element synchronizer daemon
syncfd	Sync from daemon
ncsshd	NETCONF Secure Shell (SSH) daemon
dmiauthd	Device management inteface (DMI) authentication daemon
vtyserverutild	VTY server util daemon
opdatamgrd	Operational Data Manager daemon
nginx	NGINX web server
ndbmand	NETCONF database manager

Additional References for NETCONF Protocol

Related Documents

Related Topic	Document Title
YANG data models for various release of IOS-XE, IOS-XR, and NX-OS platforms	To access Cisco YANG models in a developer-friendly way, please clone the *GitHub repository*, and navigate to the *vendor/cisco* subdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.

Standards and RFCs

Standard/RFC	Title
RFC 6020	*YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*
RFC 6241	*Network Configuration Protocol (NETCONF)*
RFC 6536	*Network Configuration Protocol (NETCONF) Access Control Model*
RFC 8040	*RESTCONF Protocol*

Technical Assistance

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

Feature Information for 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
Candidate Config Support	Cisco IOS XE Fuji 16.9.1	The Candidate Config Support feature enables support for candidate capability by implementing RFC 6241 with a simple commit option. This feature was implemented on the following platforms: • Cisco ASR 1000 Series Aggregated Services Routers • Cisco ASR 900 Series Aggregated Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco CBR-8 Series Routers • Cisco Cloud Services Router 1000V Series • Cisco ISR 4000 Series Integrated Services Routers The following command was introduced: **netconf-yang feature candidate-datastore**.

Feature Name	Release	Feature Information
NETCONF Protocol	Cisco IOS XE Denali 16.3.1	The NETCONF Protocol feature facilitates a programmatic and standards-based way of writing configurations and reading operational data from network devices. The following command was introduced: **netconf-yang**.
	Cisco IOS XE Everest 16.5.1a	This feature was implemented on Cisco Catalyst 9300 Series Switches and Cisco Catalyst 9500 Series Switches.
	Cisco IOS XE Everest 16.6.2	This feature was implemented on Cisco Catalyst 9400 Series Switches.
	Cisco IOS XE Fuji 16.7.1	This feature was implemented on the following platforms: • Cisco ASR 900 Series Aggregated Services Routers • Cisco ASR 920 Series Aggregated Services Routers • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Fuji 16.8.1a	This feature was implemented on Cisco Catalyst 9500-High Performance Series Switches.

Feature Name	Release	Feature Information
NETCONF and RESTCONF IPv6 Support	Cisco IOS XE Fuji 16.8.1a	This feature was implemented on the following platforms: • Cisco ASR 1000 Series Aggregated Services Routers • Cisco ASR 900 Series Aggregated 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 CSR 1000v Switches • Cisco ISR 1100 Series Integrated Services Routers • Cisco ISR 4000 Series Integrated Services Routers
NETCONF Global Lock and Kill Session	Cisco IOS XE Fuji 16.8.1a	This feature was implemented on the following platforms: • Cisco ASR 1000 Series Aggregated Services Routers • Cisco ASR 900 Series Aggregated 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 CSR 1000v Switches • Cisco ISR 1100 Series Integrated Services Routers • Cisco ISR 4000 Series Integrated Services Routers

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

- 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.

RESTCONF and NETCONF in IOS

Protocols and Data Models for Programmatic Device

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.

If a RESTCONF server is co-located with a NETCONF server, then there are protocol interactions with the NETCONF protocol. The RESTCONF server provides access to specific datastores using operation resources, however, the RESTCONF protocol does not specify any mandatory operation resources, each operation resource determine if and how datastores are accessed.

For more information, refer to the Protocols and Data Models for Programmatic Device section in the Catalyst 4500 Series Software Configuration Guide.

HTTPs Methods

The https-based protocol-RESTCONF (RFC 8040), which is a stateless protocol, 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

OPTIONS	SUPPORTED METHODS
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.

- The RESTCONF device uses the restconf api root resource as the initial part of the path in the request URI.

For 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 - http://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2

- fields=name – http://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=0%2F0%2F2?fields=name

- depth=1 - https://10.85.116.59:443/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet?depth=1

- Name and IP - https://10.85.116.59:443/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet/ip/address/primary;name

- MTU (fields) - https://10.104.50.97/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet(mtu)

- MTU - https://10.85.116.59:443/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=3/mtu

- Port-Channel - https://10.85.116.59:443/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:

- 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

Reserved or Unreserved Characters

Conbody

Methods

The content query parameter controls how descendant nodes of the requested data nodes are processed in the reply:

- Must be supported by the server.

- If not present in URI, the default value is: all. Allowed only for GET/HEAD method.

  A "400 Bad Request" status-line is returned if used for other methods or resource types.

Examples for allowed values are:

1. https://10.85.116.59:443/restconf/data/Cisco-IOS-XE-native:native?content=config

2. https://10.85.116.59:443/restconf/data/Cisco-IOS-XE-native:native?content=nonconfig'

Query Parameters (Fields)

- The depth-query parameter is used to limit the depth of subtrees returned by the server.

- The value of the "depth" parameter is either an integer between 1 and 65535 or the string "unbounded".

- Supported if present in the capability URI.

- If not present in URI, the default value is: "unbounded".

- Only allowed for GET/HEAD method.

   A 400 Bad Request status-line is returned if used for other methods or resource types.

Examples:

```
1) 'https://10.85.116.59:443/restconf/data/Cisco-IOS-XE-native:native?content=config&depth=65535'

2) 'https://10.85.116.59:443/restconf/data/Cisco-IOS-XE-native:native?content=nonconfig&depth=0'

>>> resp

<Response [400]>

>>> resp.text

'{"errors": {"error": [{"error-message": "invalid value for depth query parameter",
"error-tag": "malformed-message", "error-type": "application"}]}}\n'

>>>
```

Examples:

- • The "fields" query parameter is used to optionally identify data nodes within the target resource to be retrieved in a GET method.

- Supported if present in the capability URI.

   Allowed only for GET/HEAD method.

- A "400 Bad Request" status-line is returned if used for other methods or resource types.

- A value of the "fields" query parameter matches the following rule:

```
fields-expr = path "(" fields-expr ")" / path ";" fields-expr / path path = api-identifier
  [ "/" path ]
```

- **1.** ";" is used to select multiple nodes.

- **2.** Parentheses are used to specify sub-selectors of a node. Note that there is no path separator character "/" between a "path" field and a left parenthesis character "(".

- **3.** "/" is used in a path to retrieve a child node of a node.

- A value of the "fields" query parameter matches the following rule:

```
fields-expr = path "(" fields-expr ")" / path ";" fields-expr / path path = api-identifier
  [ "/" path ]
```

- **1.** ";" is used to select multiple nodes.

2. Parentheses are used to specify sub-selectors of a node. Note that there is no path separator character "/" between a "path" field and a left parenthesis character "(".

3. "/" is used in a path to retrieve a child node of a node.

Examples:

1. Server module information:
'https://10.85.116.59:443/restconf/data?fields=ietf-yang-library:modules-state/module(name;revision;schema;namespace)'

2. Name and IP:
'https://10.85.116.59:443/restconf/data/Cisco-IOS-XE-native:native/interface?fields=GigabitEthernet/ip/address/primary;name'

Query Parameters (Point)

- The "point" query parameter uses to specify the insertion point for a data resource that is being created or moved within an ordered-by user list or leaf-list.

- Must be supported by the server:

    - Only allowed for POST and PUT methods.

    The value of the "point" parameter is a string that identifies the path to the insertion point object. The format is the same as a target resource URI string.

Examples:

```
PUT:
https://10.85.116.59:443/restconf/data/Cisco-IOS-XE-native/privileg/exec/level-2/command-list-show%20terminal?insert=after&point=%2Cisco-IOS-XE-native%3Anative%2Fprivileg%2Exec%21level%3F%2Fcommand-list-show%20lo%Y

{

  "Cisco-IOS-XE-native:command-list": [

    {

      "command": "show terminal"

    }

  ]
```

Query Parameters (with defaults)

The 'with-defaults' query parameter is used to specify how information about default data nodes is returned in response to GET requests on data resources. Default basic-mode in capability is explicit.

Value	Description
Report-All	All data nodes are reported
Trim	Data nodes set to the YANG default are not reported
Explicit	Data nodes set to the YANG default by the client are reported

- The "point" query parameter uses to specify the insertion point for a data resource that is being created or moved within an ordered-by user list or leaf-list.

Examples:

```
Sync default settings (error):
 'https://10.85.116.59:443/restconf/data/cisco-self-mgmt:netconf-yang/cisco-ia:cisco-ia/cisco-ia:logging/cisco-ia:sync-log-level?with-defaults=report-all'
Intelligent sync (true):
 'https://10.85.116.59:443/restconf/data/cisco-self-mgmt:netconf-yang/cisco-ia:cisco-ia/cisco-ia:intelligent-sync?with-defaults=report-all'
```

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.

Procedure

	Command or Action	**Purpose**
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**aaa new-model** **Example:** `Device(config)# aaa new-model`	Enables AAA.

Step 4	**aaa group server radius** *server-name*
	
	Example:
	
	`Device(config)# aaa group server radius`
	`  ISE`
		Adds the RADIUS server and enters server group RADIUS configuration mode.
		
		• The *server-name* argument specifies the RADIUS server group name.

Step 5	**server-private** *ip-address* **key** *key-name*
	
	Example:
	
	`Device(config-sg-radius)# server-private`
	`  172.25.73.76 key Cisco123`
		Configures a IP address and encryption key for a private RADIUS server.

	Command or Action	Purpose
Step 6	**ip vrf forwarding** *vrf-name* **Example:** Device(config-sg-radius)# ip vrf forwarding Mgmt-intf	Configures the virtual routing and forwarding (VRF) reference of a AAA RADIUS or TACACS+ server group.
Step 7	**exit** **Example:** Device(config-sg-radius)# exit	Exits server group RADIUS configuration mode and returns to global configuration mode.
Step 8	**aaa authentication login default group** *group-name* **local** **Example:** Device(config)# aaa authentication login default group ISE local	Sets the specified group name as the default local AAA authentication during login.
Step 9	**aaa authentication login** *list-name* **none** **Example:** Device(config)# aaa authentication login NOAUTH none	Specifies that no authentication is required while logging into a system.
Step 10	**aaa authorization exec default group** *group-name* **local** **Example:** Device(config)# aaa authorization exec default group ISE local	Runs authorization to determine if an user is allowed to run an EXEC shell.
Step 11	**aaa session-id common** **Example:** Device(config)# aaa session-id common	Ensures that session identification (ID) information that is sent out for a given call will be made identical.
Step 12	**line console** *number* **Example:** Device(config)# line console 0	Identifies a specific line for configuration and enter line configuration mode.
Step 13	**login authentication** *authentication-list* **Example:** Device(config-line)# login authentication NOAUTH	Enables AAA authentication for logins.
Step 14	**end** **Example:** Device(config-line)# end	Exits line configuration mode and returns to privileged EXEC mode.

Enabling Cisco IOS HTTP Services for RESTCONF

Perform this task to use the RESTCONF interface.

Procedure

	Command or Action	**Purpose**
Step 1	**enable** **Example:** Device> enable	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** Device# configure terminal	Enters global configuration mode.
Step 3	**restconf** **Example:** Device(config)# restconf	Enables the RESTCONF interface on your network device.
Step 4	**ip http secure-server** **Example:** Device(config)# ip http secure-server	Enables a secure HTTP (HTTPS) server.
Step 5	**end** **Example:** Device(config)# end	Exits global configuration mode and enters privileged EXEC mode

Verifying RESTCONF Configuration

When a device boots up with the startup configuration, the *nginx* process will be running. However; DMI 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 serve,r and the request is transferred to the confd web server for further syntax/semantics check.

The following sample output from the **show platform software yang-management process** command shows the status of the all processes when a device is booted with the startup-configuration:

```
Device# show platform software yang-management process

confd             : Not Running
nesd              : Not Running
syncfd            : Not Running
ncsshd            : Not Running
dmiauthd          : Not Running
nginx             : Running
ndbmand           : Not Running
pubd              : Not Running
```

The *nginx* process gets restrated and DMI process are started, when the **restconf** command is configured.

The following sample output from the **show platform software yang-management process** command shows that the *nginx* process and DMI processes are up and running:

```
Device# show platform software yang-management process

confd             : Running
nesd              : Running
syncfd            : Running
ncsshd            : Not Running ! NETCONF-YANG is not configured, hence ncsshd process is
in not running.
dmiauthd          : Running
vtyserverutild    : Running
opdatamgrd        : Running
nginx             : Running ! nginx process is up due to the HTTP configuration, and it is
restarted when RESTCONF is enabled.
ndbmand           : Running
```

The following sample output from the show platform software yang-management process monitor command displays detailed information about all processes:

```
Device#show platform software yang-management process monitor

COMMAND           PID S     VSZ    RSS %CPU %MEM     ELAPSED
confd             28728 S 860396 168496 42.2  4.2     00:12
confd-startup.s   28448 S  19664   4496  0.2  0.1     00:12
dmiauthd          29499 S 275356  23340  0.2  0.5     00:10
ndbmand           29321 S 567232  65564  2.1  1.6     00:11
nesd              29029 S 189952  14224  0.1  0.3     00:11
nginx             29711 S 332288  18420  0.6  0.4     00:09
nginx             29717 S 337636  12216  0.0  0.3     00:09
pubd              28237 S 631848  68624  2.1  1.7     00:13
syncfd            28776 S 189656  16744  0.2  0.4     00:12
```

After AAA and the RESTCONF interface is configured, and nginx process and relevant DMI processes are running; the device is ready to receive RESTCONF requests.

Use the **show netconf-yang sessions** command to view the status of NETCONF/RESTCONF sessions:

```
Device# show netconf-yang sessions

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions : 1

session-id  transport    username              source-host           global-lock
```

```
--------------------------------------------------------------------------------
19          netconf-ssh  admin                2001:db8::1            None
```

Use the **show netconf-yang sessions detail** command to view detailed information about
NETCONF/RESTCONF sessions:

```
Device# show netconf-yang sessions detail

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions      : 1

session-id              : 19
transport               : netconf-ssh
username                : admin
source-host             : 2001:db8::1
login-time              : 2018-10-26T12:37:22+00:00
in-rpcs                 : 0
in-bad-rpcs             : 0
out-rpc-errors          : 0
out-notifications       : 0
global-lock             : None
```

Configuration Examples for 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 availabel in the running configuration. |

The following sample POST request uses the **logging monitor alerts** command.

```
Device:~# curl -i -k -X "POST"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor" \
>        -H 'Content-Type: application/yang-data+json' \
>        -H 'Accept: application/yang-data+json' \
>        -u 'admin:admin' \
>        -d $'{
>    "severity": "alerts"
> }'
HTTP/1.1 201 Created
Server: nginx
Date: Mon, 23 Apr 2018 14:53:51 GMT
Content-Type: text/html
Content-Length: 0
Location:
https://10.85.116.30/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 14:53:51 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-495231-97239
Pragma: no-cache

Device:~#
```

PUT: (Create or Replace) Request:

If the specified command is not present on the device, the POST request creates it ; however, if it is already present in the running configuration, the command will be replaced by this request.

The following sample PUT request uses the **logging monitor warnings** command.

```
Device:~# curl -i -k -X "PUT"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>        -H 'Content-Type: application/yang-data+json' \
>        -H 'Accept: application/yang-data+json' \
>        -u 'admin:admin' \
>        -d $'{
>    "severity": "warnings"
> }'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 14:58:36 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 14:57:46 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-495466-326956
Pragma: no-cache

Device:~#
```

PATCH: (Update) Request

The following sample PATCH request uses the **logging monitor informational** command.

```
Device:~# curl -i -k -X "PATCH"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native" \
>       -H 'Content-Type: application/yang-data+json' \
>       -H 'Accept: application/yang-data+json' \
>       -u 'admin:admin' \
>       -d $'{
>    "native": {
>      "logging": {
>        "monitor": {
>          "severity": "informational"
>        }
>      }
>    }
> }'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 15:07:56 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 15:07:56 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-496076-273016
Pragma: no-cache
Device:~#
```

GET Request (To Read)

The following sample GET request uses the **logging monitor informational** command.

```
Device:~# curl -i -k -X "GET"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>       -H 'Accept: application/yang-data+json' \
>       -u 'admin:admin'
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 23 Apr 2018 15:10:59 GMT
Content-Type: application/yang-data+json
Transfer-Encoding: chunked
Connection: keep-alive
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Pragma: no-cache

{
  "Cisco-IOS-XE-native:severity": "informational"
}
Device:~#
```

DELETE Request (To Delete the Configuration)

```
Device:~# curl -i -k -X "DELETE"
"https://10.85.116.30:443/restconf/data/Cisco-IOS-XE-native:native/logging/monitor/severity"
 \
>      -H 'Content-Type: application/yang-data+json' \
>      -H 'Accept: application/yang-data+json' \
>      -u 'admin:admin'
HTTP/1.1 204 No Content
Server: nginx
Date: Mon, 23 Apr 2018 15:26:05 GMT
Content-Type: text/html
Content-Length: 0
Connection: keep-alive
Last-Modified: Mon, 23 Apr 2018 15:26:05 GMT
Cache-Control: private, no-cache, must-revalidate, proxy-revalidate
Etag: 1524-497165-473206
Pragma: no-cache

linux_host:~#
```

Additional References for the RESTCONF Protocol

Related Documents

Related Topic	Document Title
YANG data models for various releases of IOS XE, IOS XR, and NX-OS platforms	To access Cisco YANG models in a developer-friendly way, please clone the GitHub repository, and navigate to the vendor/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
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	https://www.cisco.com/c/en/us/support/index.html

Feature Information for the RESTCONF Protocol

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

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

Table 15: Feature Information for RESTCONF Protocol

Feature Name	Releases	Feature Information
RESTCONF Protocol	Cisco IOS XE Everest 16.6.1	This chapter describes how to set-up and configure an HTTP-based protocol- 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 event notifications defined in the YANG model. This feature was introduced on the ASR 1000 Aggregation Services Routers-ASR1001-HX and ASR1002-HX, CSR 1000v Series Cloud Services Router, and Cisco 4000 Series Integrated Services Routers (ISRs). The following commands were introduced or modified: **ip http server** and **restconf**

Feature Name	Releases	Feature Information
	Cisco IOS XE Fuji 16.8.1a	This feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches

Operational Data Parser Polling

YANG data models enables you to read operational state data from devices.

Information About Operational Data Parser Polling

Operational Data Overview

You can use YANG data models to read operational state data from a device. The operational data allows you to determine the current state and behavior of a device, similar to IOS **show** commands.

You can perform NETCONF GET operations to retrieve read-only operational state data from a system. You must enable NETCONF, activate data parsers (where applicable), and then retrieve the data through an appropriate YANG model.

The *How to Configure Operational Data* section provides information on configuring operational data through a programmable interface and the CLI.

Operational Data Parsers and Corresponding YANG Models

There are two types of operational data parsers; one that is always on, and the other that must be configured to poll operational data at regular intervals. For the first type of operational data parser, no configuration is required. Data is always fetched from the device during a NETCONF GET request. These data parsers do not have a polling-interval, and operational data is updated as soon as a change occurs.

The second type of operational data parsers must be activated either via the CLI or a NETCONF message (For more information, see the *How to Enable Operational Data Parser Polling* section.). The operational data for these types of parsers is polled at regular polling intervals and this information is retrieved during a NETCONF GET request.

The following table lists the data parsers that must be activated, and the corresponding YANG model where the operational data is stored.

Table 16: Operational Data Parsers to be Activated and Corresponding Yang Models

Operational Data Parser Name	YANG Model to Access Operational Data
BGP	Cisco-IOS-XE-bgp-oper.yang
BFD	Cisco-IOS-XE-bfd-oper.yang
BridgeDomain	Cisco-IOS-XE-bridge-domain.yang **Note** Supported only on routing platforms.
DiffServ	ietf-diffserv-target.yang
EthernetCFMStats	Cisco-IOS-XE-cfm-oper.yang **Note** Supported only on routing platforms.
FlowMonitor	Cisco-IOS-XE-flow-monitor-oper.yang
IPRoute	ietf-routing.yang
MPLSLForwarding	Cisco-IOS-XE-mpls-fwd-oper.yang
MPLSLDPNeighbor	Cisco-IOS-XE-mpls-ldp.yang
MPLSStaticBinding	common-mpls-static.yang
OSPF	ietf-ospf.yang
PlatformSoftware	Cisco-IOS-XE-platform-software-oper.yang
VirtualService	Cisco-IOS-XE-virtual-service-oper.yang **Note** Supported only on routing platforms.

How to Enable Operational Data Parser Polling

Enabling Operational Data Parser Polling Through a Programmable Interface

Perform this task to enable operational data parser polling through a programmable interface:

1. After enabling NETCONF-YANG, send an <edit-config> remote procedure call (RPC) using cisco-odm.yang (available in the GitHub Repository) to enable operational data polling. When the polling is enabled, all operational data parsers are activated by default. The default polling-interval of each parser is 120 seconds (120000 milliseconds). The polling interval decides the frequency at which the parser obtains the operational data and updates the corresponding YANG model in the datastore.

2. After operational data polling is enabled, send a <get> RPC to obtain the operational data. Use the parser-to-YANG model mapping to determine which operational YANG model should be used to retrieve the operational data. The following RPC reply fetches access control list (ACL) operational data using Cisco-IOS-XE-acl-oper.yang:

```
          CORRESPONDING RPC REPLY:
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
    <data>
        <access-lists xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-acl-oper">
            <access-list>
                <access-control-list-name>TEST</access-control-list-name>
                <access-list-entries>
                    <access-list-entry>
                        <rule-name>10</rule-name>
                        <access-list-entries-oper-data>
                        <match-counter>100</match-counter>
                    </access-list-entry>
                    <access-list-entry>
                        <rule-name>20</rule-name>
                        <access-list-entries-oper-data>
                        <match-counter>122</match-counter>
                    </access-list-entry>
                </access-list-entries>
            </access-list>
        </access-lists>
    </data>
</rpc-reply>
```

Note For more information, see the cisco-odm.yang model in the *GitHub repository*.

Enabling Operational Data Parser Polling Through the CLI

After enabling NETCONF-YANG, perform this task to enable operational data parser polling and to adjust the polling interval.

Procedure

	Command or Action	**Purpose**
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**netconf-yang cisco-odm polling-enable** **Example:** `Device(config)# netconf-yang cisco-odm polling-enable`	Enables operational data polling.
Step 4	**netconf-yang cisco-odm actions** *action-name* **Example:**	Enables the specified action, and enters ODM-action configuration mode.

	Command or Action	Purpose
	`Device(config)# netconf-yang cisco-odm actions OSPF`	• Specify the operational data parser name to retrieve operational data.
Step 5	**mode poll** **Example:** `Device(config-odm-action)# mode poll`	Configures the data parser in poll mode.
Step 6	**polling-interval** *seconds* **Example:** `Device(config-odm-action)# polling-interval 1000`	Changes the default parser-polling interval. • To stop the parser from polling data, configure the **mode none** command.
Step 7	**end** **Example:** `Device(config-odm-action)# end`	Exits ODM-action configuration mode and returns to privileged EXEC mode.

What to do next

After enabling operational data polling, send a <get> RPC to obtain operational data from the device.

Additional References for Operational Data Parser Polling

Related Documents

Related Topic	Document Title
YANG data models for Cisco IOS XE	To access Cisco YANG models in a developer-friendly way, please clone the *GitHub repository*, and navigate to the *vendor/cisco* subdirectory.
	Programmability Command Reference, Cisco IOS XE Everest 16.6.1

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

Feature Information for Operational Data Parser Polling

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 17: Feature Information for Operational Data Parser Polling

Feature Name	Release	Feature Information
Operational Data Parser Polling	Cisco IOS XE Denali 16.3.1	YANG data models, enables you to read operational state data from a device.
	Cisco IOS XE Everest 16.5.1a	This feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches

Model-Driven Telemetry

Model-Driven Telemetry

Model-driven telemetry allows network devices to continuously stream real time configuration and operating state information to subscribers.

Applications can subscribe to specific data items they need, by using standard-based YANG data models over NETCONF-YANG.

Structured data is published at a defined cadence, or on-change, based upon the subscription criteria and data type.

Prerequisites for Model-Driven Telemetry

- Knowledge of NETCONF-YANG and how to use it, including:

    - Establishing a NETCONF session.

    - Sending/receiving hello and capabilities messages.

    - Sending/receiving YANG XML remote procedure calls (RPCs) over the established NETCONF session. For more information, see the *Configuration Example for NETCONF-YANG*.

    For more information on NETCONF-YANG, see the *Datamodels* chapter.

- Knowledge of XML, XML namespaces, and XML XPath.

- Knowledge of standards and principles defined by the IETF dynamic telemetry specification.

- NETCONF-YANG must be configured and running on the device. 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
```

```
vtyserverutild   : Running
opdatamgrd       : Running
nginx            : Running
ndbmand          : Running
pubd             : Running
```

Note	The process *pubd* is the model-driven telemetry process, and if it is not running, model-driven telemetry will not work.

- The *urn:ietf:params:netconf:capability:notification:1.1* capability must be listed in the hello message. This capability is advertised only on devices that support IETF telemetry.

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 receiving equipment for monitoring. Model-driven telemetry provides a mechanism to stream data from a model-driven telemetry-capable device to a destination.

Telemetry uses a subscription model to identify information sources and destinations. Model-driven telemetry replaces the need for the periodic polling of network elements; instead, a continuous request for information to be delivered to a subscriber is established upon the network element. Then, either periodically, or as objects change, a subscribed set of YANG objects are streamed to that subscriber.

The data to be streamed is driven through subscription. Subscriptions allow applications to subscribe to updates (automatic and continuous updates) from a YANG datastore, and this enables the publisher to push and in effect stream those updates.

Subscription Overview

Subscriptions are items that create associations between telemetry roles, and define the data that is sent between them.

Specifically, a subscription is used to define the set of data that is requested as part of the telemetry data; when the data is required, how the data is to be formatted, and, when not implicit, who (which receivers) should receive the data.

Even though the maximum number of supported subscriptions is platform-dependent, currently 100 subscriptions are supported. The subscriptions can be either configured or dynamic, and use any combination of transport protocols. If too many subscriptions are operating at the same time to allow all the valid configured subscriptions to be active, the removal of an active subscription will cause one of the inactive but valid configured subscriptions to be attempted. Periodic triggered subscriptions (100 centiseconds is the default minimum) and on-change triggered subscriptions are supported.

NETCONF and other northbound programmable interfaces (such as RESTCONF or gNMI) are supported to configure subscriptions.

Two types of subscriptions are used in telemetry on Cisco IOS XE systems: dynamic and configured subscriptions.

Because dynamic subscriptions are created by clients (the subscriber) that connect into the publisher, they are considered dial-in. Configured subscriptions cause the publisher to initiate connections to receivers, and as a result, they are considered dial-out.

Sample <establish-subscription> RPC

The following is a sample <establish-subscription> RPC. The stream, xpath-filter, and period fields in the RPC are mandatory.

```
<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>
```

YANG-Push

YANG-push is the subscription and push mechanism for YANG databases. YANG-push subscriptions are defined using a data model. Using YANG-push, subscriber applications can request a continuous, customized stream of updates from YANG databases. The YANG-push encompasses all data in the configuration and operational databases that is described by the YANG model installed on a device. You must provide a filter for data, as subscription to all data is not supported.

Note The yang-push stream must be specified.

XPath Filter Support

The XML Path Language (XPath) filter specifies the information element to subscribe to. It informs the telemetry parser where the required subscription information is located in the data model. The update-filter grouping of the XPath filter is supported for subscriptions.

Periodic Publication

With periodic subscription, 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.

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 interval is 100, or one second. There is no default value. This value must be explicitly set in the <establish subscription> RPC.

Subscriptions for data that does not currently exist are permitted and run as normal subscriptions. When subscribed for empty data, empty update notifications are sent at the requested period.

Periodic updates contain a full copy of the subscribed data element or table.

RPC Support in Telemetry

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	<establish-subscription> <delete-subscription>	Success
error-no-such-subscription	<delete-subscription>	The specified subscription does not exist.
error-no-such-option	<establish-subscription>	The requested subscription is not supported.
error-insufficient-resources	<establish-subscription>	A subscription cannot be created because of the following reasons: • There are too many subscriptions. • The amount of data requested is considered too large. • The interval for a periodic subscription is too small.
error-other	<establish-subscription>	Some other error.

NETCONF Sessions in Telemetry

Telemetry subscriptions and updates are transmitted over NETCONF sessions. The NETCONF session that is used to establish a telemetry subscription receives the telemetry updates. If the NETCONF session is torn down or the connection is lost, associated telemetry subscriptions are also torn down.

All sessions are NETCONF sessions and as a result, all session limitations are specific to the NETCONF implementation.

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 router. 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 subscriptions after a switchover.

Sample Model-Driven Telemetry RPCs

Creating a Subscription

Subscriptions are created using XML RPCs over an established NETCONF session. The <establish-subscription> RPC is sent from an IETF telemetry client or collector to the network device. The stream, xpath-filter, and period fields in the RPC are mandatory.

The following is a sample subscription to the operational database subscriptions table:

```
<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>
```

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:

```
<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 as 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:
```

Deleting a Subscription

You can delete a telemetry subscription in two ways. One is by sending a <delete-subscription> RPC with the subscription ID in the subscription-id tag, which only a subscriber can do. Also, a subscription is 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.

```
<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>
```

Additional References for Model-Driven Telemetry

Related Documents

Related Topic	Document Title
NETCONF-YANG patches	https://tools.ietf.org/wg/netconf/draft-ietf-netconf-yang-patch/

Related Topic	Document Title
YANG Explorer	https://github.com/CiscoDevNet/yang-explorer

Standards and RFCs

Standard/RFC	Title
NETCONF Support for Event Notifications	draft-ietf-netconf-netconf-event-notifications-01
RFC 6241	Network Configuration Protocol (NETCONF)
Subscribing to Event Notifications	draft-ietf-netconf-rfc5277bis-01
Subscribing to YANG Datastore Push Updates	draft-ietf-netconf-yang-push-04

Technical Assistance

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

Feature Information for Model-Driven Telemetry

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

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

Table 18: Feature Information for Model-Driven Telemetry

Feature Name	Release	Feature Information
Model-Driven Telemetry	Cisco IOS XE Everest 16.6.1	Model-driven telemetry allows network devices to continuously stream real time configuration and operating state information to subscribers. This feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches
	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.

In Service Model Update

This module describes how to update the YANG data models on a device through an In Service Model Update.

This module contains the following sections:

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 image basis.

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

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 2: Committing a Model Update Package
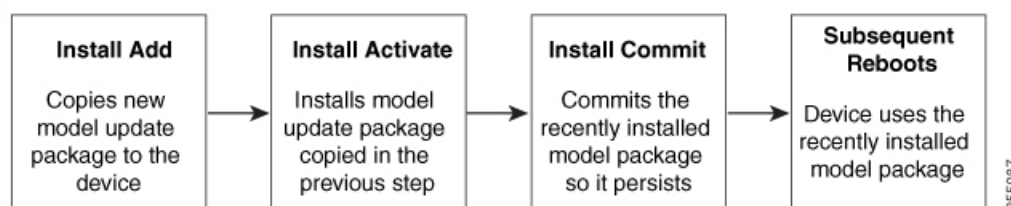
If NETCONG-YANG is enabled during package activation, NETCONF processes are restarted. All active NETCONF sessions are killed during package activation. Failure during a package verification terminates the activation process.

Deactivating the Update Package

You can deactivate an update package by using the **install deactivate** command. Enable the **install commit** command to make changes persistent.

Table 19: 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

Procedure

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.
	Example:	• Enter your password if prompted.
	Device> enable	

	Command or Action	**Purpose**
Step 2	**install add file tftp:** *filename* **Example:** `Device# install add file tftp://172.16.0.1//tftpboot/folder1/ isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`	Copies the model update package from a remote location (via FTP, TFTP) to the device, and performs a compatibility check for the platform and image versions. • You can use other methods to copy the update package from the remote location to the device, however; you still have to execute the **install add** command before the package is activated.
Step 3	**install activate file bootflash:** *filename* **Example:** `Device# install activate file bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`	Validates whether the update package is added through the **install add** command, and restarts the NETCONF processes. • Perform the **install add** operation prior to activating an update package.
Step 4	**install commit** **Example:** `Device# install commit`	Makes the changes persistent over reload. • NETCONF processes are not restarted.
Step 5	**install deactivate file bootflash:** *filename* **Example:** `Device# install deactivate file bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`	Deactivates the specified update package, and restarts the NETCONF processes.
Step 6	**install commit** **Example:** `Device# install commit`	Makes the changes persistent over reload. • NETCONF processes are not restarted.
Step 7	**install rollback to** {**base** | **committed** | **id** *commit-ID*} **Example:** `Device# install rollback to base`	Rollbacks the update to the base version, the last committed version, or a known commit ID, and restarts NETCONF processes. • Valid values for the *commit-id* argument are from 1 to 4294967295. • Older versions of data models updates are available for use.
Step 8	**install remove** {**file bootflash:** *filename* | **inactive**} **Example:** `Device# install remove file bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin`	Removes the specified update package from the bootflash. • A package must be deactivated before it is removed.
Step 9	**show install summary**	Displays information about the active package.

Command or Action	Purpose
Example: `Device# show install summary`	• 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 following is sample output from the **show install summary** command after adding an update package file to the device:

```
Device# show install summary

Active Packages:
No packages
Inactive Packages:
bootflash: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Committed Packages:
No packages
Uncommitted Packages:
No packages
Device#
```

The following example shows how to activate an added update package file:

```
Device# install activate file bootflash:
isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin

install_activate: START Sun Feb 26 05:58:41 UTC 2017
DMP package.
Netconf processes stopped
SUCCESS: install_activate /bootflash/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Sun Feb 26 05:58:58 UTC 2017*Feb 26 05:58:47.655: %DMI-4-CONTROL_SOCKET_CLOSED:
SIP0: nesd: Confd control socket closed Lost connection to ConfD (45): EOF on socket to
ConfD.
*Feb 26 05:58:47.661: %DMI-4-SUB_READ_FAIL: SIP0: vtyserverutild:
```

```
Confd subscription socket read failed Lost connection to ConfD (45):
EOF on socket to ConfD.
*Feb 26 05:58:47.667: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: syncfd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 05:59:43.269: %DMI-5-SYNC_START: SIP0: syncfd:
External change to running configuration detected.
The running configuration will be synchronized to the NETCONF running data store.
*Feb 26 05:59:44.624: %DMI-5-SYNC_COMPLETE: SIP0: syncfd:
The running configuration has been synchronized to the NETCONF running data store.
Device#
```

The following sample output from the **show install summary** command displays the status of the model package as active and uncommitted:

```
Device# show install summary

Active Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Inactive Packages:
No packages
Committed Packages:
No packages
Uncommitted Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Device#
```

The following example shows how to execute the **install commit** command:

```
Device# install commit

install_commit: START Sun Feb 26 06:46:48 UTC 2017
SUCCESS: install_commit Sun Feb 26 06:46:52 UTC 2017
Device#
```

The following sample output from the **show install summary** command displays that the update package is now committed, and that it will be persistent across reloads:

```
Device# show install summary

Active Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Inactive Packages:
No packages
Committed Packages:
bootflash:isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Uncommitted Packages:
No packages
Device#
```

The following example shows how to rollback an update package to the base package:

```
Device# install rollback to base

install_rollback: START Sun Feb 26 06:50:29 UTC 2017
7 install_rollback: Restarting impacted processes to take effect
7 install_rollback: restarting confd
*Feb 26 06:50:34.957: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: syncfd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 06:50:34.962: %DMI-4-CONTROL_SOCKET_CLOSED: SIP0: nesd:
Confd control socket closed Lost connection to ConfD (45): EOF on socket to ConfD.
*Feb 26 06:50:34.963: %DMI-4-SUB_READ_FAIL: SIP0: vtyserverutild:
```

```
Confd subscription socket read failed Lost connection to ConfD (45):
EOF on socket to ConfD.Netconf processes stopped
7 install_rollback: DMP activate complete
SUCCESS: install_rollback Sun Feb 26 06:50:41 UTC 2017
*Feb 26 06:51:28.901: %DMI-5-SYNC_START: SIP0: syncfd:
External change to running configuration detected.
The running configuration will be synchronized to the NETCONF running data store.
*Feb 26 06:51:30.339: %DMI-5-SYNC_COMPLETE: SIP0: syncfd:
The running configuration has been synchronized to the NETCONF running data store.
Device#
```

The following is sample output from the **show install package** command:

```
Device# show install package bootflash:
isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin

Name: isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Version: 16.5.1.0.199.1484082952..Everest
Platform: ISR4300
Package Type: dmp
Defect ID: CSCxxxxxxx
Package State: Added
Supersedes List: {}
Smu ID: 1
Device#
```

The following sample NETCONF hello message verifies the new data model package version:

```
Getting Capabilities: (admin @ 172.16.0.1:830)
PROTOCOL netconf
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
<capability>http://tail-f.com/ns/netconf/extensions</capability>
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=
explicit&amp;also-supported=report-all-tagged</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?
revision=2011-06-01&amp;module=ietf-netconf-with-defaults</capability>
<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-aaa?module=
Cisco-IOS-XE-aaa&amp;revision=2017-02-07</capability>
<<capability>http://cisco.com/ns/yang/Cisco-IOS-XE-native?module=
Cisco-IOS-XE-native&amp;revision=2017-01-07&amp;features=virtual-
template,punt-num,multilink,eth-evc,esmc,efp,dot1x</capability>
Device#
```

The following is sample output from the **show install log** command:

```
Device# show install log

[0|install_op_boot]: START Fri Feb 24 19:20:19 Universal 2017
[0|install_op_boot]: END SUCCESS Fri Feb 24 19:20:23 Universal 2017
[3|install_add]: START Sun Feb 26 05:55:31 UTC 2017
```

```
[3|install_add( FATAL)]: File path (scp) is not yet supported for this command
[4|install_add]: START Sun Feb 26 05:57:04 UTC 2017
[4|install_add]: END SUCCESS /bootflash/isr4300-universalk9.16.05.01.CSCxxxxxxx.dmp.bin
Sun Feb 26 05:57:22 UTC 2017
[5|install_activate]: START Sun Feb 26 05:58:41 UTC 2017
Device#
```

The sample image used in the following examples are a Cisco Catalyst 3000 Series Switch image.

The following example shows how to add a model update package file:

```
Device# install add file tftp://172.16.0.1//tftpboot/folder1/
cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin

install_add: START Sat Jul 29 05:57:04 UTC 2017
Downloading file tftp://172.16.0.1//tftpboot/folder1/
cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Finished downloading file tftp://172.16.0.1//tftpboot/folder1/
cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.SPA.smu.bin
to bootflash:cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
SUCCESS: install_add /bootflash/cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Sat Jul 29 05:57:22 UTC 2017
Device#
```

The following sample output from the **show install summary** command displays that the update package is now committed, and that it will be persistent across reloads:

```
Device# show install summary

Active Packages:
bootflash:cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Inactive Packages:
No packages
Committed Packages:
bootflash:cat3k_caa-universalk9.16.06.01.CSCxxxxxxx.dmp.bin
Uncommitted Packages:
No packages
Device#
```

Feature Information for In Service Model Update

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

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

Table 20: Feature Information for In Service Model Update

Feature Name	Release	Feature Information
In Service Model Update	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	This module describes how to update YANG data models through In Service Model Update. In Cisco IOS XE Everest 16.5.1a, this feature was implemented on the following platforms: • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco Cloud Services Router 1000v • Cisco Integrated Services Virtual Routers (ISRv) The following commands were introduced or updated: **install (Programmability)**, **show install (Programmability)**.
	Cisco IOS XE Everest 16.6.1	In Cisco IOS XE Everest 16.5.1b, this feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches